International Journal of Software Innovation
Volume 8 • Issue 4 • October-December 2020

A System for Predictive Data Analytics Using Sequential Rule Mining

Sandipkumar Chandrakant Sagare, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India
https://orcid.org/0000-0003-2774-9399

Suresh Kallu Shirgave, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India

Dattatraya Vishnu Kodavade, D.K.T.E. Society's Textile and Engineering Institute, Ichalkaranji, India

ABSTRACT

In the current scenario of the business world, the importance of data analytics is quite large. It certainly benefits the businesses in the decision-making process. Sequential rule mining can be widely utilized to extract important data having variety of applications like e-commerce, stock market analysis, etc. Predictive data analytics using the sequential rule mining consists of analyzing input sequences and finding sequential rules that can help businesses in decision making. This article presents an approach called M_TRuleGrowth that generates partially-ordered sequential rules efficiently. The authors conducted an experimental evaluation on real world dataset that provides strong evidence that M_TRuleGrowth performs better in terms of execution time.

KEYWORDS

Data Analytics, Predictive Data, Sequential Rule

1. INTRODUCTION

There are various application areas of data analytics as well as machine learning where the data to be analyzed is organized in terms of sequence of events. It is useful to identify relationships between event occurrences hidden in database as it provides a good understanding of relations of events for prediction of the next event (Mannila et al., 1999). In data mining, one of the useful techniques for discovery of temporal relations between events in discrete time series is sequential pattern mining (Mannila et al., 1999; Agrawal & Srikant 1995; Pei et al., 2004). Sequential pattern mining discovers sequences of events that frequently appear in a sequence database. That is the subsequences which appear in sequence database having support greater than or equal to threshold value of support set by the user can be found using Sequential pattern mining (Mannila et al., 1999; Agrawal & Srikant 1995; Pei et al., 2004).

There are wide varieties of algorithms developed for mining standard sequential rules. CMRules (Fournier-Viger et al., 2012) is the algorithm that mines sequential rules common to several sequences in a sequence database. The algorithm is based on association rule mining and is very efficient. It can be used to find both sequential rules and association rules in a database.

For Partially Ordered Sequential Rules (POSR), RuleGrowth algorithm (Fournier-Viger et al., 2011) was used which utilizes pattern-growth approach to find POSR that are common to several sequences. RuleGrowth (Fournier-Viger et al., 2011) does not use the existing techniques of discovering

DOI: 10.4018/IJSI.2020100107

Copyright © 2020, IGI Global. Copying or distributing in print or electronic forms without written permission of IGI Global is prohibited.

candidate rules and then testing them. Rules are discovered in incremental fashion by RuleGrowth. The process of rule discovery starts with two items and then rules grow by scanning the database for expanding the left and right part of rule. TRuleGrowth algorithm (Fournier-Viger et al., 2015) takes an extra parameter window size compared to RuleGrowth (Fournier-Viger et al., 2011). TRuleGrowth algorithm (Fournieir-Viger et al., 2015) makes use of window size for discovering the rules that occur within the sliding window. Rules of size 1*1 are enforced by this constraint. Left and right side of the sequential rule is modified accordingly. This makes TRuleGrowth algorithm (Fournier-Viger et al., 2015) an extension of RuleGrowth (Fournier-Viger, 2011) which ensures that the constraint of sliding window is taken into the consideration while generating rules. Finding rules occurring in a sliding-window has several useful advantages. First is it can reduce the time required for execution by reducing the search space. Second is it can generate a much smaller set of sequential rules which minimizes the requirement of disk space for storing sequential rules generated and makes it easy to analyze results (Fournier-Viger, 2015).

Thus, the System for mining POSR is an extension of the TRuleGrowth algorithm (Fournier-Viger et al., 2015). It uses M_TRuleGrowth approach which is multithreaded version of the preprocessing part of existing TRuleGrowth algorithm (Fournier-Viger et al., 2015). This approach analyzes the input and applies the multithreading technique. Use of multithreading minimizes the time required for preprocessing and in turn the overall execution time. Then the sequential rules generated can be used for the decision making in applications such as e-commerce, stock market analysis, etc.

1.1 Problem Statement

Discovering all valid sequential rules by analyzing the sequence database for future business predications.

1.2 Scope

The system for predictive data analytics using sequential rule mining focuses on discovering all the valid sequential rules in a sequence database. Pattern-growth approach is used to mine the partially-ordered sequential rules. This approach is extended by applying a sliding-window constraint. The approach is further extended to make use of multithreading technique and finally all the three approaches are analyzed in terms of execution times and the number of sequential rules generated.

2. LITERATURE SURVEY

2.1 Present Theory and Practices

H. Mannila, H. Toivonen, and A. I. Varian (Mannila et al., 1999) created a framework for discovering frequent episodes from a sequential data. Episodes are defined as partially-ordered sets of events by this framework, and also by looking at windows on the sequence. WINEPI is the algorithm described by them. From a given class of episodes, this algorithm finds all the episodes that are frequent enough. WINEPI algorithm focuses on finding the episodes only by taking into account an episode when all its sub-episodes are frequent, and on checking of whether or not an episode occurs inside a window in incremental manner.

R. Agrawal and R. Srikant (1995) dealt with a problem of sequential patterns mining from a customer sales transactions database and three algorithms were developed by them for solving this problem. Two of the three algorithms AprioriAll and AprioriSome have shown the comparable performance. AprioriSome algorithm showed better performance for the lower values of the number of customers that must support a sequential pattern. The limitation of these algorithms is that these algorithms were based on only calculating the support. They haven't considered the confidence value.

J. Pei, J. Han, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal, and M. Hsu (2004) developed a method for sequential pattern mining that was novel, scalable, and efficient called PrefixSpan. It analyzed the prefix subsequences and presented only the corresponding postfix subsequences from these sequences into projected databases. Sequential patterns were found by discovering only the local frequent patterns. They proposed two types of database projections: bi-level projection and level-by-level projection. Optimization approach exploring pseudo-projection is also developed. The algorithm PrefixSpan discovers complete patterns set and is said to be faster as well as efficient than both FreeSpan and Apriori-based GSP algorithm. PrefixSpan has the limitation that it cannot mine sequential patterns with time constraints, time windows and other kinds of time-based knowledge.

P. Fournier-Viger, U. Faghihi, R. Nkambou, and E. MephuNguifo (2012) presented CMRules algorithm. CMRules algorithm was 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, because both can be discovered at the same time. Also, it gives the possibility for enhancing the capability of CMRules by choosing a particular association rule mining algorithm that has this capability.

P. Fournier-Viger, R. Nkambou, and V. S. Tseng (2011) presented RuleGrowth (Fournier-Viger et al., 2011), a novel algorithm for mining sequential rules common to several sequences. It finds the 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 (Fournier-Viger et al., 2012) algorithms.

