Distant Event Prediction Based on Sequential Rules

Lina Fahed, Philippe Lenca, Yannis Haralambous, Riwal Lefort

Data Science and Pattern Recognition
Ubiquitous International
 ISSN 2520-4165
Volume **4**, Number **1**, April **2020**

Distant Event Prediction Based on Sequential Rules

Lina Fahed

ISEN Brest, L@bISEN, Yncrea Ouest, 20 Rue Cuirassé Bretagne, 29200 Brest, France
lina.fahed@isen-ouest.yncrea.fr,linafahed@gmail.com

Philippe Lenca, Yannis Haralambous

IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France
philippe.lenca@imt-atlantique.fr,yannis.haralambous@imt-atlantique.fr

Riwal Lefort

Crédit Mutuel ARKEA, Innovation and Operation Pole, Datalabs service, Brest, France
riwal.lefort@arkea.com

Abstract. Event prediction in sequence databases is an important and challenging data mining task. We focus on the specific case of prediction of *distant* events. Our aim is to mine sequential association rules with consequents that are temporally distant from their antecedents. We therefore propose two new algorithms: *D-SR-postMining* and *D-SR-inMining* (*D-SR* stands for Distant Sequential Rules). The originality of these algorithms is that they integrate a minimal gap constraint between the antecedent and the consequent of existing rules, which, as we prove, has an anti-monotonicity property. This approach allows to predict events with enough time in advance (at least as much as the gap). Both algorithms are designed to coexist with legacy rule mining algorithms: *D-SR-postMining* can be used as a post-processing step of traditional mining algorithms, and *D-SR-inMining* can be integrated into the mining process of such algorithms. Experiments on three data sets show that both algorithms are efficient for mining distant rules and scalable on large data sets. Even better, *D-SR-inMining* reduces execution time significantly (up to 9 times). Furthermore, an in-depth analysis of the rules mined from a real-world bank data set, demonstrates the efficiency of such rules for real-world applications such as churn analysis.

Keywords: Data mining, sequence mining, association rule mining, distant prediction, anti-monotonicity

1. **Introduction.** Sequential patterns and sequential rules in sequence databases have been introduced in [5] and are still an important and active research field in data mining (see for example the surveys [20, 27]). Sequences can be represented as chronologically or positionally ordered lists (in the latter case we keep only the order of events and not their timestamps) of events [39]. An *event* is usually represented by an event type (among a finite set of types) and a list of timestamps of its occurrences in the sequence database. Mining sequence databases generally comes down to mining sub-sequences, called *sequential patterns*, that are common to multiple sequences. Besides sequential pattern mining, it is also possible to mine *sequential association rules* (also called sequential rules), denoted as $R : P \rightarrow Q$ where P is the antecedent of R and Q its consequent. The existence of such a rule implies that if some events (P) occur in a specific order, some other event(s) (Q) is/are likely to follow with a given probability. Note that Q is often a singleton event. Therefore, sequential rules are generally used to predict the occurrence of events which happen to be their consequents [1, 43, 50]. Predicting future events in sequence databases is an important issue with wide applications such as stock market analysis, consumer

product recommendation, weather forecasting, e-learning, text analysis and web link recommendation [16, 40, 47, 57, 67]. For example, the analysis of users' browsing histories can be used to learn their behavior patterns in order to predict the next pages that will be visited.

In the following, we introduce the challenge of mining sequential rules with a distant consequent, based on the association rules approach. We also illustrate this challenge through churn analysis in the banking and marketing domain.

The challenge of mining sequential rules with a distant consequent. The sequential rule mining task is usually [5] decomposed into two sub-tasks: (i) the mining of frequent sequential patterns that respect a predefined support threshold and, out of these, (ii) the construction of rules that respect a predefined confidence threshold. Rules are generated by considering the last event in the sequential pattern as the consequent of the rule, and the rest of events as its antecedent. The second sub-task is quite straightforward, and thus most of the research focuses on the first one: mining frequent sequential patterns.

As mentioned above, the sequential rule $R : P \rightarrow Q$ can be used to predict Q when all (ordered) events in P have been observed. Notice that no information is given about the appearance horizon of Q (i.e., the time lapse after which the event will occur) following the appearance of the last event of P.

However, in some applications, one needs to have time to react before the occurrence of the predicted event, so it is preferable to know the temporal distance of the predicted event in advance. In other words, one needs to know not only *what* will occur, but also *when* it will occur. This is a way towards more *actionable rules* [11, 30, 53]. As an example, let us consider an online shopping framework. The flow of purchases of each client can be viewed as a single sequence of events and sequential rules can be mined from the sequence database and used to predict what will be purchased next. Store managers are interested in predicting purchases that will occur after, e.g., a month, in order to manage their supplies. In a different domain, related to banking, it may be useful to predict whether clients with a specific profile may tend to leave their bank. If this prediction can be made early enough, the bank may potentially have sufficient time to react in order to prevent the client from leaving and thus reduce churn.

We therefore focus on the mining of sequential rules $R : P \rightarrow Q$ that can be used for predicting temporally distant events. To achieve this, we model the temporal dependency between an antecedent P and its consequent Q. In relation with the churn application we mentioned, we are interested in mining rules in which Q will occur with a minimal temporal distance "*gap*" from P. We refer to these rules as *distant sequential rules*.

Several algorithms that deal with temporal constraints have been proposed and some of them will be described in Section 2. However, to the best of our knowledge, the challenge of mining distant sequential rules has not yet been carried out in the context of sequence database.

The idea of mining rules with a distant consequent has been first introduced in the context of mining *episode rules* ("minimal" episode rules) in a single long sequence of events [14]. We here extend this approach to multiple sequences and towards a more generalizable algorithm which can be integrated into any existing sequential rule mining algorithm.

In this paper, we propose two new algorithms: *D-SR-postMining* and *D-SR-inMining* (*D-SR* stands for Distant Sequential Rules). The originality of our both *D-SR* algorithms lies in the identification of the consequent of the rule, which guarantees that the consequent is temporally distant from the antecedent. This also allows to evaluate the confidence of rules, to prune the search space, and thus to stop the mining process of the current rule

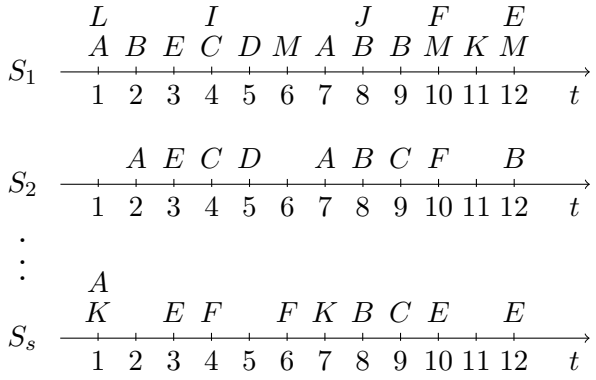


FIGURE 1. Example of sequence database.

at any time. Furthermore, due to identification of the consequent via the gap constraint, many candidate occurrences of the rules that do not respect this constraint are filtered out. Consequently, the running time of the algorithm is significantly reduced. Besides, we choose to mine rules with a consequent that contains only one event, which is common in the rule mining task [4]. As we will see, *D-SR* can be easily extended to mine rules with a consequent made up of several events. In addition, an additional originality of our algorithms is that, as they coexist with traditional algorithms, they make any mined rule more actionable and more intelligible to domain experts. Actually, while monitoring the event sequence during a prediction task, once the antecedent of a mined distant rule appears, the prediction of a distant event (the consequent) is triggered and estimated within at least gap timestamps. This allows domain experts to have potentially enough time to react depending on the nature of the future event.

The remainder of this paper is organized as follows: Section 2 presents related work about sequential patterns, sequential rule mining and relevant concepts. Algorithms *D-SR-postMining* and *D-SR-inMining* are introduced in Section 3 and experimental results are presented in Section 4. We conclude and provide some perspectives in Section 5.

2. **Related Work.**

In this section, we define some concepts and present state-of-the-art work related to temporal constraints, sequential pattern and sequential rule mining. Several definitions are inspired from state-of-the-art papers such as [14, 26, 43, 58].

2.1. **Definitions.**

As in [26, 43, 58, 66], an *event sequence* S is a list of ordered events and is defined formally as follows:

Definition 2.1. *Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of timestamped items, and $I_t \subseteq I$ the set of items that occur at timestamp t (we call I_t an **event**). An **event sequence** S_i (denoted sometimes as S for simplicity) is an ordered list of timestamped events: $S_i = <I_{t_1}, I_{t_2}, \ldots, I_{t_n}>$ with $t_1 < t_2 < \cdots < t_n$. The length of this sequence is $|S| = n$.*

A **sequence database**, denoted by $\mathscr{S}$, is a set of event sequences: $\mathscr{S} = \{S_1, S_2, \ldots, S_s\}$. Figure 1 represents a sequence database example, which we will use to illustrate various concepts (letters $A, B, C, \ldots$ represent event occurrences, when placed vertically they co-occur). Indeed, in Figure 1, S_1 is an event sequence consisting of the following events: $<\{A, L\}, B, E \ldots, \{E, M\}>$. The event $\{A, L\}$ occurs at timestamp t_1 and consists of two items: A and L (the notation $\{\ldots\}$ expresses the fact that these items occur with the same timestamp and hence are not ordered).

Definition 2.2. *A contiguous segment W of a sequence S is called a (sliding[1]) **window** of S.*

The length w of W, which is in fact a sub-sequence of S, is less than or equal to the length of S: $w \leq |S|$.

Several state-of-the-art algorithms mine rules occurring within such a sliding window [23]. The temporal constraint induced by the window is important for complexity issues but also for many real applications as users often only wish to discover patterns occurring within a maximum time interval [21]. In this paper, the window constraint is important as we will increase the window size in order to perform farther predictions.

Definition 2.3. *A **sequential pattern** $P = <p_1, p_2, \ldots, p_k>$, is a chronologically or positionally ordered list of events p_i ($p_i \subseteq I$, for $i = 1, \ldots, k$).*

*A pattern P is a **sub-pattern** of a pattern $Q = <q_1, q_2, \ldots, q_s>$ with $s \geq k$ if for each i, $\{p_i\} \subseteq \{q_i\}$ (we allow $\{p_i\}$ to be empty). In this case, Q is called a **super-pattern** of P.*

Notice that events in a pattern do not have unique, specific timestamps. When we instantiate a pattern by providing timestamps to its events we obtain an occurrence (see Definition 2.4).

