SPaR-FTR: An Efficient Algorithm for Mining Sequential Patterns-Based Rules

José Kadir Febrer-Hernández[1(✉)], Raudel Hernández-León[1], José Hernández-Palancar[1], and Claudia Feregrino-Uribe[2]

[1] Centro de Aplicaciones de Tecnologías de Avanzada (CENATAV), 7a # 21406 E/ 214 and 216, Rpto. Siboney, C.P. 12200 Playa, La Habana, Cuba
{jfebrer,rhernandez,jpalancar}@cenatav.co.cu

[2] Instituto Nacional de Astrofísica, Óptica y Electrónica (INAOE), Luis Enrique Erro No. 1, Sta. María Tonantzintla, CP:72840 Puebla, Mexico
cferegrino@ccc.inaoep.mx

Abstract. In this paper, we propose a novel algorithm for mining Sequential Patterns-based Rules, called SPaR-FTR. This algorithm introduces a new efficient strategy to generate the set of sequential rules based on the interesting rules of size three. The experimental results show that the SPaR-FTR algorithm has better performance than the main algorithms reported to discover frequent sequences, all they adapted to mine this kind of sequential rules.

Keywords: Data mining · Sequential patterns · Rule mining

1 Introduction

An important part of the Sequential Patterns-based Classification (SPaC) is the process of mining the set of classification rules, called SPaRs (Sequential Patterns-based Rules). These rules are mined from a class-transaction dataset, where a SPaR describes an implicative co-occurring relationship between a sequence α and a class c.

It is very common to confuse sequences of items with itemsets. In itemsets, an item can occur at most once but in a sequence, an itemset can occur multiple times. Additionally, in itemset mining, $(abc) = (cba)$ but in sequence mining, $\langle (ab)\ c\rangle \neq \langle c\ (ab)\rangle$.

Sequential Patterns-based Classification has been used in different tasks, for example: text classification [1], document-specific keyphrase extraction [2], text segmentation [3,4], web document classification [5,6], determination of DNA splice junction types [7], e-learning [8], automatic image annotation [9], among others.

In SPaC, it is assumed that a set of items $I = \{i_1,\ i_2,\ ...,\ i_l\}$, a set of classes C, and a set of transactions T are given, where each transaction $t \in T$ consists of a sequence $\langle \alpha_1\ \alpha_2\ ...\ \alpha_n\rangle$, so that $\alpha_i \subseteq I$, and a class $c \in C$. The Support of

A. Pardo and J. Kittler (Eds.): CIARP 2015, LNCS 9423, pp. 643–650, 2015.
DOI: 10.1007/978-3-319-25751-8_77

a sequence α, denoted as $Sup(\alpha)$, is the fraction of transactions in T containing α (see Eq. 1).

$$Sup(\alpha) = \frac{|T_\alpha|}{|T|} \tag{1}$$

where T_α is the set of transactions in T containing α (see Def. 1) and $|\cdot|$ is the cardinality operator.

Definition 1. *Let $\alpha = \langle \alpha_1\ \alpha_2\ ...\ \alpha_n \rangle$ and $\beta = \langle \beta_1\ \beta_2\ ...\ \beta_m \rangle$ be sequences, we will say that α is contained in β if there exists integers $1 \leq j_1 < j_2 < ... < j_n \leq m$ such that $\alpha_1 \subseteq \beta_{j_1}$, $\alpha_2 \subseteq \beta_{j_2}$, ..., $\alpha_n \subseteq \beta_{j_n}$, with $\beta_{j_i} \in \beta$.*

A SPaR is an implication of the form $\alpha \Rightarrow c$ where α is a sequence and $c \in C$. The size of a SPaR is defined as its cardinality, a SPaR containing k itemsets (including the class) is called a k-SPaR. The rule $\alpha \Rightarrow c$ is held in T with certain Support and Confidence (see Eqs. 2 and 3). If both Support and Confidence values of a SPaR $r : \alpha \Rightarrow c$ are greater than to the user-specified thresholds, r is declared to be an interesting SPaR.

$$Sup(\alpha \Rightarrow c) = Sup(\alpha \otimes \langle c \rangle) \tag{2}$$

where $\otimes$ is the concatenation operator (see Def. 2).

$$Conf(\alpha \Rightarrow c) = \frac{Sup(\alpha \Rightarrow c)}{Sup(\alpha)} \tag{3}$$

