Efficient Discovery of Episode Rules with a Minimal Antecedent and a Distant Consequent

Lina Fahed(✉), Armelle Brun, and Anne Boyer

Lorraine University, LORIA - KIWI Team, Campus Scientifique, BP 239,
54506 Vandoeuvre-lès-Nancy Cedex, France
{lina.fahed,armelle.brun,anne.boyer}@loria.fr

Abstract. This paper focuses on event prediction in an event sequence, particularly on distant event prediction. We aim at mining episode rules with a consequent temporally distant from the antecedent and with a minimal antecedent. To reach this goal, we propose an algorithm that determines the consequent of an episode rule at an early stage in the mining process, and that applies a span constraint on the antecedent and a gap constraint between the antecedent and the consequent. This algorithm has a complexity lower than that of state of the art algorithms, as it is independent of the gap between the antecedent and the consequent. In addition, the determination of the consequent at an early stage allows to filter out many non relevant rules early in the process, which results in an additional significant decrease of the running time. A new confidence measure is proposed, the temporal confidence, which evaluates the confidence of a rule in relation to the predefined gap. The temporal confidence is used to mine rules with a consequent that occurs mainly at a given distance. The algorithm is evaluated on an event sequence of social networks messages. We show that our algorithm mines minimal rules with a distant consequent, while requiring a small computation time. We also show that these rules can be used to accurately predict distant events.

Keywords: Data mining · Episode rules mining · Minimal rules · Distant event prediction

1 Introduction

The flow of messages posted in blogs and social networks is an important and valuable source of information that can be analyzed, modeled (through the extraction of hidden relationships) and from which information can be predicted. This last aspect is the focus of our work. Prediction of information or events in a flow of messages is of the highest importance for companies, which may be interested in what will be said about them in social networks. Prediction can also be viewed as a way to recommend items, for example, when the flow is made up of items (customer consultations or purchases, etc.).

A. Fred et al. (Eds.): IC3K 2014, CCIS 553, pp. 3–18, 2015.
DOI: 10.1007/978-3-319-25840-9_1

We consider that the sooner an event is predicted, the more useful this prediction is. When the data is a flow of messages from blogs or social networks, predicting early a negative event (a criticism of a company) allows to have enough time to act before the occurrence of thus event, or to prevent its occurrence. Predicting distant events is the focus of our work.

Temporal data mining is related to the mining of sequential patterns ordered by a given criterion such as time or position [6]. Episode mining is the appropriate pattern discovery task related to the case the data is made up of a single long sequence. An episode is a temporal pattern made up of "relatively close" partially ordered items (or events), which often appears throughout the sequence or in a part of it [9]. When the order of items is total, the episode is said to be serial.

Similarly to the extraction of association rules from itemsets, episode rules can be extracted from episodes, and used to predict events [3]. The rule mining task, whether they are association or episode rules, is usually decomposed into two sub-problems. The first one is the discovery of frequent itemsets or episodes that have a support higher than a predefined threshold. The second one is the generation of rules from those frequent itemsets or episodes, with the constraint of a confidence threshold [2]. In general, rules are generated by considering some items in the itemsets (or the last items in the case of episodes) as the consequent of the rule, and the rest of the items as the antecedent. Since the second subproblem is quite straightforward, most of the researches focus on the first one: the extraction of itemsets or episodes. Episode and episode rules mining are used in many areas, such as telecommunication alarm management [9], intrusion detection [8], discovery of relation between financial events [12], etc.

Not only predicting distant events is more useful than predicting close events, but also predicting these events as soon as possible is of higher usefulness. Therefore, we aim at mining serial episode rules with a consequent distant from the antecedent. Due to complexity reasons, traditional episode rules mining algorithms form episode rules with a consequent close to the antecedent, through the use of a predefined span. To mine rules with a distant consequent, these algorithms have to use a larger span and perform a post-processing step: the rules are filtered to keep only rules with a consequent that may occur far from the antecedent. This post-processing is time consuming.

We propose a new algorithm that serial episode rules with a consequent temporally distant from the antecedent, and with a small antecedent (in number of events and in time), to be able to predict events as early as possible. This algorithm has a complexity lower than that of traditional algorithms.

An example of rules we focus on is presented below (from a sequence of annotated messages of blogs about finance issues, where each event includes a sentiment polarity): *R: (interest rate, neutral) (credit, negative) (waiting, neutral)* $\rightarrow$ *(concurrence, negative); the antecedent occurs within 5 days, the gap between the antecedent and the consequent is 15 days.*

The rest of this paper is organized as follows: Sect. 2 presents related works about episode rules mining. Our algorithm is introduced in Sect. 3, followed by experimental results in Sect. 4. We conclude and provide some perspectives in Sect. 5.