From now on we will focus on *sequential patterns* and may call them *patterns* for the sake of simplicity. Let us take an example in Figure 1: The pattern $P = <\{I, C\}>$ is a sub-pattern of $Q = <A, \{I, C\}>$; on the other hand, the pattern $P' = <\{L, A\}>$ is not a sub-pattern of Q.

Definition 2.4. *A pattern P can occur in more than one sequences in the sequence database. The **occurrence** of P in a sequence S_i, denoted by $\mathrm{occ}_i(P)$, is the list of timestamps of the events of S_i that compose the pattern P. In most state-of-the-art papers, the occurrence of a pattern is often represented solely by the timestamp of the first event (prefix) and the timestamp of the last event (suffix) of the pattern. The list of all occurrences of a pattern P will be referred to as (capitalized) $\mathrm{Occ}(P)$.*

For example, the pattern $<A, E>$ in Figure 1 occurs in sequences S_1, S_2 and S_s, where $\mathrm{occ}_1(<A, E>) = (1, 3)$, $\mathrm{occ}_2(<A, E>) = (2, 3)$, $\mathrm{occ}_s(<A, E>) = (1, 3)$. The list of all occurrences of this pattern is $\mathrm{Occ}(<A, E>) = [(1, 3), (2, 3), (1, 3)]$.

Definition 2.5. *The **support** of a pattern P, denoted by $\mathrm{supp}(P)$, represents the number of its occurrences in the sequence database: $\mathrm{supp}(P) := |\mathrm{Occ}(P)|$. Usually a pre-defined minimal threshold $\mathrm{minsupp}$ is fixed, in order to mine only **frequent patterns**: P is said to be a frequent pattern if $\mathrm{supp}(P) \geq \mathrm{minsupp}$.*

Let us take an example in Figure 1. The support of the pattern $<A, E>$ is calculated as follows: $\mathrm{supp}(<A, E>) = |\mathrm{Occ}(<A, E>)| = |[(1, 3), (2, 3), (1, 3)]| = 3$. If, for example, we set $\mathrm{minsupp} = 2$, the pattern $<A, E>$ is a frequent pattern since $\mathrm{supp}(<A, E>) \geq \mathrm{minsupp}$.

In state-of-the-art approaches and depending on research or applications goals, the number of occurrences of a pattern in a given sequence may be counted in different ways using various predefined anti-monotone frequency measures [4, 41]. In some approaches one considers the exact number of occurrences while in others one counts only a single occurrence even if the pattern appears several times. Moreover, several frequency measures are proposed, such as non-overlapped frequency [38], minimal occurrence-based frequency, windows-based frequency [43], etc. As the algorithms we propose in this paper are based on an anti-monotone measure (namely the minimal occurrence-based frequency measure), they can be integrated into any traditional algorithm [41].

[1] A sliding window is the most common type of windows.

Definition 2.6. *The **concatenation** of two sequential patterns* $P = <p_1, \ldots, p_k>$ *and* $Q = <q_1, \ldots, q_\ell>$*, where* $t_{p_k} < t_{q_1}$ *in all occurrences of these patterns, is the pattern* $P \oplus Q := <p_1, \ldots, p_k, q_1, \ldots, q_\ell>$*, of length* $k + \ell$*.*

For example, let us consider from Figure 1 the two patterns $<A, E>$ and $<F>$. The concatenation of both patterns forms a new pattern as follows: $<A, E> \oplus <F> = <A, E, F>$. Notice that, this new pattern is a frequent pattern since $\text{supp}(<A, E, F>) = |\text{Occ}(<A, E, F>)| = |[(1, 3, 10), (2, 3, 10), (1, 3, 4)]| = 3 \geq \text{minsupp} = 2$.

Definition 2.7. *Let* P*,* Q *be two sequential patterns. A **sequential association rule*** $R : P \rightarrow Q$ *is an association rule such that the consequent* Q *tends to occur after the antecedent* P*. This tendency is often represented by the probability that* Q *occurs after* P *knowing that* P *occurs. This conditional probability is the **confidence** of the rule* R*. It is calculated as the support of the pattern* $P \oplus Q$*, divided by the support of the pattern* P*:*

$$\text{conf}(P \rightarrow Q) := \frac{\text{supp}(P \oplus Q)}{\text{supp}(P)} \tag{1}$$

The rule $P \rightarrow Q$ *is said to be **confident** if*

$$\text{conf}(P \rightarrow Q) \geq \text{minconf},$$

where minconf *is a predefined confidence threshold.*

Problem formulation. One of our objectives is to extract sequential rules $P \rightarrow Q$ in sequence databases that are both frequent and confident as defined in [56]:

- $\text{supp}(P \oplus Q) \geq \text{minsupp}$
- $\text{conf}(P \rightarrow Q) \geq \text{minconf}$

Let us, once again, take an example from Figure 1. The rule $A, E \rightarrow F$ consists of two patterns: $<A, E>$ and $<F>$. We have $\text{supp}(A, E \rightarrow F) = \text{supp}(A, E, F) = 3$ (it is a frequent rule for minsupp = 2) and $\text{conf}(A, E \rightarrow F) = \text{supp}(A, E, F)/\text{supp}(A, E) = 3/3 = 1$. When, for example, minconf = 0.5, this rule is considered as confident.

2.2. **Sequential Pattern Mining.** Numerous algorithms have been designed to discover sequential patterns in sequence databases. Among the most popular ones are GSP [56], Spade [63], PrefixSpan [46], CM-Spam [15], and CM-Spade [15]. These sequential pattern mining algorithms all take a sequence database and a minimum support threshold (fixed by the user) as input, and provide the set of frequent sequential patterns as output. It is important to mention that—for general algorithms without constraints—there is always only a single correct answer (for a given sequence database and a given support threshold value). Hence, algorithms differ in their strategies of searching the sequential patterns (depth-first search, breadth-first search, etc.) and thus in their performance in terms of time and memory consumption.

2.3. **Mining Patterns under Constraints.** Since the search space of all possible subsequences in a database can be very large, sequential pattern mining is computationally expensive in terms of execution time and memory [22, 56]. This is even more the case for dense databases, or databases with a large number or long sequences.

Proposing an efficient algorithm for sequential pattern mining requires thus the integration of pruning techniques aiming to restrict exploration of the entire search space. The basic techniques for pruning the search space is based on the anti-monotonicity property of the frequency measure (also called Apriori property or downward-closure property) as it is proposed mainly for association rules [4].

However, using only anti-monotone frequency measures may not be enough in order to prune the search space and to produce the required rules. For this reason, other constraints can be added to progressively reduce execution time and memory consumption [18, 20, 33].

For example, many studies have focused on the mining of *concise patterns* (patterns that respect predefined constraints). Among the traditional concise patterns, we can cite high utility patterns [25, 61], popular patterns [42], maximal patterns [32, 59, 62] (i.e., frequent patterns that have no frequent super-patterns), closed patterns [7, 64, 65] (i.e., frequent patterns that have no super-patterns with the same support value), and periodic patterns [6, 8, 19, 35, 36, 54, 55] (where the periodicity of a pattern is measured on the basis of the temporal distance between consecutive occurrences of it). However, these types of constraints are applied between the different occurrences of a pattern and not within a single occurrence of a pattern, which is part of our objectives.

Last but not least there is the approach of considering maximal or minimal temporal distances between occurrences of two consecutive events in a pattern, referred to as a *gap* [2, 45, 48]—this approach is the closest to our approach. Note that the *maximal gap* is frequently used as the upper bound of the temporal distance between two events within the pattern [45], while, in our case, we are interested in a *minimal gap*.

2.4. **Sequential Rule Mining.** Recently, efficient sequential rule mining algorithms have been proposed, such as RuleGrowth [23] and ERMiner [17]. They adopt a pattern-growth approach (resp. a vertical representation of the database) for discovering rules. It should be noticed that these algorithms do *not* rely on a pattern mining step, probably because a pattern mining step is very time-consuming.

A fixed gap constraint between the antecedent and the consequent of sequential rules has been proposed for mining *precise-positioning*—as called by the authors—association rules [9]. Using a strict gap may be a requirement in time-sensitive applications, such as security trading. However, a strict gap is not suitable for our churn application.

2.5. **Event Prediction.** Based on sequential rules in a sequence database, event prediction algorithms aim at predicting the next event [50]. Several classifiers with high precision for early prediction have been proposed [13, 44, 51, 52], they use as little information as possible to make a prediction. However, theses techniques do not allow modeling the occurrence horizon of the future event and often do not offer a satisfactory *explanation* of the prediction.

Deep learning techniques for modeling sequence databases, such as RNNs, have been proposed in the literature [10, 34]. Such models are known to be efficient for future event prediction. However, recently it has become clear that such predictive models are only applicable if they provide some degree of usable intelligence, i.e., being able to extract and to provide explanations intelligible to domain experts [12, 28, 60]. Numerous explanation approaches have been proposed and applied on images and text data [37, 49]. However, to the best of our knowledge, there are no approaches built specifically for temporal sequential data. Consequently, despite their high performance level, deep learning methods are not suitable for our objective of event prediction in temporal sequence databases as no explanation can be provided. On the other hand, rule-based predictive models are known to be inherently explainable to domain experts [3, 29], which is essential to our application.

Conclusion. Fighting against churn, as discussed with our bank partner, requires to be able to make early predictions. Also, in order to be able to take decisions and to provide a satisfactory explanation of churn behavior, it has been decided to rely on sequential rules-based approaches that are suitable for these objectives. As shown in the state-of-the-art

section above, no existing approaches fully satisfy these goals. Consequently, we propose to extend sequential rule mining with a minimal gap constraint between the antecedents and the consequents of rules in order to be able to perform the challenging task of early predictions in sequence databases. In the next section we describe our proposal.

