Sequence mining using FCA and the NextPriorityConcept algorithm

Salah Eddine Boukhetta, Christophe Demko, Jérémy Richard, and Karell Bertet

Laboratory L3i, La Rochelle University, La Rochelle, France

Abstract. In this paper, we are interested in sequence mining using GALACTIC, a new library based on Formal Concept Analysis (FCA) for generating a concept lattice from heterogeneous and complex data. Inspired from pattern structure theory, data in Galactic are described by predicates according to their types and a system of plugins which allows an easy integration of new characteristics, descriptions and strategies. We present new plugins to describe and analyze a sequential dataset with predicates: the KCS description (where a set of sequences are described by the set of K-Common Subsequences), the MCS description (where Maximal Common Subsequences are used to describe a sequential dataset) and the PCS description (where the Prefix Common Subsequence is used). Experimentation on both real and synthetic datasets shows the effectiveness of our plugins in terms of the number of generated predicates and concepts.

Keywords: Formal Concept Analysis, Lattice · Pattern structure · Sequences · Sequence mining

1 Introduction

Sequences appear in many areas: sequences of words in a text, trajectories, sequences of life stage, sequences of surfing on the internet, or sequences of buying products in a supermarket. A sequence is an ordered list of symbols, sets or events. Sequence mining is a topic of data mining which aims at finding patterns in a database of sequences that appears more frequently, where patterns can be subsequences, prefixes, suffixes, subsequences according to a sliding window [23].

The first generation of algorithms emerged in the 90s with the article of Agrawal and Srikant [1] that extends the well-known Apriori algorithm to sequence mining. The Apriori algorithm can be used to discover frequent subsequences such as $\langle\{Milk, Coffee\}, \{Sugar\}\rangle$ in a customer transaction database, indicating that {Milk, Coffee}, followed by {Sugar}, are frequently purchased together by most customers. Many algorithms were proposed to improve the time computation of Apriori algorithm, and could be categorized in three classes [26]: horizontal formatting methods (GSP [27], PSP [24]), vertical formatting methods (Spade [31], Spam [2]) and projection-based methods (FreeSpan [19],

 Published in Francisco J. Valverde-Albacete, Martin Trnecka (Eds.): Proceedings of the 15th International Conference on Concept Lattices and Their Applications, CLA 2020, pp. 209–222, 2020.

PrefixSpan [25]). All these algorithms take as input a database of sequences and a minimum support threshold, and generate the set of frequent subsequences. Algorithms with horizontal formatting databases require multiple scans of the database, while using vertical formatting needs at most three database scans for the construction of the ID-list tables, which allows a fast calculation of the support [31]. Projection-based methods use projected databases by means of many types of extensions [19].

For big databases and a short minimum support, these algorithms take a huge computing time, and generate a too large number of patterns that are difficult to interpret and contain redundancy. For example, subsequences $\langle a, b, c\rangle$, $\langle a, b\rangle$, $\langle a, c\rangle$, $\langle b, c\rangle$, $\langle a\rangle$, $\langle b\rangle$, *and* $\langle c\rangle$, all with the same support, are redondant while $\langle a, b, c\rangle$ clearly represents all the others.

A second generation of algorithms focuses on maximal patterns also denoted closed patterns because they verify a well-known property of closure. Many algorithms directly address this problem (CloSpan [30], ClaSP [18] and BIDE [28]). Some algorithms also appear within

Formal Concept Analysis (FCA) frameworks [15] and their extensions to pattern structures[14] [21], where the lattice property of closed patterns is promoted. We can mention here an article for analysing titles of papers using suffix tree [10], mining medical care trajectories using pattern structures [7], sequence mining to discover rare patterns [8]. Also other studies on demographic sequences with FCA and pattern structures, [16] and [17].

Formal Concept Analysis (FCA) appears in 1982 [29], then in the Ganter and Wille's 1999 work [15]. It is based on mathematical ordered sets theory, and the theory of lattices introduced by Birkhoff in 1940 [5]. Starting from a binary relation between a set of objects and a set of attributes, formal concepts are built as maximal sets of objects in relation with maximal sets of attributes. The whole set of concepts equipped with a specialization/generalization relation forms a partially ordered set with the lattice property called the concept lattice. This lattice represents the initial data where concepts are clusters of similar objects, and the concept lattice is a hierarchy of clusters.

FCA is issued from a branch of applied lattice theory that first appeared in the Barbut and Monjardet's 1970 work [3]. By means of derivation operators between objects and attributes forming a Galois connection and whose composition is a closure operator [4], nice bijections between (reduced) binary data, lattices and bases of rules have been stated, providing efficient methods of data analysis based on these canonical structures, thus with few tuning. The lattice property guarantees both a hierarchy of clusters, and a complete and consistent navigation structure for interactive approaches [12].