Philippe Fournier-Viger, Cheng-Wei Wu, Vincent S. Tseng, Longbing Cao, and Roger Nkambou (2015) developed two algorithms. First is RuleGrowth (Fournier-Viger et al., 2011) which is an algorithm to mine the sequential rules common to multiple sequences. It makes use of pattern-growth approach to discover valid sequential rules such that it avoids considering the rules that does not appear in the database. The second algorithm TRuleGrowth (Fournier-Viger et al., 2015) allows the user to give a sliding-window constraint for the rules to be mined. To evaluate the RuleGrowth (Fournier-Viger et al., 2011) algorithm and TRuleGrowth (Fournier-Viger et al., 2015) algorithm, several experiments were performed to assess the influence of minsup, minconf parameters on the performance of each algorithm. Also, RuleGrowth (Fournier-Viger et al., 2011) was compared with TRuleGrowth (Fournier-Viger et al., 2015) for different window size values for evaluating the benefits of using the window size constraint.

D. Lo, S.-C. Khoo, and L. Wong (2009) presented a problem of investigating syntactic characterization of non-redundant sequential rules. This was built on compact set of representative patterns. If rules that have same support and confidence can be found from one another, they are considered as redundant rules. When we use the set of discovered rules as a composite filter, replacement of a rules set with a non-redundant subset of the rules doesn't impact the accuracy of the filter. Several sets of rules were considered based on closed patterns, projected-database closed patterns, projected-database generators, and composition of various types of pattern sets-generators. D. Lo, S.-C. Khoo and L. Wong (2009) examined the completeness and tightness of these rule sets. A complete and tight set of Non-redundant rules are characterized by defining it based on the composition of two pattern sets. Furthermore, compressed set of non-redundant rules were proposed a in a similar spirit to how closed patterns serve as a compressed representation of a full set of patterns. Lastly, they proposed an algorithm to mine this compressed set of non-redundant rules. A performance study showed the improved runtime and compactness of mined rules both over mining a full set of sequential rules.

A. Pitman and M. Zanker (2011) presented an empirical study of different design variants of sequential pattern mining for the recommendation of items of interest to shallow user profiles. They demonstrated how integrating multidimensional product knowledge can be used to increase recommender recall using the historical dataset obtained from a leading e-tailor of nutritional supplements. Also Δ-closed frequent sequences concept was introduced and they showed how this can be applied for reducing the volume of patterns generated without compromising accuracy of recommender results. For less than 5 view events, it achieved a recall of 28% for short user navigation sequences.

G. Das, K.-I. Lin, H. Mannila, G. Renganathan, and P. Smyth (1998) discovered two problems for extraction of the rules from time-series data. First was that since rules are nothing but a symbolic representation, signal data must be transformed into an abstract symbolic alphabet. This was achieved using data-driven clustering of signal windows. Second problem was of finding the rules from symbolic sequences. There is a trade-off between quality of rules and abstraction quality. Different Parameters affect the types of rules induced. It was more useful when coupled with human interpretation of the rules rather than automated approach.

S. K. Harms, J. Deogun, and T. Tadesse (2002) found the approach MOWCATL for generating episodical association rules in multiple data sets. Gen-REAR approach was compared with this approach. It proved how the MOWCATL and Gen-REAR approach complemented each other to address complex real-life problems like drought risk management. They carried experiments that demonstrated the fact that these methods were efficiently finding the relationships between climatic episodes and droughts by using the constraints, the time lags, closures and the representative episodical association rules. It is said that this approach could also benefit Other problem domains especially when there are groupings of the events that occur close together in time, but does not occur relatively frequently over entire dataset, with possible time delays between the occurrences. Temporal associations between episodes of events can be evaluated by using the analysis techniques developed in this work. Also, it can facilitate the decision support systems by incorporating this knowledge. The limitation of this work was that it did not consider the spatial extent of the relationships. Also, these approaches could have been incorporated into the advanced geospatial decision support system for drought risk management.

P. Fournier-Viger and V. S. Tseng (2013) pointed out two important problems in the classical sequential rule mining algorithm that were, first, it is difficult and time-consuming to select the parameters for generating a desired number of rules and second, the results can be redundant. P. Fournier-Viger and V. S. Tseng (2013) addressed these problems by developing an efficient algorithm called TNS to mine the top-k non-redundant sequential rules. To ensure the efficiency of TNS, TNS relied on an approximate technique that guaranteed exact results under certain conditions. Performance of TNS algorithm has been compared with TopSeqRules on three real datasets and it was shown that TNS had excellent performance and scalability comparable to TopSeqRules. But the advantage of TNS was that it eliminated redundancy. This was an improvement because rules generated by TopSeqRules were redundant.

P. Fournier-Viger, T. Gueniche, and V. S. Tseng (2012) discovered the Prediction of the next elements of a sequence which is a research problem having wide applications such as consumer product recommendation, Web link recommendation and stock market prediction. It 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 (2001) developed Sequential Pattern Discovery using Equivalence classes (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; seconds can 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.

It is observed that the initial work focused on mining of sequential patterns based on only the support parameter. The confidence factor was not considered in many approaches. We know how much important confidence factor is for making the decisions or predictions. Later the sequential rule mining algorithms involved support as well as confidence as important factors for generation of sequential rules. But it is found that the standard sequential rules were too specific. So, using these standard sequential rules for making important predictions was inefficient. A new type of sequential rules has been developed later called partially-ordered sequential rules (POSR) (Harms et al., 2002). Usually users only want to find rules occurring within a maximum amount of time (Mannila et al., 1999; Fournier-Viger et al., 2015; Lo et al., 2009). This motivated to find the sequential rules occurring within a sliding-window, which can be utilized for applications such as analyzing stock market data. Interest in mining rules common to multiple sequences came from the fact that it has many useful applications. For example, mining sequential rules common in customer transaction databases to recommend the products. Also, ability to achieve faster execution motivated to use multithreading technique for preprocessing of the data.

3. PROPOSED SYSTEM

3.1 System Architecture

Figure 1 shows the architecture of the proposed System for Predictive Data Analytics Using Sequential Rule Mining using M_TRuleGrowth technique, which consists of three major modules RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth. Survey of existing techniques has motivated for discovering the sequential rules occurring within a sliding window. This can be very useful for the discovery of temporal patterns for many real-life applications such as analyzing stock market data. Also, ability to achieve faster execution motivated us to use multithreading technique for mining sequential rules.

3.1.1 Sequence Database

A sequence database D (Mannila et al., 1999; Fournier-Viger et al., 2015) is nothing but a set of sequences $D = \{s_1, s_2, \ldots, s_m\}$ and a finite set of items $I = \{ i_1, i_2, .., i_n\}$ occurring in these sequences, where each sequence has been assigned a unique identifier ID (Sequence ID).

A sequence is nothing but an ordered list of item sets $s_x = I_1, I_2, \ldots, I_m$ such that $I_1, I_2, \ldots, I_m$ are subsets of I. For example, Table 1 shows a sequence database which consists of four sequences s_1, s_2, s_3 and s_4. In this example, each single letter represents a single item. An Itemset is represented by item(s) within the curly braces. That is, the sequence s_1 means that an item *a* occurred first, then after that items *b* and *c* occurred both at the same time and was followed successively by item *g*.

