Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction

Philippe Fournier-Viger[1], Ted Gueniche[1], and Vincent S. Tseng[2]

[1] Dept. of Computer Science, University of Moncton, Canada
[2] Dept. of Computer Science and Inf. Eng., National Cheng Kung University, Taiwan
philippe.fournier-viger@umoncton.ca, ted.gueniche@gmail.com,
tsengsm@mail.ncku.edu.tw

Abstract. Predicting the next element(s) of a sequence is a research problem with wide applications such as stock market prediction, consumer product recommendation, and web link recommendation. To address this problem, an effective approach is to mine sequential rules from a set of training sequences to then use these rules to make predictions for new sequences. In this paper, we improve on this approach by proposing to use a new kind of sequential rules named partially-ordered sequential rules instead of standard sequential rules. Experiments on large clickstream datasets for webpage recommendation show that using this new type of sequential rules can greatly increase prediction accuracy, while requiring a smaller training set.

Keywords: symbolic sequence prediction, sequential rules, partial order

1 Introduction

Predicting the next element(s) of a sequence is an important research problem with wide applications such as stock market analysis, consumer product recommendation, weather forecasting, text generation and web link recommendation. Techniques for sequence prediction can be categorized according to the types of sequences on which they are applied. There are two main types. On one hand, time series are sequences of numeric data typically recorded at an equal time interval (e.g. sale data and temperature data). On the other hand, symbolic sequences are sequences of events or nominal data generally recorded at unequal time intervals (e.g. customer shopping sequences, program execution sequences and web click streams). Previous research on sequence prediction has mainly focused on predicting time-series. This is usually done by applying statistical methods to find mathematical functions that fit the data. These functions are then used to make predictions [3]. In this paper, we are interested by the case of symbolic sequences, which has many applications [13]. Because the data in symbolic sequences is not numeric, techniques for time-series forecasting cannot be applied to this problem.

To predict symbolic sequences, researchers have used techniques such as recurrent neural networks [11] and Markov models [2] (see [13] for a comprehensive

survey). Limitations of those techniques are that they (1) require labeled training data, (2) require defining a reward function and/or (3) assume that the next element of a sequence only depends on the previous element or requires building a model that is exponentially large if more than one element has to be considered [13], [2]. These assumptions are unrealistic or unpractical for many real applications [13]. To perform symbolic sequence prediction without making these assumptions, an effective solution is to use algorithms to discover sequential rules occurring in a set of training sequences. These rules are then used to make predictions [14], [8], [9], [12]. This approach has the advantage of being unsupervised, scalable, and to generate accurate predictions [10], [9], [12]. In this paper, we improve on this approach by proposing to use a new type of sequential rules named partially-ordered sequential rules [5], [6], [7] that we have proposed in previous works. We compare the prediction accuracy of these rules with standard sequential rules [14], [10], [9], [12]. for the task of webpage recommendation for large clickstream datasets. Experimental results show that using the new type of sequential rules can greatly improve prediction accuracy, while requiring a smaller training set.

The rest of this paper is organized as follows. Section 2 defines the problem of sequence prediction. Section 3 defines the two types of sequential rules that are compared. Section 4 presents our comparison framework that we have named the Predictor. Section 5 reports experimental results. Finally, section 6 draws a conclusion and discusses future works.

2 The Problem of Sequence Prediction

To define the problem of sequence prediction, we first need to define what are a sequence and a sequence database.

An *itemset* $I = \{i_1, i_2, ..., i_m\}$ is an unordered set of items, where an item is a symbolic value. For example, $\{a, b, c\}$ represents the sets of items a, b and c.

A *sequence database* SDB is a set of sequences $S = \{s_1, s_2 ... s_s\}$and a set of items $I = \{i_1, i_2, ..., i_m\}$ occurring in these sequences [3], [4], [14].

A *sequence* is an ordered list of itemsets $s = \langle I_1, I_2, ... I_n \rangle$ such that $I_k \subseteq I$ for all $1 \leq k \leq n$. For instance, the sequence $s1 = \langle \{a, b\}, \{c\}, \{f\}, \{g\}, \{e\} \rangle$ represents that items a and b occurred at the same time, and were followed successively by c, f, g and lastly e. For example, consider the sequence database SDB depicted in Figure 1. It contains four sequences named s_1, s_2, s_3 and s_4. In this example, each single letter represents an item. Items between curly brackets represent an itemset. The sequence of SDB could encode, for example, the sequence of webpages visited by four users or transactions of four customers in a store.

