JMLR: Workshop and Conference Proceedings 25:159–174, 2012 Asian Conference on Machine Learning

Learning Temporal Association Rules on Symbolic Time Sequences

Mathieu Guillame-Bert MATHIEU.GUILLAME-BERT@INRIA.FR
James L. Crowley JAMES.CROWLEY@INRIA.FR
INRIA Rhône-Alpes Research Center

Editor: Steven C.H. Hoi and Wray Buntine

Abstract

We introduce a temporal pattern model called Temporal Interval Tree Association Rules (Tita rules or Titar). This pattern model can express both uncertainty and temporal inaccuracy of temporal events. Among other things, Tita rules can express the usual time point operators, synchronicity, order, and chaining, as well as temporal negation and disjunctive temporal constraints. Using this representation, we present the Titar learner algorithm that can be used to extract Tita rules from large datasets expressed as Symbolic Time Sequences. The selection of temporal constraints (or time-frames) is at the core of the temporal learning. Our learning algorithm is based on two novel approaches for this problem. This first one is designed to select temporal constraints for the head of temporal association rules. The second selects temporal constraints for the body of such rules. We discuss the evaluation of probabilistic temporal association rules, evaluate our technique with two experiments, introduce a metric to evaluate sets of temporal rules, compare the results with two other approaches and discuss the results.

1. Introduction

Data mining of Temporal Association Rules has been successfully applied in numerous domains including medical, trading, robotics, social analysis, fraud detection, marketing and assisted design (Guimares, 2000; Keogh and Smyth, 1997). Because of the variety of domains and types of applications, several temporal data models have proven to be useful (numeric time series, symbolic time series, symbolic time sequences, symbolic interval series, item set sequences, etc.)(Roddick and Spiliopoulou, 2002; Antunes and Oliveira, 2001; Mörchen, 2007). Similarly, different types of models have been studied (Episodes, sequential rules, first order logic based rules, chronicles, etc.). For each of these data and pattern models (or subsets of pattern models), efficient learning algorithms have been proposed.

In this paper we are using the data model called Symbolic Time Sequence (multi set of time points associated with a symbol) to represent time. Temporal Interval Tree Association Rules (Tita rules) are a rule model that express both uncertainty and temporal inaccuracy of temporal events. Tita rules can also express the usual time point operators, synchronicity, order, chaining and temporal negation. The Titarl algorithm (Temporal Interval Tree Association Rule Learner) is a temporal learning algorithm able to efficiently extract Tita rules from symbolic time sequences. This paper presents an improved version of the temporal learning algorithm called Titarl (Temporal Interval Tree Association

Rule Learner) and introduced in the short paper (Guillame-Bert and Crowley, 2011). The main improvements of this new version of Titarl are the exploration and the suppression of duplicates in the division of rules' heads, the smoothing of the temporal constraints, the sampling of the input dataset instead of the complete scanning, and several implementation improvements which are not presented in this article.

Most of the algorithms that learn temporal rules begin with the learning of *frequent patterns*, and then, they 'convert' these patterns into association rules. We can show that a frequent pattern is not guaranteed to give a good rule, especially in the case of datasets with *rare events* (or unbalanced dataset). Therefore, we design our algorithm to directly learn temporal rules without going through a generation of frequent patterns.

Most temporal patterns can be seen as sets of temporal constraints between events. The core of learning temporal pattern is the learning of these temporal constraints. For the large majority of temporal patterns, these constraints are binary constraints (constraints over two events). Also, the constraints of a majority of temporal patterns can be expressed as a subset of $\mathbb{R}$. A constraint $c \subset \mathbb{R}$ over two time points t_1 and t_2 holds if and only if $(t_1 - t_2) \in c$. Different algorithms allow different degree of freedom on the constraints, and different solutions to learn them (we call this problem the *Temporal Constraint Selection Problem*). Most of the techniques found in the literature ask for the user to specify this interval as an input parameter. Our approach does not need for the interval constraints to be fixed.

The Titarl algorithm is evaluated on both real world and computer generated datasets. The real world dataset evaluation supports the concrete usefulness of the technique while the evaluation on the computer generated dataset allows precise understanding of the algorithm on various and controlled situations. In addition, the computer generated dataset contains a 'ground truth'. This point is discussed in the evaluation section. The computer generated dataset is available online (Guillame-Bert, 2011).

A significant part of temporal learning algorithms are extracting sets of patterns. The evaluation of the 'value' of a set of patterns is a complex problem. We propose a new tool called Global Support Map (GSM) to evaluate such 'value'.

The next section is a quick introspection in the related literature. The third section defines the Titar representation. Simple examples are presented. The fourth section gives an overview of the Titarl algorithms and explains the main improvements. The fifth section shows and discusses the results of the application of the algorithm on several test cases. The results of two other algorithms are presented for comparison purpose. The last section discusses several aspects of the algorithms.

2. Related work

A simple and well studied temporal data model is the symbolic time series. This problem, also known as *sequential patterns mining* problem, is to extract sequences of events e.g. '$ab(cd)e$'. In such sequences, the relations between the events are limited to be 'before' or 'unconstrained'. The first problem has been fairly well studied, and powerful solutions are existing (Ayres et al., 2002; Zaki, 1998; Pei et al., 2001; Zhao and Bhowmick, 2003).

Symbolic time sequence is an another type of data model where events are 'time sampled'. Learning on time sequences (which is different from *time sequences learning/mining*)

extends *sequential patterns mining* with a much richer grammar for temporal pattern e.g. the relations between the events of a pattern is not limited to be 'before' or 'unconstrained'. In this problem, the challenged is not only to select the events of the patterns, but also to extract the relations (or constraints) between them.

A *Temporal Constraint System* (TCS) as defined by Balaban et al. (Balaban and Rosen, 1999) is a collection of existing conditions (of events) and a set of binary constraints on these events. Binary constraint systems can be represented by directed graphs where each node is labelled with a symbol (i.e. existing condition of an event with such symbol), and each edge is labelled with a temporal constraint. The simplest binary constraints are *before*, *equals* and *after*. All temporal pattern presented in this paper can be reduced to a TCS.