2 Related Works

We first start by introducing few concepts. Let $I = \{i_1, i_2, ..., i_m\}$ be a finite set of items. I_t is the set of items that occur at a timestamp t, referred to as an **event**. An **event sequence** S is an ordered list of events, $S =< (t_1, I_{t_1}), (t_2, I_{t_2}), ..., (t_n, I_{t_n}) >$ with $t_1 < t_2 < ... < t_n$, (see Fig. 1). The **serial episode** $P =< p_1, p_2, ..., p_k >$ on I^k is an ordered list of events. Its support, denoted by $supp(P)$, represents the number of occurrences of P, according to a frequency measure. P is said to be a frequent episode if $supp(P) \geq minsupp$ where $minsupp$ is the predefined minimal threshold. An **occurrence window** of the episode P is a segment $< I_{t_s}, ..., I_{t_e} >$ of the sequence, denoted as $OW(S, t_s, t_e)$ that starts at timestamp t_s and ends at timestamp t_e, where $P \subseteq < I_{t_s}, ..., I_{t_e} >$, $p_1 \subseteq I_{t_s}$ and $p_k \subseteq I_{t_e}$. It represents the interval that bounds the episode. Let P and Q be two episodes. An **episode rule** $R : P \rightarrow Q$ means that Q appears after P. The **confidence** of this episode rule is the probability to find Q after P: $conf\ (P \rightarrow Q) = supp(P{\cdot}Q)/supp(P)$. The rule is said to be confident if its confidence exceeds a predefined threshold $minconf$.

$Winepi$ and $Minepi$ are seminal episode mining algorithms [9], and are the basis of many recent algorithms [5,7]. To extract episodes, both algorithms start by extracting 1-tuple episodes (made up of one item/event), then iteratively seek larger ones by merging items/events on their right side.

In the frequent pattern mining task (which includes episode mining), anti-monotonicity is a common property that has to be respected by any frequency measure [2]. Several frequency measures for episode mining have been proposed. In [9], window-based and minimal occurrence-based frequency measures are introduced in both $Winepi$ and $Minepi$. $Winepi$ evaluates the frequency of an episode as the number of windows of length w that contain the episode. $Minepi$ evaluates the frequency of an episode as the number of minimal windows that contain the episode. A minimal window is a window such that no sub-window contains the episode. [7] has introduced the non-overlapped occurrence-based frequency measure. Two occurrences of an episode are non-overlapped if no item of one occurrence appears in between items of the other. It is shown that the non-overlapped occurrence-based algorithms are much more efficient in terms of space and time needed.

When mining serial episodes, additional constraints on the episodes may be imposed, mainly to reduce the complexity of the algorithms. The span constraint [1] imposes an upper bound (of distance or time) between the first and last event of the occurrence of an episode. The gap constraint [10] imposes an upper bound between successive events in the occurrence of an episode. If the extracted serial episodes have to represent causative chains, such constraints are important.

Traditional episode rules mining algorithms construct episode rules with a large antecedent (made up of many events) [13].

Discovering rules with a small antecedent was introduced for association rules, called "minimal association rules discovery" [14], where it is assumed that

no new knowledge is given by larger antecedents. This algorithm has the constraint that the consequents are fixed in advance by the user. Minimal rules have also been studied with the aim to reduce time and space complexity of the mining task, as well as to avoid redundancy in the resulting set of rules [11]. These works focus on association rules; recall we want to form episode rules.

Mining episodes in an event sequence is a task which has received much attention [4]. In an event sequence each data element may contain several items (an event). In [5], the algorithm *Emma* encodes the event sequence with frequent itemsets then serial episodes are mined. In [4], episodes are first extracted, then non-derivable episodes rules are formed (where no rule can be derived from another).

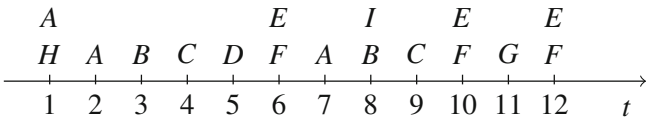

Fig. 1. Example of an event sequence S.

3 The Proposed Algorithm

3.1 Principle

Our goal is to mine episode rules that can be used to efficiently predict distant events. To achieve this goal, the episode rules formed have to hold the characteristic that the consequent is temporally distant from the antecedent. Traditional algorithms are not designed to form such rules. Recall that they first form episodes from left to right by iteratively appending events to the episode being formed. Due to the limited of the predefined span, these events are close to the episode. Second, the episode rules are built by considering the last element(s) of the episodes as consequent of the rules. When forming these episodes, it is impossible to know if the event being appended will be part of the consequent or not. So, it is impossible to know if a distance to other events has to be imposed, while forming the episode. The only way to mine rules with a consequent distant from the antecedent is to use a larger span, mine all rules and then filter out the occurrences that do not respect this distance. Both modifications make the algorithm more time consuming.

We propose to mine episode rules without any episode mining phase. To be able to constrain the distance between the antecedent and the consequent, the consequent is determined early in the mining process. We think that, by determining the consequent at an early stage, the occurrence windows of an episode rule will be filtered early, thus the search space will be pruned, and no post-processing is required. We also aim at predicting events as early as possible. We assume that the more the antecedent of a rule is small in number of events and in time, the earliest it ends, and the earliest the consequent can be

predicted. Therefore, we propose to extract episode rules that have an additional characteristic: an antecedent as small as possible (in number of events and in time), which we call "minimal episode rules". For that, we apply the traditional minimal occurrence-based frequency measure. Determining the consequence early in the mining process allows to get such rules, without relying on a post-processing phase. Indeed, knowing the consequence makes the confidence of a rule computable. The construction of the episode rule can be stopped as yet as the confidence threshold is reached.