The *problem of sequence prediction* is defined as follows (adapted from [6], [7] [9]). Let $s = \langle I_1, I_2 ..., I_{v-1}, I_v, I_{v+1}, I_{w-1}, I_w \rangle$ be a sequence such that $\langle I_1, I_2, I_{v-1} \rangle$ has been observed and that $\langle I_v, ... I_{w-1}, I_w \rangle$ has not yet been observed. Lets $prefix\_size$ and $suffix\_size$ be positive integers. The problem of sequence pre-

ID	Sequences
s_1	⟨{a, b},{c},{f},{g},{e}⟩
s_2	⟨{a, d},{c},{b},{a, b, e, f}⟩
s_3	⟨{a},{b},{f},{e}⟩
s_4	⟨{b},{f, g, h}⟩

Fig. 1. A sequence database SDB

diction is to predict an item from the subsequence $< I_v, I_{v+1}, ... I_{(v+suffix\_size-1)} >$ based on the subsequence $\langle I_{(v-prefix\_size)}, ...I_{v-2}, I_{v-1}\rangle$.

Consider a sequence of webpages $s = \langle\{c\},\{a\},\{b\},\{e\},\{g\},\{c\}\rangle$ that a new user visits. Let $prefix\_size = 2$, $suffix\_size = 2$ and $v = 4$. This represents the problem of predicting an item from $\langle\{e\},\{g\}\rangle$ based on $\langle\{a\},\{b\}\rangle$. For this problem a good prediction is e or g. Any other prediction is considered wrong.

3 Two Types of Sequential Rules

To perform prediction with sequential rules, two steps needs to be performed [9], [12]. First, sequential rules are extracted from a training set of sequences (a sequence database). Second, these rules are used to make predictions for new sequences. For example, consider the problem of predicting the next webpages that a user will visit. The approach consists of first extracting sequential rules from logged sequences of webpages visited by previous users. Then, these rules are used to predict the webpages that new users will visit. In this paper, our contribution is to propose to use a new type of sequential rules for improving the quality of predictions. We thereafter define the two types of sequential rules that are compared in this paper.

3.1 Standard Sequential Rules

The first type is *standard sequential rules* [14], [8], [9], [12]. It is the most common type of sequential rules used in the literature. It is defined as follows.

A *sequential pattern* [4], [14] is a sequence that is a subsequence of one or more sequences of a sequence database SDB. Formally, a sequence $s_a = \langle A_1, A_2, ...A_e\rangle$ is said to be a subsequence of a sequence $s_b = \langle B_1, B_2, ...B_f\rangle$ if and only if there exists integers $1 \leq x_1 < x_2 ... < x_e \leq f$ such that $A_1 \subseteq B_{x1}$, $A_2 \subseteq B_{x2}$, $...A_e \subseteq B_f$. For instance, consider the sequence database of Figure 1 as SDB. The sequence $\langle\{b\}, \{f\}\rangle$ is a sequential pattern occurring in sequences s_1, s_2, s_3 and s_4. Another example is $\langle\{b\},\{f\},\{e\}\rangle$. It is a sequential pattern occurring in sequences s_1 and s_3.

A *standard sequential rule* $s_a \Rightarrow s_b$ is a sequential relationship between two sequential patterns s_a and s_b. The interpretation of a rule $s_a \Rightarrow s_b$ is that if s_a occurs in a sequence, it is likely to be followed by s_b in the same sequence. Formally, a standard sequential rule $\langle A_1, A_2, ... A_e\rangle \Rightarrow \langle B_1, B_2, ...B_f\rangle$ occurs in a sequence $\langle C_1, C_2, ...C_g\rangle$ if and only if there exists integers $1 \leq x_1 < x_2 \ < x_e$

$< y_1 < y_2 < y_e \leq f$ such that $A1 \subseteq C_{x1}$, $A_2 \subseteq C_{x2}$, ...$A_e \subseteq C_{xe}$ and $B_1 \subseteq C_{y1}$, $B_2 \subseteq C_{y2}$, ...$B_f \subseteq C_{ye}$. For example, the rule $\langle\{b\}, \{f\}\rangle \Rightarrow \langle\{e\}\rangle$ is a sequential rule occurring in sequences s_1 and s_3 of the database depicted in Figure 1. This rule is interpreted as if b is followed by f, it will be followed by e.