Chen et al. (Chen et al., 2003) extend the sequential patterns used in symbolic time series mining to the mining of symbolic time sequence. Symbols expressing laps of time are introduced e.g. the time sequence $< A, T_{2,5}, B >$ describes A followed, between 2 and 5 time units, by B. The two classical sequential patterns mining algorithms Aprori and PrefixSpan are extended to this grammar. Hirate (Hirate and Yamana, 2006) extends the work of Chen et al. to the case of sequential patterns with partial orders.

WinEpi is a well known algorithm developed by Mannila et al. (Mannila et al., 1997) to learn Episodes and association rules (based on the episodes). Basically, WinEpi discretizes a symbolic time sequence into a transaction dataset with a fixed window length (parameter of the algorithm), and apply the Apriori algorithm. A Mannila et al.'s episode is defined as a partially ordered collection of events such that, the distance between any two events is bounded. This maximum distance is called the 'window size'.

The *MinEpi algorithm* is an improvement of the WinEpi algorithm proposed by Mannila et al. (Mannila et al., 1997). Unlike WinEpi, it is not based on a sliding window. MinEpi's grammar is richer than the one used by Winepi (MinEpi's rules allows two 'window sizes'), and MinEpi does not need to go through the dataset several times. However, MinEpi is more space consuming than WinEpi.

The *Face algorithm* has been developed by Dousson et al. (Dousson and Duong, 1999) to extract Chronicles from symbolic time sequences. A Chronicle is defined as a collection of events such that, the temporal distance between any two events e_i and e_j is restricted to be in a subsets $c_{i,j} \subset \mathbb{R}$. Chronicles are particularly interesting because they are more expressive than Episodes (which is the commonly used pattern model). This algorithm has the particularity of not needing the user to specify (as a parameter) the interval constraints.

The *EpiBF algorithm* has been developed by Garriga (Casas-Garriga, 2003). This algorithm is inspired from WinEpi and MinEpi. One of the main differences between EpiBF and WinEpi/MinEpi is that, EpiBF uses local constraints between couples of elements (called 'time unit separation') instead of having a global window constraint between all the elements;

Pablo Hernandez and his colleagues (Hernandez-Leal et al., 2011) are developing an algorithm to learn Temporal Nodes Bayesian Networks (TNBNs). TNBNs are relatively limited to express temporal pattern with more than two events, but the algorithm is an interesting solution to select the temporal intervals (Gaussian mixture models).

The core of temporal patterns presented in this section is the set of temporal constraints between the events. We call this problem the *Temporal Constraint Selection problem*. Our main critic on current literature is the lack of rigorous learning/estimation of these con-

straints. Several techniques are simply fixing the temporal constraints as a user parameter, or they are proposing an exhaustive evaluation of an arbitrarily subset of candidates. Since the number of patterns grows exponentially with the number of constraints and the number candidate constraints, these solutions are not viable. Other techniques are proposing non-robust estimations of the temporal constraints with strong suppositions on the data (shape of the distribution, type and level of noise, etc.). Unfortunately, real world data do not satisfy these suppositions, and the techniques can not be applied.

Our approach to this problem has three major differences with the current literature: First, we do not suppose any properties on the dataset. Second, we are not doing any exhaustive tests on the candidate constraints. Third, because of the fundamental difference between the temporal constraints of the body of an association rule and the temporal constraints of the head of an association rule, we develop two different and specialized techniques for these two cases.

3. Tita rules

Several notations are used in this paper:

A *probability distribution* describes the probability of each value (or interval of values) of a random variable. The uniform probability distribution between two points a and b is noted $\mathbb{U}_{a,b}$.

A *(temporal) event* e is a symbol (called type and noted symbol_e) and a time of occurrence (time_e). The writing convention is $e := \text{symbol}_e\,[\text{time}_e]$. A *state* s is a function $\mathbb{R} \to \{0, 1\}$ that maps a value for every time location (i.e. real number). If $s(t) = 1$, s is said to be true at time t. Otherwise, s is said to be false at time t. A *boolean function* is a function $\mathbb{R} \to \{0, 1\}$.

A Temporal Interval Tree Association Rule (Tita rule or Titar) is a temporal pattern with the semantic of a rule i.e. a body and a head. Several graphical examples of rules are given in fig. 1 and detailed after the formal definition.

Definition 1 *A* type 1 condition c *is a symbol (*symbol_c*) and a set (possibly empty) of type 2 conditions (*conds_c*). The writing convention is* $c := \langle \text{symbol}_c, \text{conds}_c \rangle$*. Given a set of events* E*, a type 1 condition* c *is true at time* t *if:*

- *E contains an event e of symbol* symbol_c *and time t i.e.* $\text{symbol}_e = \text{symbol}_c$ *and* $\text{time}_e = t$.
- *All type 2 conditions* $c' \in \text{conds}_c$ *are true at time t (see definition bellow).*

Definition 2 *A* type 2 condition c *is either:*

- *The negation of a type 2 condition c_2 (written $c := \text{not}\, c_2$). Here, c is true at time t if and only if c_2 is false at time t.*
- *A condition over a state s (written $c := s$). Here, c is true at time t if and only if s is true at time t i.e. $s(t) = 1$.*
- *An association between a boolean function m and a type 1 condition c_3 (written $c := [m, c_3]$). Here, c is true at time t if and only if $\exists t'$ with $m(t' - t) = 1$ and c_3 is true at time t'. The boolean function is the temporal constraint of the condition.*

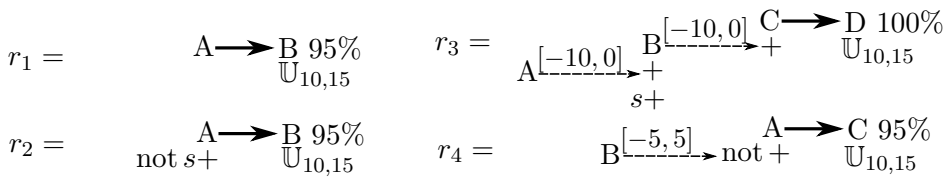


Figure 1: Four examples of Titar rules

Definition 3 *A* Temporal Interval Tree Association Rule *(Titar) r is a symbol (*symbol_r*), a confidence (*conf_r*), a non null temporal distribution (*dist_r*) called the head, and a type 1 condition (*cond_r*) called the body.* $\text{dist}_r(t - t')$ *is the probability density of having an event of symbol* symbol_r *at time t while the condition* cond_r *being true at time t'. The temporal distribution is the temporal constraint of the rule's head. The writing convention is* $r := \text{cond}_r \Rightarrow \text{head}_r \langle \text{conf}_r, \text{dist}_r \rangle$.