3. Mining Sequential Association Rules for Distant Events Prediction.

In the following, we present the algorithms *D-SR-postMining* and *D-SR-inMining* for mining distant sequential rules. The main contribution of these algorithms is to integrate a minimal gap constraint within sequential rule-mining algorithms. This minimal gap, located between the antecedent and the consequent of the rule, guarantees the fact that the consequent is distant from the antecedent. As a consequence, the resulting rules can be used for distant event prediction (the consequent of the rule), and this represents an early prediction.

The gap constraint that we propose is a minimal temporal distance constraint imposed on each occurrence of the rule. This constraint is imposed between the suffix of the antecedent and the prefix of the consequent of the rule. As we do not need to take into account each timestamp, our first proposal consists in modifying the traditional representation of the occurrence of a rule R (see definition 2.4).

Definition 3.1. *Let $R : P \rightarrow Q$ be a rule made of pattern $P \oplus Q = <p_1, \dots p_k, q_1, \dots q_e>$. An **occurrence** of R corresponds to an occurrence of $P \oplus Q$ and is represented by the timestamps of the suffix of P and the prefix of Q as follows:* $\text{occ}(R : P \rightarrow Q) = \text{occ}(P \oplus Q : <p_1, \dots p_k, q_1, \dots q_e>) = (t_{p_k}, t_{q_1})$. *The occurrence list of R is built by performing a temporal join between* $\text{Occ}(p_1, \dots p_k)$ *and* $\text{Occ}(q_1, \dots q_e)$.

As it is customary in the literature [63], a *temporal join* is an operation that allows to join the occurrences of two patterns while respecting the temporal order.

Let us take an example from Figure 1. The occurrence list of the rule $(A, E \rightarrow F)$ is represented as follows: $\text{Occ}(A, E \rightarrow F) = [(3, 10), (3, 10), (3, 4)]$.

Let us now define the notions of minimal gap constraint and of distant sequential rule as follows:

Definition 3.2. *We say that* $\text{occ}(P \oplus Q)$ *satisfies a **minimal gap constraint** of length equal to* gap *and denote it by* $\text{occ}_{\text{gap}}(P \oplus Q)$, *when* $(t_{q_1} - t_{p_k}) \geq \text{gap}$. *The list of all occurrences of $P \oplus Q$ that respect the gap is denoted as* $\text{Occ}_{\text{gap}}(P \oplus Q)$.

Definition 3.3. *The rule $R : P \rightarrow Q$ is a **distant sequential rule** if it is frequent and confident when we consider solely occurrences that respect the gap constraint:*

$R : P \rightarrow Q$ is a distant sequential rule if:

$$\begin{cases} \text{supp}(P \oplus Q) = |\text{Occ}_{\text{gap}}(P \oplus Q)| \geq \text{minsupp}, \\ \text{conf}(P \oplus Q) = \frac{|\text{Occ}_{\text{gap}}(P \oplus Q)|}{\text{supp}(P)} \geq \text{minconf}. \end{cases} \tag{2}$$

For example, let us check whether the occurrences of the rule $\text{Occ}(A, E \rightarrow F)$ satisfy the gap constraint gap = 5. The occurrences $(3, 10)$ (in S_1) and $(3, 10)$ (in S_2) satisfy this gap. However, the occurrence $(3, 4)$ does not satisfy this gap as $(t_{q_1} - t_{p_k}) = 4 - 3 = 1 <$ gap. The support of this rule is $\text{supp}(A, E \rightarrow F) = |(3, 10), (3, 10)| = 2$. Its confidence is $\text{conf}(A, E \rightarrow F) = |(3, 10), (3, 10)|/3 = 0.666\dots$ If we set minsupp = 2 and minconf = 0.5, then the rule is frequent and confident and thus, a distant sequential rule.

We will now prove that the gap constraint as we define is an anti-monotonicity property.

Definition 3.4. *Let there be two rules:*

$$R : P \rightarrow Q = <p_1, \dots p_k, q_1, \dots q_e>$$
$$R' : P' \rightarrow Q = <p_1, \dots, p_k, \dots, p_{k+i}, q_1, \dots q_e> \quad (i \geq 0).$$

We say that R' is a ***super-antecedent rule of*** *R when the antecedent P' is a super pattern of P, i.e., P' is made of P with one or more events added to its right-hand side.*

Lemma 3.1. *Let R be a rule and R' a super-antecedent rule of R. Then if R does not respect the minimal gap constraint, R' also does not respect this constraint.*

Proof. As R does not respect the minimal gap constraint, we have: $(t_{q_1} - t_{p_k}) <$ gap. We know also that P' is a super pattern of P, which means that $(t_{q_1} - t_{p_{k+i}}) \leq (t_{q_1} - t_{p_k})$, and thus obviously $(t_{q_1} - t_{p_{k+i}}) \leq (t_{q_1} - t_{p_k}) <$ gap. As a consequence, according to definition 3.2, R' also does not respect the minimal gap constraint. □

Corollary 3.1. *The minimal gap constraint is an* anti-monotonicity *property.*

We use this property to prune the search space during the mining process and it contributes to the computational efficiency of our algorithms.

As we have mentioned before, state-of-the-art algorithms for mining sequential rules can be divided into two types: (i) algorithms that mine patterns and then form rules, and (ii) algorithms that mine directly rules by fixing the consequent early during the mining process. Addressing these two types, we propose two algorithms *D-SR-postMining* and *D-SR-inMining* as extensions of traditional algorithms. Our algorithms extract distant rules by incorporating into them the minimal gap constraint. Thus, the originality of our algorithms is that they represent a general framework, that we call *D-SR*, for mining sequential rules (for example if the gap is set to 0 then both proposed algorithms behave like classical sequential rule mining algorithms).

We next present the two algorithms for mining distant rules: *D-SR-postMining* in § 3.1 and *D-SR-inMining* in § 3.2.

3.1. *D-SR-postMining*: Distant Rule Mining Algorithm during the Post-processing Step.

Most traditional rule mining algorithms are not directly designed to build distant rules, as they first form patterns and then form rules. This fact has two drawbacks. First, due to complexity issues, the window W (see Definition 2.2) used by these algorithms is generally relatively small, so that the rules have a consequent that is close to the antecedent. Second, when an event is appended to a pattern while it is formed, the mining algorithm is unable to know whether this event will be part of the consequent of the rule or part of its antecedent. In other words, the algorithm lacks the information whether it has to constrain or not the distance of the appended event to the previously formed pattern during the mining process. Consequently, integrating the gap constraint needed for mining distant rules cannot be performed during the mining process. For this reason, we propose to integrate the gap constraint in a post-processing step.

A pseudo-code of the algorithm *D-SR-postMining* is presented below (Algorithm 1). It has two main steps, presented in order of execution:

1. During the mining of a sequential pattern $P \oplus Q : <p_1, \dots p_k, q_1>$ by a traditional mining algorithm (which consists in appending the event q_1 to its right hand side), we keep track of the temporal distance between p_k and q_1 (as in Definition 3.1). This can be performed by saving the occurrences and their timestamps (Algorithm 1, lines 5 and 6).

   At timestamp t_{q_1}, we are not able to know whether the new event q_1 will be the consequent of a frequent and confident rule. However, the temporal distance between the antecedent and the consequent of each occurrence of the rule is already known.

2. (Gap and support pruning.) Once the rule $R : P \rightarrow Q = <p_1, \dots p_k, q_1>$ is constructed, we perform a post-processing step in order to filter out the occurrences of R that do not respect the predefined minimal gap constraint (see Definition 3.2, Lemma 3.1 and Algorithm 1, line 10). The support and confidence values are updated during this step (see Definition 2.5, equation 1, Definition 3.3 and Algorithm 1, line 12). We then keep only frequent and confident rules, all occurrences of which respect the gap constraint $\text{Occ}_{\text{gap}}(R)$. They represent the distant sequential rules we are seeking (see Definition 3.3).

Despite the use of the anti-monotonicity property of the gap constraint, the efficiency of this post-processing approach is nevertheless limited by the efficiency of the sequential rule mining algorithm used.

Algorithm 1: *D-SR-postMining*

input : a sequence database $\mathscr{S}$, values of minsupp, minconf, gap
output: R_{DS}: list of distant sequential rules

```
1  Main Procedure Mining distant sequential rules:
2      Pattern mining then rule construction via a traditional algorithm ;
3      Gap pruning ;
4  Procedure Pattern mining then rule construction via a traditional algorithm:
5      foreach P ⊕ Q ∈ set of frequent patterns do
6          save occ(<p_1, ... p_k, q_1>) = (t_{p_k}, t_{q_1})
7  Procedure Gap pruning
8      foreach P → Q ∈ set of frequent and confident rules do
9          foreach occ(P ⊕ Q) = occ(<p_1, ... p_k, q_1>) = (t_{p_k}, t_{q_1}) do
10             if (t_{q_1} − t_{p_k}) ≥ gap then
11                 add occ(P ⊕ Q) to Occ_gap(P ⊕ Q)
12         if P → Q is frequent and confident based on Occ_gap(P ⊕ Q) then
13             add P → Q to R_DS
```

3.2. ***D-SR-inMining*: Distant Rule Mining Algorithm during the Mining Process.** Recently, traditional algorithms have been proposed [14, 23] that mine directly sequential rules without relying on a pattern mining step. Such algorithms start by joining two events to form a rule where both the antecedent and the consequent are made of a single event: $R : <p_1> \rightarrow <q_1>$. If the support and the confidence of this rule exceed the predefined thresholds, then the rule is extended by adding one event, either to its antecedent or to its consequent, in order to mine longer rules, and so on.

Below, we describe algorithm *D-SR-inMining* that can be integrated into the mining process of such existing algorithms. We consider rules with a single-event consequent as it is often the case in the literature. Our algorithm uses a recursive depth-first search based approach. Its three main steps are detailed below and the pseudo-code is shown in Algorithm 2:

1. (Gap pruning.) At each step of the rule mining process, when considering a candidate rule $R : p_1, \dots p_k, p_{k+1} \rightarrow q_1$ (see Algorithm 2, line 2 for a candidate rule $p_1 \rightarrow q_1$ and line 14 for a longer rule), we construct the list of its occurrences that respect the minimal gap constraints $\text{Occ}_{\text{gap}}(p_1, \dots p_k, p_{k+1}, q_1)$ (see Definition 3.2 and Algorithm 2, lines 3, 15).