Before presenting our algorithm in details, we introduce the new concepts of **Sub-windows of** $Win(S, t_s, w)$. Let $Win(S, t_s, w)$ be a window in the sequence S of length w that starts at t_s, with its first element containing the prefix of an episode rule (the first event) to be built. In order to mine episode rules with a distant consequent and a minimal antecedent, we split this window into three sub-windows as follows (see Fig. 2):

- Win_{begin} is a segment of $Win(S, t_s, w)$ of length $w_{begin} < w$, starting at t_s. Win_{begin} can be viewed as an expiry time for the antecedent of an episode rule. It represents the span of the antecedent of an episode rule to guarantee that the antecedent occurs within a determined time.
- Win_{end} is a segment of $Win(S, t_s, w)$ of length $w_{end} < w$, that ends at $t_s + w$. Win_{end} represents the time window of occurrence of the consequent.
- $Win_{between}$ is the remaining sub-window of length $w_{between}$, in which neither the antecedent nor the consequent can appear. $Win_{between}$ guarantees the temporal distance between the antecedent and the consequent of an episode rule. It represents a minimal gap between the antecedent and the consequent to guarantee that the consequent is far from the antecedent.

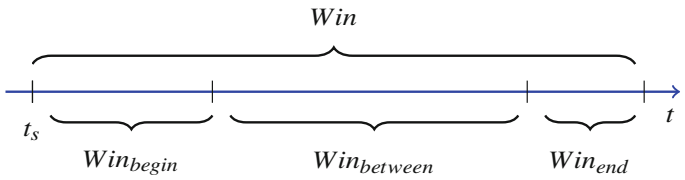

Fig. 2. Sub-windows of $Win(S, t_s, w)$.

3.2 Steps of the Algorithm

We present now the different steps of our algorithm.

Initialization. The algorithm we propose starts by an initialization phase, which reads the event sequence to extract all frequent events and their associated occurrence timestamps. An event represents a 1-tuple episode and will be denoted by P. Table 1 presents the list of 1-tuple episodes of the sequence S (Fig. 1) and their associated occurrence windows when $minsupp = 2$ (see Algorithm line 2).

Table 1. 1-tuple episodes of S.

1-tuple episode p	List of occurrence windows
A	[1,1], [2,2], [7,7]
B	[3,3], [8,8]
C	[4,4], [9,9]
E	[6,6], [10,10], [12,12]
F	[6,6], [10,10], [12,12]
EF	[6,6], [10,10], [12,12]

Prefix Identification. Episode rules are built iteratively by first fixing the prefix (the first event of the antecedent). The antecedent is denoted by *ant*. Each 1-tuple episode p is viewed as a prefix of the antecedent of an episode rule R to be built. Once the prefix R is fixed, the set of its occurrence windows $OW(S, t_s, t_e)$ is known. For example, let $minsupp = 2$, A can be considered as a prefix of an episode rule. The list of occurrence windows of $A = ([1, 1], [2, 2], [7, 7])$ (see Table 1).

Consequent Identification. A candidate consequent of an antecedent *ant* (at this step of the algorithm, *ant* corresponds to a single element, the prefix) is chosen in the windows $Win(S, t_s, w)$ where $[t_s, t_s]$ is in the set of occurrence windows of *ant*. Recall we want to form episode rules with a consequent distant from the antecedent. Thus, the candidate consequents are not searched in the entire windows of size w, they are searched only in the last part of these windows: Win_{end}, where the farthest candidates are. We construct $\mathcal{P}_{end}(ant)$, the ordered list of 1-tuple episodes that occur frequently in Win_{end}.

Let $p_j \in \mathcal{P}_{end}(ant)$ be a candidate consequent of *ant*, forming the candidate episode rule $R : ant \rightarrow p_j$. Its set of occurrence windows is formed. The frequency of this rule is computed as the number of minimal occurrence windows. For example, let $w_{begin} = 2$, $w_{end} = 2$ and $w = 6$. The episode rule $R : A \rightarrow E$ has three occurrence windows: $([1, 6], [2, 6], [7, 12])$. The first occurrence window $[1, 6]$ is not minimal. So, the support of $R : A \rightarrow E$ is 2. Notice that all occurrence windows are kept in memory, they will be used to complete the antecedent. This allows to guarantee to discover all interesting episode rules, which could be missed if we kept only the minimal occurrences in memory.

If $R : ant \rightarrow p_j$ is not frequent, p_j cannot be a consequent of *ant*. This iteration is stopped and the rule is discarded. There is no need to complete *ant* as whatever are the events that may be appended to the antecedent, the resulting rule will not be frequent. The algorithm will iterate on another consequent. If $R : ant \rightarrow p_j$ is frequent, its confidence is computed.

If the rule $R : ant \rightarrow p_j$ is confident, this rule is added to the set of rules formed by the algorithm. It is minimal and has a consequent far from

the antecedent; it fulfills our goal. If the rule is frequent but not confident, the antecedent of the rule $R : ant \rightarrow p_j$ is further completed (see next subsection).

For example, let $w = 6$, $w_{begin} = 2$ and $w_{end} = 2$. For the episode rule R with prefix A, $\mathcal{P}_{end}(A) = [E, F, EF, A]$. We first construct the episode rule R with the consequent E. Thus, for $R : A \rightarrow E$, $supp(R) = 2$ and $conf(R) = 2/3 = 0.67$. For $minsupp = 2$ and $minconf = 0.7$, R is frequent but not confident, so its antecedent has to be completed. If $minconf = 0.5$, $R : A \rightarrow E$ is confident, its construction is stopped; it is a frequent, confident and minimal episode rule.