When the body cond_r of a Tita rule r is true at time t (also written as $\text{cond}_r(t)$), r is said to predict an event of symbol head_r with a probability of conf_r and with a temporal distribution of $t + \text{dist}_r$. An event e of symbol $\text{symbol}_e = \text{head}_r$ is said to verify such prediction if the density of the prediction is not equal to zero at time time_e i.e. $f'(\text{time}_e) > 0$ with $f' := t + \text{dist}_r$.

The *support* of a rule r is the percentage of events of type head_r which are predicted by the rule. By convention the *standard deviation* std_r of a rule r is the standard deviation of its probability distribution dist_r. The *temporal precision* of a rule r is defined as $\frac{1}{\text{std}_r}$. By convention the *prediction range* (written range_r) of a rule r is the range of its temporal distribution dist_r. The standard deviation and the prediction range of a rule are two measures of the "temporal accuracy" of a rule. Depending on the application of the rules, one measure can be more interesting to use than the other.

Definition 4 *A* unit Tita rule *is a Tita rule with the pattern $\langle \_, \emptyset \rangle \Rightarrow \_ \langle \_, \_ \rangle$ with $\_$ the unbound term i.e. a unit rule is a direct correlation between two events. Unit Tita rule expresses direct correlation such as: An event A at time t implies an event B between $t + 5$ and $t + 10$.*

Definition 5 *A* trivial unit Tita rule *is a unit rule with the pattern $\langle x, \emptyset \rangle \Rightarrow y \langle 100\%, \mathbb{U}_{-\infty,+\infty} \rangle$. This type of rule is called trivial because as soon as there is at least one occurrence of x and one occurrence of y, this rule's confidence and support are 100%.*

We present four examples of Tita rules. A graphical representations of next example rules are given in fig. 1. This representation is made to help the understanding and reading of rules.

Example 1 *Suppose the rule $r_1 := \langle A, \emptyset \rangle \Rightarrow B \langle 95\%, \mathbb{U}_{10,15} \rangle$. Literally, r_1 expresses that if an event of type A occurs at time t, then, an event of type B will occur between $t + 10$ and $t + 15$ with 95% chance. This rule is a* unit rule. *It expresses a simple and direct correlation between two events.*

Example 2 *Suppose the rule $r_2 := \langle A, \{\text{not } s\} \rangle \Rightarrow B \langle 95\%, \mathbb{U}_{10,15} \rangle$. Literally, r_2 expresses that if an event of type A occurs at time t and the state s is not true at time t, then, an event of type B will occur between $t + 10$ and $t + 15$ with 95% chance. This rule uses the negation of a state as a condition.*

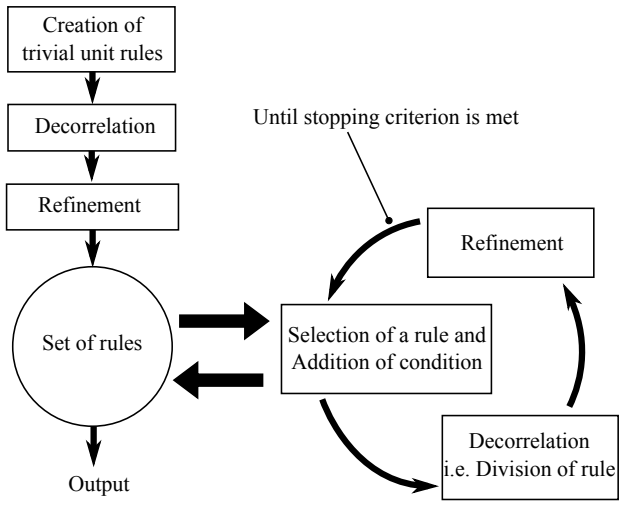


Figure 2: Structure of the Titarl algorithm.

Example 3 *Suppose the rule* $r_3 := \langle C, \{[T_{-10,0}, \langle B, \{s, [T_{-10,0}, \{\langle A, \emptyset\rangle\}]\}\rangle]\}\rangle \Rightarrow D \langle 100\%, \mathbb{U}_{10,15}\rangle$ *. Literally,* r_3 *expresses that if an event of type* C *occurs at time* t_c *followed by an event of type* B *at time* t_b *(with a maximum interval of 10 seconds i.e* $t_b - 10 \leq t_c \leq t_b - 0$*) followed by an event of type* A *at time* t_a *(with a maximum interval of 10 seconds i.e* $t_a - 10 \leq t_b \leq t_a - 0$*) and* s *is true at time* t_b*, then, an event of type* D *will occur between* $t_a + 10$ *and* $t_a + 15$ *with* 100% *chance. This rule is the chain of conditions* $A \rightarrow B \rightarrow C \overset{then}{\Longrightarrow} D$*.*

Example 4 *Suppose the rule* $r_4 := \langle A, \{\text{not}\,[T_{-5,5}, \langle B, \emptyset\rangle]\}\rangle \Rightarrow C \langle 95\%, \mathbb{U}_{10,15}\rangle$*. Literally,* r_4 *expresses that if an event of type* A *occurs at time* t *and no events of type* B *occur between* $t-5$ *and* $t+5$ *i.e. there are no events of type* B *around the event of type* A*, then, an event of type* C *will occur between* $t+10$ *and* $t+15$ *with* 95% *chance. This rule shows a negation of the occurrence of an event.*

4. Learning algorithm

The structure of the algorithm has been proposed and presented in (Guillame-Bert and Crowley, 2011). This section presents an overview of this algorithm, and details the improvements. The underling idea is the following one: Titarl begins by computing the set of trivial unit rules: Given n symbols, n^2 trivial unit rules are created. These rules are 'decorrelated', 'refined' and stored in a set R (R is initially empty). Next, until a stopping criterion is meet (maximum number of rules, maximum duration of learning, etc.) the algorithm picks a rule r in the set R, it adds a condition to r, 'decorrelate' it, 'refine' it, and it adds the result back to the set R (the result can be several rules). The figure 2 shows the global architecture of the Titarl algorithm.