Definition 2. *Let $\alpha = \langle \alpha_1\ \alpha_2\ ...\ \alpha_n \rangle$ and $\beta = \langle \beta_1\ \beta_2\ ...\ \beta_m \rangle$, we will call the sequence $\langle \alpha_1\ \alpha_2\ ...\ \alpha_n\ \beta_1\ \beta_2\ ...\ \beta_m \rangle$ the concatenation of α and β, and we will use the operator $\otimes$ to indicate it.*

In this paper, we introduce an efficient strategy to generate the set of SPaRs based on the interesting rules of size three. The rest of the paper is organized as follows. The next section describes the related work. Our proposal are presented in Section three. In the fourth section the experimental results are shown. Finally, the conclusions and future works are given in section five.

2 Related Work

In the last decades, some works have used sequential patterns to increase the accuracy of classifiers. In these works, the extracted sequential patterns are considered to be important features and are used to build the classification model. However, there are not reported algorithms (with pseudo code or source code included) that directly compute the set of SPaRs. We assume that this is due to these algorithms can be obtained from the sequential pattern mining algorithms, without any algorithmic complications.

In general, most of the sequential pattern mining algorithms can be split into two main groups: (1) apriori-like algorithms (AprioriAll, AprioriSome and

DynamicSome [10], GSP [11], SPIRIT [12]) and (2) pattern-growth based algorithms (PrefixSpan [13], LAPIN [14], PRISM [15]).

In [11], the authors proposed the GSP algorithm, which includes time constrains and taxonomies in the mining process. In the experiments, the authors show that GSP runs 2 to 20 times faster than apriori-like algorithms [10]. Following similar ideas, the use of regular expressions was introduced in the SPIRIT algorithm [12].

The PrefixSpan algorithm, proposed in [13], is based on recursively constructing the patterns by growing on the prefix, and simultaneously, restricting the search to projected datasets. This way, the search space is reduced at each step, allowing for better performance in the presence of small support thresholds.

PRISM, the algorithm introduced by Karam Gouda in [15], uses a vertical approach for enumeration and support counting, based on the novel notion of primal block encoding, which is based on prime factorization theory.

The LAPIN (LAst Position INduction) algorithm [14] uses an item-last-position list and a prefix border position set instead of the tree projection or candidate generate-and-test techniques introduced so far.

Our proposal also stores a list of occurrence positions but unlike LAPIN that stores the last position of each single item in each transaction, SPaR-FTR stores for each interesting sequence α, and for each transaction t, a list with the occurrence positions of α in t.

3 SPaR-FTR Algorithm

In this section, we describe the SPaR-FTR algorithm, which uses the Support and Confidence measures to evaluate the candidate SPaRs and generates all candidate SPaRs from the set of interesting 3-SPaRs. Let $r : \alpha \Rightarrow c$ be an interesting SPaR and T be a transactional dataset, SPaR-FTR stores for each $t \in T$, a list L_t with the occurrence positions of α in t (see Def. 3).

Definition 3. *Let $\alpha = \langle \alpha_1\ \alpha_2\ ...\ \alpha_n \rangle$ and $\beta = \langle \beta_1\ \beta_2\ ...\ \beta_m \rangle$ be sequences such that α is contained in β (i.e. exists integers $1 \leq j_1 < j_2 < ... < j_n \leq m$ such that $\alpha_1 \subseteq \beta_{j_1}$, $\alpha_2 \subseteq \beta_{j_2}$, ..., $\alpha_n \subseteq \beta_{j_n}$), we will call occurrence position of α in β ($occP(\alpha, \beta)$) to:*

- *the set of positions of all possible β_{j_n} in β, if $\mid \alpha \mid \leq 2$;*
- *the least position of all possible β_{j_n} in β, if $\mid \alpha \mid > 2$.*

In Table 1, five transactions and the occurrence positions of three sequences of different sizes are shown. Notice that when $\mid \alpha \mid > 2$ (*e.g.* $\langle a\ f\ b \rangle$ in transaction 2) we could also have several β_{j_n} (*e.g.* $(b : 4)$ and $(b : 6)$) but the proposed strategy to generate the candidate rules, only require the least of all.