Antecedent Completion. In this step, the antecedent ant is iteratively completed with 1-tuple episodes from the initialization step, placed on its right side in the limit of the predefined sub-window Win_{begin}. At the first iteration, ant is a unique element (the prefix) (see Algorithm 1, line 15). Recall that we aim at forming rules having the last event of the antecedent as far as possible from the consequent, so as close as possible of the prefix. We construct $\mathcal{P}_{begin}(ant)$, the ordered list of 1-tuple episodes that occur frequently after ant in the windows of R, in Win_{begin}. The new candidate rule $R : ant, p_i \rightarrow p_j$ is formed. Similarly to the consequent identification step, the occurrence windows of R are computed, as well as its support. We apply the same support and confidence verifications, similarly to the previous step.

To speed up the episode rules mining process we use a heuristic. We sort the list of candidates $\mathcal{P}_{begin}(ant)$ in descending order of the number of windows in which each candidate appears. We assume that this number is highly correlated with the support of the corresponding episode rules. So, in the traversal of this list, when we observe that candidates tend to form infrequent episode rules (several consecutive 1-tuple episodes lead to infrequent episode rules), we stop the traversal. We consider that the remaining candidates in this list will lead to infrequent episode rules. This heuristic is used to reduce the number of iterations. Although it may discard interesting rules, it allows to reduce the number of iterations thus allows to increase the size of the span of the rule (Win).

For example, let $minsupp = 2$ and $minconf = 0.7$, for $R : A \rightarrow E$, $\mathcal{P}_{begin}(A) = [B, C, A]$. The antecedent of R is completed with B and forms the episode rule $R : A, B \rightarrow E$. Thus, $supp(R) = 2$ and $conf(R) = 2/2 = 1$. The episode rule R is now confident, the phase of completing its antecedent is stopped.

3.3 Temporal Confidence

$Win_{between}$ has been introduced to guarantee that the consequent of an episode rule occurs in Win_{end}, after at least $w_{begin} + w_{between}$ events from the prefix of the episode rule. However, the consequent of a rule may also occur closer to the antecedent, in the window $Win_{between}$, which should affect the confidence of the rule. This information may be important in some applications. In the previous example about social networks, where the prediction of a negative event allows the company to prevent its occurrence, it is important to mine rules with

a consequent that never occurs in $Win_{between}$. Indeed, predicting a consequent at a given distance, which may appear closer is useless, even dangerous. Consequently, the occurrence of the consequent in $Win_{between}$ has to be considered. We introduce a new confidence measure, the temporal confidence, which represents the probability that the consequent occurs in Win_{end} and only in it. For an episode rule $R : P \rightarrow Q$, the temporal confidence $conf_t(R)$ is equal to the ratio between the support of $P \cdot Q$ when Q occurs only in Win_{end} (and not in $Win_{between}$) and the support of $P \cdot Q$. $conf_t(R) = 1$ if no occurrence of the consequent is found in $Win_{between}$. The rules with a temporal confidence above $minconf_t$ are kept. For example, let $w = 6$, $w_{begin} = 2$ and $w_{end} = 2$, the temporal confidence of the frequent confident episode rule $R : A \rightarrow E$ depends on the occurrences of E in $Win_{between}$ which is equal to 1 (E appears in the timestamp t_{10} in $Win_{between}$). Thus, $conf_t(R : A \rightarrow E) = 1/2 = 0.5$. For $minconf_t = 0.5$, R is temporally confident and is a rule formed by our algorithm.

Algorithm 1. Episode rules mining

input : S: event sequence, *minsupp*, *minconf*, $minconf_t$, w, w_{begin}, $w_{between}$, w_{end}
output: ER : List of episode rules

```
1  Procedure Episode rules mining
2      extract frequent 1-tuple episodes;
3      foreach p_i ∈ 1-tuple episodes do
4          ant ← p_i ;
5          Construct ordered lists
           P_end(ant), P_begin(ant) ;
6          Consequent (ant, P_end(ant), P_begin(ant))
7  Procedure Consequent (ant, P_end(ant), P_begin(ant))
8      foreach p_j ∈ P_end(ant) do
9          if ant → p_j is frequent then
10             if ant → p_j is confident then
11                 if ant → p_j is temporally confident
                   then
12                     Add ant → p_j to ER
13             else
14                 Antecedent (ant, p_j, P_begin(ant))
15 Procedure Antecedent (ant, p_j, P_begin(ant))
16     foreach p_k ∈ P_begin(ant) do
17         if ant, p_k → p_j is frequent then
18             if ant, p_k → p_j is confident then
19                 if ant, p_k → p_j is temporally
                   confident then
20                     Add ant, p_k → p_j to ER
21             else
22                 ant ←< ant, p_k > ;
23                 Antecedent (ant, p_j, P_begin(ant))
```

4 Experimental Results

We now evaluate the algorithm through the study of the characteristics of the episode rules formed, as well as its performance in a prediction task and its running time in comparison to traditional algorithm.