The improvement of a rule is based on three different operations (Addition of condition, division (or decorrelation) and refinement). These three operations are presented through three examples in Figs. 3, 4, 5 and 6. Given a rule, the number of different parameters for the improvements (Addition of condition, division and refinement) is infinite. Therefore, for each of these three operations, we associate an *improvement policy*. These improvement policies are the core of the algorithm, and they are our solution to the Temporal Constraint Selection Problem. They are presented in the next sub-section.

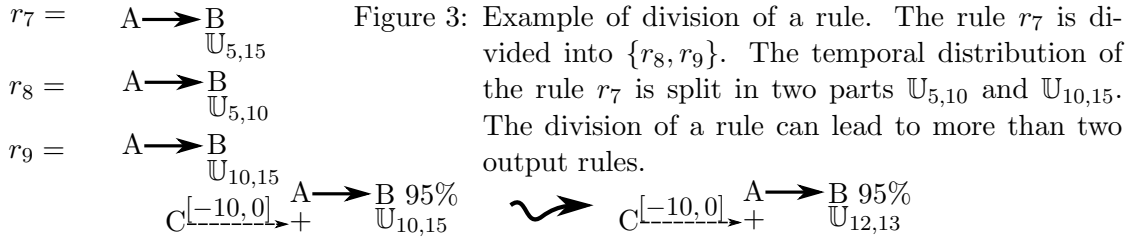


Figure 3: Example of division of a rule. The rule r_7 is divided into $\{r_8, r_9\}$. The temporal distribution of the rule r_7 is split in two parts $\mathbb{U}_{5,10}$ and $\mathbb{U}_{10,15}$. The division of a rule can lead to more than two output rules.

Figure 4: Example of head of rule refinement. The distribution of the rule is reduced from $\mathbb{U}_{10,15}$ to $\mathbb{U}_{12,13}$. The result rule has a greater temporal precision (i.e. lower prediction range), but the confidence and the support of the result rule may be lower than the confidence and support of the original rule.

4.1. Improvement policies

This subsection presents the three improvement policies used in the algorithm. Each of these policies take as input a rule (or a set of rules), and give in return a parameter for the corresponding operation. For example, given a rule r, the policy for the division of rule returns the 'best' *division function* for r. Each of these policies need to scan the input dataset of events. In the case of large datasets, the input can be randomly sampled.

4.1.1. Policy for the Addition of condition

The policy for the addition of condition relies on the *information gain* of addition of condition. The selection of the highest information gain is inspired from the ID3 algorithm used to generate decision trees (Quinlan, 1986). A given percentage of the time (fixed at 90% in the experiments), the process selects the rule and the condition to add in order to maximise the information gain. The other ten percent of the time, the process selects a random rule and a random condition.

4.1.2. Policy for the Division of rule

We begin the presentation of this policy with an example.

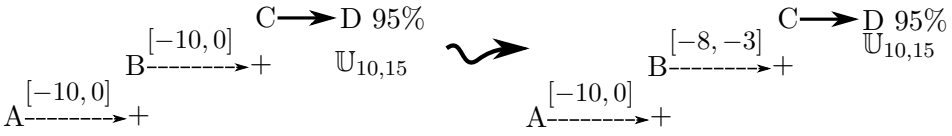


Figure 5: Example of body of rule refinement. The boolean function of one of the condition of the the rule is reduced from $\mathbb{U}_{-10,0}$ to $\mathbb{U}_{-8,-3}$.

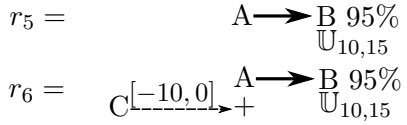
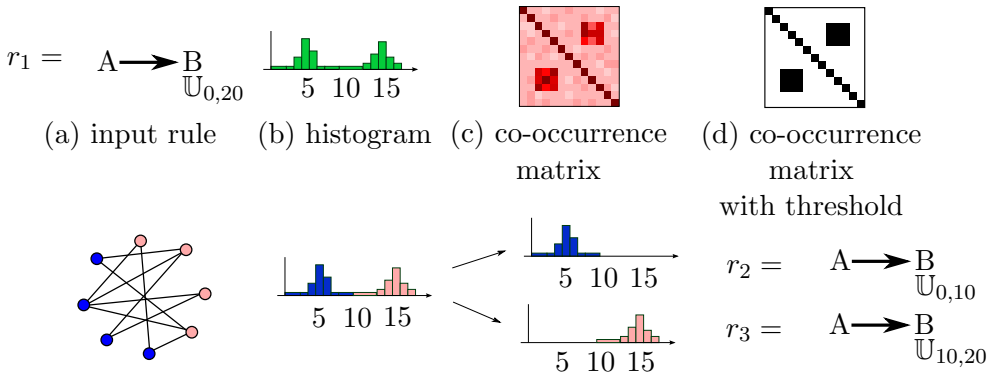


Figure 6: Example of addition of the condition $[T_{-10,0}, \langle C, \emptyset \rangle]$ to the rule $r_5 := \langle A, \emptyset \rangle \Rightarrow B\,\langle 95\%, \mathbb{U}_{10,15} \rangle$. The result is $r_6 := \langle A, \{[T_{-10,0}, \langle C, \emptyset \rangle]\} \rangle \Rightarrow B\,\langle 95\%, \mathbb{U}_{10,15} \rangle$.



(a) input rule (b) histogram (c) co-occurrence matrix (d) co-occurrence matrix with threshold

(e) colored independence graph (f) colored distribution (g) output rules

Figure 7: The seven steps of rule division policy.

Example 5 *Suppose a rule r that produces predictions that always match two events or none. Suppose, we divide this rule into two rules r_1 and r_2, and that r_1 and r_2 produce predictions which always match one event or none. We have,* $\text{conf}_{r_1} = \text{conf}_{r_2} = \text{conf}_r$ *and* $\text{std}_{r_1} \leq \text{std}_r$ *and* $\text{std}_{r_2} \leq \text{std}_r$*. To conclude r_1 and r_2 are 'better' than r because r_1 and r_2 are more precise than r but their confidence is similar to the confidence of r.*

The goal of this policy is to produce more temporally precise rules (decrease of rule standard deviation) while loosing as less as possible confidence and support (like the division of r into r_1 and r_2). Several possible solutions have been studied to chose the 'best divisions' including the analysis of the shape of histograms (detection of maximums, detection of inflexions points, etc.) and the maximisation of the likelihood on models mixtures (Gaussian, exponential, uniform, log-normal, etc.). These solutions rely on strong hypothesis on the data. The final solution we propose does not rely on such hypothesis.