3.1.2 minsup

It is a minimum support threshold to be satisfied for generating the sequential rule.

3.1.3 minconf

It is a minimum confidence threshold to be satisfied for generating the sequential rule.

3.1.4 window_size

It is a size of sliding window which is nothing but a window that is assumed to move from beginning of a sequence to the end of it one itemset at a time.

3.1.5 R1, R2, and R3

It is a set of sequential rules generated by RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms respectively.

Figure 1. System architecture

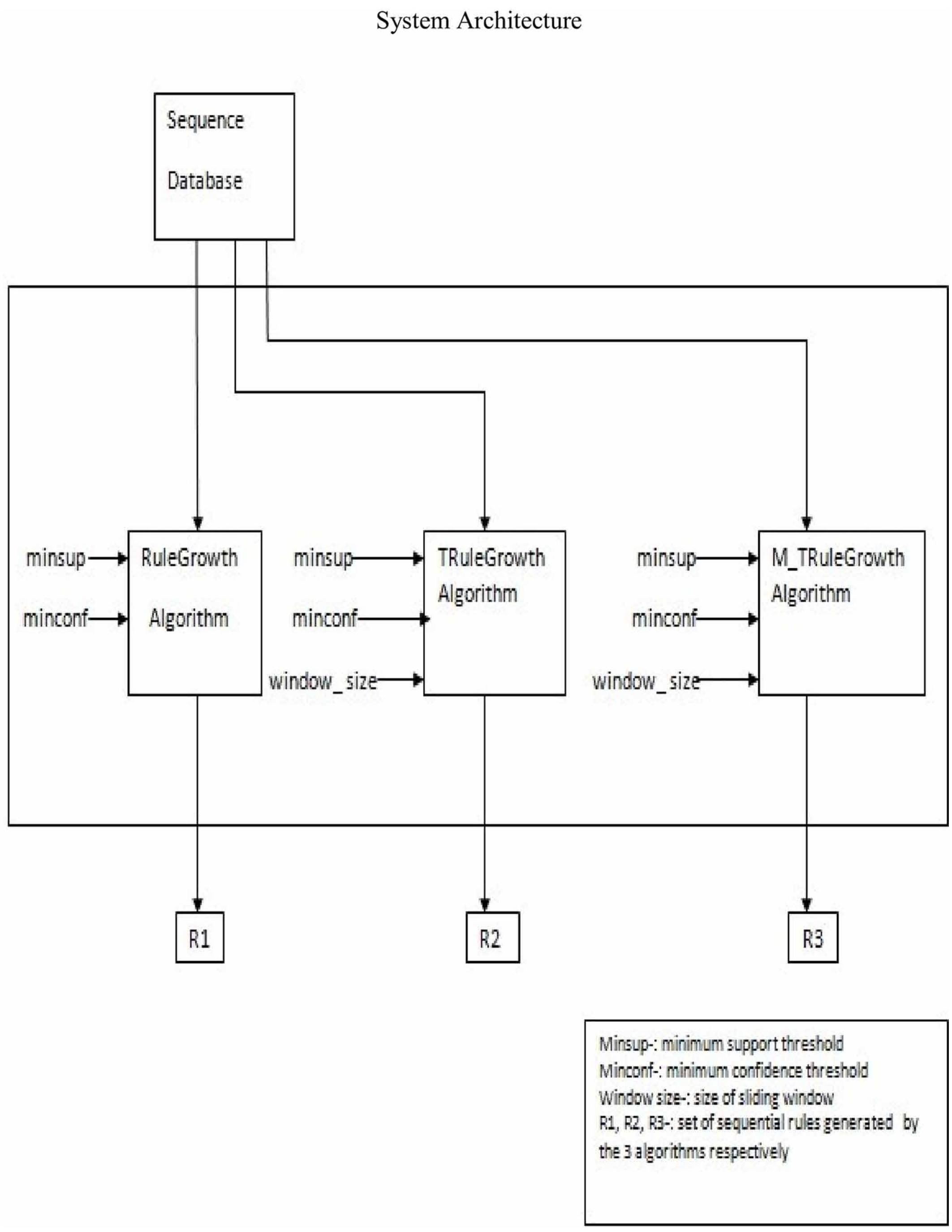

System Architecture
Sequence Database
minsup
minconf
RuleGrowth Algorithm
minsup
minconf
window_size
TRuleGrowth Algorithm
minsup
minconf
window_size
M_TRuleGrowth Algorithm
R1
R2
R3
Minsup-: minimum support threshold
Minconf-: minimum confidence threshold
Window size-: size of sliding window
R1, R2, R3-: set of sequential rules generated by the 3 algorithms respectively

Table 1. A Sequence databasel

SID	SEQUENCES
s_1	{a},{b, c},{g}
s_2	{a, c},{b},{a, e}
s_3	{a},{b},{c}
s_4	{d},{c},{g}

4. METHODOLOGY

4.1 The RuleGrowth Module

RuleGrowth (Fournier-Viger et al., 2011) is the first module that takes as input the sequence database (SD) and the threshold values for minimum support (minsup) and minimum confidence (minconf). It avoids the problems of candidate generation and depends on a pattern-growth approach used in the PrefixSpan algorithm (Pei et al., 2004) for sequential pattern mining. RuleGrowth (Fournier-Viger et al., 2011) first finds sequential rules of size 1*1 and then it grows them recursively 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 sequence database are considered as the valid rules by the algorithm. Two processes for expanding rules in RuleGrowth (Fournier-Viger et al., 2011) are named left expansion (EXPANDLEFT) and right expansion (EXPANDRIGHT) (Fournier-Viger et al., 2011). RuleGrowth algorithm (Fournier-Viger et al., 2011) finds larger rules by adding the items into rules by scanning through the sequence database. Another feature of the RuleGrowth (Fournier-Viger et al., 2011) 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.

4.2 The TRuleGrowth Module

TRuleGrowth (Fournier-Viger et al., 2015) is the second module which is based on the RuleGrowth (Fournier-Viger eet al., 2011) approach. The RuleGrowth algorithm is extended to take an extra input in the form of window-size. This module considers the TRuleGrowth algorithm (Fournier-Viger et al., 2015) that includes one particular extension which is to discover the partially-ordered sequential rules occurring within a sliding-window, i.e. within a maximum number of consecutive itemsets in each sequence. The modifications will ensure that the sliding-window constraint is enforced for rules of size 1*1. EXPANDLEFT procedure (Fournier-Viger et al., 2015) and EXPANDRIGHT procedure (Fournier-Viger et al., 2015) were modified to take this constraint into account for generating larger rules (Fournier-Viger et al., 2015).