Standard sequential rules can be discovered by algorithms such as RuleGen [14]. These algorithms take two thresholds named *minsup* and *minconf* as parameters, which are set by the user. The algorithms return all sequential rules such that their support and confidence are respectively higher than these thresholds. The support and confidence of standard sequential rules are defined as follows.

The support of a standard sequential rule $s_a \Rightarrow s_b$ for a database SDB is denoted as $sup(s_a \Rightarrow s_b)$. It is defined as the number of sequences from SDB where the rule occurs divided by the number of sequences in SDB. For instance, consider the sequence database of Figure 1 as SDB. The rule $\langle\{b\}, \{f\}\rangle \Rightarrow \langle\{e\}\rangle$ has a support of 0.50 because it appears in sequences s_1 and s_3 and there are four sequences in SDB.

The confidence of a standard sequential rule $s_a \Rightarrow s_b$ for a database SDB is denoted as $conf(s_a \Rightarrow s_b)$. It is defined as the number of sequences from SDB, where s_a is followed by s_b, divided by the number of sequence where s_a occurs in SDB. For instance, consider the sequence database of Figure 1 as SDB. The rule $\langle\{b\}, \{f\}\rangle \Rightarrow \langle\{e\}\rangle$ appears in sequences s_1 and s_3 and its antecedent $\langle\{b\}, \{f\}\rangle$ occurs in s_1, s_2, s_3 and s_4. The rule $\langle\{b\}, \{f\}\rangle \Rightarrow \langle\{e\}\rangle$ has therefore a confidence of 2 / 4 = 0.5.

3.2 Partially-Ordered Sequential Rules

The second type of sequential rules that we consider is *partially-ordered sequential rules*, a new type of sequential rules that we have proposed recently [5], [6], [7]. We call these rules partially-ordered because the requirements of a sequential ordering inside the antecedent and inside the consequent of rules are eliminated. But the requirement of a sequential relationship between the antecedent and consequent of a rule is preserved.

A *partially-ordered sequential rule* is a sequential relationship between two unordered itemsets $I_a \Rightarrow I_b$ such that $I_a \cap I_b = \emptyset$ and $I_a, I_b \neq \emptyset$. The interpretation of a partially-ordered sequential rule $I_a \Rightarrow I_b$ is that if the items of I_a occur in a sequence (in any order), the items in I_b will occur afterward in the same sequence (in any order). Formally, we say that a rule $I_a \Rightarrow I_b$ occurs in a sequence $s = \langle I_1, I_2, ...I_n\rangle$ if and only if there exists an integer k such that $1 \leq k < n$, $I_a \subseteq \bigcup_{i=1}^{k} I_i$ and $I_b \subseteq \bigcup_{i=k+1}^{n} I_i$. For instance, consider the sequence database of Figure 1 as SDB. The rule $\{a, b, f\} \Rightarrow \{e\}$ appears in sequence s_1 and s_3. It means that if items a, b and f appears in a sequence in any order, it will be followed by e.

Partially-ordered sequential rules have the interesting property of being more general than standard sequential rules [5], [6], [7]. Several standard sequential rules can be represented by a single partially-ordered sequential rule. For example, the standard sequential rules $\langle\{a, b\}, \{c\}\rangle \Rightarrow \langle\{f\}\rangle$ and $\langle\{a\}, \{c\}, \{b\}\rangle \Rightarrow$

$\langle\{f\}\rangle$ are represented by a single partially-ordered sequential rule $\{a, b, c\} \Rightarrow \{f\}$.

Algorithms for mining partially-ordered sequential rules [5] take two thresholds named *minsup* and *minconf* as parameters, which are set by the user. The algorithms return all rules having a support and confidence respectively higher than these thresholds. Support and confidence for partially-ordered sequential rules are defined as follows.

