SPOSR: A System for Mining Partially-Ordered Sequential Rules

Mr. Sandipkumar C. Sagare[1], Prof. Dr. S. K. Shirgave [2]
M.E (CSE) Student, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji[1].
Professor, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji[2].
sagaresandip999@gmail.com[1]
skshirgave@yahoo.com[2]

Abstract ─ Sequential rule mining is used to extract important data in various application such as stock market analysis, e-commerce. It generally includes identifyingsequential rules from given sequence database which will be common in multiple sequences. Partially Ordered Sequential rules (POSR) is a type of sequential rules in which the items in left and right side of the sequential rule does not need to be ordered. The existing approaches used for mining POSRinclude RuleGrowth Algorithm, TRuleGrowth Algorithm.But, these approaches either not use sliding window size constraint (RuleGrowth) or take more execution time even after using the sliding window size constraint (TRuleGrowth).This paper presents SPOSR, a system for mining partially –ordered sequential rules based on these techniques called RuleGrowth and TRuleGrowth These techniques take the sequence database as input and applies minimum support, minimum confidence, and window size constraints respectively to generate partially-ordered sequential rules. The experimental evaluation in terms of number of rules generated and execution time is conducted to compare these techniques. It is found that the TRuleGrowth performs better in terms of sequential rule count and the execution time.

Keywords ─ Sequential rules, window size.

1. INTRODUCTION

There are important data mining and machine learning application areas where the data to be analyzed consists of a sequence of events.Identifying relationships between occurrences of events stored in database is important in various domains as it provides understanding of relations of events to predict the next event [2]. In the field of data mining, one of the most popular set of techniques for discovering temporal relations between events in discrete time series is sequential pattern mining [1], [3] which consists of finding sequences of events that appear frequently in a sequence database.That is it discovers the subsequences that appear in sequence database having support not less than threshold value of support set by the user [1][2][3].

Many algorithms were developed for mining standard sequential rules. CMRules [4] is an algorithm for mining sequential rules common to several sequences in sequence databases. Because the algorithm is based on association rule mining it is very efficient.It can be used to discover both association rules and sequential rules in a databaseat the same time. Moreover, it gives possibility to enhance the capability of CMRules by choosing a particular association rule mining algorithm having this capability.

For mining POSR, RuleGrowth approach [7] was presented which includespattern-growth approach for mining POSR common to several sequences. This approach does not use the previous techniques of generating candidate and then testing them. Instead it uses pattern-growth approach for discovering more efficient and valid rules. It finds rules in incremental fashion. It starts with two items and then grows by scanning the database to expand their left or right part of rule. RuleGrowth takes as input a sequence database S and a minsup and minconf thresholds. TRuleGrowth approach [8] is proposed to takes extra input to RuleGrowth in the form of window size. As in extension to RuleGrowth algorithm, TRuleGrowth algorithm uses window size to discover the rules occurring within the sliding window. This constraint enforced for rule size of 1*1. Accordingly left and right side of the rule is modified. This can be said as anextension of RuleGrowth algorithm as some modifications have been made in the RuleGrowth algorithm to ensure that the sliding window constraint is taken into the account when generating rules.The pattern growth approach [7] is extended by applying a constraint, called sliding-window constraint [8]. This extended approach discovers the sequential rules while verifying that they occur in a sliding-window. Discovering rules appearing in a sliding-window has several important benefits. First, it can decrease the execution time by several orders of magnitude by pruning the search space. Second, it can produce a much smaller set of rules, thus reducing the disk space requirement for storing rules found and making it easier for the user to analyze results [8].

The System for mining partially-ordered sequential rules accepts the sequence database as input and generates the sequential rules based on the given values of minimum support, minimum confidence and window size.Finally these two approaches are to be analyzed by applying on real-life dataset containing set of sequences.

2. RELATED WORK

2.1 Present Theory and Practices:

P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. MephuNguifo [4] presented CMRules algorithm. CMRules algorithm can be used to mine sequential rules which will be common to several sequences in sequence databases. The algorithm is based on association rule mining. So, it is very efficient if one wants to generate both association rules as well as sequential rules in a database, as both can be discovered at the same time. Moreover, it gives the possibility to enhance the capability of CMRules by choosing a particular association rule mining algorithm that has this capability.

P. Fournier-Viger, T. Gueniche, and V. S. Tseng [5] investigated the Prediction of the next element(s) of a sequence which is a research problem having wide applications such as consumer product recommendation, Web link recommendation and stock market prediction. They suggested discovering sequential rules as symbolic sequence recommendation to make predictions from generated rules on training set of sequences. They explored a new type of sequential rules called partially-ordered sequential rules for prediction of the sequence. The prediction accuracy of these rules was compared with standard sequential rules for the task of webpage recommendation under multiple scenarios. They showed the improvement in prediction accuracy and matching rate after using partially-ordered sequential rules instead of standard sequential rules. The limitation of this work was that the execution time was not considered in these experiments.