4.1 Dataset

The dataset we use is made up of 27,612 messages extracted from blogs about finance. Messages are annotated using the *Temis*[1] software. Each message is represented by its corresponding set of annotations (items). For example, the message: *"Not only my bank propose the best, but also online banks, so how to optimize my savings? accounts in other banks? life insurance? i need more info."* is annotated with the following items, each one being associated with an opinion degree: {*(online banks, negative), (savings, neutral), (life insurance, neutral), (needs, neutral)*} .

In this dataset, the messages are annotated with 4.8 items on average, ranging from 1 to 50 and the median is equal to 4. There are about 4,000 distinct annotations (items), with an average frequency of 88.5. 1,981 of these items have a frequency equal to 1. These items will be automatically filtered out in the initialization phase of our algorithm.

4.2 Characteristics of the Resulting Rules

Initialization Phase. In this phase, frequent 1-tuple episodes are extracted (a 1-tuple episode is made up of one or more items). We fix *minsupp* to 30. This phase results in 652 frequent 1-tuple episodes. Table 2 shows that the 1-tuple episodes are made up of one to three items only. They are mainly made up of one item (76 % of them), which also have a high support (on average 149.7).

Table 2. 1-tuple episodes : length and support.

1-tuple episode length (#items)	Number(%)	Support			
		Min	Max	Mean	Median
1	498(76.4)	30	797	149.7	89.5
2	147(22,5)	30	376	71.5	55
3	7(1)	32	54	44.4	46

In order to study the episode rules formed, we make vary *minsupp*, $minconf$ and $w_{between}$ one at a time, while fixing others.

Making Vary *minsupp***.** We make vary *minsupp* from 10 to 35 to study the number of rules, represented in Fig. 3. Other parameters remain fixed: $minconf = 0.4$, $w = 40$ (with $w_{begin} = 20$, $w_{between} = 10$ and $w_{end} = 10$). As expected, the smaller *minsupp*, the higher the number of rules. When *minsupp* is fixed to 10, the number of rules is high (about 10^4). Recall that the number of 1-tuple episodes is only 652. However, the number of rules is dramatically decreased when *minsupp* is increased: only 1,200 when $minsupp = 15$ and 250 when $minsupp = 20$. These low values are due to the small average frequency of 1-tuple episodes (about 88).

[1] http://www.temis.com.

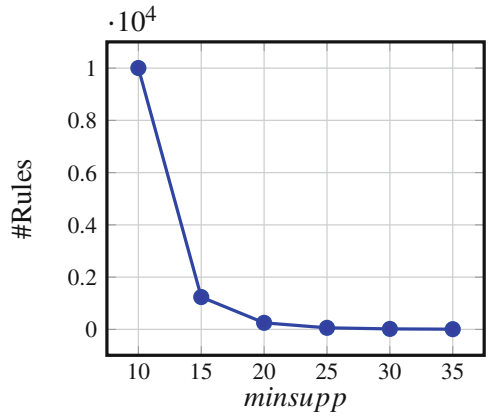

Fig. 3. Number of rules (10^4) vs. $minsupp$.

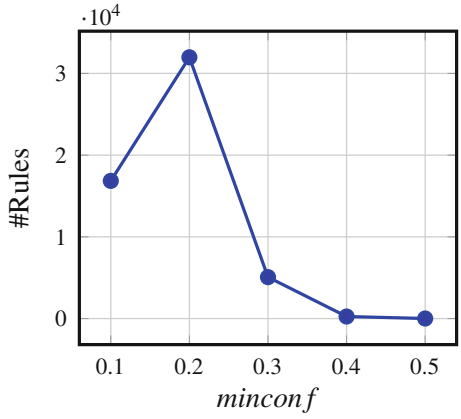

Fig. 4. Number of rules (10^4) vs. $minconf$.

In addition, these rules represent a temporal dependence between the antecedent and the consequent which, in these experiments, is at least $w_{between} = 10$. This may explain the low number of rules. A thorough study shows that their average confidence increases with $minsupp$.

Making Vary $minconf$**.** We make vary $minconf$ from 0.1 to 0.5. Figure 4 presents the number of episode rules according to the value of $minconf$ ($minsupp = 20$, $w = 40$ ($w_{begin} = 20$, $w_{between} = 10$ and $w_{end} = 10$)). The number of rules is particularly high with $minconf = 0.2$. This is explained by the way our rules are formed: when the confidence of a rule does not exceed $minconf$, its antecedent is extended. Given an antecedent ant, events in $\mathcal{P}_{begin}(ant)$ (up to 652) are appended to it, resulting in a large number of candidate rules. Some of them are confident, which explains the increase of the number of rules.

Table 3 presents the length of the antecedent of the rules (in number of events), according to $minconf$. The maximum length of an antecedent is three. Thus, our algorithm, forms rules with a small antecedent, which was one of its goals. The average length of the antecedent increases with $minconf$: when $minconf = 0.1$, most of the rules have an antecedent of length 1, whereas when $minconf = 0.3$, most of the rules have an antecedent of length 2. This was expected, as minimal antecedents are searched. Indeed, when a frequent rule has a confidence below $minconf$, its antecedent is extended, till it is confident or not frequent. So, the higher $minconf$, the larger the antecedents. A thorough study shows that the average length of occurrence windows of the antecedents is 8 timestamps (for antecedents of length 2 and 3), which is smaller than the span of the antecedent (w_{begin}). We conclude that our algorithm succeeds in forming rules with a distant consequent, a small antecedent (in length and time) and a relatively high confidence.