$\text{Occ}_{\text{gap}}(p_1, \dots p_k, p_{k+1}, q_1)$ is built by performing a temporal join between $\text{Occ}_{\text{gap}}(p_1, \dots p_k, q_1)$ and $\text{Occ}(p_{k+1})$ (Algorithm 2, line 8). Notice that for a single event, such as p_{k+1}, we have $\text{Occ}_{\text{gap}}(p_{k+1}) = \text{Occ}(p_{k+1})$.

Thanks to the anti-monotonicity property of the minimal gap constraint (see Lemma 3.1), when the rule does not respect the gap constraint there is no need to extend its antecedent, as the resulting rule will also refrain from respecting this gap. It can thus be pruned and it is not extended anymore, i.e., none of its super-antecedent rules enter the mining process (and this reduces significantly the search space).

Recall that in this step we mine only rules with a single event in the consequent. However, if we would like to mine rules with longer consequents, we do not need to perform a gap verification, as the distance between the suffix of the antecedent and the prefix of the consequent does not change when we extend the consequent.

2. (Support pruning.) Support and confidence verification (see Equation 3.3) are performed on rules that respect the gap constraint in order to obtain distant sequential rules (Algorithm 2, lines 4, 5, 16, 17). If the rule is not frequent, then the process stops, i.e., the rule is not extended and no candidate rules are generated from it, according to the anti-monotonicity property of support. This is the second step of candidate rule pruning.
3. (Antecedent extension.) For each frequent rule the occurrences of which respect the minimal gap constraint, the mining process continues by extending the antecedent of the rule (Algorithm 2, lines 7, 19).

It is important to mention that the anti-monotonicity property of both the gap constraint and the frequency measure is a key factor for candidate rule pruning during the mining process and thus for the efficiency of the algorithm in terms of execution time and memory consumption.

3.3. **Application of distant rules in a distant event prediction task.** Let us now discuss the way mined distant rules are used in a distant event prediction task and how they can be actionable and intelligible to domain experts. Before starting the mining process, the unit used to represent timestamps should be determined as it depends on the application domain and especially on the velocity of the sequence generation: it may be *minutes* for Twitter applications, *days* for purchase frameworks, *months* for Earth observation, or just the position of the event (regardless of time) for click-prediction application, etc.

For a prediction task, during the monitoring of the event sequence database, we wait until a rule matches the sequence, i.e., until the events of its antecedent appear in the sequence, keeping the same order. Once a rule matches a sequence, the prediction of its consequent is immediately triggered and estimated within at least gap timestamps. Let us consider the online shopping framework, the temporal unit used to represent timestamps being days. An example of mined rule could be: R: camera, lens, tripod $\rightarrow$ printer, gap $=$ 10 days, $\text{conf}(R) = 0.9$, in other words: when monitoring the purchase flow, once a client buys a camera, then a lens and then a tripod, the prediction of the printer purchase is triggered. With a probability of 90%, this purchase is estimated within at least 10 days. Another example related to churn application is presented in Section 4.4.1.

In the following section, we present experiments performed to evaluate the two proposed algorithms.

Algorithm 2: *D-SR-inProcess*

input : a sequence database $\mathscr{S}$, values of minsupp, minconf, gap
output: R_{DS}: List of distant sequential rules

Procedure *Rule construction via traditional algorithm*
  foreach $p_1 \rightarrow q_1 \in$ *set of candidate rules under construction* **do**
    $\text{Occ}_{\text{gap}}(p_1, q_1) :=$ *Gap Pruning*$(\text{Occ}(p_1), \text{Occ}(q_1))$
    if $|\text{Occ}_{\text{gap}}(p_1, q_1)| \geq$ minsupp **then**
      if $\text{conf}(p_1 \rightarrow q_1) \geq$ minconf **then**
        add $p_1 \rightarrow q_1$ to R_{DS}
      Extend Antecedent$(p_1 \rightarrow q_1)$
Procedure *Gap Pruning* $(\text{Occ}(p_{k+1}), \text{Occ}_{\text{gap}}(p_1, \ldots, p_k, q_1))$
  foreach $\text{occ}(p_1, \ldots p_k, q_1) \in \text{Occ}_{\text{gap}}(p_1, \ldots, p_k, q_1)$ **do**
    construct $\text{occ}(p_1, \ldots p_k, p_{k+1}, q_1)$
    if $(t_{q_1} - t_{p_{k+1}}) \geq$ gap **then**
      add $(t_{p_{k+1}}, t_{q_1})$ to $\text{Occ}_{\text{gap}}(p_1, \ldots, p_{k+1}, q_1)$
  return $\text{Occ}_{\text{gap}}(p_1, \ldots, p_{k+1}, q_1)$
Procedure *Extend Antecedent* $(p_1, \ldots, p_k \rightarrow q_1)$
  $\text{Occ}_{\text{gap}}(p_1, \ldots, p_k, p_{k+1}, q_1) :=$ *Gap Pruning*$(\text{Occ}(p_{k+1}), \text{Occ}_{\text{gap}}(p_1, \ldots, p_k, q_1))$
  if $|\text{Occ}_{\text{gap}}(p_1, \ldots, p_k, p_{k+1}, q_1)| \geq$ minsupp **then**
    if $\text{conf}(p_1, \ldots, p_k, p_{k+1} \rightarrow q_1) \geq$ minconf **then**
      add $p_1, \ldots, p_k, p_{k+1} \rightarrow q_1$ to R_{DS}
    Extend Antecedent$(p_1, \ldots, p_k, p_{k+1} \rightarrow q_1)$

4. **Experimental results.** This section is dedicated to the evaluation of the two algorithms : *D-SR-postMining* and *D-SR-inMining* (two algorithms of the general framework *D-SR*).

As already mentioned, our algorithms run as extensions of traditional state-of-the-art algorithms. For the experiments we used the state-of-the-art algorithms *MINEPI+* [31] that first mines patterns and then rules, and *TRuleGrowth* [24] that mines directly rules.

We use the same publicly available data sets and settings used by the authors of *TRuleGrowth* in [16, 23] to evaluate algorithm performances. In addition, we present some results on a real-world data set for churn analysis in banking.

Experiments are performed on a laptop computer with a 2.6 GHz Intel Core i7 Processor and 16 GB of RAM, running on an OS X system. All the algorithms are implemented in Java. *MINEPI+* has been implemented by ourselves and *TRuleGrowth* has been downloaded from their authors' website [2] and adapted to our needs.

In the remainder of this section, we present the data sets used, as well as the performance of our algorithms in terms of execution time, scalability and in the frame of a prediction task.

4.1. **Data sets.** Table 1 shows the characteristics of the three data sets used in the experiments. They are briefly described below:

[2] `http://www.philippe-fournier-viger.com/spmf/index.php?link=download.php`, downloaded on February 2018

TABLE 1. Characteristics of the data sets

Data set			# Sequences	# Events	Average event number per sequence (average sequence length)
KOSARAK	very large	sparse	990,000 (70,000)	21,144	7.97 (max= 796)
BMS1	small	sparse	59,601	497	2.51 (max= 267)
BANKING	very large	sparse	825,741	71	25.00 (max= 80)

1. KOSARAK[3] is a sparse data set that contains a large number of sequences of click-stream data from an online news portal. This data set fits perfectly for the evaluation of the scalability of the algorithms when varying the number of sequences in the data set. For some experiments, as will be shown later, only 70,000 sequences have been used, like in [23].
2. BMS-WEBVIEW1 (BMS1)[4] is a very sparse data set that contains sequences of click-stream e-commerce data.
3. BANKING is an anonymized real-world banking data set provided by *Crédit Mutuel ARKEA*[5]. It represents the history of actions (events) of a large number of clients. Each sequence is related to a single client and represents his/her actions ordered by their timestamps. Notice that events can occur multiple times within each sequence and that the average frequency of the 71 distinct events is huge (more than 200,000 times each). This data set is used to show the utility of the mined rules for churn prediction in a real-world bank expert context.

Tables 2 and 3 show the results of all experiments performed on the KOSARAK data set and on the BMS1 data set. These experiments allow to compare algorithms *MINEPI+*, *TRuleGrowth*, *D-SR-postMining* and *D-SR-inMining* in terms of scalability (number of mined rules and execution time). Furthermore, they present the performance of our algorithms in a prediction task. The values of algorithm parameters are shown in the first five columns of Tables 2 and 3. For each experiment, we vary only one parameter in order to study its impact (represented in blue or in red), while all other parameters remain fixed (represented in black). Among the varying values of the parameter we select the one that provides us with a sufficient number of rules to pursue experiments, and this value we represent in red.

4.2. **Scalability Evaluation.** Tables 2 and 3 present the results of evaluation experiments on the scalability of our algorithms. Scalability is evaluated by observing the amount of mined rules as well as execution time, when varying:

- the number of sequences (this is relevant only for the KOSARAK data set, as it contains a large number of sequences that allows us to evaluate scalability: we vary the number of sequences from 10,000 to 200,000 with an incremental step of 20,000),
- the gap size (for both data sets) and
- the window size (for both data sets).

Notice that *D-SR-postMining* can be used with both *MINEPI+* and *TRuleGrowth* as post-processing step. However, *D-SR-inMining* can be used only with *TRuleGrowth* (during the mining process), as *TRuleGrowth* mines rules directly.

[3]Kosarak data set: `http://goo.gl/4B6ve5`, downloaded on February 2018. The name *kosarak* is Hungarian, it is pronounced *kósharak* and means "shopping baskets".

[4]BMS1 data set: `http://www.philippe-fournier-viger.com/spml/index.php?link=datasets.php`, downloaded on February 2018.

[5]This data set has been anonymized within the Datalabs service of Crédit Mutuel ARKEA in order to be used for research purposes. It is confidential and cannot be distributed online.