M. J. Zaki [6] developed SPADE algorithm for fast mining of sequential patterns in large databases. Approaches that have been used before SPADE have made multiple database scans and used complex hash-tree structures that can have sub-optimal locality; SPADE algorithm decomposes the given original problem into smaller sub-problems. For that, it uses equivalence classes on frequent sequences. Here each of the equivalence class is solved independently. Also it is very likely that it can be processed in main-memory. Thus SPADE algorithm usually makes use of only three scans of the database – first scan for frequent 1-sequences, second scan for frequent 2-sequences, and third scan to generate all other frequent sequences. In the case in which supports of 2-sequences are available then one scan is sufficient.

P. Fournier-Viger, R. Nkambou, and V. S. Tseng [7] presented RuleGrowth, a novel algorithm for mining sequential rules common to several sequences. Unlike previous algorithms, it does not use a generate-candidate-and-test approach. Instead, it uses a pattern-growth approach for discovering valid rules such that it can be much more efficient and scalable. It first finds rules between two items and then recursively grows them by scanning the database for single items that could expand their left or right parts. This approach is evaluated by comparing it with the CMDeo and CMRules [4] algorithms.

Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng, Longbing Cao, and Roger Nkambou [8] developed two algorithms. First is RuleGrowth which is an algorithm to mine thesequential rules common to multiple sequences. It makes use of pattern-growth approach todiscover valid sequential rules suchthat it avoids considering the rules that does not appear inthe database. The second algorithm TRuleGrowth allows the userto give a sliding-windowconstraint for the rules to be mined. To evaluate the RuleGrowth algorithm and TRuleGrowth algorithm, several experiments were performed to assessthe influence of minsup, minconfparameters on the performance of each algorithm. Also, RuleGrowth was compared withTRuleGrowth for different window size valuesfor evaluating the benefits of using the windowsize constraint.

2.2 Summary of Discussion:

It is observed that the initial work focused on mining of sequential patterns based on only the support parameter. It didn’t consider the confidence factor which can be important for making the decisions or predictions. Later the sequential rule mining algorithms came into picture involving both support and confidence as important factors for generation of rules. But it is found that the standard sequential rules were too specific to be used for making important decisions or predictions. So, a new type of sequential rules has been developed later called partially-ordered sequential rules (POSR) [10].Users often only wish to discover rules occurring within a maximum amount of time [2] [6] [7]. This motivated to discover sequential rules occurring within a sliding-window, which can be very useful for the discovery of temporal patterns for many real-life applications such as analyzing stock market data. Interest in mining rules common to multiple sequences is motivated by the fact that it has many potential applications such as mining sequential rules common to several customer transactions to make product recommendations.

3. SYSTEM ARCHITECTURE:

Figure 3.1 shows architecture of the proposed system**.** The system consists of two major components:

I. RuleGrowth algorithm
II. TRuleGrowth algorithm

I. RuleGrowth:

RuleGrowth [7] is the first component of system which takes sequence database, minsup and minconf as an input. A sequence database SD [1] [8] is nothing but a set of sequences SD = {s1, s2, …, sm} and a finite set of items I={ i1, i2,.., in} occurring in these sequences, where each sequence has been assigned a unique identifier SID (Sequence ID).A sequence is nothing but an ordered list of item sets Sx = I1, I2,.., Ip such that I1, I2,.., Ip are subsets of I. minsup is a minimum support threshold and minconf is a minimum confidence threshold [8].