4.3 The M_TRuleGrowth Module

This module takes as input the sequence database (SD) and the threshold values for minimum support. In this module the TRuleGrowth (Fournier-Viger et al., 2015) approach will be further extended to apply multithreading technique on the preprocessing part of the algorithm used in the TRuleGrowth (Fournier-Viger et al., 2015) module. This module is called M_TRuleGrowth. That is, Multithreaded TRuleGrowth (Fournier-Viger et al., 2015). This module accepts the same input as in the TRuleGrowth (Fournier-Viger et al., 2015) and makes use of the multithreading technique by splitting the data and task for loading the input sequence database into parallel subtasks. It lets the underlying architecture manage how threads run, either concurrently on single core, or in parallel on multiple cores.

M_TRuleGrowth approach that makes use of multithreading technique can be evaluated by comparing with RuleGrowth (Fournier-Viger et al., 2011) and TRuleGrowth (Fournier-Viger et al., 2015) in terms of execution time. For this evaluation, the system time can be retrieved before executing the M_TRuleGrowth module and after getting the results, again the system time can be retrieved. The difference between the initial and final system time can give the execution time of the M_TRuleGrowth module. The same procedure can be repeated for the RuleGrowth (Fournier-Viger et al., 2011) and TRuleGrowth (Fournier-Viger et al., 2015) modules and finally these execution times can be compared with that of M_TRuleGrowth. This can give us the better idea of the method that generates the partially-ordered sequential rules with faster execution time.

5. EXPERIMENTAL RESULTS

In order to evaluate the system, Kosarak dataset (Fournier-Viger et al., 2015) is used. The first part of this section discusses the statistics of Kosarak dataset used for evaluation of the system. It is followed by the evaluation methodology and the evaluation of various parameters used to generate the partially-ordered sequential rules. Comparative evaluation of the RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth is represented in the form of graphs showing the performance of each of the technique used to generate the sequential rules.

5.1 Dataset

Experiments were carried on the real-life dataset Kosarak (Fournier-Viger et al., 2015) . The Kosarak dataset is a click-stream dataset from a Hungarian news portal. It consists of 990,000 sequences of click-stream data from an online news portal from Hungary. The first 70,000 sequences out of the total 990,000 sequences are used for the evaluation of the system. The Kosarak dataset (Fournier-Viger et al., 2015) is present in the form of a .dat file which is to be taken as an input to the System for Predictive Data Analytics Using Sequential Rule Mining. Preprocessing is done to convert the original dataset into required format. The converted dataset is then further processed using the three approaches RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth respectively. Evaluation results are represented using the graphs.

5.2 Experimentation

Performance of the three algorithms RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth is analyzed by performing experiments on the Kosarak real-life dataset (Fournier-Viger et al., 2015) .

Experiments have been performed to analyze the influence of minsup, minconf and window size as follows:

The first experiment is performed by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with varying values of minsup and fixed values of minconf and window size. This assessed the influence of minsup on performance of RuleGrowth, TRuleGrowth, and M_TRuleGrowth algorithms.

The algorithms were run with minconf = 0.2, and for TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth, using window size = 3.Finally, to assess the influence of minsup, the minsup values were varied in range 0.004 to 0.001.The sequential rule count and execution times of each algorithm are compared and analyzed.

The second experiment is performed by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with varying values of minconf and fixed values of minsup and window size. This is useful to assess the influence of minconf on performance of RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms.

The algorithms were run with minsup = 0.004, and for TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth, using window size = 3. Finally, to assess the influence of minconf, the minconf values were varied in range 0.2 to 0.8. The effect of variation of minconf is evaluated in terms of number of sequential rules generated and execution time.

Since RuleGrowth (Fournier-Viger et al., 2011) does not include window size constraint, the third experiment is performed by running TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of window size and fixed values of minsup and minconf respectively. This assessed the influence of window size on performance of TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms.

Since the task of choosing appropriate window size is dependent on the dataset, various values have been used for window size. The algorithms were run with minsup = 0.001 and minconf = 0.2 for TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth.Finally, to assess the influence of window size, the window size values were varied in range 2 to 8.The effect of variation of window size is evaluated in terms of number of sequential rules generated and execution time.

5.3 Evaluation of Parameters

Two parameters are evaluated while generating the partially-ordered sequential rules. First, the evaluation in terms of number of sequential rules generated and second, evaluation in terms of execution time. Comparative evaluation of the RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth is represented in the form of graphs showing the performance of each of the technique used to generate the sequential rules.

5.3.1 Evaluation in Terms of Number of Sequential Rules Generated

This is the first parameter used for evaluation of the system which generates the rules using RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms. For this evaluation, different experiments are performed as follows

5.3.2. Experiment for Assessing Influence of Minsup on Rule Count

Figure 2 shows the evaluation of rule count by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of minsup and fixed values of minconf and window size respectively. This assessed the influence of minsup on performance of RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms respectively.

RuleGrowth (Fournier-Viger et al., 2011) algorithm generates a greater number of sequential rules as compared to TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth algorithms respectively for the varying values of minsup. This is because of the fact that the TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth algorithms take an extra input as a constraint in the form of window size. So, obviously the rules generated will be limited due to this constraint.

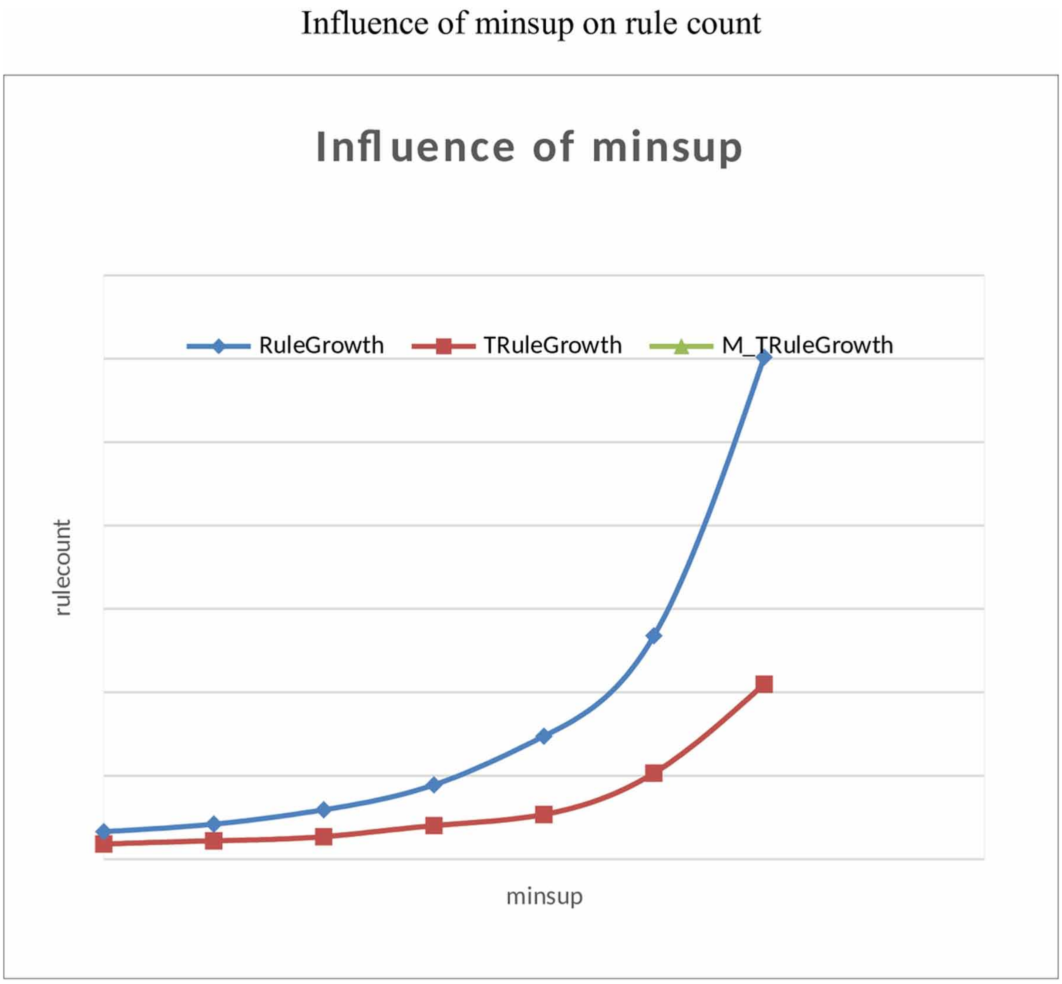
5.3.3. Experiment for Assessing Influence of Minconf on Rule Count