The formalism of pattern structures [14] [21] and abstract conceptual navigation [11,13] extend FCA to deal with non binary data, where data is described by patterns such that the patterns space must be organised as a semi-lattice in order to maintain a Galois connection between objects and their descriptions. By FCA framework, pattern lattice and bases of rules are defined, where a concept is composed of a subset of objects together with their common patterns, and

a rule possesses patterns in premises and conclusions. However, pattern lattices are huge, often untractable [20], and the need for approaches to drive the search towards the most interesting patterns is a current challenge.

Inspired from pattern structures, the NextPriorityConcept algorithm, introduced in a recent article [9], proposes a user-driven patterns mining approach for heterogeneous and complex data as input. This algorithm allows a generic pattern computation through specific *descriptions* of objects by predicates. It also proposes to reduce predecessors of a concept by refinement of a set of objects into a fewer one through specific user exploration *strategies*, thus a reduction of the number of generated patterns.

In this article, we propose to specialize this NextPriorityConcept algorithm for new sequence mining approaches with three descriptions and two strategies dedicated to sequences. As descriptions, we propose the classical maximal common subsequences (MCS), the maximal prefix (PCS), and the k-subsequences (KCS). Therefore a set of sequences can be mined by one of these three descriptions in a generic way. We also propose two strategies of pattern exploration in order to reduce a given set of sequences into their predecessors in the pattern lattice. The first strategy (simple) considers all the possible subsets of sequences as in classical approaches, while the second one (AAS) proposes a refinement of a set of sequences by adding only one item to their common description.

Section 2 introduces a short description of the NextPriorityConcept algorithm and basic definitions related to sequence mining. Section 3 will be dedicated to our new sequence descriptions and strategies. Experimental results are presented in Section 4.

2 Preliminaries

2.1 Description of the NextPriorityConcept algorithm

The NextPriorityConcept algorithm [9] computes concepts for heterogeneous and complex data G as inputand its main characteristics are:

Heterogeneous data as input, described by specific predicates. The algorithm introduces the notion of *description* δ as an application to provide predicates describing a set of objects $A \subseteq G$ according to their characteristics, each predicate being specific to one type of characteristic. Predicates describing the objects A are locally computed by a specific treatment for each type of characteristics, depending on whether it is digital, discrete or more complex, and the final description δ is the union of these predicates. Each concept $(A, \delta(A))$ is then composed of a subset of objects A together with a set of predicates $\delta(A)$ describing them. Such generic use of predicates makes it possible to consider heterogeneous data as input, i.e. digital, discrete or more complex data. As in pattern structutres, the descriptions must verify $\delta(A) \sqsubseteq \delta(A')$ for $A' \subseteq A$ in order to obtain a lattice. However, unlike classical pattern structures, predicates are not globally computed in a preprocessing step, but locally for each concept.

Concept lattice generation. NextPriorityConcept algorithm is inspired from Bordat's algorithm[6], also found in Linding's work [22], that recursively computes the immediates successors of a concept, starting with the bottom concept. It is a dual version that computes the immediate predecessors of a concept, starting with the top concept $(G, \delta(G))$ containing the whole set of objects, until no more concepts can be generated. The use of a priority queue ensures that each concept is generated before its predecessors, and a mechanism of propagation of constraints ensures that meets will be computed. NextPriorityConcept computes a concept lattice and therefore is positioned in FCA framework, with the possibility of extraction of rules, closure computations or navigation in the lattice, that can be useful in many fields of pattern mining and discovery.

Predecessors selection by specific strategies. The algorithm also introduces the notion of *strategy* σ to provide predicates (called *selectors*) generating the predecessors of a concept $(A, \delta(A))$ according to each characterisitc. A selector proposes a way to refine the description $\delta(A)$ to a reduced set $A' \subseteq A$ of objects. Selectors can be defined either by a restriction on the description $\delta(A)$ or on the set A. Several strategies are possible to generate predecessors of a concept, going from the naive strategy classically used in FCA that considers all the possible predecessors, to strategies reducing the number of predecessors in order to obtain smaller lattices. Let us observe that selectors are only used for the predecessors generation, they are not kept either in the description or in the final set of predicates. Therefore, choosing or testing several strategies at each iteration in a user-driven pattern discovery approach would be interesting. It is also possible to introduce a filter (or meta-strategy) on the selector as for example maximal support (number of objects) of a concept, or entropy (according to a class information).

Galactic. The use of predicates regardless of the data allows a generic implementation of our algorithm which, mixed with a system of plugins, makes it possible easy integration of new kinds of data (description and strategies). `Galactic`[1] (**GA**lois **LA**ttices, **C**oncept **T**heory, **I**mplicational systems and **C**losures) is a development platform of our algorithm allowing easy integration of new plugins.