Making Vary $w_{between}$**.** We now focus on the number of rules formed, according to $w_{between}$ (w and w_{begin} remain fixed), presented in Fig. 5 where $minsupp = 20$ and $minconf = 0.4$. Two values of w are studied: $w = 40$ (with $w_{begin} = 20$)

Table 3. Antecedent length when making vary $minconf$.

$minconf$	#Rules	%Rules		
		Ant 1	Ant 2	Ant 3
0.1	16,850	57.2	42.84	0
0.2	31,972	4.2	95.6	0.2
0.3	5,072	0.9	97.5	1.6
0.4	251	0.8	90.9	8.3
0.5	9	0	100	0

and $w = 100$ ($w_{begin} = 20$). Notice that the cases $w_{between} = 0$ represent similar cases than state of the art. We note that the larger $w_{between}$, the smaller the number of rules. Two reasons may explain this decrease. First, when $w_{between}$ increases, w_{end} (the window in which the consequent is searched) decreases, as well as the number of consequents studied. Second, the larger $w_{between}$, the more distant the consequent, thus the lower the probability of having a dependence between the antecedent and the consequent. However, even with a large value of $w_{between}$, some rules are formed: 210 rules when $w_{between} = 70$. We conclude that there is actually a temporal dependence between messages in blogs. When the minimal distance between the antecedent and the consequence is 50, more than 140k confident rules are formed: there is a strong dependence between messages with such a distance. When $w = 100$, an episode rule: *(price, positive), (information, positive)* $\rightarrow$ *(buy, positive)*, means that when someone talks about the price of an article, then asks for information, he/she will buy this article after some time. Thus, we have time to recommend him similar articles or to propose to him a credit to buy it. **Influence of $minconf_t$:** In this section we study the temporal confidence of the resulting rules. Table 4 presents the evolution of the temporal confidence according to $w_{between}$. We remark that the smaller $w_{between}$, the higher the temporal confidence: the consequent does rarely occur between the antecedent and the consequent when the gap between them is small, which was expected. When $w_{between} = 30$, the average temporal confidence is 0.6, which is quite high. A thorough study shows that among $2.7 \cdot 10^5$ rules (see Fig. 5), about 40 ones have a temporal confidence equal to 1 (the consequent never occurs in $Win_{between}$) and 1,400 rules have a temporal confidence higher than 0.9 (the consequent appears in $Win_{between}$ in less than 10 % of the cases). This shows that in this dataset there is a strong temporal dependence between events, and that some events are interdependent at a distance of 30. So, when exploiting the temporal confidence as a filter, a great number of rules remain.

As mentioned before, one may be interested in the extraction of rules with a consequent that occurs in Win_{end} most of the time (with a high temporal confidence). Therefore, a minimal threshold ($minconf_t$) is fixed to keep only rules having a temporal confidence that exceeds this threshold. In Fig. 6, we make vary $minconf_t$ when varying $w_{between}$ with the same parameters as in Fig. 5. It is intuitive that the higher $minconf_t$, the lower the number of rules. We remark

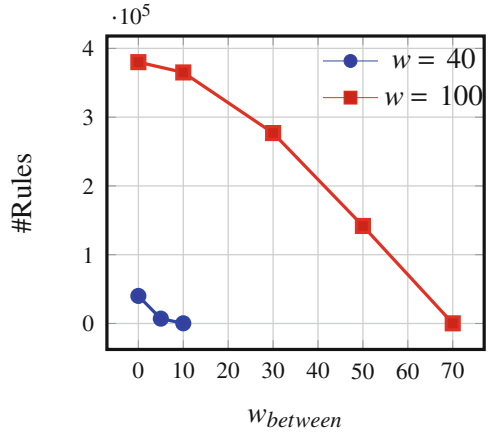

Fig. 5. Number of rules (10^5) vs. $w_{between}$.

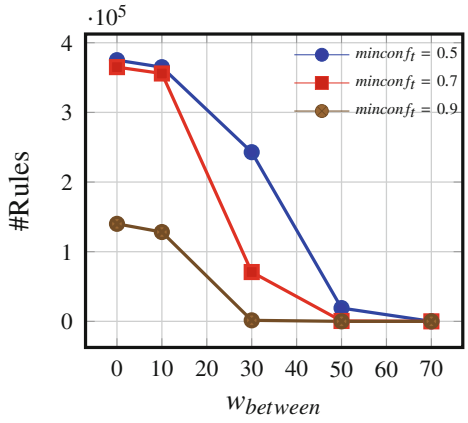

Fig. 6. Number of rules (10^5) vs. $w_{between}$, $minconf_t$.

also that the larger $Win_{between}$, the lower the number of temporally confident rules. This decrease is intuitive since the farthest the consequents are searched (for large $Win_{between}$), the lower the dependence between events. Despite this strong filter of rules, a considerable number of temporally confident rules is extracted. For example, when $w_{between} = 30$ (which represents a quite large distance), about 1,400 rules are kept when $minconf_t = 0.9$. 70,800 rules are kept when $minconf_t = 0.7$ and about 242,900 rules when $minconf_t = 0.5$, which is quite high. Once again, this confirms the existence of a strong temporal dependence between events in these experiments.

Table 4. $w_{between}$ vs. Temporal confidence ($conf_t$).