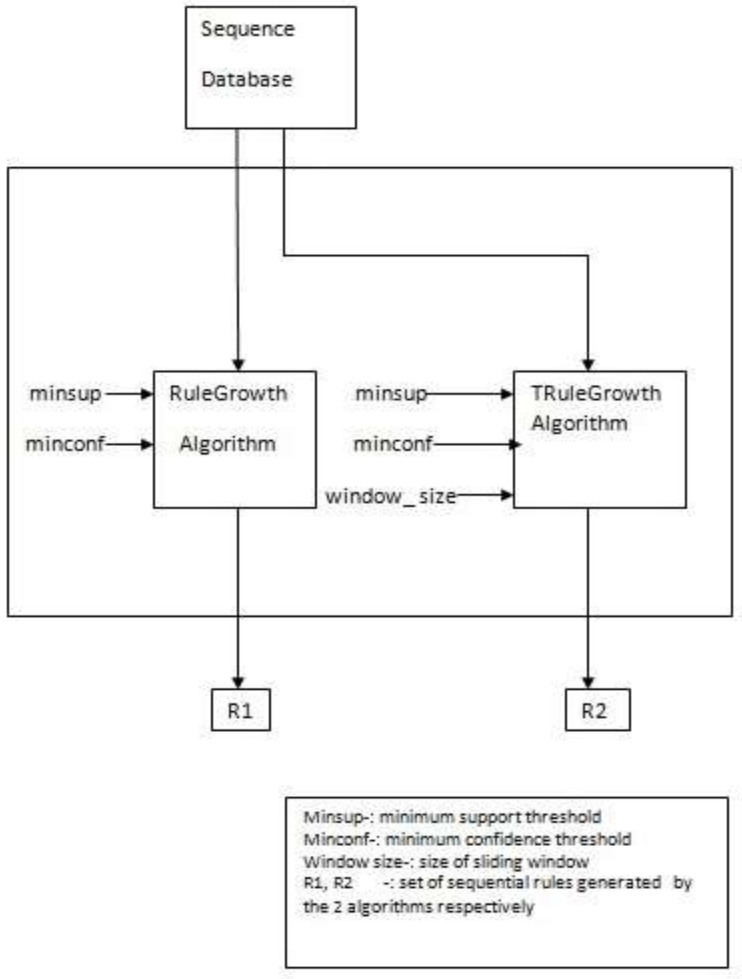
Sequence Database
minsup
minconf
RuleGrowth Algorithm
minsup
minconf
window_size
TRuleGrowth Algorithm
R1
R2
Minsup-: minimum support threshold
Minconf-: minimum confidence threshold
Window size-: size of sliding window
R1, R2 -: set of sequential rules generated by the 2 algorithms respectively

Figure 3.1: System Architecture

RuleGrowth algorithm [7] uses pattern-growth approach for discovering more efficient and valid rules. It finds rules in incremental fashion. It starts with two items and then grows by scanning the database to expand their left or right part of rule. RuleGrowth is the first module that will take as input the sequence database (SD) and the threshold values for minimum support (minsup) and minimum confidence (minconf) respectively. It avoid the problem of candidate generation and rely on a pattern-growth approach inspired by the one used in the PrefixSpan algorithm [3] for sequential pattern mining. RuleGrowth [7] will first find sequential rules of size 1*1 and then it will recursively grow them by scanning the sequences containing them to find single items that will expand their left or right sides. This approach ensures that only the partially-ordered sequential rules which occur in the database are considered as the valid rules by the algorithm. Two processes for expanding rules in RuleGrowth are named left expansion (EXPANDLEFT) and right expansion (EXPANDRIGHT) [7].RuleGrowth algorithm finds larger rules by adding the items into rules by scanning through the sequence database. Another feature of the RuleGrowth is it simultaneously keeps track of first and the last occurrences of each item, and also of antecedents and consequents for avoiding the complete scanning of sequences. It relies on the use of sequence id sets for calculation of support and confidence of the generated rules obtained by expanding left and right sides of the rules. Finally, the R1 block in Figure 3.1 represents the generated sequential rules using RuleGrowth algorithm.

II. TRuleGrowth:

TRuleGrowth [8] is a second component of the system SPOSR which is an algorithm that takes extra input to RuleGrowth in the form of window size which is the size of sliding window. As in extension to RuleGrowth algorithm, TRuleGrowth algorithm uses this window size constraint to discover the rules occurring within the sliding window. This constraint is enforced for rule size of 1*1. Accordingly left and right side of the rule is modified. That is, to take this constraint into account in the generation of larger rules, EXPANDLEFT and EXPANDRIGHT procedures have been modified accordingly [8]. Finally, the R2 block in Figure 3.1 represents the generated sequential rules using TRuleGrowth algorithm.

4. EXPERIMENTAL RESULTS

Performance of the two algorithms RuleGrowth and TRuleGrowth is analyzed by performing experiments on the Kosarak real-life dataset [8].

Experiments have been performed to analyze the influence of minsup as follows:

- **Experiment for measuring the effect of minsup:**

The experiments were performed on Kosarak dataset [8] by executing RuleGrowthand TRuleGrowth algorithms with various values of minsup and fixed values of minconf and window size respectively. This assessed the effect of minsup on performance of RuleGrowth and TRuleGrowth algorithms.

The algorithms were executed with minconf value of 0.2, and for TRuleGrowth, using window size value of 3 .Finally, to assess the effect of minsup on number of rules generated, the minsup values were taken in range between 0.004 and 0.001.The sequential rule count and execution times of each algorithm are compared and analyzed. Figure 4.1 shows the graph showing the analysis.