The main result in [9] states that the NextPriorityConcept algorithm computes the formal context $\langle G, P, I_P \rangle$ and its concept lattice (where P is the set of predicates describing the characteristics, and $I_P = \{(a, p),\ a \in G, p \in P\ :\ p(a)\}$ is the relation between objects and predicates) if the description δ verifies $\delta(A) \sqsubseteq \delta(A')$ for $A' \subseteq A$. The run-time of NextPriorityConcept algorithm has a complexity $O(|\mathcal{B}|\,|G|\,|P|^2\,(c_\sigma + c_\delta))$ (where $\mathcal{B}$ is the number of concepts, c_σ is the cost of the strategy and c_δ is the cost of the description), and a space memory in $O(w\,|P|^2)$ (where w is the width of the concept lattice).

[1] https://galactic.univ-lr.fr

2.2 Sequences

A *sequence* $s = \langle s_1, s_2, \ldots s_n \rangle$ is a list of ordered events belonging to an alphabet Σ of event types. The length $|s|$ of a sequence s is the number of its elements, here $|s| = n$.

Let us denote G a database of sequences.

Example 1 in Table 1 represents a database G which contains 4 sequences with the alphabet $\Sigma = \{A, C, G, T\}$.

A sequence $a = \langle a_1 a_2 \ldots a_n \rangle$ is a *subsequence* of a sequence $b = \langle b_1 b_2 \ldots b_m \rangle$, and we write[2] $a \sqsubseteq_s b$ if there are integers $1 \leq i_1 < i_2 < \ldots < i_n \leq m$ such that $a_j = b_{i_j}$ with $j \leq n$. It is also said that b is a *super-sequence* of a, or b contains a. From the example in Table 1, we have $3 \sqsubseteq_s 4$.

A sequence $p = \langle p_1, p_2 \ldots p_m \rangle$, is a *prefix* of a sequence $s = \langle s_1, s_2, \ldots s_n \rangle$ if $m < n$ and $p_i = s_i$ for any $i \leq m$. For example, $\langle A, G \rangle$ is a prefix of sequence 1. A *k-subsequence* is a subsequence of length k. For a subset $A \subseteq G$ of sequences, s is a *common subsequence* of A if, $\forall x \in A$, we have $s \sqsubseteq_s x$.

The *support* of a subsequence s is $support(s) = |\{x \in G \ : \ s \sqsubseteq_s x\}|$. A *closed subsequence* c is a subsequence that has no super-sequences with the same support, i.e if $\exists c' \in \Sigma^*$ such that $support(c) = support(c')$ then $c' \sqsubset_s c$ or $c' = c$.

1	$\langle A, G, T \rangle$
2	$\langle G, C, G \rangle$
3	$\langle T, C, A \rangle$
4	$\langle T, G, C, A \rangle$

Table 1. DNA examples of sequences

3 NextPriorityConcept for sequences

NextPriorityConcept is a generic algorithm that considers any characteristic as input, that must be supplied with a description δ and a strategy σ. In order to mine sequences with this algorithm, we have to define descriptions and strategies for a set G of input sequences defined on an alphabet Σ.

A description δ is a mapping $\delta : 2^G \rightarrow 2^P$ which defines a set of predicates $\delta(A)$ describing any subset $A \subseteq G$ of sequences.

A strategy σ is an mapping $\sigma : 2^G \rightarrow 2^P$ which defines a set of selectors $\sigma(A)$ to select strict subset A' of A corresponding to the predecessors of any concept $(A, \delta(A))$.

[2] The symbol $\sqsubseteq_s$ denotes a subsequence, to distinguish with the symbol $\sqsubseteq$ classicaly used for the subsumption

For sequences, predicates are based on subsequences, and are of the form *"a matches x"* meaning $x \sqsubseteq_s a$. For better readability, the sets $\delta(A)$ and $\sigma(A)$ will be treated either as sets of predicates or as sets of subsequences in the rest of this document, they can reciprocally be deduced from each other with a predicate on the form *"matches x"* for each subsequence x. These predicates could be specialized according to the type of subsequences given by the description. For example, for a description that computes prefixes, predicates are *"starts with"*.