Figure 3 shows the evaluation of rule count by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of minconf and fixed values of minsup and window size respectively. This assessed the influence of minsup on performance of RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms respectively.

5.3.4. Experiment for Assessing Influence of Window Size on Rule Count

Figure 4 shows the evaluation of rule count by running TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of window size and fixed values of minsup and minconf respectively. This assessed the

Figure 2. Influence of minsup on rule count



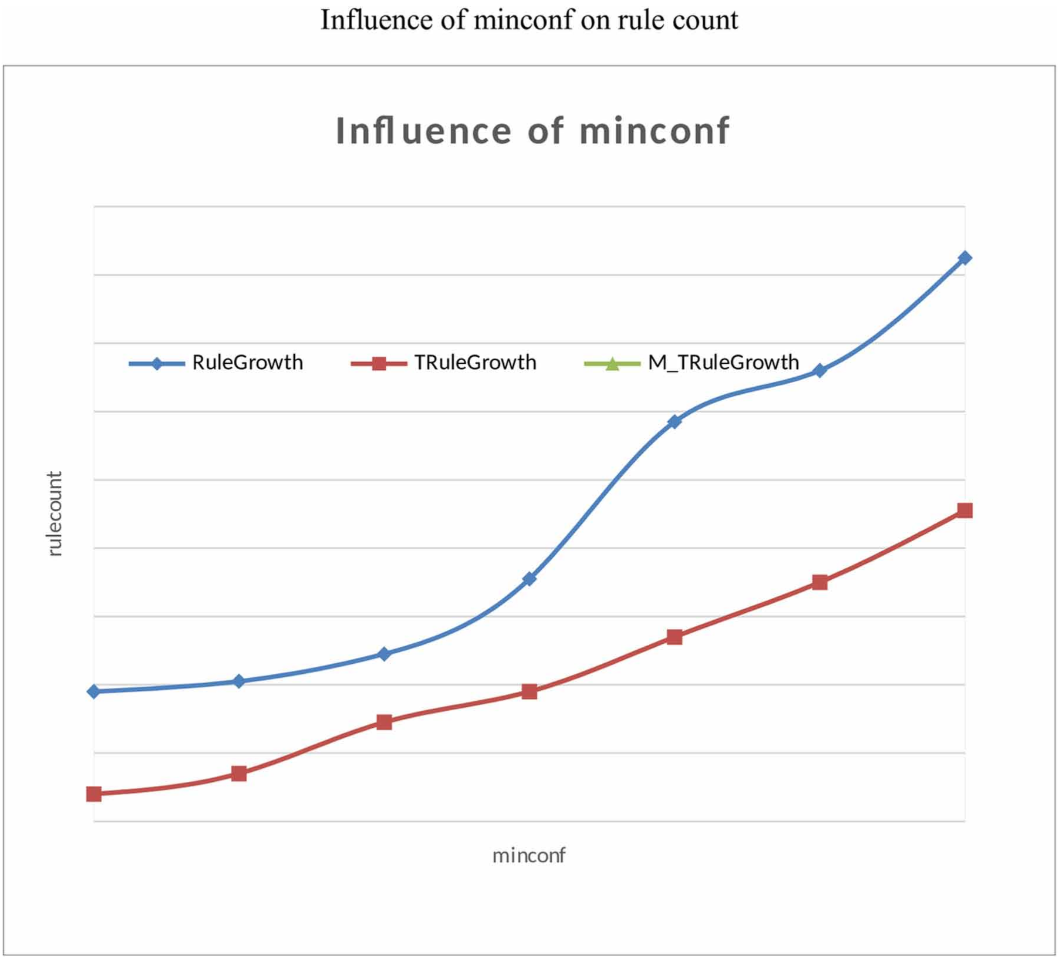
influence of window size on the performance of only TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms since RuleGrowth does not have window size constraint.

TRuleGrowth algorithm (Fournier-Viger et al., 2015) and M_TRuleGrowth algorithm generate same number of sequential rules for the varying values of window size. M_TRuleGrowth algorithm only scans the sequence database and logically divides the number of sequences into number of parts and assigns a part to each thread to store the sequences in the data structure. The main logic of generation of rules is same as that of TRuleGrowth algorithm (Fournier-Viger et al., 2015).So; the window size constraint will not affect the number of rules generated by the M_TRuleGrowth algorithm.

5.3.5. Evaluation in Terms of Execution Time

This is the second parameter used for evaluation of the system which generates the rules using RuleGrowth, TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms. For this evaluation, different experiments are performed as follows.