TABLE 2. Performance on KOSARAK data set

minsupp	minconf	w	gap	#seq	*MINEPI+*		D-SR *-postMining*		Whole process	*TRuleGrowth*		D-SR *-postMining*		Whole process	D-SR *-inMining*		Whole process	D-SR prediction	
					time	#R	time	#R	time	time	#R	time	#R	time	time	#R	time	accuracy	coverage
0.001	0.5	10	5	70k	948	70,000	21.4	700	969.4	500	70,000	21.4	700	521.4	5	700	50		
0.003	0.5	10	5	70k	331	1,700	1	450	332	70	1,700	1	450	71	1	450	25		
0.005	0.5	10	5	70k	201	270	0.2	20	201.2	50	270	0.2	20	50.2	0.8	20	15		
0.003	0.2	10	5	70k	951	3,200	1.1	703	952.1	628	3,200	1.1	703	629.1	6	703	78		
0.003	0.5	10	5	70k	331	1,700	1	450	332	70	1,700	1	450	71	1	450	25		
0.003	0.8	10	5	70k	146	68	0.1	5	146.1	30	68	0.1	5	30.1	0.5	5	8		
0.003	0.5	10	5	70k	331	1,700	1	450	332	70	1,700	1	450	71	1	450	25	29	45
0.003	0.5	15	5	70k	1,145	7,000	1.6	650	1,146.6	300	7,000	1.6	650	301.6	1	650	70	35	46
0.003	0.5	20	5	70k	1,988	17,000	8	701	1,989	900	17,000	8	701	908	5	701	85	40	46
0.003	0.5	20	0	70k	1,988	17,000	11	3,922	1,999	900	17,000	11	3,922	911	20	3,922	402	40	35
0.003	0.5	20	5	70k	1,988	17,000	8	701	1,996	900	17,000	8	701	908	5	701	85	38	32
0.003	0.5	20	15	70k	1,988	17,000	3.1	102	1,991.1	900	17,000	3.1	102	903.1	1	102	34	30	20
0.003	0.5	20	20	70k	1,988	17,000	1	23	1,989	900	17,000	1	23	901	1	23	12	25	15
0.003	0.5	10	5	10k	243	1,900	1	450	244	30	1,900	1	450	31	1	450	25		
0.003	0.5	10	5	30k	250	1,750	1	420	251	40	1,750	1	420	41	1.2	420	26		
0.003	0.5	10	5	50k	300	1,600	0.9	405	300.9	50	1,600	0.9	405	50.9	1	405	20		
0.003	0.5	10	5	70k	331	1,700	1	450	332	70	1,700	1	450	71	1	450	25		
0.003	0.5	10	5	90k	339	1,550	0.9	401	339.9	80	1,550	0.9	401	80.9	1.3	401	19		
0.003	0.5	10	5	110k	487	1,500	0.9	391	487.9	100	1,500	0.9	391	100.9	1.2	391	18		
0.003	0.5	10	5	130k	499	1,550	0.9	401	499.9	150	1,550	0.9	401	150.9	1.2	401	19		
0.003	0.5	10	5	150k	514	1,600	0.9	405	514.9	200	1,600	0.9	405	200.9	1.2	405	20		
0.003	0.5	10	5	170k	549	1,650	0.9	401	549.9	408	1,650	0.9	408	408.9	1	401	19		
0.003	0.5	10	5	190k	596	1,700	1	450	597	270	1,700	1	450	271	1	450	25		

TABLE 3. Performance on BMS1 data set

minsupp	minconf	w	gap	#seq	*MINEPI+*		D-SR *-postMining*		Whole process	*TRuleGrowth*		D-SR *-postMining*		Whole process	D-SR *-inMining*		Whole process	D-SR prediction	
					time	#R	time	#R	time	time	#R	time	#R	time	time	#R	time	accuracy	coverage
0.72	0.5	10	5	70k	25	5,000	1	450	26	70	5,000	1	450	71	1	450	30	40	69
0.72	0.5	12	5	70k	110	18,000	1.6	650	111.6	300	18,000	1.6	650	301.6	1.1	650	100	42	70
0.72	0.5	16	5	70k	170	20,000	8	701	178	900	20,000	8	701	908	1.3	701	140	43	71
0.72	0.5	20	5	70k	210	25,000	8	701	218	900	25,000	8	701	908	1.3	701	200	44	72
0.003	0.5	20	0	70k	1,988	17,000	11	3,922	1,999	900	17,000	11	3,922	911	10	3,922	402	44	72
0.003	0.5	20	5	70k	1,988	17,000	8	701	1,996	900	25,000	8	701	908	1	701	200	35	60
0.003	0.5	20	15	70k	1,988	17,000	3.1	102	1,991.1	900	17,000	3.1	102	903.1	1	102	34	30	50
0.003	0.5	20	20	70k	1,988	17,000	1	23	1,989	900	17,000	1	23	901	0.8	23	12	26	30

4.2.1. *Performance of Each Individual Algorithm.* We first compare the execution time and the number of mined rules of each algorithm, on each data set separately for scalability evaluation.

Recall that our algorithms *D-SR-postMining* and *D-SR-inMining* coexist with state-of-the-art algorithms by extending them, namely by adding an additional step in order to mine distant rules. Therefore, their performance in terms of execution time is not directly comparable to the one of state-of-the-art algorithms.

We notice that both traditional algorithms *MINEPI+* and *TRuleGrowth* mine a large number of rules (as can be seen in Table 2 for the KOSARAK data set: 1,700 rules when #seq = 70k). These rules are strictly filtered by our algorithms, and thus we manage to have 450 distant rules when #seq = 70k for the KOSARAK data set. In the same way, the number of final rules mined by *D-SR-inMining* algorithm is significantly lower than the one of *TRuleGrowth* on the KOSARAK data set. This is due to the fact that our algorithm *D-SR-inMining* imposes more constraints during the mining process (i.e., the gap constraint), and thus stops the completion of a rule at an earlier stage of the mining process. However, it results in rules with lower support which are potentially filtered out.

We also notice that *MINEPI+* needs a large amount of time to mine a large amount of rules (331 seconds to mine 1,700 rules when #seq = 70k, as can be seen in Table 2). *TRuleGrowth* needs less time (only 70 seconds) to mine the same set of rules. However, our algorithms *D-SR-postMining* and *D-SR-inMining* perform even better, and take nearly 1 second to get distant rules, regardless of the traditional algorithms in which they are incorporated.

Let us now turn to execution time. As expected, execution time of both traditional algorithms, *MINEPI+* and *TRuleGrowth*, increases linearly with respect to the size of the data set (the number of sequences). However, the execution time needed to mine distant rules remains approximately constant and depends rather on the number of mined rules. For example, in Table 2 for the KOSARAK data set, we can notice that for the extreme case of minsupp = 0.001, the number of rules increases significantly. However, this impacts only slightly the execution time of our algorithms: *D-SR-postMining* takes 21.4 seconds, while *D-SR-inMining* only takes 5 seconds.

Also, we notice that the execution time of both algorithms *TRuleGrowth* and *D-SR-inMining* grows with the size of the data set. We also notice that *D-SR* is nearly constant and its execution time is constant while the number of sequences varies from 10,000 to 200,000. However, the execution time of *TRuleGrowth* is multiplied by 9 in the same situation. Once again, the low execution time of our algorithm *D-SR-inMining* is due to the fact that it imposes more constraints during the mining process (i.e., the gap constraint), and thus stops the completion of a rule at an earlier stage of the mining process.

We conclude that our algorithms for mining distant rules are more scalable than traditional algorithms.

4.2.2. *Performance of the Whole Process of Distant Rule Mining.* Let us now turn to the evaluation of the whole process in terms of time needed to mine distant rules, in the case of traditional algorithms combined with one of our two algorithms. It is important to mention that, in the case of mining distant rules using *D-SR-postMining*, the total execution time represents the sum of execution time of the traditional algorithm (*MINEPI+* or *TRuleGrowth*) and execution time of *D-SR-postMining*, since *D-SR-postMining* runs once the traditional algorithm has finished its mining process. In the case of mining distant rules using *D-SR-inMining*, only *TRuleGrowth* can be used as it mines directly rules. Thus, in this case, the total execution time represents the sum of the new execution time of *TRuleGrowth* (after pruning during the mining process) and the execution time of *D-SR-inMining*. Recall that *D-SR-inMining* runs through different levels of *TRuleGrowth* which prunes the search space. Consequently, no clear separation between both algorithms can be done.

In Tables 2 and 3, the total time is represented by the blue, green and pink columns and is calculated from execution times (gray columns) of the traditional algorithm and of the *D-SR* algorithm on its left-hand side. We notice that the total execution time needed

to mine distant rules via *D-SR-postMining* is quite high because it depends on the performance of the traditional algorithm. For example, in Table 2 for the KOSARAK data set, it is equal to 332 seconds for #seq = 70k when using *MINEPI+*, and to 71 seconds when using *TRuleGrowth*. Notice that, due to application constraints, in case one doesn't have the choice of the distant-rule algorithm to use and no directly-mining-rules algorithm is available, one may have to use *D-SR-postMining*. It is noteworthy that the total execution time is significantly reduced when the process is based on a traditional algorithms that mine rules directly. For example, in Table 2, the total time is only 25 seconds when #seq = 70k.

Also, as expected, when increasing minsupp and minconf, the execution time and the number of rules decrease linearly for all algorithms. The same conclusion is valid for both data sets.

We can conclude that for mining distant rules, when we have the choice of the traditional algorithm on which *D-SR* algorithms car run, it is more efficient in terms of execution time to use an algorithm that mines rules directly with *D-SR-inMining*.

4.2.3. *Impact of w and gap on Scalability.* Tables 2 and 3 show the performance of our algorithms when varying the window size from 10 to 20. To accelerate the experiment on the KOSARAK data set, we use the first 70,000 sequences of this data set. While execution time of both *D-SR* algorithms stays nearly constant, execution times of *MINEPI+* and *TRuleGrowth* increase significantly for larger window sizes. *D-SR-inMining* has a scalability coefficient of 5, while the scalability coefficient of *TRuleGrowth* reaches 12. This is due to the fact that *D-SR-inMining* results in much less rules, since the integrated constraints (= the gap) stop the mining process early.

Tables 2 and 3 show the impact of the gap size on the execution time and on the number of mined rules of *D-SR* (notice that since the gap parameter is not available for traditional algorithms, their execution times remain the same). As expected, the execution time of both *D-SR* algorithms decreases significantly when the gap size grows, as significantly less rules are mined for large values of the gap.