3.1 Descriptions for sequences

We define three descriptions for a set A of sequences. The first one generates the classical maximal common subsequences of A. The second one is the restriction to prefix sub-sequences. The third one offers the possibility to limit the size of subsequences according to a given length:

The Maximal Common Subsequences (MCS) description is defined, for a subset $A \subseteq G$ of sequences by:

$$\delta_{mcs}(A) = \{x \in \Sigma^* \ : \ \forall a \in A, x \sqsubseteq_s a \text{ and } \nexists x' \in \delta_{mcs}(A) \ : \ x \sqsubseteq_s x'\} \quad (1)$$

For this description we use the BIDE algorithm [28] to generate all maximal common subsequences, with a time complexity $c_\delta^{mcs} = O(2^{|A|}) \leq O(2^{|G|})$

The Prefix Common Subsequence (PCS) description is defined for a subset $A \subseteq G$ of sequences by:

$$\delta_{pcs}(A) = \{x \in \Sigma^* \ : \ \forall a \in A, x \text{ prefix of } a \text{ and } \nexists x' \in \delta_{pcs}(A) \ : \ x \text{ prefix of x'}\} \quad (2)$$

The complexity of this description is $c_\delta^{pcs} = O(|G|.|\Sigma|.l_{max})$, where l_{max} is the length of the maximal sequence in A.

The K-Common Subsequence (KCS) description is defined for a subset $A \subseteq G$ and a length k by:

$$\delta_{kcs}(A, k) = \{x \in \Sigma^* \ : \ \forall a \in A, x \sqsubseteq_s a \text{ and } |x| = k\} \quad (3)$$

This description has the same time complexity of c_δ^{mcs} because maximal common subsequences have to be computed, thus $c_\delta^{kcs} = O(2^{|A|}) \leq O(2^{|G|})$.

Some examples of descriptions for the sequences in Table 1 are given in Table 2.

To ensure that NextPriorityConcept will generate a concept lattice, these three descriptions must verify $\delta(A) \sqsubseteq \delta(A')$ for $A' \subseteq A$ where $\sqsubseteq$ is the subsumption relation between sets of subsequences defined by:

$$\delta(A) \sqsubseteq \delta(A') \iff \forall x \in \delta(A), \exists x' \in \delta(A') \text{ such that } x \sqsubseteq_s x'$$

Proposition 1 *For $A' \subseteq A \subseteq G$ and a length k, we have the three following properties:*

1. $\delta_{mcs}(A) \sqsubseteq \delta_{mcs}(A')$
2. $\delta_{pcs}(A) \sqsubseteq \delta_{pcs}(A')$
3. $\delta_{kcs}(A, k) \sqsubseteq \delta_{kcs}(A', k)$

A	$\delta_{mcs}(A)$	$\delta_{pcs}(A)$	$\delta_{kcs}(A, k=2)$	$\delta_{kcs}(A, k=3)$
$\{2,4\}$	$\{\langle G, C\rangle\}$	$\{\langle\rangle\}$	$\{\langle G, C\rangle\}$	$\{\}$
$\{3,4\}$	$\{\langle T, C, A\rangle\}$	$\{\langle T\rangle\}$	$\{\langle T, C\rangle, \langle T, A\rangle, \langle C, A\rangle\}$	$\{\langle T, C, A\rangle\}$
$\{1,3,4\}$	$\{\langle A\rangle, \langle T\rangle\}$	$\{\langle\rangle\}$	$\{\}$	$\{\}$

Table 2. Some MCS, PCS and KCS descriptions for sequences in Table 1

Proof: Let A and A' be two subsets of G such that $A' \subseteq A$.

1. Let $c \in \delta_{mcs}(A)$, i.e. c is a maximal subsequence of A. By $A' \subseteq A$ we can deduce that c is also a subsequence of sequences in A', but c is not necessarily a maximal subsequence for A'. If c is a maximal subsequence in A' then $c \in \delta_{mcs}(A')$. Otherwise, there exists $c' \in \delta_{mcs}(A')$ such that c is a subsequence of c'. In these two cases, we can deduce that, $\delta_{mcs}(A) \sqsubseteq \delta_{mcs}(A')$
2. The proof is similar for the prefix common subsequence description δ_{pcs}.
3. For δ_{kcs}, the subsumption relation can be refined by the inclusion relation because it is obvious that k-subsequences of a set A of sequences also are k-subsequences of any subset A' of A. Then $\delta_{kcs}(A) \subseteq \delta_{kcs}(A')$ and therefore $\delta_{kcs}(A) \sqsubseteq \delta_{kcs}(A')$.

□

3.2 Strategies and selectors for sequences

NextPriorityConcept refines each concept $(A, \delta(A))$ into predecessors with fewer sequences and more specific descriptions using an input strategy that gives a means of reducing the set of objects from the concept to its predecessors. In this section, we introduce two strategies for sequences. The first one considers all the possible subsets of A, whereas the second one consists in adding one element of the alphabet to the current common subsequences of the description.

The Simple strategy σ_{simple} is defined for a subset A of sequences and a description δ by:

$$\sigma_{Simple}(A, \delta) = \{x \ : \ x \in \delta(A'), A' = A \backslash \{a\}, \ \text{ for all } a \in A\} \tag{4}$$

The cost for this strategy depends on the cost of the description $\delta : c_{\sigma}^{Simple} = O(c_{\delta}.|G|)$.