Similar to the reported algorithms for frequent sequence mining [10,13–15], in a first step, SPaR-FTR computes all the interesting 2-SPaRs using the Support and Confidence measures to evaluate them. As we mentioned above, SPaR-FTR stores for each interesting SPaR $r : \alpha \Rightarrow c$ (of any size) and for each transaction

Table 1. Example of five transactions and the occurrence positions of three sequences off different sizes.

Tid	Sequence	⟨b⟩	⟨a f⟩	⟨a f b⟩
1	$\langle a\ \ b\rangle$	(b:2)		
2	$\langle cd\ \ a\ \ ef\ \ b\ \ cd\ \ ab\rangle$	(b:4), (b:6)	(f:3)	(b:4)
3	$\langle af\ \ \ f\rangle$		(f:2)	
4	$\langle af\ \ ef\ \ bf\rangle$	(b:3)	(f:2), (f:3)	(b:3)
5	$\langle b\rangle$	(b:1)		

$t \in T$, a list with the occurrence positions of α in t. Later, in a second step, SPaR-FTR obtains the set of 3-SPaRs (see Alg. 1) by combining the 2-SPaRs belonging to the same class. Unlike the reported algorithms mentioned above, which generates the k-SPaRs either by combining the interesting $(k-1)$-SPaRs with a common $k-2$ prefix or using a depth first search strategy, SPaR-FTR computes the k-SPaRs ($k > 3$) by combining the interesting $(k-1)$-SPaRs and the interesting 3-SPaRs obtained in the second step (see Alg. 2).

Algorithm 1. Pseudo code for computing the interesting 3-SPaRs.

Input: Transactional dataset T, Support threshold $minSup$ and Confidence threshold $minConf$.
Output: Set of interesting 3-SPaRs.

1 $L_1 \leftarrow \{twoInterestingSPaR(T)\}$
2 $L_2 \leftarrow \emptyset$
3 **foreach** $c \in C$ **do**
4 **foreach** $(r_1 : \langle i\rangle \Rightarrow c) \in L_1$ **and** $(r_2 : \langle j\rangle \Rightarrow c) \in L_1$ **do**
5 **foreach** $t \in T$ **do**
6 **if** $\exists op_1 > op_2$ $(op_1 \in occP(\langle j\rangle, t)$ **and** $op_2 \in occP(\langle i\rangle, t))$ **then**
7 $r_3 \leftarrow \langle i\rangle \otimes \langle j\rangle \Rightarrow c$
8 Computes support Sup and confidence $Conf$ of r_3
9 **if** $(r_3.Sup > minSup)$ **and** $(r_3.Conf > minConf)$ **then**
10 $L_2 \leftarrow L_2 \cup \{r_3\}$
11 **end**
12 **end**
13 **end**
14 **end**
15 **end**
16 **return** L_2

The main differences between algorithms 1 and 2 are in lines 4 and 6. In line 4 of Algorithm 1, the 2-SPaRs of the same class are combined to generate the candidate 3-SPaRs while in Algorithm 2, the $(k-1)$-SPaRs are combined with

Algorithm 2. Pseudo code for computing the interesting k-SPaRs

Input: Set of interesting $(k-1)$-SPaRs, set of interesting 3-SPaRs, Support threshold $minSup$ and Confidence threshold $minConf$.
Output: Set of interesting k-SPaRs.

```
1  L1 ← (k − 1)-SPaRs
2  L2 ← 3-SPaRs
3  L3 ← ∅
4  foreach c ∈ C do
5      foreach (r1 : ⟨α1 ... αk−1⟩ ⇒ c) ∈ L1 and (r2 : ⟨αk−1 β⟩ ⇒ c) ∈ L2 do
6          foreach t ∈ T do
7              if ∃op1 (op1 ∈ occP(⟨αk−1 β⟩, t) and op1 > occP(⟨α1 ... αk−1⟩, t))
               then
8                  r3 ← ⟨α1 ... αk−1⟩ ⊗ ⟨β⟩ ⇒ c
9                  Computes support Sup and confidence Conf of r3
10                 if (r3.Sup > minSup) and (r3.Conf > minConf) then
11                     L3 ← L3 ∪ {r3}
12                 end
13             end
14         end
15     end
16 end
17 return L3
```

the 3-SPaRs to generate the candidate k-SPaRs. In case of line 6, the difference is a direct consequence of the definition of occurrence position (see Def. 3 in this section.