The support of a partially-ordered sequential rule $I_a \Rightarrow I_b$ for a database SDB is denoted as $sup(I_a \Rightarrow I_b)$. It is defined as the number of sequences from SDB where the items in the rule occur, divided by the number of sequences in SDB.The confidence of a partially-ordered sequential rule for a database SDB is denoted as $conf(I_a \Rightarrow I_b)$. It is defined as the number of sequences in SDB where items the rule occur divided by the number of sequence where items in I_a appears.For instance, consider the sequence database of Figure 1 as SDB. The rule $\{a, b, c\} \Rightarrow \{e\}$ has a support of 0.5 because it appears in s_1 and s_2. Moreover, its confidence is 1 because its antecedent only appears in s_1 and s_2. Another example is the rule $\{a\} \Rightarrow \{b, c, e\}$, which has a support of 0.5 and a confidence of 0.6.

4 The Comparison Framework

We now present our comparison framework that we have designed to compare the prediction accuracy of standard sequential rules and partially-ordered sequential rules. We named this framework the *Predictor*. It a framework for sequence prediction that can be used with the two types of sequential rules. The predictor works in two phases.

1) Training. The *first phase* consists of mining sequential rules from a sequence database containing a set of training sequences. The user has to provide the sequence database and has to choose the types of sequential rules to be mined. If standard sequential rules are chosen, the RuleGen [14] algorithm is applied. If partially-ordered sequential rules are selected, TRuleGrowth [5] is applied. Note that TRuleGrowth allow specifying a parameter named *window_size*, which is not found in RuleGen. It allows specifying that patterns have to occur within a maximum number of consecutive itemsets. To provide the same functionality for RuleGen, we have modified it to also accept this parameter. As it will be shown in the experimental section, considering this additional constraint can improve the accuracy of predictions. Besides, note that we here only consider sequential rules containing a single item in the consequent because we are interested in predicting one item at a time for the application of web recommendation described in this paper.

2) Prediction. The *second phase* consists of predicting an item that will follow a sequence provided by the user. This phase is accomplished in two substeps. The *first step* is to scan all the rules to identify those that match with the sequence provided by the user. Here, *a rule is said to match with a sequence* if the antecedent appears in the sequence.

The *second step* is to generate a prediction based on the matching rules. This is performed by selecting one of the matching sequential rules. To select a rule, several criteria can be used. We have tested several of them and found that the criterion that gives the best results is to choose the rule with the highest score. We define the score of a rule p as $Score(p) = (c_1 conf(p) + c_2 sup(p)) \times length(p)$, where c_1 and c_2 are constants and $length(p)$ is the number of items contained in p that match with the sequence. In our tests, $c_1 = 0.7$ and $c_2 = 0.3$ generated the best results. Note however that other values of c_1 and c_2 could be used for other datasets and may provide better results. After a sequential rule has been selected based on the score, the *Predictor* makes the prediction by choosing the item in the rule consequent.

5 Experimentation

We have implemented the RuleGen and TRuleGrowth sequential rule mining algorithms, and the Predictor framework in Java.

Experiments were carried with two public click-stream datasets commonly used in the sequential pattern mining literature. The first dataset is **Kosarak** (`http://fimi.cs.helsinki.fi/data/`). It contains 990,000 sequences of click-stream data from an online news portal. To make the experiment faster, we only used the first 50,000 sequences of Kosarak. Each sequence has an average length of 7.97 items from 21,144 different items. The second dataset is **BMSWebView1** (BMS). It contains 59,601 sequences of click-stream data from an e-commerce website (`http://www.ecn.purdue.edu/KDDCUP/`). BMS differs from Kosarak in that sequences are shorter and that the set of different items is much smaller (497 items compared to 21,144 items). The average length of sequences in BMS is short with 2.51 items (σ =4.85). But it also contains several long sequences.

We have performed several experiments with the datasets. For each experiment, we have randomly divided each dataset into two sets: a training set and a test set. The division was made according to a parameter *training_ratio* that we have initially set to 50%. This parameter indicates the percentage of sequences from a dataset that is used for training. The training set is used for generating sequential rules. These rules are then used to generate predictions for each sequence of the test set. Statistics are recorded about the number of correct predictions and the total number of predictions generated. This allows computing two measures.

The first measure is the *accuracy*. We define it as the number of good predictions divided by the number of sequences in the test set. The second measure is named *matching rate*. It is defined as the number of sequences where a prediction was generated divided by the number of sequences in the test set. It is important to consider this measure because no prediction is generated for a sequence if there is no matching rule.