From Table 3 on the BMS1 data set, we can draw the same conclusions as those for the KOSARAK data set, and this is as expected, since the BMS1 data set is only slightly denser than the KOSARAK one.

We conclude that both *D-SR* algorithms are significantly more scalable when varying the window size than when varying the number of sequence. This proves the high impact of the number of sequences on the running time of the mining algorithms. Both *D-SR* algorithms not only run faster than *MINEPI+* and *TRuleGrowth* during these experiments, but also have good scalability coefficients. As we mentioned before, this high performance of *D-SR* is related to the early pruning that the algorithm performs in order to mine rules.

4.3. **Performance in a Prediction Task.** In order to evaluate the performance in prediction of an association rule mining algorithm, each data set is split into two subsets: one used for training (generation of distant rules using a *D-SR* algorithm), and the other for testing (prediction task). In order to perform training, we use, for both data sets, the sets of sequences already used for scalability evaluation in the experiments described in Tables 2 and 3. In particular, for KOSARAK, the first 70k sequences are used for training and the 70k subsequent sequences are used for testing. For BMS1, the 29,800 first sequences are used for training and an equal number of subsequent sequences is used for testing.

The prediction is accomplished by first scanning all mined rules to identify those that match the sequence in the test data set (a rule matches a sequence when its antecedent appears in the sequence), and then getting a prediction that corresponds to the consequent

of the matching rules. This is performed by selecting the matching sequential rule (@10) with the best confidence in the given experiment.

In addition, the task of distant prediction in a sequence is to predict the occurrence of the consequent following the antecedent of the rule at a gap-minimum distance.

It is important to notice that for the same parameter values and whatever the algorithm used to mine distant rules (*D-SR-postMining* or *D-SR-inMining*), the final number of distant rules remains the same. For this reason, in this section we refer indifferently to *D-SR-postMining* or *D-SR-inMining* as *D-SR*.

In order to evaluate the prediction performance, we measure the accuracy and define it as the number of good predictions divided by the number of sequences in the test set. We measure also the coverage (or the match rate) and define it as the number of sequences for which it was possible to make a prediction, i.e., when the antecedent finds a match in the sequence. In this experiment, we evaluate the prediction performance when varying the gap and the window size w, these two values are proper to *D-SR* algorithms.

4.3.1. *Impact of w on performance in a prediction task.* Tables 2, 3 and Figure 2 show the evaluation of prediction performance of algorithms when varying w for the Kosarak and BMS1 data sets. The initial parameters for all the experiments in this sub-section are shown in Tables 2 and 3.

Overall, we observe that using *D-SR* always provides an (up to 30%) higher accuracy but a lower coverage. As expected, the coverage of *D-SR* is significantly lower than the one of *TRuleGrowth*, as rules mined by *D-SR* are more specific and include more constraints (i.e., the order of events).

When $w = 20$ on the Kosarak data set, *D-SR* provides 40% accuracy and 46% coverage, while *TRuleGrowth* provides 20% accuracy and 50% coverage. On the BMS1 data set, *D-SR* provides 45% accuracy and 70% coverage, while *TRuleGrowth* provides 30% accuracy and 90% coverage. Even though *D-SR* has a low coverage, it is able to predict more accurately the distant events, which represents a very satisfactory result in the context of distant events prediction. Actually, in several applications where a risky or costly action should follow each prediction, it is preferable and safer to correctly predict few events instead of badly predicting a large number of them.

The experiment has also shown that using the window-size constraint is beneficial. When increasing w, specific rules are potentially more frequent and thus can potentially be retained, which, in turn, can increase the coverage resulting by these rules.

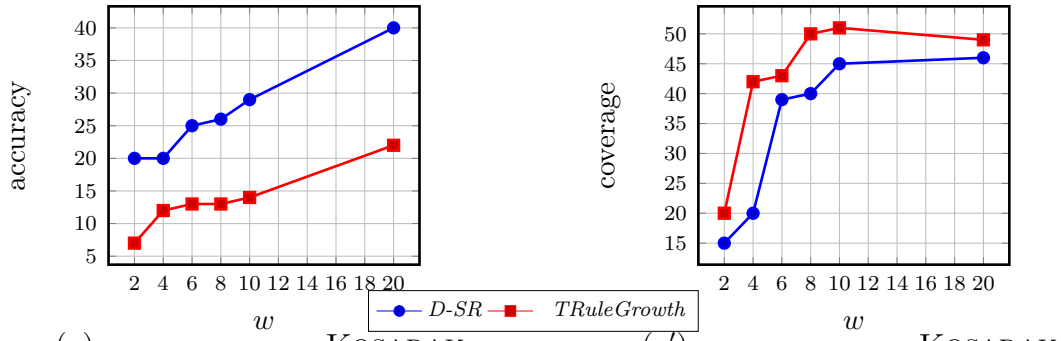
The same conclusions can be drawn for the BMS1 data set.

We conclude that, for both used data sets, *D-SR* performs better than traditional algorithms in a prediction task, even though the coverage is lower.

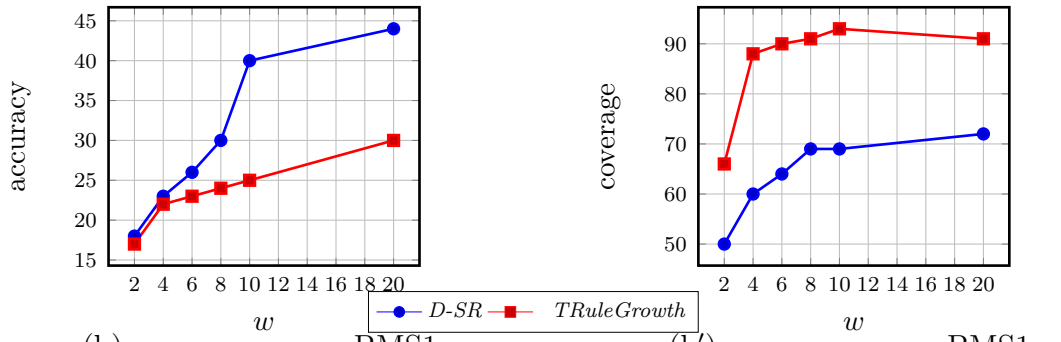
4.3.2. *Impact of the gap on performance in a prediction task.* In Tables 2, 3 and in Figure 3, we evaluate the accuracy of *D-SR* when varying the gap constraint on the Kosarak and BMS1 data sets. When gap = 0, we can compare *D-SR* and *TRuleGrowth* (accuracy and coverage values for the algorithm *TRuleGrowth* are set to 0 when gap > 0, as this algorithm does not include the gap constraint).

We notice that accuracy decreases when the gap grows. This is expected since the larger the gap, the more difficult the prediction is, as we predict events occurring after a long time interval. We also notice that accuracy remains relatively acceptable for high values of gap. For example, when gap = 20, accuracy is about 25% for Kosarak, and about 26% for the BMS1 data set. We consider this as a high performance, as rules here aim at predicting events after 20 time units.

We conclude that, for both used data sets, *D-SR* succeeds in predicting distant consequents and performs better than *TRuleGrowth* in the prediction task.

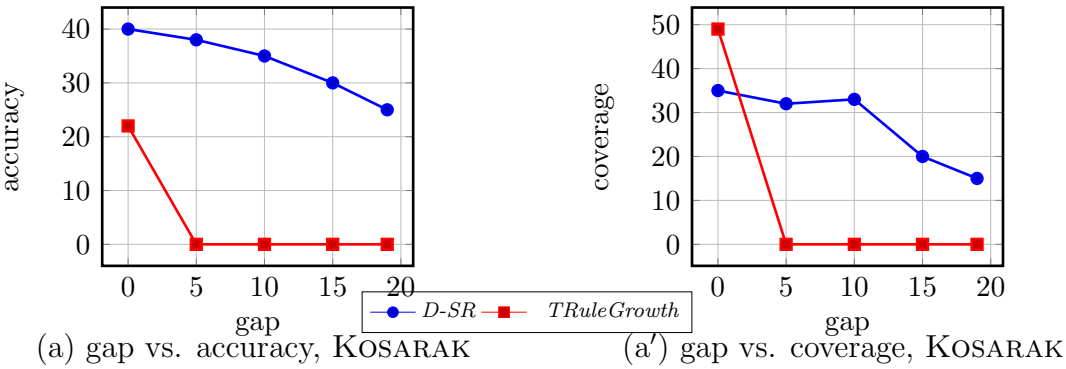


(a) w vs. accuracy, KOSARAK (a′) w vs. coverage, KOSARAK

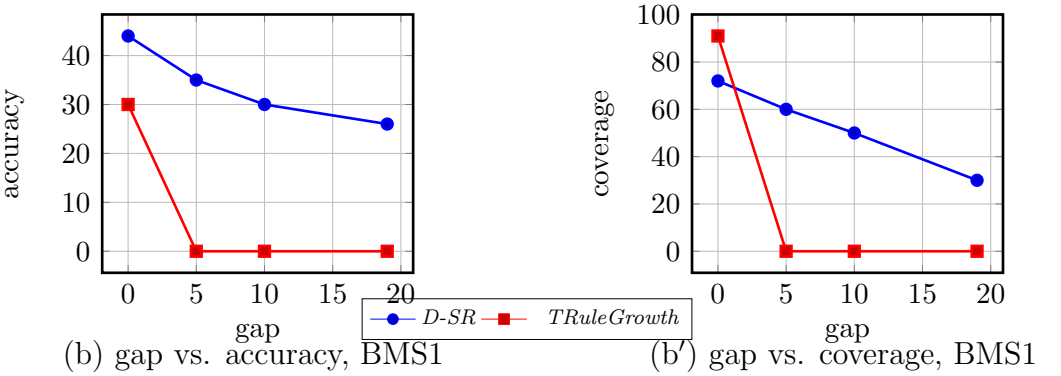


(b) w vs. accuracy, BMS1 (b′) w vs. coverage, BMS1

FIGURE 2. Window size impact on prediction: KOSARAK, BMS1 data sets



(a) gap vs. accuracy, KOSARAK (a′) gap vs. coverage, KOSARAK



(b) gap vs. accuracy, BMS1 (b′) gap vs. coverage, BMS1

FIGURE 3. Gap impact on prediction: KOSARAK, BMS1 data sets

4.4. **Evaluation on a Real-World Banking Data Set.** In this section we evaluate our algorithm *D-SR* on the real-world BANKING data set, with the main goal to present real-world examples of mined rules and their utility in an early churn prediction.