Given a rule r, the policy selects a 'good division' to split r into a set of rules $\{r_i\}$ such that the rules $\{r_i\}$ are optimally matching always only one event. We perform this operation with a graph colouration technique. Suppose a rule r, and cond_r the condition of r. By convention $\text{cond}_r(t)$ is a Boolean predicate which is true if and only if cond_r is true at time t. The policy relies on the analysis of the probability distribution of the rule $\text{dist}_r = P(t' - t'' | \, \text{head}_r[t] \text{ and } \text{cond}_r(t'))$. Fig. 7 shows a step by step example of the process.

1. The first step is to compute H, an N categories histogram of dist_r (dist_r is the distribution of a rule r). The bounds of this histogram are defined by the user parameter **histogramBounds** (H is not necessarily uniform). Each category i of the histogram corresponds to the interval I_i.
2. The second step is to compute an N by N co-occurrence matrix M such as: Given a prediction $t' + \text{dist}_r$ of the rule r ($\text{cond}_r(t')$ is true) and an event $\text{head}_r[t_1]$ matching this prediction $((t' + \text{dist}_r)(t_1) > 0)$ with $t_1 - t' \in I_i$, $M_{i,j}$ is the probability of having an event $\text{head}_r[t_2]$ also matching this prediction with $t_2 - t' \in I_j$.

$$M_{i,j} = P(\, \text{head}_r[t_2] \wedge (t' + \text{dist}_r)(t_2) > 0 \wedge (t_2 - t') \in I_j \\ | \, \text{cond}_r(t') \wedge \text{head}_r[t_1] \wedge (t' + \text{dist}_r)(t_1) > 0 \\ \wedge (t_1 - t') \in I_i)$$

3. Next, we apply a set of thresholds $\{\sigma_i\}$ on the covariance matrix $M_{i,j}$. We apply alternatively the thresholds to the matrix $M_{i,j}$. The next operations will be applied on each of these result matrices.
4. For each matrix $M_{i,j}$, we compute the graph $G := (\{v_i\}, \{e_i\})$ using $M_{i,j}$ as an *adjacency matrix*. $\{v_i\}$ are the vertices of the graph G. $\{e_i\}$ are the edges of the graph G. An adjacency matrix is defined a follow: If $M_{i,j} = 1$, then there is an edge between the vertices v_i and v_j. Otherwise, if $M_{i,j} \neq 1$, then there is no an edge between the vertices v_i and v_j.
5. We compute the vertices colouring $c : V \to \mathbb{N}$ of G i.e. labelling of the graph's vertices with colours such that no two vertices sharing the same edge have the same colour. In this context, two vertices of different colours represent independent intervals of the probability distribution of the rule r.
6. Finally, the division function d is defined as follow:

$$d(x) = c(i) \text{ with } x \in I_i$$

The division function defines the division of the rule's head. We merge all the division function and remove the duplicates. The Thresholds are defined between 0 and 1. In the next the experiments, σ_i are set to be $\sigma_i = \alpha \exp^{\frac{x-p}{2}}$ for $i \in [0, p[$ with $p = 10$ and $\alpha = 0.2$. Decreasing the number of thresholds increases the speed of the algorithm, but in the case of noisy datasets, the algorithm will produce rules with less accurate temporal constraints (rule's confidence, support and temporal accuracy will decrease). If the dataset is noisy, increasing the number of thresholds increases the number of generated rules and reduce the speed of the algorithm.
In the worse case (datasets with a very specific type of noise), the number generated rules is bounded by $\min(\frac{N(N+1)}{2}, \frac{N(N+1)-(N-p)((N-p)+1}{2})$ with N the number of categories of the histogram. In our real world experiment, the average number of rules created during this step is 1 and in the computer simulated dataset, it was about 10 in the part with the highest level of noise.

4.1.3. Policy for the Refinement of a rule

The refinement of a rule r is the modification of the temporal constraint of the conditions of r, or the modification of the temporal distribution of r. The refinement has two objectives: First, in the case of a modification of the temporal distribution, it can increase a rule precision without decreasing significantly the support and the confidence. Second, in the case of the modification of the condition, it can improve a condition, and therefore, increase a rule confidence.Such operation is needed to get rid of the noise of initial estimation of temporal constraints.

Head refinement

The head refinement consists in:

1. Compute a histogram of the distribution of a rule's head.
2. Apply a small Gaussian filter on the distribution.
3. Threshold this histogram with the **ruleRefinementThreshold** parameter.
4. Set the new distribution of a rule's head to be the thresholded histogram.

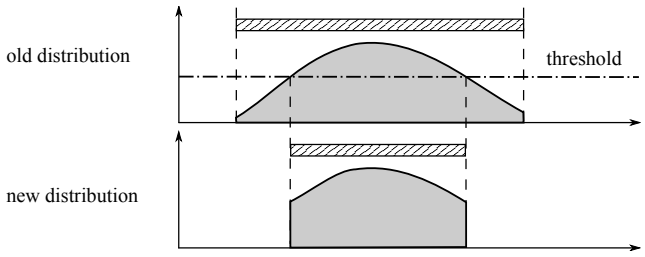


Figure 8: Graphical representation of the head refinement.

The figure 8 presents graphical this operation.

Body refinement
The body refinement consists in:

1. Compute all the successful occurrences of the rule i.e. successful predictions. For each prediction, we need to keep track of each events that make the condition true i.e. for each predictions, we need to keep track of the mapping 'type 2 condition of the rule' $\rightarrow$ 'matching time'.
2. For each type 2 conditions of the rule c, compute the histogram h of $D := \{\text{time}_c - \text{time}_{\text{parent}(c)}\}$, where given a prediction, time_c is the time of matching of the condition c, and $\text{parent}(c)$ is the type 2 parent condition of c.
   Note: $\forall d \in D, m(d) = 1$ with $c := [m, c']$.
3. Threshold the histogram h with the **ruleRefinementThreshold** parameter.
4. Set the new boolean function of the condition c to be m' with $m'(t) = 1$ if $h(t) > 0$, $m'(t) = 0$ otherwise.

5. Results

We present two evaluations of our algorithm. First, the algorithm is applied on a computer generated dataset that constraints 100 sub-parts. The second experiment is performed on a real world dataset. This data set is based on a one month recording of human activity. We conclude these experiments with a discussion. A new metric call *Global Support Map* (or GSMap) is introduced and used to evaluate our algorithm and compares it with related techniques.