$w_{between}$	min-$conf_t$	max-$conf_t$	mean-$conf_t$	median-$conf_t$
70	0	0.5	0.2	0.2
50	0.2	0.9	0.4	0.4
30	0.3	1	0.6	0.6
10	0.2	1	0.9	0.9

4.3 Performance

In this section we focus on the accuracy of the rules formed, when they are used to predict events. we also perform a comparison of these rules with those of a traditional algorithm.

We evaluate the accuracy of our algorithm with the traditional recall and precision measures. The episode rules are trained on the first 75 % messages (in the temporal order of publication) and are tested on the 25 % of messages left. Figure 7 presents the resulting precision and recall at 20. We fix $minsupp = 20$,

$minconf = 0.4$, $w = 100$ and $w_{begin} = 20$. We make vary $w_{between}$ from 10 to 70. First of all, mention that two precision and recall values with two different values of $w_{between}$ are not directly comparable as they are not computed on the same data (the windows Win_{end}, on which they are computed vary in size). Both precision and recall curves decrease as $w_{between}$ increases. This was expected as the number of rules decreases. When $w_{between} = 70$ (and $w_{end} = 10$), both precision and recall values are quite low. This was expected as the rules aim at predicting events distant to at least 70, in an occurrence window of length w_{end}=10. The prior probability of predicting events accurately is low. Let us now consider $w_{between} = 30$, as in the previous section. We can see that both precision and recall values are quite high. When an event is predicted, in 37 % of the cases, it actually occurs and events that occur in the sequence are predicted by our rules in 70 % of the cases.

Comparison of the Rules Formed with Traditional Algorithms. Contrary to traditional algorithms, our algorithm forces a minimum gap of length $w_{between}$ between the antecedent and the consequent of an episode rule. Our algorithm can be compared to traditional algorithms when $w_{between} = 0$. Since we apply the minimal occurrence-based frequency, we choose to compare it to the well-known $Minepi$ [9].

Table 5 compares the number of rules formed by our algorithm and by $Minepi$, for $minsupp = 20$, $minconf = 0.4$, $w = 40$, $w_{begin} = 20$ $w_{end} = 20$ and $w_{between} = 0$ (the only parameter used by $Minepi$ is $w = 40$).

$Minepi$ forms more than 136,000 episode rules, whereas our algorithm (when $w_{between} = 0$) extracts about 40,000 episode rules (70 % less). This decrease is due to two reasons. First, the constraint about the position of consequent of the episode rules from our algorithm (in this case the distance between the antecedent and the consequent is at least 20, even if $w_{between} = 0$), makes the number of rules resulting from our algorithm lower (also their support is lower). Second, our algorithm aims at forming minimal rules, thus few rules with a large antecedent are formed. We remark that 25 % of the rules extracted by $Minepi$ have an antecedent larger or equal to 3, whereas this rate is only 1.8 % for our algorithm.

Here is an example of an episode rule extracted by both our algorithm ($w_{between} = 0$) and by $Minepi$: *(credit, positive), (consultant, positive)* $\rightarrow$ *(loan subscription, positive).*

Here is a rule that has not been extracted by our algorithm, as it does not satisfy the desired characteristics of episode rule (minimal antecedent): *(consultant, neutral), (interest rate, positive)* $\rightarrow$ *(request interest rate 0, positive), where the antecedent occurs in 5 timetstamps and consequent occurs in the* 7^{th} *timestamp.* This rule is useful in traditional cases of event prediction (prediction of close events). However, it does not fit our objective of early prediction of distant events, as the antecedent is so long both in time and in number and the consequent is too close to the antecedent.

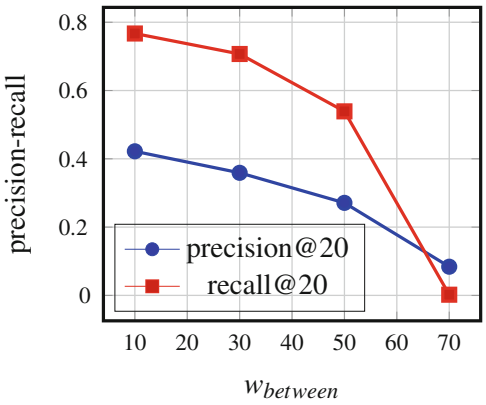

Fig. 7. Precision, recall vs. $w_{between}$.

Table 5. *Our algorithm* vs. *Minepi*.

Algorithm	#Rules	#R/ant 1	#R/ant 2	#R/ant 3	#R/ant 4
This paper	40,061	578	38,742	741	0
Minepi	136,225	9,300	93,389	33,505	31

4.4 Running Time

In this last section, we are interested in the running time of our algorithm according to the minimal gap between the antecedent and the consequent ($w_{between}$), in comparison to the running time of *Minepi*. We consider the two previously studied values of w: 40 and 100. The running time is presented in Fig. 8, in terms of relative running time compared to that of *Minepi*. First, we can see that our algorithm runs faster than *Minepi*. The most comparable points are those corresponding to $w_{between} = 0$: no minimal gap between the antecedent and the consequent, where our algorithm runs 5 times faster than *Minepi* when $w = 100$, and 4 times faster when $w = 40$. This decrease is due to the same two factors than presented below. The first one is related to the consequent, which is fixed at an early stage of the algorithm and which allows to filter infrequent rules. The second one is due to the fact that the proposed algorithm mines rules with a minimal antecedent, which avoids some iterations once a confident rule is found.