4.4.1. *Comparison of the Rules Formed.* As an extension of traditional algorithms, *D-SR* integrates a minimal gap between the antecedent and the consequent of the sequential rule. When gap = 0, the situation is almost similar to the one of traditional algorithms. As in the previous section, we choose to compare *D-SR* to *TRuleGrowth*.

Table 4 displays the prediction performance and the number of rules produced by our algorithm and by *TRuleGrowth* on the BANKING data set, for minsupp = 0.109, minconf = 0.8, w = 360, and gap = 180 days. The gap parameter is not used for *TRuleGrowth*.

D-SR builds about 200 sequential rules, whereas the algorithm *TRuleGrowth* mines about 1,000 rules, this represents a 80% decrease of the number of mined rules. As shown in the section on scalability evaluation, this decrease is due to the fact that *D-SR* imposes additional constraints (gap and order of events), so that these rules are more specific. *D-SR* is more efficient than *TRuleGrowth* in predicting distant events (events at more than gap = 180 distance) with accuracy of 64%. Actually, due to the criticality of the churn application in this paper, it is important for us to outperform state-of-the-art algorithms especially for this real-world banking data set. We also notice that *TRuleGrowth* covers more rules than *D-SR*. However, these rules have not been designed to mine distant rules as they do not model the distance between events, so, as expected, they are less accurate than those of *D-SR*.

TABLE 4. *D-SR* vs. *TRuleGrowth* on BANKING data set

Algorithm	#Rules	Accuracy	Coverage
D-SR	200	64%	52%
TRuleGrowth	1,000	55%	80%

We present now some examples of sequential rules extracted by *D-SR*. Recall that for confidentiality reasons imposed by Crédit Mutuel ARKEA (the provider of this real-world Banking data set), we are not allowed to show all experiments or more examples of rules.

`(House credit simulation, cancellation of house insurance contract, exceptional external money transfer)`
$\rightarrow$
`(client attrition)`, with gap = 180 days.

Application of this rule in a distant event prediction task. While monitoring the event sequence during a prediction task, once all (ordered) events of the antecedent of the rule appear, they trigger the prediction of rule consequent within at least gap timestamps. In the same way, during the monitoring of banking actions of clients (that represent a sequence database), we can choose days as timestamp unit (the same unit as when mining the rules). Thus, when detecting a client profile that (a) does a house loan simulation, (b) cancels his/her house insurance contract, and then (c) transfers a given amount of money to an account in another bank, then he/she is likely to leave the bank within at least 180 days. The bank thus may potentially benefit from these 180 days to contact the client in order to propose solutions to eventually prevent him/her from leaving the bank.

Here is a rule that has been extracted by *TRuleGrowth*, but not by *D-SR*, as it does not satisfy the desired sequential-rule characteristics:

`(Salary no more received on the account, exceptional external money transfer, closure of savings account, exceptional external money transfer, decreased online activities)`
$\rightarrow$
`(client attrition)`, within N time units ($N \ll 180$) between antecedent and consequent.

This rule means that when detecting a client profile for which the monthly salary is no more received on his/her account, and he/she transfers a given amount to an external account, then closes his/her savings account, transfers again a given amount to an external account, and finally connects only scarcely to his/her account, this means it is likely that he/she may leave the bank within at least N days. The interval of $N \ll 180$ days is considered as very short and probably not sufficient for the bank to react and eventually prevent the client's departure.

This rule is useful in applications where the goal is to predict close events. However, in the context of early prediction of distant events, this rule does not fit, as its consequent is too close to its antecedent.

5. **Conclusion and Perspectives.** In this paper, we have proposed two original algorithms (*D-SR-postMining* and *D-SR-inMining*) that mine sequential rules with distant consequents under an anti-monotone minimal gap constraint. These algorithms represent a general framework allowing the use of existing traditional rule mining algorithms in order to mine distant rules. Both algorithms were evaluated on three data sets. Experiments show that our algorithms are scalable and more efficient than other state-of-the-art algorithms in terms of execution time for the used data sets. They are also more accurate in a prediction task for the real-world banking data set, which is very important due to the criticality of the churn application in this paper.

The algorithms mine rules with a minimal gap constraint between the antecedent and the consequent. This constraint can be very strict in several applications, when occurrences of rules that do not strictly respect the gap are not accepted. For future work, we aim at introducing interestingness measures in order to provide more flexibility on the gap constraint.

As we have shown in this paper, a window is used to constrain the research space of the mining algorithms. The value of this window constraint is fixed on the level of each run of the algorithm. We notice that the larger the window, the more distant is the consequent of the mined rules. For this reason, in future work we aim at removing this window constraint in order to be able to mine simultaneously rules with consequents at all distances, namely near, distant and very distant consequents. The challenge will be to propose a solution that does not explode the search space.

As we have shown, the anti-monotonicity property is a key factor for pruning rules using our algorithms. In future work, we intend to propose new frequency-constraint or temporal-constraint measures, again relying on the anti-monotonicity property. We assume that will allow to mine more precise rules and will improve the efficiency of the algorithm in terms of time and memory consumption.

Acknowledgments. This research has been supported by the group Crédit Mutuel ARKEA. We would like to thank members of the Datalabs service at the Innovation

and Operation Pole at Crédit Mutuel ARKEA for their collaboration and for providing the anonymized confidential BANKING data set used in this paper.

REFERENCES

[1] A. Achar, S. Laxman, R. Viswanathan, and P. Sastry, "Discovering injective episodes with general partial orders," *Data Mining and Knowledge Discovery*, vol. 25(1), pp. 67–108, 2012.
[2] A. Achar, A. Ibrahim, and P. S. Sastry, "Pattern-growth based frequent serial episode discovery," *Data & Knowledge Engineering*, vol. 87, pp. 91–108, 2013.
[3] A. Adadi and M. Berrada, "Peeking inside the black-box: A survey on explainable artificial Intelligence (XAI)," *IEEE Access*, vol. 6, pp. 52138–52160, 2018.
[4] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *ACM SIGMOD Record*, vol. 22, pp. 207–216, 1993.
[5] R. Agrawal and R. Srikant, "Mining sequential patterns," *The International Conference on Data Engineering*, pp. 3–14, 1995.
[6] S. Akther, M. R. Karim, M. Samiullah, and C. F. Ahmed, "Mining non-redundant closed flexible periodic patterns," *Engineering Applications of Artificial Intelligence*, vol. 69, pp. 1–23, 2018.
[7] K. Amphawan and P. Lenca, "Mining top-k frequent-regular closed patterns," *Expert Systems with Applications*, vol. 42(21), pp. 7882–7894, 2015.
[8] K. Amphawan, A. Surarerks, and P. Lenca, "Mining periodic-frequent itemsets with approximate periodicity using interval transaction-IDs list tree," *The International Conference on Knowledge Discovery and Data Mining*, pp. 245–248, 2010.
[9] X. Ao, P. Luo, J. Wang, F. Zhuang, and Q. He, "Mining precise-positioning episode rules from event sequences," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30(3), pp. 530–543, 2018.
[10] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," *Advances in Neural Information Processing Systems*, pp. 1171–1179, 2015.
[11] L. Cao, "Combined mining: Analyzing object and pattern relations for discovering and constructing complex yet actionable patterns," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 3(2), pp. 140–155, 2013.
[12] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad, "Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1721–1730, 2015.
[13] V. Chaurasia and S. Pal, "Early prediction of heart diseases using data mining techniques," *Caribbean Journal of Science and Technology*, vol. 1, pp. 208–217, 2013.
[14] L. Fahed, A. Brun, and A. Boyer, "DEER: Distant and Essential Episode Rules for early prediction," *Expert Systems with Applications*, vol. 93, pp. 283–298, 2018.
[15] P. Fournier-Viger, A. Gomariz, M. Campos, and R. Thomas, "Fast vertical mining of sequential patterns using co-occurrence information," *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 40–52, 2014.
[16] P. Fournier-Viger, T. Gueniche, and V. S. Tseng, "Using partially-ordered sequential rules to generate more accurate sequence prediction," *The International Conference on Advanced Data Mining and Applications*, pp. 431–442, 2012.
[17] P. Fournier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, "ERMiner: sequential rule mining using equivalence classes," *The International Symposium on Intelligent Data Analysis*, vol. 8819, pp. 108–119, 2014.
[18] P. Fournier-Viger, J. C. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7(4), e1207, 2017.
[19] P. Fournier-Viger, J. C. W. Lin, Q. H. Duong, and T. L. Dam, "PHM: mining periodic high-utility itemsets," *The Industrial Conference on Data Mining*, pp. 64–79, 2016.
[20] P. Fournier-Viger, J. C. W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1(1), pp. 54–77, 2017.
[21] P. Fournier-Viger and V. S. Tseng, "TNS: mining top-k non-redundant sequential rules," *The Symposium on Applied Computing*, pp. 164–166, 2013.
[22] P. Fournier-Viger, C. W. Wu, A. Gomariz, and V. S. Tseng, "VMSP: Efficient vertical mining of maximal sequential patterns," *The Canadian Conference on Artificial Intelligence*, vol. 8436, pp. 83–94, 2014.

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

[24] P. Fournier-Viger, C. W. Wu, V. S. Tseng, and R. Nkambou, "Mining sequential rules common to several sequences with the window size constraint," *The Canadian Conference on Artificial Intelligence*, vol. 7310, pp. 299–304, 2012.

[25] P. Fournier-Viger, P. Yang, J. C. Lin, and U. Yun, "HUE-Span: fast high utility episode mining," *The International Conference on Advanced Data Mining and Applications*, vol. 11888, pp. 169–184, 2019.

[26] M. Gan and H. Dai, "Fast mining of non-derivable episode rules in complex sequences," *The International Conference Modeling Decision for Artificial Intelligence*, vol. 6820, pp. 67–78, 2011.