4 Experimental Results

In this section, we present the results of our experimental comparison between SPaR-FTR and the main sequence mining algorithms reported in the literature (GSP [10], PrefixSpan [13], LAPIN [14] and PRISM [15]), all them adapted to compute the interesting SPaRs. All codes (implemented in ANSI C standard) were provided by their authors and adapted by us to compute the interesting SPaRs.

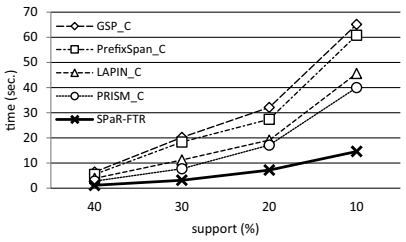
The experiments were conducted using several document collections, three in our case: AFP (*http://trec.nist.gov*), TDT (*http://www.nist.gov*) and Reuter (*http://kdd.ics.uci.edu*). The characteristics of these datasets are shown in Table 2. Our tests were performed on a PC with an Intel Core 2 Quad at 2.50 GHz CPU with 4 GB DDR3 RAM, running on Windows 7 system.

In the same way as in other works [10], for all used datasets, sentences are distinguished and ordered in each document. This means that the document is

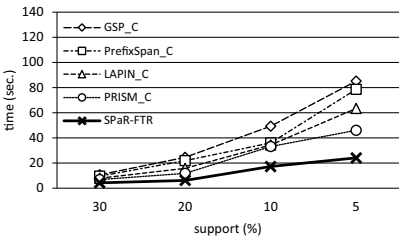
Table 2. Tested datasets characteristics.

Dataset	#instances	#classes
AFP	711	22
TDT	2978	92
Reuter	21578	115

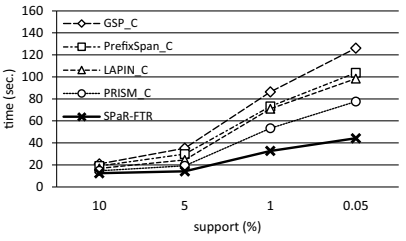
considered as being an ordered list of sentences. Each sentence is considered as being an unordered set of words. If we compare the market basket analysis problem with our approach, then a document plays the role of a client, the sentences from a document play the role of all the transactions for this client, the position of the sentence within the document plays the role of the date, and the set of words from a sentence plays the role of a list of bought items. Therefore, we represented the document as a sequence of itemsets where each one corresponds with the set of words of each sentence.



(a) AFP.



(b) TDT.



(c) Reuter.

Fig. 1. Runtime comparison using AFP, TDT and Reuter document collections.

In order to evaluate the performance of the SPaR-FTR algorithm, we process the three document collections with different support thresholds. In general, document collections are very sparse (with low transaction overlapping degree). Therefore, low Support thresholds are required, mainly in Reuter collection, where there are 21578 transactions and 115 classes.

In figures 1(a), 1(b) and 1(c), we show the result of all evaluated algorithms using different Support thresholds and a Confidence threshold set to 0.5. We do

not test different Confidence values because the volume of the SPaRs depends on the Support threshold and we are evaluating the efficiency of our algorithm to generate the set of SPaRs. Notice that we add the characters "_C" to the name of the algorithms to specify that they are the adaptation of the original sequence mining algorithms mentioned above.

In the three experiments, the SPaR-FTR algorithm shows the best performance of all evaluated algorithms. The main reason of this result is that the candidates generation strategy, introduced in SPaR-FTR, generates less candidate rules than the other algorithms. As another experiment, we count the number of candidate SPaRs generated for each evaluated algorithm. In Table 3, we show the approximate results, in thousands, obtained on Reuter collection.

Table 3. Approximate number of candidate SPaRs, in thousands, obtained on Reuter collection.

	Support thresholds (%)			
Algorithms	10	5	1	0.05
GSP_C	25.3	46.2	80.9	112.3
PrefixSpan_C	23.9	41.1	73.6	104.2
LAPIN_C	21.3	37.4	68.7	97.5
PRISM_C	19.3	34.6	64.6	91.2
SPaR-FTR	16.2	28.8	52.1	71.9

Notice that SPaR-FTR generates 15 % less candidate rules (for all Support thresholds) than PRISM_C algorithm, which has the second better performance. Therefore, based on our experiments we can conclude that SPaR-FTR has good scalability with respect to the number of transactions and with respect to the decreasing of the Support threshold.