The initial parameters for all the experiments are as follows. The parameter *minsup* is set to 0.00055 for BMS and 0.002 for Kosarak. These values

were determined has the best values (giving the highest accuracy and matching rate) after executing several preliminary experiments. Similarly, we have found that 0.5 was the best value for $minconf$ for both datasets. The parameters $prefix\_size$ and $suffix\_size$ (cf. section 2) were set to 3. This means that the problem of sequence prediction is to predict an item from the last three itemsets of a sequence given the three preceding itemsets. Because we used these parameters, we only kept sequence containing at least 6 itemsets in each dataset. Lastly, the $window\_size$ parameter (cf. section 4) was set to 5.

5.1 Influence of $prefix\_size$.

The first experiment consists of varying $prefix\_size$ to assess its influence on the accuracy and matching rate for the two types of sequential rules (*SSR = Standard Sequential Rules, POSR = Partially-Ordered Sequential Rules*). As previously explained, $prefix\_size$ represents the number of preceding itemsets that are used for making a prediction (cf. section 2). Figure 2 respectively show the impact of varying this parameter from 1 to 10 for BMS and Kosarak. It can be seen that partially-ordered sequential rules can improve accuracy by up to 28% and matching rate by up to 60% compared to standard sequential rules.

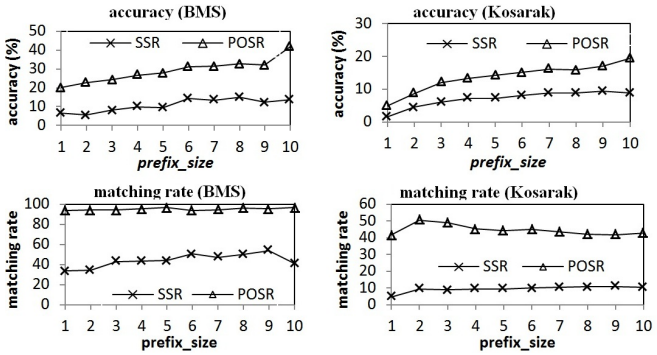

Fig. 2. Influence of $prefix\_size$ on accuracy and matching rate

5.2 Influence of $suffix\_size$.

The second experiment consists of varying $suffix\_size$ to assess its influence on prediction accuracy and matching rate. As explained in section 2, $suffix\_size$ indicates the number of itemsets that should be considered for making a prediction. Figure 3 respectively show the results obtained of varying $suffix\_size$ from 1 to 10 for BMS and Kosarak. These results show that partially-ordered sequential rules can improve accuracy by up to 26% and matching rate by up to 60% compared to standard sequential rules.

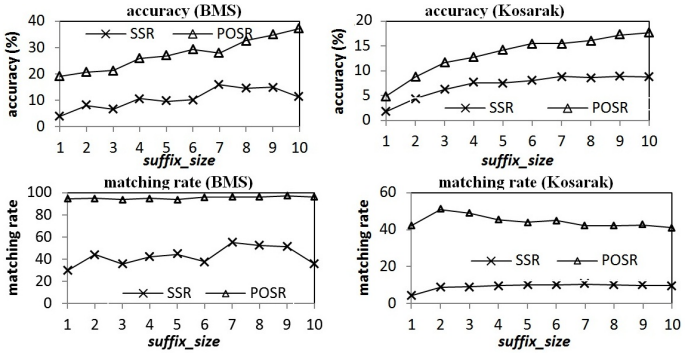

Fig. 3. Influence of $suffix\_size$

5.3 Influence of *training_ratio*

The third experiment consists of assessing the impact of the size of the training set on the accuracy of predictions by varying *training_ratio* from 10% to 90%. Figure 4 respectively show the results obtained with BMS and Kosarak. Predictions based on partially-ordered sequential rules were from 11% to 19% more accurate than predictions based on standard sequential rules. It can be observed that this is true even if a smaller training set is used for partially-ordered sequential rules. Note that for BMS, no results are available for $training\_ratio = 60$ for standard sequential rules (SSR) because not enough rules were found during the mining process.