Figure 3. Influence of minconf on rule count



5.3.6. Experiment for Assessing Influence of Minsup on Execution Time

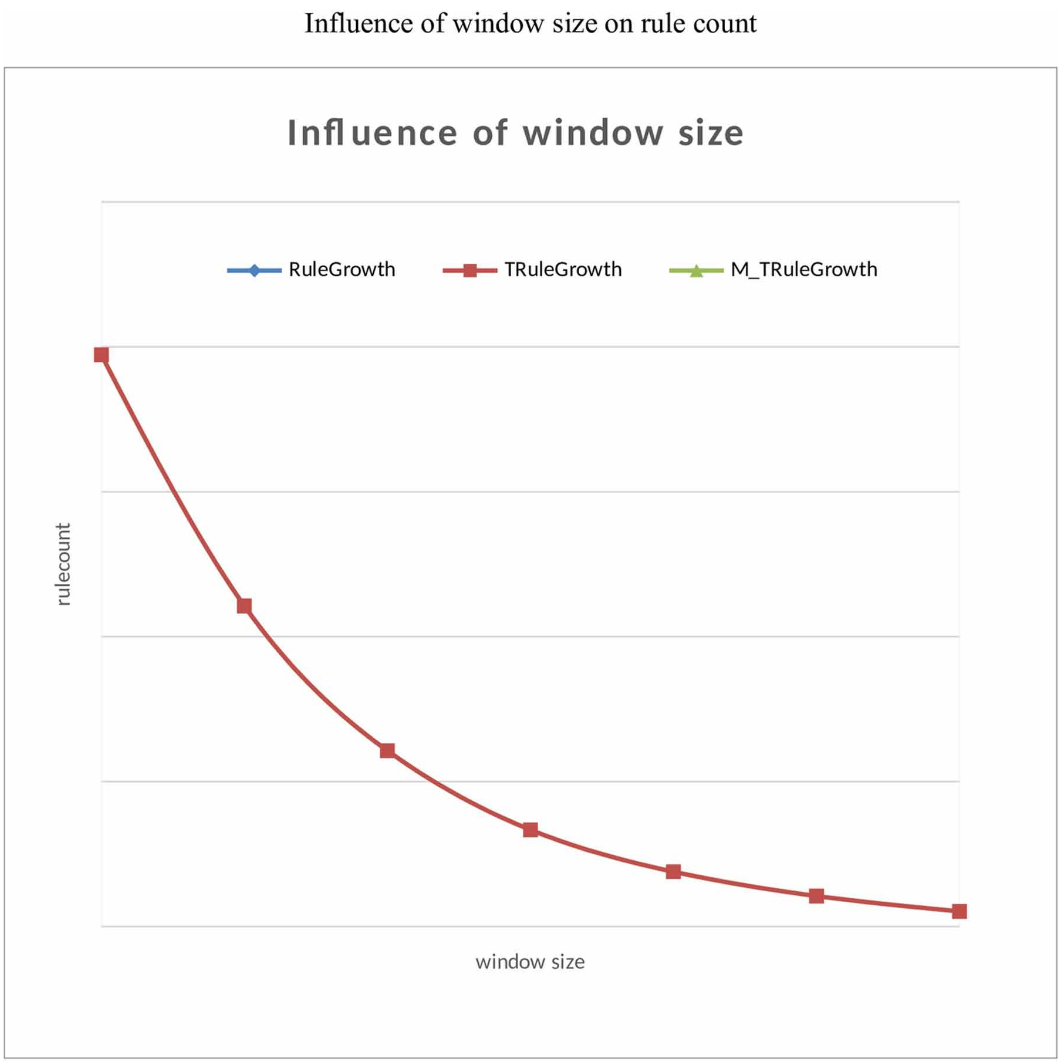
Figure 5 shows the evaluation of execution time by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of minsup and fixed values of minconf and window size respectively. This assessed the influence of minsup on performance of RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms respectively.

TRuleGrowth algorithm (Fournier-Viger et al., 2015) takes more execution time as compared to RuleGrowth (Fournier-Viger et al., 2011) and M_TRuleGrowth algorithms respectively for the varying values of minsup.

5.3.7. Experiment for Assessing Influence of Minconf on Execution Time

Figure 6 shows the evaluation of execution time by running RuleGrowth (Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of minconf and fixed values of minsup and window size respectively. This assessed the influence of minconf on the performance of RuleGrowth

Figure 4. Influence of window size on rule count



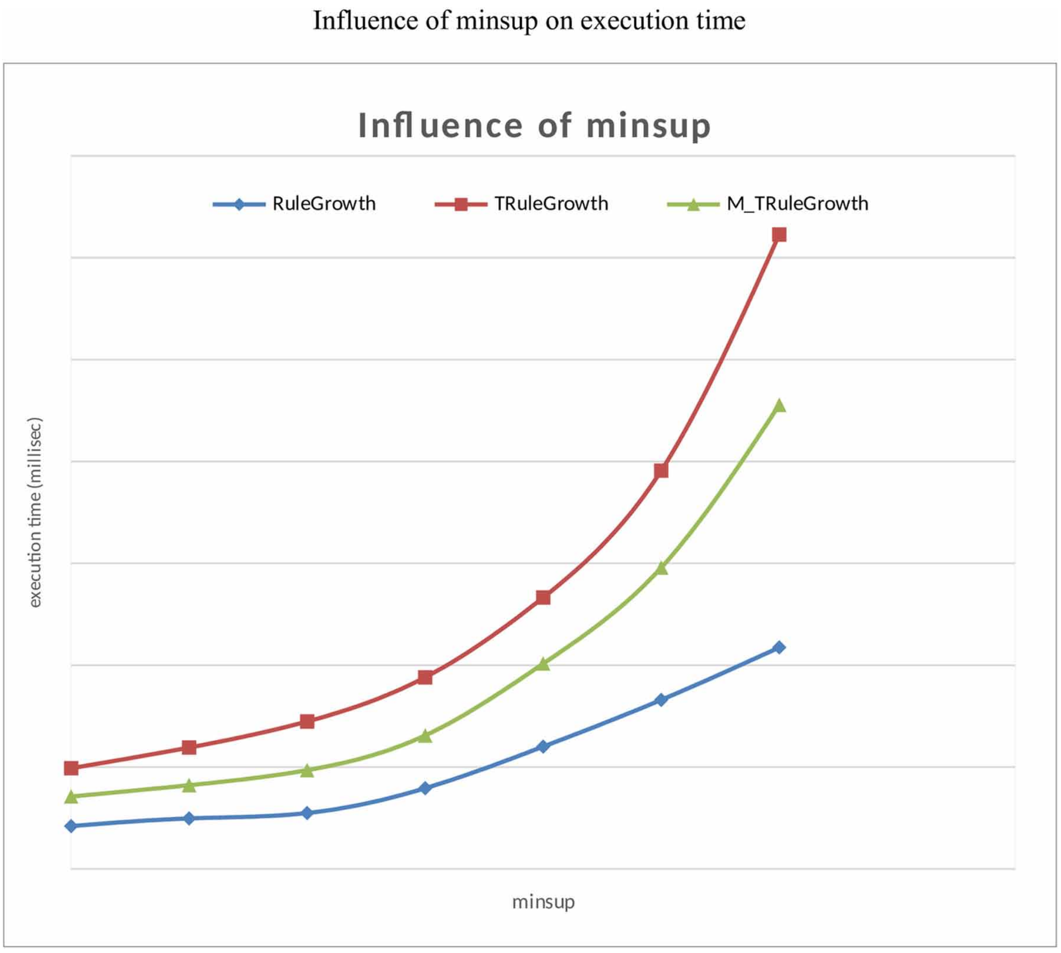
(Fournier-Viger et al., 2011), TRuleGrowth (Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms respectively.

TRuleGrowth algorithm (Fournier-Viger et al., 2015) takes more execution time as compared to RuleGrowth (Fournier-Viger et al., 2011) and M_TRuleGrowth algorithms respectively for the varying values of minconf.