The Augmented Alphabet Subsequence (AAS) strategy σ_{aas} is defined for a subset A and a descirption δ of sequences by:

$$\sigma_{aas}(A, \delta) = \{x + a \ : \ x \in \delta(A), \ a \in \Sigma\} \tag{5}$$

Note that the $x + a$ notation means the concatenation of the item a to the subsequence x. The cost for this strategy is $c_{\sigma}^{aas} = O(|\delta(A)|.|\Sigma|)$

A	$\sigma_{AAS}(A, \delta_{mcs})$	$\sigma_{Simple}(A, \delta_{kcs}(k=2))$
$\{2,4\}$	$\delta_{mcs}(A) = \{\langle G, C\rangle\}$	$\delta_{kcs}(A, k=2) = \{\langle G, C\rangle\}$
	$\{\langle G, C, \mathbf{A}\rangle, \langle G, C, \mathbf{C}\rangle,$ $\langle \mathbf{G}, \mathbf{C}, \mathbf{G}\rangle \mathbf{\$8}, \langle G, C, \mathbf{T}\rangle\}$	$\{\langle \mathbf{G}, \mathbf{G}\rangle \mathbf{\$5}, \langle \mathbf{C}, \mathbf{G}\rangle \mathbf{\$5}, \langle \mathbf{T}, \mathbf{G}\rangle \mathbf{\$4},$ $\langle T, C\rangle, \langle T, A\rangle, \langle C, A\rangle, \langle \mathbf{G}, \mathbf{A}\rangle \mathbf{\$4}\}$

Table 3. Some examples of Simple and AAS strategies for descriptions in Table 2

Table 3 gives some generated predicates for $A = \{2, 4\}$ (concept 5 in Figure 1, concept 2 in Figure 3) and with these two strategies. For our example in Table 2, those which generated predecessors are mentioned by adding the $ symbol with the concept number, from Figures 1 and 3.

Figure 2 displays the concept lattice generated with the AAS strategy and the PCS description. Figure 1 displays the concept lattice generated with the Simple strategy and the MCS description. Figures 3 and 4 display the two concept lattices generated with the simple strategy and the KCS description with resp. $k = 2$ and $k = 3$.

The concept $1 in Figure 2 shows that sequences 3 and 4 both start with the prefix T. We can observe that we obtain fewer concepts with $k = 3$ than with $k = 2$. In Figure 3, concept $2 contains the subsequence $\langle G,\ C\rangle$ which is the only 2-common subsequence of sequences 2 and 4.

Clearly the predicates in the concept lattice in Figure 1 corresponds to all the closed subsequences since the MCS description generates maximal subsequences, and the simple strategy considers all the possible subsets A of sequences. We can observe that the other lattices contain fewer concepts.

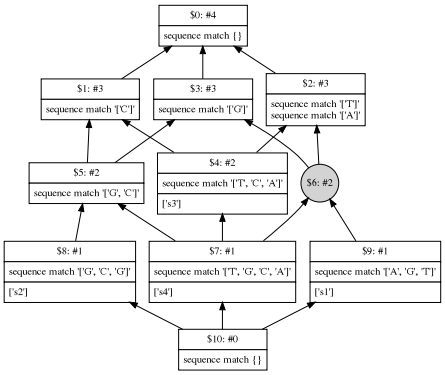


Fig. 1. Hasse diagram of the reduced concept lattice for the Simple strategy and the MCS description

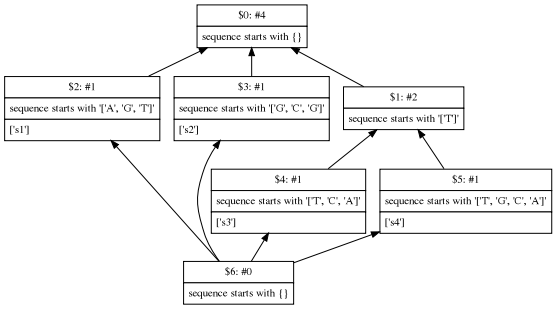


Fig. 2. Hasse diagram of the reduced concept lattice for the AAS strategy and the PCS description

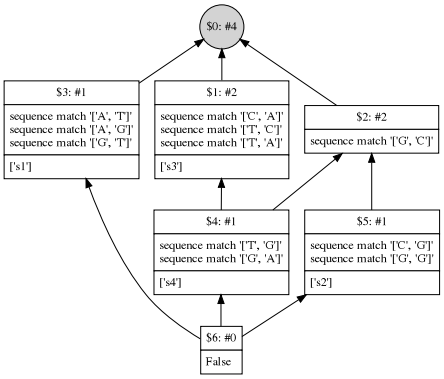


Fig. 3. Hasse diagram of the reduced concept lattice for the Simple strategy and the KCS description with $k = 2$