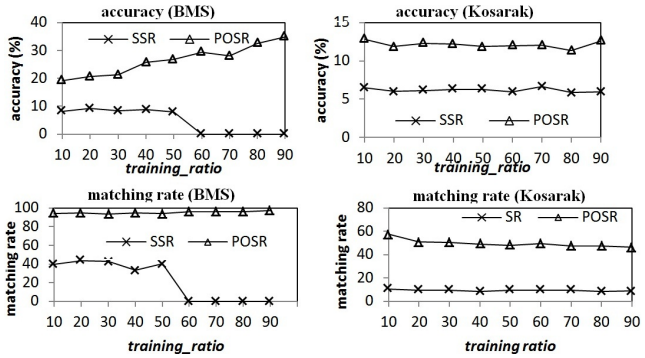

Fig. 4. Influence of the training ratio

5.4 Influence of the rule selection criterion

The fourth experiment consists of assessing the influence of the rule selection criterion to make a prediction when several rules match with a sequence. The

criterion previously suggested in section 4 was to select the matching rule with the highest score. An alternative approach that we have tested is to keep the top W matching rules with the highest score, and then to perform a majority vote on the items predicted by the W rules. The item with the most votes is then chosen as the prediction. Note that if W is set to 1, the result is the same as before. Results from his experiment are shown on Figure 5 for $W = 1, 10, 20,$ ...90 for BMS and Kosarak. Results show that using a majority vote improves accuracy by up to 6% for partially-ordered sequential rules, and up to 5% for standard sequential rules.

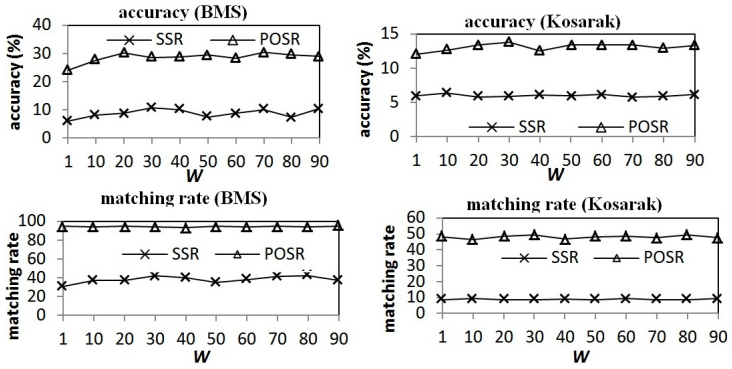

Fig. 5. Influence of the rule selection criterion

5.5 Influence of *window_size*

The fifth experiment consists of assessing the influence of *window_size*on the prediction accuracy by varying *window_size* from 2 to 10. Results of this experiment are shown on Figure 6. Results indicate that setting *window_size* to a value from 3 to 6 provided reasonable accuracy and that using larger values did not provide a major improvement.

5.6 Influence of making multiple predictions

For the sixth experiment, we have tested the possibility of making multiple predictions for each sequence instead of only one. This experiment is motivated by the fact that for some real applications more than one recommendation can be made to the user. For example, for webpage recommendation, more than one webpage recommendation can be made to the user by displaying several links on the same page. To assess the benefits of making multiple predictions, we have added a parameter Q, which is the number of predictions. The definition of accuracy has been adjusted as follows for multiple predictions. The accuracy is calculated as the number of sequences where at least one prediction is correct

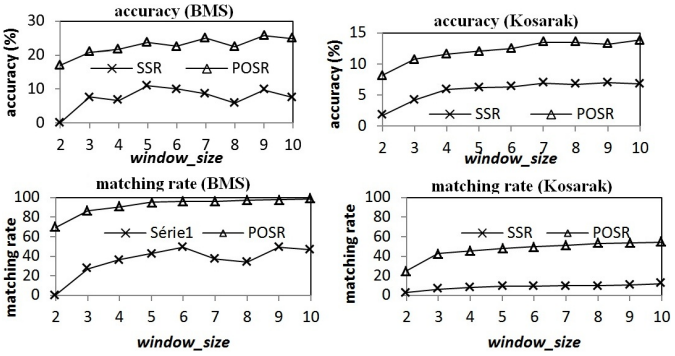

Fig. 6. Influence of *window_size*

divided by the total number of sequences in the test set. Results of varying Q from 1 to 10 are shown on Figure 7. As it was expected, allowing multiple predictions largely increased prediction accuracy (by up to 24% for partially ordered sequential rules and 7% for standard sequential rules).