[27] W. Gan, J. C. W. Lin, P. Fournier-Viger, H. C. Chao, and P. S. Yu, "A survey of parallel sequential pattern mining," *ACM Transactions on Knowledge Discovery from Data*, vol. 13(3), pp. 1–34, 2019.

[28] I. Giurgiu and A. Schumann, "Explainable failure predictions with rnn classifiers based on time series data," Preprint `https://arxiv.org/pdf/1901.08554`, 2019.

[29] D. Gunning and D. Aha, "DARPA's Explainable Artificial Intelligence (XAI) Program," *AI Magazine*, vol. 40(2), pp. 44–58, 2019.

[30] Z. He, X. Xu, S. Deng, and R. Ma, "Mining action rules from scratch," *Expert Systems with Applications*, vol. 29(3), pp. 691–699, 2005.

[31] K. Y. Huang and C. H. Chang, "Efficient mining of frequent episodes from complex sequences," *Information Systems*, vol. 33, pp. 96–114, 2008.

[32] K. Iwanuma, R. Ishihara, Y. Takano, and H. Nabeshima, "Extracting frequent subsequences from a single long data sequence a novel anti-monotonic measure and a simple on-line algorithm," *IEEE International Conference on Data Mining*, pp. 186–193, 2005.

[33] M. S. M. Jabbar, C. Bellinger, O. R. Zaïane, and A. Osornio-Vargas, "Discovering co-location patterns with aggregated spatial transactions and dependency rules," *The International Journal of Data Science and Analytics*, vol. 5(2-3), pp. 137–154, 2018.

[34] A. N. Jagannatha and H. Yu, "Structured prediction models for rnn based sequence labeling in clinical text," *The Conference on Empirical Methods in Natural Language Processing*, vol. 2016, pp. 856–865, 2016.

[35] R. U. Kiran, A. Anirudh, C. Saideep, M. Toyoda, P. K. Reddy, and M. Kitsuregawa, "Finding periodic-frequent patterns in temporal databases using periodic summaries," *Data Science and Pattern Recognition*, vol. 3(2), pp. 24–46, 2019.

[36] R. U. Kiran, M. Kitsuregawa, and P. K. Reddy, "Efficient discovery of periodic-frequent patterns in very large databases," *Journal of Systems and Software*, vol. 112, pp. 110–121, 2016.

[37] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol, 2016, pp. 1675–1684, 2016.

[38] S. Laxman, P. Sastry, and K. Unnikrishnan, "A fast algorithm for finding frequent episodes in event streams," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 410–419, 2007.

[39] S. Laxman and P. S. Sastry, "A survey of temporal data mining," *Sadhana*, vol. 31(2), pp. 173–198, 2006.

[40] S. Laxman, V. Tankasali, and R. W. White, "Stream prediction using a generative model based on frequent episodes in event sequences," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 453–461, 2008.

[41] Y. Le Bras, P. Lenca, and S. Lallich, "Mining classification rules without support: an anti-monotone property of Jaccard measure," *The International Conference on Discovery Science*, vol. 6926, pp. 179–193, 2011.

[42] H. Li, Z. Li, S. Peng, J. Li, and C. E. Tungom, "Mining the frequency of time-constrained serial episodes over massive data sequences and streams," *Future Generation Computer Systems*, 2019. `doi:10.1016/j.future.2019.11.008`.

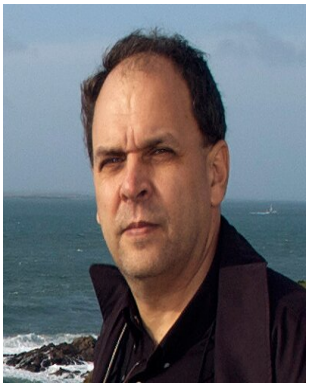
[43] H. Mannila, H. Toivonen, and A. I. Verkamo, "Discovery of frequent episodes in event sequences," *Data Mining and Knowledge Discovery*, vol. 1(3), pp. 259–289, 1997.

[44] N. Master, Z. Zhou, D. Miller, D. Scheinker, N. Bambos, and P. Glynn, "Improving predictions of pediatric surgical durations with supervised learning," *The International Journal of Data Science and Analytics*, vol. 4(1), pp. 35–52, 2017.

[45] N. Méger and C. Rigotti, "Constraint-based mining of episode rules and optimal window sizes," *The European Conference on Principles and Practice of Knowledge Discovery in Databases*, vol. 3202, pp. 313–324, 2004.
[46] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16(11), pp. 1424–1440, 2004.
[47] Y. J. M. Pokou, P. Fournier-Viger, and C. Moghrabi, "Authorship attribution using small sets of frequent part-of-speech skip-grams," *The International Florida Artificial Intelligence Research Society Conference*, pp. 86–91, 2016.
[48] I. Railean, P. Lenca, S. Moga, and M. Borda, "Closeness Preference - A new interestingness measure for sequential rules mining," *Knowledge-Based Systems*, vol. 44, pp. 48–56, 2013.
[49] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?: Explaining the predictions of any classifier," *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016.
[50] C. Rudin, B. Letham, A. Salleb-Aouissi, E. Kogan, and D. Madigan, "Sequential event prediction with association rules," *The Annual Conference on Learning Theory*, pp. 615–634, 2011.
[51] M. Rußwurm, S. Lefèvre, N. Courty, R. Emonet, M. Körner, and R. Tavenard, "End-to-end learning for early classification of time series," Preprint `http://arXiv.org/abs/1901.10681`, 2019
[52] T. Santos and R. Kern, "A literature survey of early time series classification and deep learning," *The International Workshop on Science, Application and Methods in Industry 4.0*, vol. 1793, 2016.
[53] J. Shao, J. Yin, W. Liu, and L. Cao, "Actionable combined high utility itemset mining," *AAAI Conference on Artificial Intelligence*, pp. 4206–4207, 2015.
[54] J. Soulas and P. Lenca, "Periodic episode discovery over event streams," *Portuguese Conference on Artificial Intelligence*, vol. 9273, pp. 547–559, 2015.
[55] J. Soulas, P. Lenca, and A. Thépaut, "Unsupervised discovery of activities of daily living characterized by their periodicity and variability," *Engineering Applications of Artificial Intelligence*, vol. 45, pp. 90–102, 2015.
[56] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," *The International Conference on Extending Database Technology*, pp. 1–17, 1996.
[57] S. Srinivasulu and P. Sakthivel, "Extracting spatial semantics in association rules for weather forecasting image," *Trends in Information Sciences & Computing*, pp. 54–57, 2010.
[58] N. Tatti and B. Cule, "Mining closed strict episodes," *Data Mining and Knowledge Discovery*, vol. 25(1), pp. 34–66, 2012.
[59] B. Vo, S. Pham, T. Le, and Z. H. Deng, "A novel approach for mining maximal frequent patterns," *Expert Systems with Applications*, vol. 73, pp. 178–186, 2017.
[60] D. S. Weld and G. Bansal, "The challenge of crafting intelligible intelligence," *Communications of the ACM*, vol. 62(6), pp. 70–79, 2019.
[61] J. M. T. Wu, J. C. W. Lin, and A. Tamrakar, 'High-utility itemset mining with effective pruning strategies," *ACM Transactions on Knowledge Discovery from Data*, vol. 13(6), Article No. 58, 2019.
[62] U. Yun and G. Lee, "Incremental mining of weighted maximal frequent itemsets from dynamic databases," *Expert Systems with Applications*, vol. 54, pp. 304–327, 2016.
[63] M. J. Zaki, "SPADE: an efficient algorithm for mining frequent sequences," *Machine learning*, vol. 42(1-2), pp. 31–60, 2001.
[64] J. Zhang, Y. Wang, and D. Yang, "CCSpan: Mining closed contiguous sequential patterns," *Knowledge-Based Systems*, vol. 89, pp. 1–13, 2015.
[65] W. Zhou, H. Liu, and H. Cheng, "Mining closed episodes from event sequences efficiently," *Advances in Knowledge Discovery and Data Mining*, pp. 310–318, 2010.
[66] H. Zhu, P. Wang, X. He, Y. Li, W. Wang, and B. Shi, "Efficient episode mining with minimal and non-overlapping occurrences," *IEEE International Conference on Data Mining*, pp. 1211–1216, 2010.
[67] S. Ziebarth, I. A. Chounta, and H. U. Hoppe, "Resource access patterns in exam preparation activities," *Design for Teaching and Learning in a Networked World*, pp. 497–502, 2015.

Lina Fahed is associate professor at ISEN, Artificial Intelligence and Emergent data department (Brest, France). She has received her PhD in Computer Science from Lorraine University, LORIA laboratory (France, 2016). Her research interests include data mining, pattern mining, predictive modeling, explainable artificial intelligence, complex data.

Philippe Lenca, PhD & HDR (Accreditation to Supervise Research), is Professor in Computer Science and head of the Department "Logics in Uses, Social Science and Information Science" at IMT Atlantique, France. He is a member of the Laboratory "Information and Communication Science and Technology" (UMR CNRS 6285 Lab-STICC). He created and led the "DECision aId and knowleDge discovEry" team from 2008 to 2015. He also created and co-chaired the Task Force "Evaluation and Quality" of the IEEE Computational Intelligence Society (Data Mining Technical Committee) from 2011 to 2016. Philippe Lenca's research interests include mainly data science, machine learning, data mining, decision aiding and artificial intelligence. He has a particular focus on interestingness measures, new algorithms for data mining, especially for rule-based patterns, and real-world applications of data science.



Yannis Haralambous, PhD, is Professor in Computer Science at the Computer Science Department of IMT Atlantique, France. He is a member of the Laboratory of "Information and Communication Science and Technology" (UMR CNRS 6285 Lab-STICC). He is the organizer and chair of the biennial interdisciplinary conference *Grapholinguistics in the 21st Century*. Yannis Haralambous's research interests include natural language processing, text mining, knowledge representation, grapholinguistics and digital typography. He is, or has been, member of the scientific committee of several conferences such as CICLing, ICALP, FedCSIS, TSD and CICM.

Riwal Lefort is data scientist and project manager at team IA-Factory at Crédit Mutuel ARKEA, France. He received his PhD in Computer Science From Telecom Bretagne, Bretagne Loire University (2010). His main research interests include machine learning, data analysis and modeling, and big data.