5.3.8. Experiment for Assessing Influence of Window Size on Execution Time

Figure 7 shows the evaluation of execution time by running TRuleGrowth(Fournier-Viger et al., 2015), and M_TRuleGrowth algorithms for Kosarak dataset (Fournier-Viger et al., 2015) with various values of window size and fixed values of minsup and minconf respectively. This assessed the influence of window size on the performance of TRuleGrowth (Fournier-Viger et al., 2015) and M_TRuleGrowth algorithms only since RuleGrowth(Fournier-Viger et al., 2011) does not have window size constraint.

Figure 5. Influence of minsup on execution time



M_TRuleGrowth algorithm shows much better execution time than TRuleGrowth algorithm (Fournier-Viger et al., 2015) for the varying values of window size. As the window size increases, M_TRuleGrowth gets better in terms of execution time.

6. CONCLUSION

System for Predictive Data Analytics Using Sequential Rule Mining generates the POSR which can be used for making important business decisions. Also, analysis of three approaches for the generation of partially-ordered sequential rules is performed on the real-life dataset. Finally, performance of all the three algorithms is evaluated in terms of the number of sequential rules generated and execution time taken by each algorithm. It is found that RuleGrowth (Fournier-Viger et al., 2011) generates more number of sequential rules and has a less execution time. But, when the window size constraint is applied, M_TRuleGrowth outperforms the TRuleGrowth algorithm (Fournier-Viger et al., 2015) in terms of execution time.

Figure 6. Influence of minconf on execution time

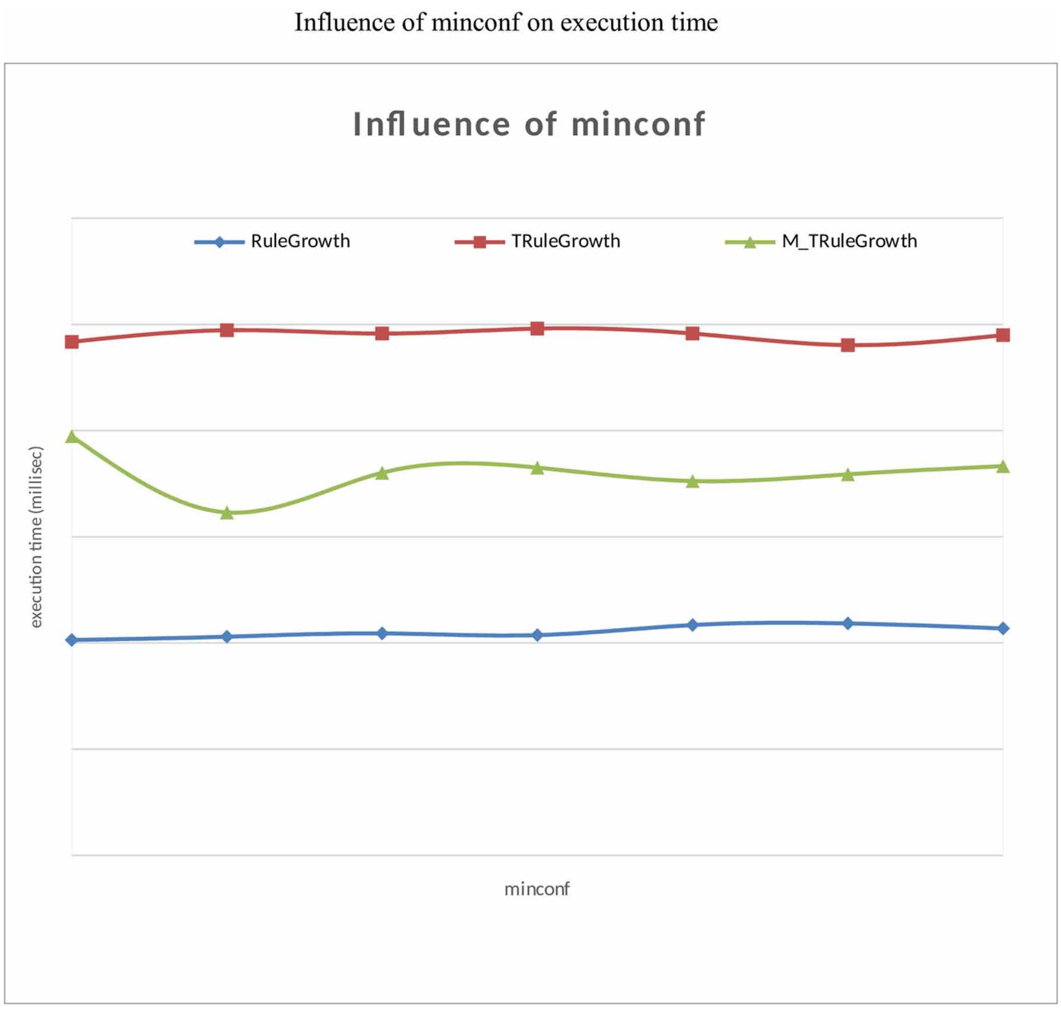
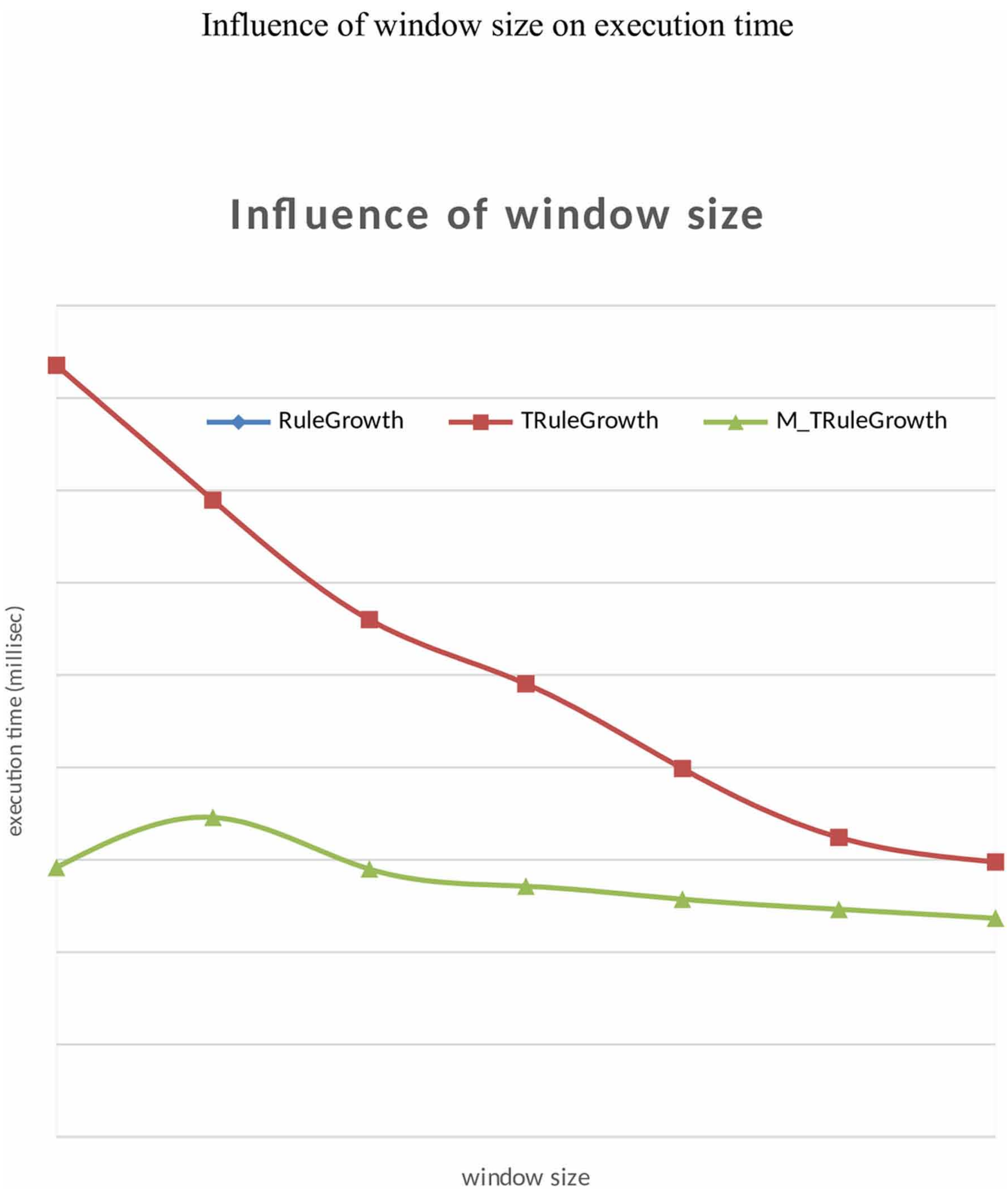