5.1. Global Support Map

A *Global Support Map* (GSMap) is a tool used to evaluate a set of predictors (such as association rules) on a temporal dataset. A support map $m : [0, 1] \times [0, \infty] \rightarrow [0, 1]$ is a function mapping a *minimum confidence* (probability of the predictions to be true) and a *maximum prediction range* (or size of prediction's window) to a *global support*. The *global support* is the percentage of events predicted by rules with a confidence greater or equal than the minimum confidence and a prediction range lower or equal than the maximum prediction range. The main advantage of GSMaps over classical metrics (e.g. confidence or support) is to do not require an arbitrary fixed window size as it is usually done in related literature. A GSMap can be represented as two dimensional picture. Two GSMaps can also be compared together.

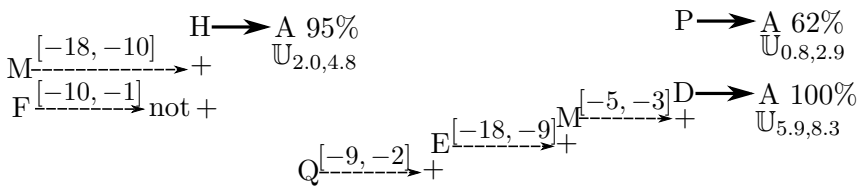


Figure 9: Three examples of rules from the generated dataset (experiment 1).

5.2. Experiment 1

We apply the Titarl algorithm on a symbolic time sequence generated on a computer. The dataset is divided into 100 *parts*. Each part has a 'ground truth'. Each part is constructed in this following way: The algorithm randomly selects a temporal association rule (structure and metrics) called *reference rule*. The confidence is chosen between 60% and 90%, the support is chosen between 40% and 90%, the number of conditions is chosen between 1 and 4, the number of uses of the pattern is chosen between 1000 and 50000. The ratio of noisy events/useful events is selected between 0% and 1000%. Several types of noises are considered: A noise non-correlated with the pattern, and noises with different types of correlations with the pattern. Next, the algorithm generates a dataset's part with the selected rule and parameters. This process is an heuristic, and the metrics of the rule need to be re-evaluated on the generated part. The generated part contains and infinite number of temporal rules (e.g. a dataset where the reference rule is $A \dashrightarrow B \rightarrow C$ also contains the rule $B \rightarrow C$ with a lower confidence). However, the generative algorithm guaranties that the *reference rule* is the rule with the highest score according to Equ. 1.

For a symbolic time sequence, a ground truth is not a set of the actual events to predict, but a set of *the best possible predictions to do*. Such ground truth is required for the proper evaluation of predictors on symbolic time sequences. This is especially true while dealing with patterns with low confidence/support. For example, an algorithm producing predictions with 10% confidence when the ground truth has a 15% confidence can be considered better than an algorithm producing predictions with 50% confidence when the ground truth has a 100% confidence.

The goal of this dataset is to precisely analysis of the robustness, completeness and power of expression of temporal learning algorithm. As far as we know, there is currently no available symbolic time sequence datasets associated with a 'ground truth'. The reference predictions are (with a small error rate) the best predictions that can be expected to do. Fig. 9 shows three examples of patterns of the dataset. The convention is the same as the one used for the fig. 1. The dataset is available online (Guillame-Bert, 2011).

We apply the Titarl algorithm for 60 seconds on each part of the dataset. We applied the Minepi (Mannila and Toivonen, 1996) until it reaches the fourth loop (since the maximum number of conditions is 4). The time bound is fixed to 100 time units and the window sizes parameter is fixed to all integer between 1 and 100 i.e.
$(w1, w2)|(w1, w2) \in \mathbb{N}^2$ with $w1 \leq w2 \leq 100$. We applied the Face (Dousson and Duong, 1999) algorithm until it stops. On every part of the dataset, Face is run ten times with the pMin parameter fixed between 0.1 and 1 i.e. $pMin \in \{0.1, 0.2, 0.3, \cdots, 0.9, 1\}$.

Each generated rule is scored according to the equation 1. Since each part of the dataset is based on only one temporal rule (called reference rule), and because the temporal distribution of this rule is always uniform distribution (the dataset is designed in such way),

algorithm	Normalized confidence /reference	Normalized support /reference	A.v.g. range /reference	N.o.p. /reference
Face	0.49	0.71	16.29	2.21
Minepi	0.51	0.78	3.75	1.43
Titarl (old version)	**0.83**	**0.95**	**3.18**	**1.16**
Titarl (new version)	**0.90**	**0.99**	**2.09**	**1.11**
Reference	1.00	1.00	1.00	1.00

Table 1: Evaluation of Titarl, Minepi and Face algorithms on the computer generated dataset. The table shows the average of the ratio measure/reference measure for the normalized confidence, normalized support, average range and number of predictions. 'n.o.p.' stands for 'number of predictions'.

then the reference rule is guaranteed to be the rule with the highest score in a given dataset. We select the best rule from each set of leaned sets of rules according to the 'score'.

$$score(r) := \frac{\text{conf}_r^4 \, \text{supp}_r^2}{\text{range}_r} \tag{1}$$

The cross validation is performed in the following way: (a) We extract rules for each part of the dataset (b) We select the rule with the highest score. (c) Finally, we evaluate this rule on another part of the dataset that contains the same pattern. We know the exact patterns (reference rules) for each parts of the dataset. The ranges of predictions are normalized to the ranges of the reference predictions in order to compare supports and confidences (called normalised support and normalised confidences). For example, suppose a part where all reference predictions have a range of 4. In addition, suppose the prediction 'A will occurs between times 16 and 22'. This prediction has a range of $22 - 16 = 6$. The normalized prediction will be 'A will occurs between times 17 and 21' (range of 4). The normalized confidence of a predictor is the confidence of its normalized predictions.

The table 1 shows the average ratio between the normalized confidence, normalized support, average range and number of predictions of the learned rules and the confidence, support, average range and number of predictions (n.o.p.) of the reference rules.

Minepi gives a slightly better normalized confidence and normalized support than Face. We also observe that our algorithm outperforms Minepi and Face on normalized confidence and normalized support. Minepi prediction ranges are better than Face prediction ranges. Our algorithm outperform Minepi and Face on prediction range. The new version of Titarl performs better than the old version. Detailed observation shows that the new version of Titarl especially outperform the old version of Titarl in the part of the dataset with the highest level of noise.