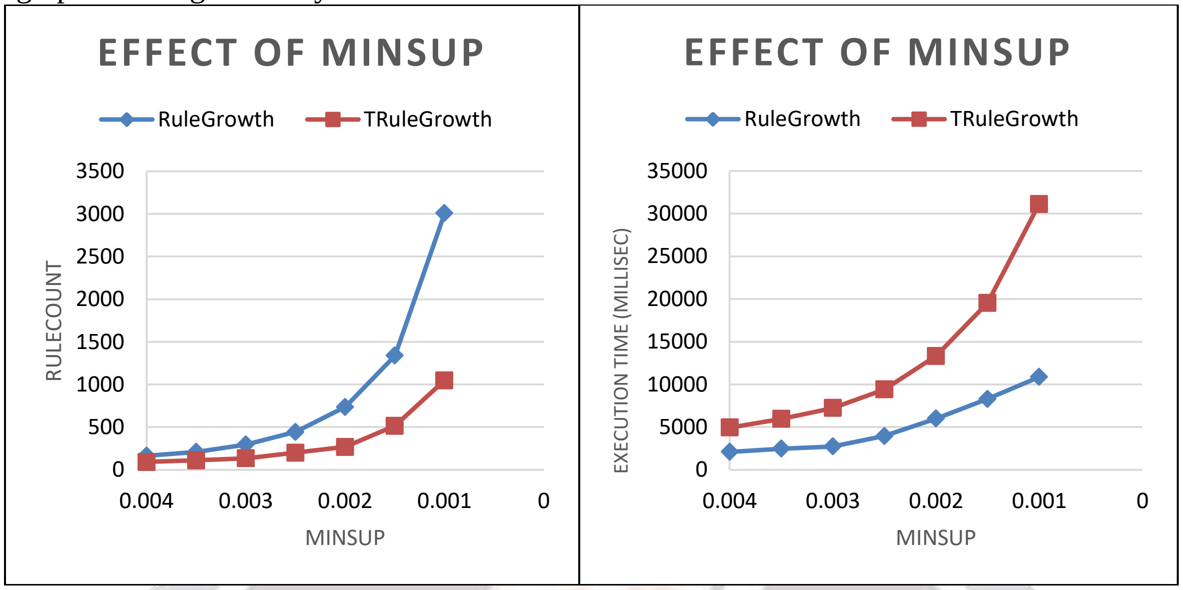

Figure 4.1 Effect of minsup on Rule count and Execution time on Kosarak dataset

The important Comparison is between sequential rule count and the execution times of RuleGrowth and TRuleGrowth algorithms. It is found that the sequential rule count and execution time of TRuleGrowth algorithm is much less than that of RuleGrowth algorithm.

Overall, in absence of window size constraint, RuleGrowth generates more number of sequential rules than the TRuleGrowth algorithm. The window size constraint is applied on TRuleGrowth algorithm, which performs better than RuleGrowth algorithm in terms of sequential rule count andexecution time.

5. CONCLUSION

In this paper, we have presented a system for mining partially–ordered sequential rules. TRuleGrowth accepts the window size as an extra constraint for generation of partially-ordered sequential rules. The future work includes further optimization of TRuleGrowth by applying multithreading technique on the algorithm.

REFERENCES

[1] *R. Agrawal and R. Srikant*, “Mining sequential patterns,” in Proc. 11th Int. Conf. Data Eng., 1995, pp. 3–14.

[2] *H. Mannila, H. Toivonen, and A. I. Verkano*, “Discovery of frequent episodes in event sequences,” Data Mining Knowl. Discovery, vol. 1, no. 3, pp. 259–289, 1999.

[3] *J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu*, “Mining sequential patterns by pattern-growth: The prefixspan approach,” IEEE Trans. Knowl. Data Eng., vol. 16,no. 10, pp. 1–17, Oct. 2004.

[4] *P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. MephuNguifo*, “CMRules: An efficient algorithm for mining sequential rules common to several sequences,” Knowl. Based Syst., vol. *25, no. 1, pp. 63–76, 2012.*

[5] *P. Fournier-Viger, T. Gueniche, and V. S. Tseng*, “Using partially ordered sequential rules to generate more accurate sequence prediction,” in Proc. 8th Int. Conf. Adv. Data Mining *Appl., 2012, pp. 431–442.*

[6] *M. J. Zaki*, “SPADE: An efficient algorithm for mining frequent sequences,” Mach. Learning, vol. 42, no. 1–2, pp.31–60, 2001.*P.*

[7] *Fournier-Viger, R. Nkambou, and V. S. Tseng*, “RuleGrowth: Mining sequential rules common to several sequences by patterngrowth,” in Proc. 26th ACM Symp. Appl. Comp., 2011, pp. 954–959.

[8] *Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng, Longbing Cao, and Roger Nkambou*, “Mining Partially-Ordered Sequential Rules Common to Multiple Sequences” IEEE Trans. Knowl. Data Eng., vol. 27, no. 8, Aug. 2015.

[9] *Mr.Sandipkumar C. Sagare, Prof. Dr. S. K. Shirgave*, “Survey on Mining Partially-Ordered Sequential Rules” IJCERT ,Volume 4, Issue 5, May-2017, pp. 169-170.