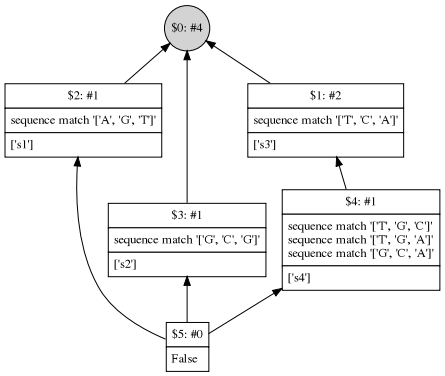


Fig. 4. Hasse diagram of the reduced concept lattice using the Simple strategy and the KCS description with $k = 3$

4 Experiments

To experimentally evaluate the effectiveness of our descriptions and strategies, we use `Galactic`[3] (**GA**lois **LA**ttices, **C**oncept **T**heory, **I**mplicational systems and **C**losures), a development platform of the NextPriorityConcept algorithm which, mixed with a system of plugins, makes it possible easy integration of new kinds of data (description and strategies). We have implemented new plugins for the sequences: `MCS Match`, `PCS Match` and `KCS Match` as description plugins, and `Simple Match` and `AAS Match` as strategies plugins. We consider the following synthetic and real datasets that are summarized in Table 4:

3 groups of synthetic datasets. They are generated by the IBM Quest Dataset Generator software hosted in the SPMF Website[4]. These datasets are generated by varying three parameters; the dataset size $|G|$, the alphabet size $|A|$, and the sequences size $|S|$. Generated datasets are named as follows, G100A30S7 for the dataset where $|G| = 100$, $|A| = 30$ and $|S| = 7$. In total, 10 datasets were generated : G200A30S7, G100A30S7, G75A30S7, G50A30S7, G25A30S7, G100A20S7, G100A10S7, G100A30S5, G100A30S6, G100A30S8

2 real datasets. The Daily-actions dataset is a small database of sequences that represents daily actions of 25 individuals of L3i laboratory[5], where daily actions can be $[Wakeup, Breakfast, Work, Dinner]$. The Wine city dataset is issued from the museum data "La cité du vin" in Bordeaux, France[6], gathered from the visits on a period of one year (May 2016 to

[3] https://galactic.univ-lr.fr
[4] http://www.philippe-fournier-viger.com/spmf/
[5] https://l3i.univ-larochelle.fr/
[6] https://www.laciteduvin.com/en

May 2017). The museum is a large "open-space", where visitors are free to explore the museum the way they want, without predetermined path. When they arrive at the museum, they receive a small personal device to detect whenever a visitor is close to an animation spot called a *module*. The museum contains 20 modules, so we can say that extracted sequences for this data is quite short. By extracting the sequences of activation of each module, we end up with a precise enough idea of what the visit looked like for each visitor of the museum.

datasets	dataset size	alphabet size	average sequence size
Group 1	$\{25, 50, 75, 100, 200\}$	30	7
Group 2	100	$\{10, 20, 30\}$	7
Group 3	100	30	$\{5, 6, 7, 8\}$
DAILY-ACTIONS	25	12	8
WINE CITY	$\{25, 50, 75, 100, 200\}$	30	

Table 4. Datasets

Closed subsequences with the MCS description

Table 5 shows the number of predicates and concepts generated by $\delta_{mcs}/\sigma_{simple}$ and $\delta_{mcs}/\sigma_{aas}$ for real and synthetic datasets, and the number of closed subsequences generated by Clospan[30].

Using $\delta_{mcs}/\sigma_{simple}$, we clearly generate all the closed subsequences since the MCS description generate closed subsequences, and the simple strategy considers all the possible subsets of sequences. We can indeed observe that the number of predicates is the same as the number of closed subsequences with Clospan. The number of concepts is greater than the number of predicates/closed subsequences because many concepts are generated from their successors without new predicates.

We can also observe that the AAS strategy generates less closed subsequences since only the right augmented subsequences are considered. But this difference is only observed with more than 100 sequences in G and more than 20 elements in A, meaning that the AAS strategy is close to the simple strategy, but with a better time complexity. This illustrates the possibility of reducing the closed subsequences generated according to a specific strategy.

KCS description and the DAILY-ACTIONS dataset

Figure 5 and Table 6 show a comparison between MCS and KCS descriptions for synthetic and the DAILY-ACTIONS datasets.