5.7 Influence of the number of rules

The last experiment consisted of varying *minsup* to assess the influence of the sequential rule count on the prediction accuracy. For BMS, we varied *minsup* from 0.0006 to 0.0005. For Kosarak, we varied *minsup* from 0.003 to 0.001. These intervals were chosen because they provide an interesting view of the results. The prediction accuracy and the number of rules generated for BMS and Kosarak are shown in Figure 8. From these results, we can observe that the number of rules for the two types of rules varies differently because their definitions are different. Second, we can see that for the two types of rules, as soon as there are approximately 1000 to 10,000 rules, the accuracy remains more or less the same if the number of rules increase. This result is interesting because it provides a solution to the problem of choosing the *minsup* value. The solution is to use an algorithm for mining the top-k sequential rules such as the one that we have designed in previous work [6]. This algorithm can discover the top-k partially-ordered sequential rules where k is set by the user. The advantage of using this algorithm is that the user does not need to find a suitable value for the *minsup* thresholds by hand. For example, the user could set $k = 10,000$ to find the top $10,000$ rules and use them to make predictions.

6 Conclusion

Predicting the next element(s) of a sequence is a research problem with wide applications such as stock market prediction, consumer product recommendation, and web link recommendation. A popular approach to symbolic sequence recommendation is to discover sequential rules in a training set of sequences to then use

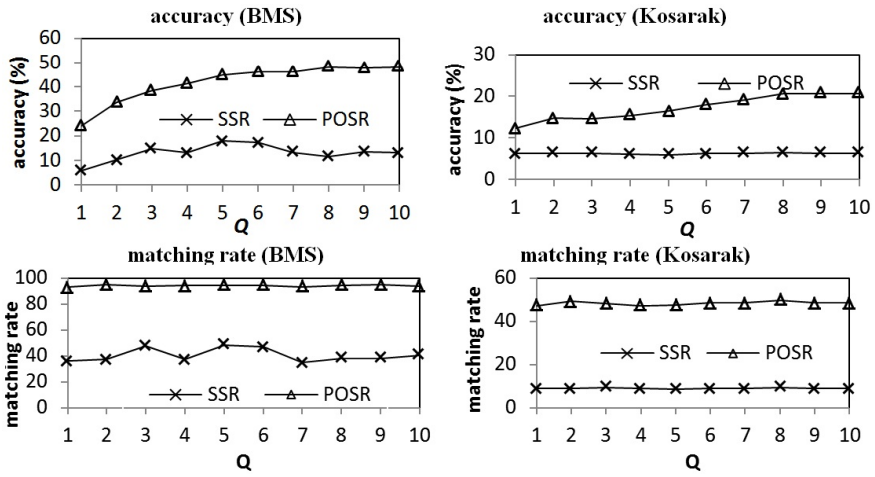

Fig. 7. Influence of making multiple predictions

these rules to make predictions. In this paper, we have explored the possibility of using a new type of sequential rules named partially-ordered sequential rules [5], [6] for sequence prediction. We have compared the prediction accuracy of these rules with standard sequential rules for the task of webpage recommendation under multiple scenarios (different prefix sizes, different suffix sizes, allowing multiple predictions, varying the size of the training set, performing a majority vote, choosing different window sizes and different minsup values). Overall, results show that using partially-ordered sequential rules instead of standard sequential rules can improve the prediction accuracy and matching rate by more than 30%, depending on the scenario.

In this paper, we have focused on improving prediction accuracy and matching rate. For future work, we plan to work on enhancing the performance in terms of execution time. We are working on designing a storage structure that will allow an efficient storage and retrieval of sequential rules to test if they are matching with a sequence. We also plan to consider the problem of sequence prediction with a user profile as it was done for standard sequential rules in [12]. We are also considering integrating ideas such as rule prioritization and rule pruning from works on associative classification to further improve the results (e.g. [8], [1]). The source code of the software presented in this paper is available for free at `http://www.philippe-fournier-viger.com/spmf/`.