Figure 7. Influence of window size on execution time



REFERENCES

Mannila, H., Toivonen, H., & Verkano, A. I. (1999). Discovery of frequent episodes in event sequences. *Data Mining and Knowledge Discovery*, *1*(3), 259–289. doi:10.1023/A:1009748302351

Agrawal, R., & Srikant, R. (1995). Mining sequential patterns. In *Proc. 11th Int. Conf. Data Eng.* (pp. 3–14). Academic Press. doi:10.1109/ICDE.1995.380415

Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., & Hsu, M. (2004, October). Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, *16*(10), 1–17.

Fournier-Viger, P., Faghihi, U., Nkambou, R., & Nguifo, E. M. (2012). CMRules: Mining sequential rules common to several sequences. *Knowledge-Based Systems*, *25*(1), 63–76. doi:10.1016/j.knosys.2011.07.005

Fournier-Viger, P., Nkambou, R., & Tseng, V. S. M. (2011, March). RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In *Proceedings of the 2011 ACM symposium on applied computing* (pp. 956-961). ACM.

Fournier-Viger, P., Wu, C. W., Tseng, V. S., Cao, L., & Nkambou, R. (2015). Mining partially-ordered sequential rules common to multiple sequences. *IEEE Transactions on Knowledge and Data Engineering*, *27*(8), 2203–2216.

Lo, D., Khoo, S.-C., & Wong, L. (2009). Non-redundant sequential rules- Theory and algorithm. *Information Systems*, *34*(4/ 5), 438–453. doi:10.1016/j.is.2009.01.002

Pitman, A., & Zanker, M. (2011). An empirical study of extracting multidimensional sequential rules for personalization and recommendation in online commerce. In *Proc. 10th Int. Conf. Wirtschaf-informatik* (pp. 180–189). Academic Press.

Das, G., Lin, K.-I., Mannila, H., Renganathan, G., & Smyth, P. (1998). Rule discovery from time series. In *Proc. 4th ACM Int. Conf. Knowl. Discovery Data Mining* (pp. 16–22). ACM.

Harms, S. K., Deogun, J., & Tadesse, T. (2002). Discovering sequential association rules with constraints and time lags in multiple sequences. In *Proc. of the 13th Int. Symp. Method. Intell. Syst.* (pp. 373–376). Academic Press. doi:10.1007/3-540-48050-1_47

Fournier-Viger, P., & Tseng, V. S. (2013). TNS: Mining Top-K Non-redundant sequential rules. In *Proc. of the 28th Symp. Appl. Comput.* (pp. 164–166). Academic Press.

Fournier-Viger, P., Gueniche, T., & Tseng, V. S. (2012). Using partially ordered sequential rules to generate more accurate sequence prediction. In *Proc. of the 8th Int. Conf. Adv. Data Mining Appl.* (pp. 431–442). Academic Press. doi:10.1007/978-3-642-35527-1_36

Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine Learning*, *42*(1–2), 31–60. doi:10.1023/A:1007652502315

Agrawal, R., Imielminski, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proc. of the 13th ACM SIGMOD Int. Conf. Manage. Data* (pp. 207–216). ACM. doi:10.1145/170035.170072

Fournier-Viger, P., Lin, C. W., Gomariz, A., Gueniche, T., Soltani, A., Deng, Z., & Lam, H. T. (2016). The SPMF Open-Source Data Mining Library Version 2. In *Proc. of the 19th European Conference on Principles of Data Mining and Knowledge Discovery (PKDD 2016) Part III* (pp. 36-40). Springer.

Sagare, S., & Shirgave, S. (2017). Survey on Mining Partially-Ordered Sequential Rules. *International Journal of Computer Engineering In Research Trends*, *4*(5), 169–170.

Sandipkumar C. Sagare is currently working as Assistant Professor in the Computer Science and Engineering Department of D.K.T.E. Society's Textile and Engineering Institute Ichalkaranji, India. He received the B.E., M.E.in Computer Science and Engineering from the Shivaji University, Kolhapur, Maharashtra, India. His research interests include sequential rule mining, data analytics, and machine learning. He is also a nominee member of the Computer Society of India from D.K.T.E. Society's Textile and Engineering Institute Ichalkaranji.

Suresh K. Shirgave (PhD) is an Associate Professor in the Computer Science & Engineering department at D.K.T.E. Society's Textile and Engineering Institute Ichalkaranji, India. He received his B.E., M.E., and PhD in Computer Science and Engineering from Shivaji University, Kolhapur, Maharashtra, India. He has published many research papers in national and international conferences and journals. His research interests include data mining, web usage mining recommender systems, social networks, and Internet security.

Dattatraya V. Kodavade (PhD) received B.E., ME., and PhD degree from Shivaji University, Kolhapur. He is presently working as a Professor and Head of the Computer Science & Engineering department at D.K.T.E. Society's Textile and Engineering Institute Ichalkaranji, India. He has 27 years of teaching experience and has published more than 20 papers in international journals. He is working as a reviewer for the International Journal on Super Computing from Springer USA. He is also the Editor in Chief for International Journal on Knowledge Based Computer Systems, New Delhi. He is working as a member of the technical program committee for many national and internationally reputed conferences in computing. His areas of research are artificial intelligence, machine learning, IoT, robotics, and high-performance computing.