Clearly, the KCS description does not generate closed subsequences because we set the size of the subsequences that are not maximal. Therefore we obtain

Table 5. Synthetic Datasets

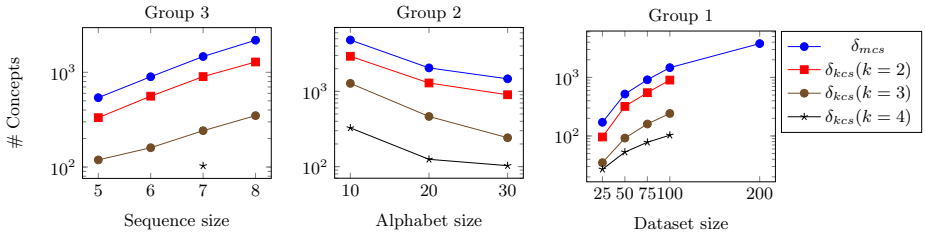
		A30S7					G100S7		G100A30		
		G25	G50	G75	G100	G200	A10	A20	S5	S6	S8
CloSpan	# closed subsequences	**132**	**331**	**541**	**801**	**1647**	**1094**	**880**	**360**	**552**	**1031**
$\sigma_{Simple}(\delta_{mcs})$	# predicates	**133**	**332**	**542**	**802**	**1648**	**1095**	**881**	**361**	**553**	**1032**
	# concepts	171	520	911	1468	3780	4810	2049	540	898	2182
$\sigma_{AAS}(\delta_{mcs})$	# predicates	**132**	**332**	**541**	785	1639	1075	872	**360**	**552**	**1031**
	# concepts	170	520	910	1458	3771	4788	2037	539	897	2181

more predicates, but fewer concepts. Sequences are likely to share more subsequences of small length, but when we increase the length, concepts sharing the same subsequences became fewer. We can also observe that we get fewer concepts as the k parameter increases. The KCS description is clearly adapted to the DAILY-ACTIONS dataset because small subsequences make more sense to analyze daily actions than the maximal / closed subsequences. The possibility to fix k allows to avoid the generation of all possible subsequences.

Table 6. # concepts and predicates for KCS and MCS descriptions for the DAILY-ACTIONS dataset with different value for the k parameter for KCS

	σ_{Simple}	δ_{mcs}	$\delta_{kcs}(k=2)$	$\delta_{kcs}(k=3)$	$\delta_{kcs}(k=4)$
DAILY-ACTIONS	# predicates	272	107	503	1406
	# concepts	373	275	255	197

Fig. 5. # Concepts for synthetic datasets using the Simple strategy and descriptions MCS and KCS(2,3,4)



PCS description and the WINE CITY dataset

Table 7 compared the MCS and the KCS descriptions for the WINE CITY dataset. For this special dataset, we found it more accurate to propose a description by

prefixes since we were focused on the flow of visitors at the entrance of the museum, that is which paths were visited first. Here we can observe that we obtain fewer concepts and predicates with the PCS description compared to the MCS description that generates a very large number of closed subsequences.

Table 7. # concepts and predicates for PCS and MCS descriptions for the WINE CITY dataset

Datasets		WINE CITY			
		25	50	75	100
$\sigma_{AAS}(\delta_{mcs})$	# concepts	576	2172	3580	8233
	# predicates	575	2171	3579	8232
$\sigma_{AAS}(\delta_{pcs})$	# concepts	33	74	114	146
	# predicates	32	75	113	145

5 Conclusion

In this paper, we present a sequence mining approach using the NEXTPRIORITY-CONCEPT algorithm that generated pattern concepts in a user-driven patterns mining approach for heterogeneous and complex data as input. This algorithm allows a generic patterns computation through specific *descriptions* and *strategies* by predicates.

We present three descriptions and two strategies dedicated to the analysis of sequential data. As descriptions we propose the classical maximal common subsequences (MCS), the maximal prefix (PCS), or the k-subsequences (KCS). Therefore a set of sequences can be mined by one of these three descriptions in a generic way. We also propose two strategies of pattern exploration. The first strategy (simple) considers all the possible subsets of sequences as in classical approaches, while the second one (AAS) proposes a refinement of a set of sequences by adding only one item to their common description.

Sequence analysis of real world data may require a specific way of describing the sequences, and a specific strategy of exploration of the predecessors, and the NEXTPRIORITYCONCEPT algorithm could be very useful since it allows several descriptions and strategies in a generic way. For example the prefix match description was triggered by the need to know which spaces are visited first in a museum, which common paths are chosen by visitors, or which spaces may be forgotten within the visit.

For future work, we are interested in finding new ways of describing sequences to make it more specific to the application, one of which is the use of distances between elements in the sequence, or specify a given *window* parameter to deal with long sequences.