5.3. Experiment 2

The 'Home activities dataset' created by Tim van Kasteren et al. (van Kasteren et al., 2008) is a record of 28 days of sensor data and activity annotations about one person performing activities within an apartment. The apartment is equipped with sensors on doors, cupboard, fridge, freezer, etc. Activities of the person are annotated (prepare breakfast, dinner, having a drink, toileting, sleeping, leaving the house, etc.). The dataset is divided into two cat-

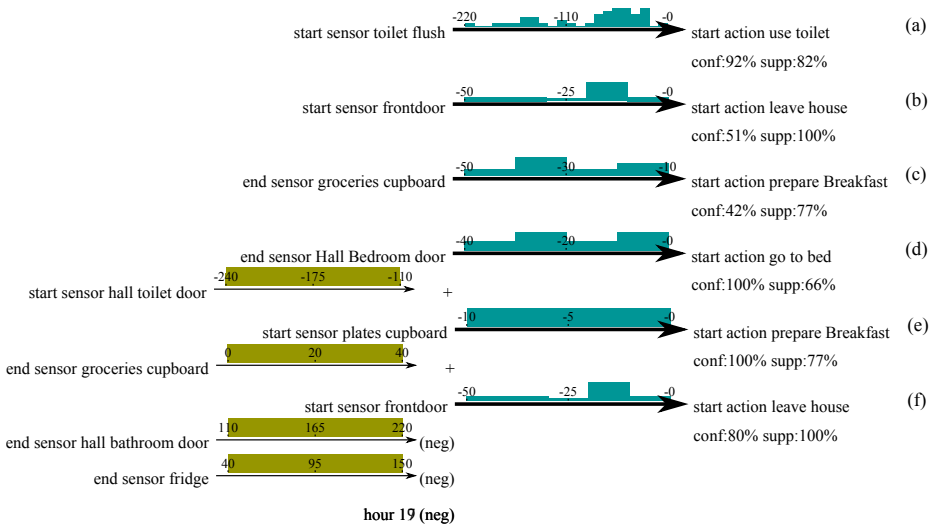


Figure 10: Example of learned rules from the HomeDataSet (experiment 2)

egories: sensor events (start_sensor_fridge, end_sensor_fridge, start_sensor_frontdoor, etc.) and change of activities (start_action_get_drink, end_action_get_drink, start_action_prepare_dinner, etc.). In addition, twenty-four states describing the time of the day (it_is_1am, it_is_2am, it_is_3am, etc.) is available. In this experiment, the Titarl algorithm is applied in order to predict activity change events according to sensor events and states.

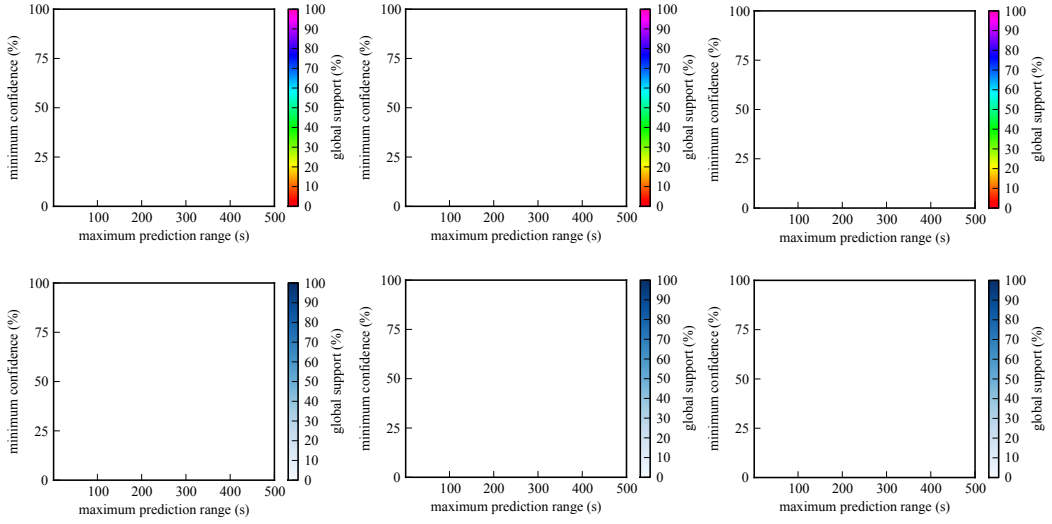
The algorithm is executed for 120 seconds for every type of event to predict (beginning and ending of each activities).

The learned rules that predict all the uses of bathroom with at least 58% confidence and a temporal precision of less than 100 seconds, or with 100% confidence and a temporal precision range of 400 seconds. Direct correlations between events (unit rules) often have good support but average confidence. The algorithm learns the direct implication between the use of the toilet flush and the action of using toilets (confidence: 92%, support: 82%, prediction range: 220 seconds and standard deviation: 46 seconds), or the use of the front door and the action of leaving the flat (confidence: 51%, support: 100%, prediction range: 50 seconds and standard deviation: 11 seconds). Fig. 10 shows some example of learned rules. The Fig. 11 shows the average GSMaps of the rules learned with the Face, Minepi and our algorithm. Table 2 shows the statistics of the subtraction of these GSMaps.

Table 2 shows that Face is more efficient than Minepi: With the same confidence and range constraint, Face explains an average of 41% more of events than Minepi. Titarl outperform the Face and Minepi algorithm: With the same confidence and range constraint, Titarl explains an average of 10% more of events than Face and 51% more of events than Minepi.

Remark 6 *The given percentage represents means of differences of global support. If the techniques A and B have respectively a global support of* 50% *and* 100%*, the B explains* +50% *of events than A.*

Fig. 12 shows the evolution of the average of the GSMap for the Titarl algorithm with the training time. Most of the events can be explained with the *trivial rules* learned during the first loop (spike at time = 2). Next, the rules are refined and the metrics are improved until they stabilize.



(a) Face algorithm (b) Minepi algorithm (c) Titarl algorithm (new version)

Figure 11: Global support maps for the 'Home activities dataset' represented with mono-color and multi-colour map. GSMap shows the percentage of explained events given a minimum confidence and a maximum prediction range. It is easy to read global tendencies when using a mono-colour colour map. It is convenient to read punctual support value when using a multi-colour map.