References

1. Antonie, M.-L., Chodos, D., Zaiane, O.: Variations on Associative Classifiers and Classification Results Analyses. In: Zhao, Y., Zhang, C., Cao, L.: Post-Mining of Association Rules: Techniques for Effective Knowledge Extraction (2008)
2. Begleiter, R., El-Yaniv, R., Yona, G.: On Prediction Using Variable Order Markov Models, Journal of Artificial Intelligence Research, vol. 22, pp. 385-421 (2004)

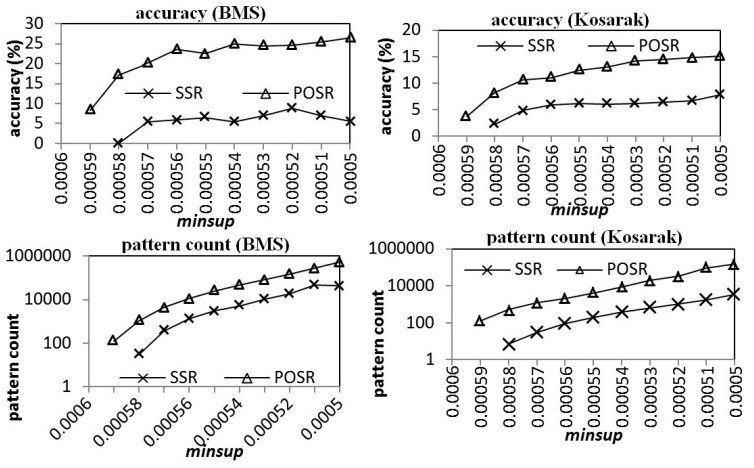

Fig. 8. Influence of the number of rules

3. Han, J., Kamber, M.,: Data Mining: Concepts and Techniques, 3rd ed. Morgan Kaufmann, San Francisco (2011).
4. Pei, J. et al.: Mining Sequential Patterns by Pattern Growth: The PrefixSpan Approach. IEE Trans. on Knowledge and Data Engineering. 16(11), 1420–1440 (2004)
5. Fournier-Viger, P., Wu, C.-W., Tseng, V.S., Nkambou, R.: Mining Sequential Rules Common to Several Sequences with the Window Size Constraint. In: L. Kosseim, D. Inkpen (eds.) AI 2012. LNCS, vol. 7310, pp. 299–304. Springer, Heidelberg (2012)
6. Fournier-Viger, P., Tseng, V. S.: Mining Top-K Sequential Rules. In: J. Tang, I. King, L. Chen, J. Wang (eds.) ADMA 2011. LNCS, vol. 7310, pp. 180–194. Springer, Heidelberg (2011)
7. Fournier-Viger, P., Nkambou, R., Tseng, V. S.: RuleGrowth: Mining Sequential Rules Common to Several Sequences by Pattern-Growth. In: ACM SAC 2011. ACM Press, pp. 954-959. (2011)
8. Liu, B., Hsu, W., Ma, Y.: Integrating Classification and Association Rule Mining. In: Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), pp.80–86. AAAI Press, New York (1998)
9. Liu, D.-R., Lai, C.-H.: A hybrid of sequential rules and collaborative filtering for product recommendation. Information sciences, 179, 3505-3519 (2009).
10. Lo, D., Khoo, S.-C., Wong, L.: Non-redundant sequential rules Theory and algorithm. Information Systems. 34(4-5), 438-453 (2009)
11. Frez-Ortiz, J. A., Calera-Rubio, J., Forcada, M. L., Dorer, G., Bischof, H., Hornik, K.: Online symbolic-sequence prediction with discrete-time recurrent neural networks. In: G. Dorffner, H. Bischof, K. Hornik (eds.) ICANN 2001 LNCS, vol. 2001, pp. 719–724. Springer, Heidelberg (2001)
12. Pitman, A. Zanker, M.: An Empirical Study of Extracting Multidimensional Sequential Rules for Personalization and Recommendation in Online Commerce. In: 10th Intern. Conf. on Wirtschaftsinformatik(2011)
13. Sun, R., Giles, C. L.: Sequence Learning: From Recognition and Prediction to Sequential Decision Making. IEEE Intelligent Systems, 16 (4) (2001)
14. Zaki, M. J.: SPADE: An Efficient Algorithm for Mining Frequent Sequences. Machine Learning. 42(1/2), 31-60 (2001)