References

1. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Data Engineering, 1995. Proceedings of the Eleventh International Conference on. pp. 3–14. IEEE (1995)
2. Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 429–435. ACM (2002)
3. Barbut, M., Monjardet, B.: Ordres et classifications : Algèbre et combinatoire. Hachette, Paris (1970), 2 tomes
4. Bertet, K., Demko, Ch., Viaud, J.F., Guérin, C.: Lattices, closure systems and implication bases: A survey of structural aspects and algorithms. Theoretical Computer Science **743**, 93–109 (2018)
5. Birkhoff, G.: Lattice theory, vol. 25. American Mathematical Soc. (1940)
6. Bordat, J.P.: Calcul pratique du treillis de Galois d'une correspondance. Mathématiques et Sciences humaines **96**, 31–47 (1986)
7. Buzmakov, A., Egho, E., Jay, N., Kuznetsov, S.O., Napoli, A., Raïssi, C.: Fca and pattern structures for mining care trajectories (2013)
8. Codocedo, V., Bosc, G., Kaytoue, M., Boulicaut, J.F., Napoli, A.: A proposition for sequence mining using pattern structures. In: International Conference on Formal Concept Analysis. pp. 106–121. Springer (2017)
9. Demko, C., Bertet, K., Faucher, C., Viaud, J.F., Kuznetsov, S.: Next priority concept: A new and generic algorithm computing concepts from complex and heterogeneous data. arXiv preprint arXiv:1912.11038 (2019)
10. Ferré, S.: The efficient computation of complete and concise substring scales with suffix trees. In: International Conference on Formal Concept Analysis. pp. 98–113. Springer (2007)
11. Ferré, S.: Reconciling Expressivity and Usability in Information Access - From Filesystems to the Semantic Web. Habilitation, University of Rennes 1, France (november 2014)
12. Ferré, S.: Reconciling expressivity and usability in information access from file systems to the semantic web (2014)
13. Ferré, S.: Systèmes d'information logiques : un paradigme logico-contextuel pour interroger, naviguer et apprendre. Doctorat, University of Rennes 1, France (Oct 2002)
14. Ganter, B., Kuznetsov, S.O.: Pattern structures and their projections. In: LNCS of International Conference on Conceptual Structures (ICCS'01). pp. 129–142 (2001)
15. Ganter, B., Wille, R.: Formal Concept Analysis, Mathematical foundations. Springer Verlag, Berlin (1999)
16. Gizdatullin, D., Baixeries, J., Ignatov, D.I., Mitrofanova, E., Muratova, A., Espy, T.H.: Learning interpretable prefix-based patterns from demographic sequences. In: International Conference on Intelligent Data Processing: Theory and Applications. pp. 74–91. Springer (2016)
17. Gizdatullin, D., Ignatov, D., Mitrofanova, E., Muratova, A.: Classification of demographic sequences based on pattern structures and emerging patterns. In: Supplementary Proceedings of 14th International Conference on Formal Concept Analysis, ICFCA. pp. 49–66 (2017)
18. Gomariz, A., Campos, M., Marin, R., Goethals, B.: Clasp: An efficient algorithm for mining frequent closed sequences. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. pp. 50–61. Springer (2013)

19. Han, J., Pei, J., Mortazavi-Asl, B., Chen, Q., Dayal, U., Hsu, M.C.: Freespan: frequent pattern-projected sequential pattern mining. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 355–359. ACM (2000)
20. Kaytoue, M.: Contributions to Pattern Discovery. Habilitation, University of Lyon, France (february 2020)
21. Kaytoue, M., Codocedo, V., Buzmakov, A., Baixeries, J., Kuznetsov, S.O., Napoli, A.: Pattern structures and concept lattices for data mining and knowledge processing. In: In Proceedings of ECML-PKDDl (2015)
22. Linding, C.: Fast concept analysis. In: Working with Conceptual Structures-Contributions to ICC. pp. 235–248 (2002)
23. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data mining and knowledge discovery **1**(3), 259–289 (1997)
24. Masseglia, F., Cathala, F., Poncelet, P.: The psp approach for mining sequential patterns. In: European Symposium on Principles of Data Mining and Knowledge Discovery. pp. 176–184. Springer (1998)
25. Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: icccn. p. 0215. IEEE (2001)
26. Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: Proceedings of the eleventh international conference on Information and knowledge management. pp. 18–25 (2002)
27. Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements, edbt (1996)
28. Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: Proceedings. 20th international conference on data engineering. pp. 79–90. IEEE (2004)
29. Wille, R.: Restructuring lattice theory : an approach based on hierarchies of concepts. Ordered sets pp. 445–470 (1982), i. Rival (ed.), Dordrecht-Boston, Reidel.
30. Yan, X., Han, J., Afshar, R.: Clospan: Mining: Closed sequential patterns in large datasets. In: Proceedings of the 2003 SIAM international conference on data mining. pp. 166–177. SIAM (2003)
31. Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine learning **42**(1-2), 31–60 (2001)