X \ Y	**Face**	**Minepi**	**Titarl** (old version)	**Titarl** (new version)
Face		mean : 41.65% median : 47.33% std : 17.51% strict pos. : 97.70%	mean : -11.76% median : -2.31% std : 15.60% strict pos. : 5.53%	mean : -15.83% median : -9.82% std : 17.68% strict pos. : 26.84%
Minepi	mean : -41.65% median : -47.33% std : 17.51% strict pos. : 1.72%		mean : -53.41% median : -54.48% std : 12.40% strict pos. : 1.41%	mean : -57.42% median : -59.59% std : 12.20% strict pos. : 1.78%
Titarl (old version)	mean : 11.76% median : 2.31% std : 15.60% strict pos. : 69.90%	mean : 53.41% median : 54.48% std : 12.40% strict pos. : 98.02%		mean : -4.05% median : -1.92% std : 4.96% strict pos. : 36.77%
Titarl (new version)	mean : 15.83% median : 9.82% std : 17.68% strict pos. : 73.16%	mean : 57.42% median : 59.59% std : 12.20% strict pos. : 98.22%	mean : 4.05% median : 1.92% std : 4.96% strict pos. : 63.23%	

Table 2: Statistics of the subtraction of the GSMaps display in Fig. 11. The 'strict pos.' is the percentage of the global support strictly greater than 0. The GSMaps are computed with a prediction range between 0s an 500s with 100×100 cells.

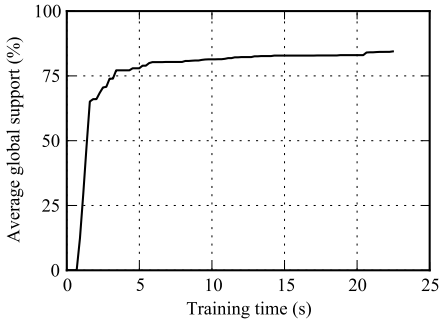


Figure 12: Evolution of the GSMap average with the training time i.e. the curve shows the average (on the different actions) of the average (for a given GSMap) of the GSMaps according to training time.

6. Summary and Conclusions

We identify one of the core problems of the learning on symbolic time sequences (the Temporal Constraint Selection Problem) and presented a new solution to tackle it.

We introduce a temporal model for temporal associate rules called Tita rules. Based on our solution for the *Temporal Constraint Selection Problem*, we present an efficient algorithm (Titarl) able to learn such rules. A solution to represent performance of sets of temporal rules is presented (GSMaps). The algorithm is evaluated on two datasets, and compared with other approaches. The computer generated dataset shows the robustness of the approach while the real world dataset shows the usability in practical problems.

The result of our algorithm leads us to believe that the combination of the new grammar of temporal rules (Titar) and our techniques to mine them is a powerful tool.

References

C. M. Antunes and A. L. Oliveira. Temporal data mining: An overview. *KDD 2001 Workshop on Temporal Data Mining*, 2001.

Jay Ayres, Johannes Gehrke, Tomi Yiu, and Jason Flannick. Sequential pattern mining using a bitmap representation. ACM Press, 2002.

Mira Balaban and Tzachi Rosen. Stcsp : structured temporal constraint satisfaction problems. *Annals of Mathematics and Artificial Intelligence*, 25, January 1999.

Gemma Casas-Garriga. Discovering unbounded episodes in sequential data. In *Knowledge Discovery in Databases: PKDD 2003*, volume 2838. Springer Berlin / Heidelberg, 2003.

Yen-Liang Chen, Mei-Ching Chiang, and Ming-Tat Ko. Discovering time-interval sequential patterns in sequence databases. *Expert Systems with Applications*, 25(3):343 – 354, 2003.

Christophe Dousson and Thang Vu Duong. Discovering chronicles with numerical time constraints from alarm logs for monitoring dynamic systems. In *Proceedings of the 16th international joint conference on Artifical intelligence - Volume 1*, 1999.

M. Guillame-Bert. Symbolic time sequence dataset. 2011. "available at http://www-prima.imag.fr/guillame-bert/?page=database".

Mathieu Guillame-Bert and James L. Crowley. New approach on temporal data mining for symbolic time sequences: Temporal tree associate rules. In *ICTAI*, 2011.

G. Guimares. The induction of temporal grammatical rules from multivariate time series. In *Grammatical Inference: Algorithms and Applications*, volume 1891. Springer Berlin / Heidelberg, 2000.

Pablo Hernandez-Leal, Luis Enrique Sucar, and Jesus A. Gonzalez. Learning temporal nodes bayesian networks. In *FLAIRS Conference*, 2011.

Yu Hirate and Hayato Yamana. Generalized sequential pattern mining with item intervals. *JCP*, pages 51–60, 2006.

Eamonn J. Keogh and Padhraic Smyth. A probabilistic approach to fast pattern matching in time series databases. In *KDD*, 1997.

Heikki Mannila and Hannu Toivonen. Discovering generalized episodes using minimal occurrences. In *In Proceedings of the 2nd International Conference on Knowledge Discovery in Databases and Data Mining*. AAAI Press, 1996.

Heikki Mannila, Hannu Toivonen, and A. Inkeri Verkamo. Discovery of frequent episodes in event sequences. *Data Mining and Knownledge Discovery*, 1, 1997.

Fabian Mörchen. Unsupervised pattern mining from symbolic temporal data. *SIGKDD Explor. Newsl.*, 9, June 2007.

Jian Pei, Jiawei Han, Behzad Mortazavi-asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei chun Hsu. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. 2001.

J. R Quinlan. Induction of decision trees. In *Machine Learning*, 1986.

J.F. Roddick and M. Spiliopoulou. A survey of temporal knowledge discovery paradigms and methods. *Knowledge and Data Engineering, IEEE Transactions on*, 14(4), 2002.

Tim van Kasteren, Athanasios Noulas, Gwenn Englebienne, and Ben Kröse. Accurate activity recognition in a home setting. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*. ACM, 2008.

Mohammed J. Zaki. Efficient enumeration of frequent sequences. In *CIKM '98: Proceedings of the seventh international conference on Information and knowledge management*. ACM, 1998.

Q. Zhao and S. S. Bhowmick. Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore*, 2003.