We can also see that the larger $w_{between}$, the faster the algorithm. This trend is intuitive since large $w_{between}$ values imply small w_{end} values (as w remains fixed), so a smaller number of consequent are studied in Win_{end} for each potential rule. For example, when $w = 100$, the running time is divided by almost 2 when $w_{between}$ ranges from 10 to 50.

Last, when comparing running times when $w = 40$ and $w = 100$, in comparable configurations: $w_{begin} = 20$ and $w_{end} = 10$ ($w_{between}$ is equal to respectively 10 and 70), we remark that they have nearly the same running time (these figures are not presented in Fig. 8 as it shows relative time compared to *Minepi*). This confirms that the running time of our proposed algorithm is independent of w.

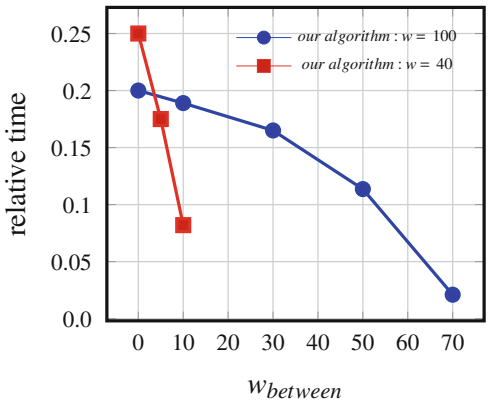

Fig. 8. Relative time according to $w_{between}$ ($relative\ time = 1$ represents running time of $Minepi$)

It only dependents on w_{begin} and w_{end}, in which antecedents and consequents are searched.

5 Conclusion

In this paper, we have proposed an algorithm that mines episode rules, in order to predict distant events. To achieve our goal, the algorithm mines serial episode rules with a distant consequent. We determine two main characteristics of the episode rules formed: minimal antecedent and a consequent temporally distant from the antecedent. A new confidence measure, the temporal confidence, is proposed to evaluate the confidence on distant consequents. Our algorithm is evaluated on an event sequence of annotated social networks messages. We show that our algorithm is efficient in extracting episode rules with the desired characteristics and in predicting distant events.

Since we use data from social networks, we aim to use multi-thread sequences. This means that we intend to construct a sequence for each thread of messages: user messages thread, topic messages thread and discussion thread, etc. and the algorithm is run on each one. Using multi-thread sequences allows to build more diverse episode rules which are all together more significant. The presence of a rule in several threads will increase its confidence.

Acknowledgements. This research is supported by Crédit Agricole S.A.

References

1. Achar, A., Sastry, P., et al.: Pattern-growth based frequent serial episode discovery. Data Knowl. Eng. **87**, 91–108 (2013)
2. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD Record, vol. 22, pp. 207–216. ACM (1993)

3. Cho, C.W., Wu, Y.H., Yen, S.J., Zheng, Y., Chen, A.L.: On-line rule matching for event prediction. VLDB J. **20**(3), 303–334 (2011)
4. Gan, M., Dai, H.: Fast mining of non-derivable episode rules in complex sequences. In: Torra, V., Narakawa, Y., Yin, J., Long, J. (eds.) MDAI 2011. LNCS, vol. 6820, pp. 67–78. Springer, Heidelberg (2011)
5. Huang, K.Y., Chang, C.H.: Efficient mining of frequent episodes from complex sequences. Inf. Syst. **33**(1), 96–114 (2008)
6. Laxman, S., Sastry, P.S.: A survey of temporal data mining. Sadhana **31**(2), 173–198 (2006)
7. Laxman, S., Sastry, P., Unnikrishnan, K.: A fast algorithm for finding frequent episodes in event streams. In: 13th ACM SIGKDD. ACM (2007)
8. Luo, J., Bridges, S.M.: Mining fuzzy association rules and fuzzy frequency episodes for intrusion detection. Int. J. Intell. Syst. **15**(8), 687–703 (2000)
9. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Min. Knowl. Discov. **1**(3), 259–289 (1997)
10. Méger, N., Rigotti, C.: Constraint-based mining of episode rules and optimal window sizes. In: Boulicaut, J.-F., Esposito, F., Giannotti, F., Pedreschi, D. (eds.) PKDD 2004. LNCS (LNAI), vol. 3202, pp. 313–324. Springer, Heidelberg (2004)
11. Neeraj, S., Swati, L.S.: Overview of non-redundant association rule mining. Res. J. Recent Sci. **1**(2), 108–112 (2012). ISSN 2277-2502
12. Ng, A., Fu, A.W.: Mining frequent episodes for relating financial events and stock trends. In: Whang, K.-Y., Jeon, J., Shim, K., Srivastava, J. (eds.) PAKDD 2003. LNCS, vol. 2637, pp. 27–39. Springer, Heidelberg (2003)
13. Pasquier, N., Bastide, Y., Taouil, R., Lakhal, L.: Discovering frequent closed itemsets for association rules. In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 398–416. Springer, Heidelberg (1998)
14. Rahal, I., Ren, D., Wu, W., Perrizo, W.: Mining confident minimal rules with fixed-consequents. In: 16th IEEE ICTAI 2004 (2004)