5 Conclusions

In this paper, we have proposed a novel algorithm for mining Sequential Patterns-based Rules, called SPaR-FTR, which introduces a new efficient strategy to generate the set of SPaRs based on the interesting rules of size three. The experimental results show that the SPaR-FTR algorithm has better performance than the main algorithms reported to discover frequent sequences, all they adapted to mine this kind of sequential rules.

As future work, we are going to study the problem of producing SPaRs with multiple labels, it means rules with multiple classes in the consequent. This kind of rules could be useful for problems where some documents can belong to more than one topic.

References

1. Buddeewong, S., Kreesuradej, W.: A new association rule-based text classifier algorithm. In: Proceedings of the 17th IEEE International Conference on Tools with Artificial Intelligence, pp. 684–685 (2005)

2. Xei, F., Wu, X., Zhu, X.: Document-specific keyphrase extraction using sequential patterns with wildcards. In: Proceedings of the IEEE 14th International Conference on Data Mining (2014)
3. Cesario, E., Folino, F., Locane, A., Manco, G., Ortale, R.: Boosting text segmentation via progressive classification. Knowl. Inf. Syst. **15**(3), 285–320 (2008)
4. García-Hernández, R.A., Martínez-Trinidad, J.F., Carrasco-Ochoa, J.A.: A fast algorithm to find all the maximal frequent sequences in a text. In: Sanfeliu, A., Martínez Trinidad, J.F., Carrasco Ochoa, J.A. (eds.) CIARP 2004. LNCS, vol. 3287, pp. 478–486. Springer, Heidelberg (2004)
5. Shettar, R.: Sequential Pattern Mining from Web Log Data. International Journal of Engineering Science and Advanced Technology **2**, 204–208 (2012)
6. Haleem, H., Kumar, P., Beg, S.: Novel frequent sequential patterns based probabilistic model for effective classification of web documents. In: 2014 International Conference on Computer and Communication Technology (ICCCT), pp. 361–371 (2014)
7. Berzal, F., Cubero, J.C., Sánchez, D., Serrano, J.M.: ART: A Hybrid Classification Model. Mach. Learn. **54**(1), 67–92 (2004)
8. Faghihi, U., Fournier-Viger, P., Nkambou, R., Poirier, P.: A generic episodic learning model implemented in a cognitive agent by means of temporal pattern mining. In: Chien, B.-C., Hong, T.-P., Chen, S.-M., Ali, M. (eds.) IEA/AIE 2009. LNCS, vol. 5579, pp. 545–555. Springer, Heidelberg (2009)
9. Teredesai, A.M., Ahmad, M.A., Kanodia, J., Gaborski, R.S.: CoMMA: A Framework for Integrated Multimedia Mining Using Multi-relational Associations. Knowl. Inf. Syst. **10**(2), 135–162 (2006)
10. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Proceedings of the Eleventh International Conference on Data Engineering, pp. 3–14 (1995)
11. Srikant, R., Agrawal, R.: Mining sequential patterns: generalizations and performance improvements. In: Proceedings in the 5th International Conference Extending Database Technology, pp. 3–17 (1996)
12. Garofalakis, M., Rastogi, R., Shim, K.: SPIRIT: Sequential pattern mining with regular expression constraints. In: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 223–234 (1999)
13. Pei, J., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal U., Hsu, M.: PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proceedings of the 17th International Conference on Data Engineering, pp. 215–224 (2001)
14. Yang, Z., Wang, Y., Kitsuregawa, M.: LAPIN: effective sequential pattern mining algorithms by last position induction for dense databases. In: Kotagiri, R., Radha Krishna, P., Mohania, M., Nantajeewarawat, E. (eds.) DASFAA 2007. LNCS, vol. 4443, pp. 1020–1023. Springer, Heidelberg (2007)
15. Gouda, K., Hassaan, M., Zaki, M.J.: Prism: An effective approach for frequent sequence mining via prime-block encoding. J. Comput. Syst. Sci. **76**(1), 88–102 (2010)
16. Yu, X., Li, M., Lee, D.G., Kim, K.D., Ryu, K.H.: Application of closed gap-constrained sequential pattern mining in web log data. In: Zeng, D. (ed.) Advances in Control and Communication, LNEE, vol. 137, pp. 649–656. Springer, Heidelberg (2012)
17. Liao, V., Chen, M.: An efficient sequential pattern mining algorithm for motifs with gap constraints. In: Proceedings of the 2012 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (2012)