Noname manuscript No.
(will be inserted by the editor)

Exploring Multi-Relational Temporal Databases with a Propositional Sequence Miner

Carlos Abreu Ferreira · João Gama · Vítor Santos Costa

Received: date / Accepted: date

Abstract In this work we introduce the *MuSer*, a propositional framework that explores temporal information available in multi-relational databases. At the core of this system is an encoding technique that translates the temporal information into a propositional sequence of events. By using this technique we are able to explore the temporal information using a propositional sequence miner. With this framework we mine each class partition individually and we do not use classical aggregation strategies, like window aggregation. Moreover, in this system we combine feature selection and propositionalization techniques to cast a multi-relational classification problem into a propositional one.

We empirically evaluate the *MuSer* framework using two real databases. The results show that mining each partition individually is a time and memory efficient strategy that generates a high number of highly discriminative patterns.

Keywords Sequence Mining; Discriminative Patterns; Multi-relational Temporal Databases

School of Engineering, Polytechnic of Porto
Rua Dr. António Bernardino de Almeida, 431, 4200-072 Porto, Portugal
Tel.: +351228340500
Fax: +351228321159
E-mail: cgf@isep.ipp.pt

LIAAD - INESC TEC LA
Rua Dr. Roberto Frias, 378, 4200 - 465 Porto, Portugal
E-mail: joao.jgama@gmail.com

CRACS - INESC TEC LA
Rua do Campo Alegre, 1021-1055, 4169-007 Porto, Portugal
E-mail: vsc@dcc.fc.up.pt

1 Introduction

Quite often, multi-relational databases accumulate data over long periods of time. To explore such temporal records we can use aggregation operators, like window aggregation, or use special purpose techniques that explore the time order of the events. In this work we assume that the temporal data stored in a subset of tables are sequences of events that reflect the evolution of a phenomenon of interest. As an example, consider a multi-relational dataset in a medical domain. A patient is subject to a sequence of examinations and a set of measurements, corresponding to the results of different analysis, are taken. For each patient, we get a sequence of measurements over time. Our hypothesis is that the evolution of these measurements might encode relevant information for the diagnosis of a certain disease. The problem we address here is therefore *how can we explore such information?* In more general terms, can we extract *implicit* multi-relational temporal information that can be used to learn predictive and high accurate decision models? We believe that both intra-table and inter-table temporal patterns can be used for this purpose. To do so, we develop an architecture that is grounded in the propositionalization methodology [10, 30].

In this work we propose *MuSer* (MUlti-relational SEquential patteRn knowledge learning), a framework that discovers sequence patterns available in a subset of relational tables. The algorithm is organized as a pipeline. We start by converting the multi-relational temporal data into what we call a *sequence database*. In this new database the temporal information of each example is represented by a heterogeneous sequence, that can include events registered in one or more tables. In a second phase, we run a sequence miner to

find frequent sequences in each database partition. Next, in phase three, we select a subset of sequence patterns. We extract the discriminative frequent ones, e.g. those that appear in only one class, and using a chi-square filter we remove class unrelated sequences. In phase four we map the selected patterns into a binary set of attributes, that indicate the presence or absence of a sequence in a particular example. Last, we induce a classification model using a propositional algorithm.

To evaluate our methodology, we use two multi-relational datasets and compute the number of patterns, the run-time, the peak memory usage and the generalization accuracy of the learned classification models. Moreover, we compare the performance of our system against the RUSE-WARMR system [10], that finds first-order logic itemsets and aggregates temporal data using a set of statistics.

The contributions of this work are therefore:

- We developed *MuSer*, a framework that explores heterogeneous sources of time data recorded in a multi-relational database. Our methodology explores temporal records by abstracting patterns over *time*. We do not deal with explicit time, instead we explore the order of the events.
- The *MuSer* architecture searches for sequence patterns in each class partition individually. With this strategy we find a large number of in-class and discriminative patterns.

In the next section we present related work that motivated the development of the *MuSer* algorithm. In Section 3 we present a detailed description of the *MuSer* framework. Next, in Section 4, we present and discuss the obtained results. In the last section we conclude and point to the development of new methodologies that can include valuable domain knowledge.

2 Related Work

In this section we present an overview of the related work that inspired us and contributed to the development of the *MuSer* framework.

Let $I = \{i_1, i_2, \ldots, i_n\}$ be a set of items and e an event such that $e \subseteq I$. A *sequence* is an ordered list of events $e_1 e_2 \ldots e_m$ where each $e_i \subseteq I$. Given two sequences $\alpha = a_1 a_2 \ldots a_k$ and $\beta = b_1 b_2 \ldots b_t$, sequence α is called a *subsequence* of β if there exists integers $1 \leq j_1 < j_2 < \ldots < j_l \leq t$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_k \subseteq b_{j_l}$. A sequence database is a set of tuples $< sid, \alpha >$ where *sid* is the sequence identification and α is a sequence. The *count* of a sequence α in a database of sequences D, denoted $count(\alpha, D)$, is the number of examples in D that contain the α subsequence. The *support* of a sequence α is the ratio between $count(\alpha, D)$ and the number of sequences in D. We denote support of a sequence as $support(\alpha, D)$. Given a sequence database D and a minimum support value λ, the problem of sequence mining is to find all subsequences in D having a support value equal or higher than the user-defined value, the λ value. Each one of the obtained sequences is also known as a frequent sequence or a sequential pattern.

There exists a wide range of algorithms that can explore propositional sequential data efficiently. To the best of our knowledge, Agrawal and Srikant introduced the problem of mining sequential patterns in [2]. In [26] these two authors introduce the GSP algorithm, an algorithm that generalizes the original sequential pattern mining problem and that is inspired in search procedure of the well-known Apriori algorithm [1]. GSP uses a levelwise candidate generation strategy to find all frequent sequences. The algorithm uses the lattice structure to generate all candidate item sequences. It starts by finding all frequent 1-items that constitute the lattice base. Then, in an iterative process, at each lattice level the frequent sequences of the previous level are combined to generate new frequent sequence candidates. Notice that at each lattice level l only the frequent subsequences of level $l - 1$ are used to generate new candidates.

The problem with this approach is the huge number of findings, especially when we run the sequence miner to find patterns with a low support value. In such a case, and if working with large datasets, the complexity of the search space can make it impossible to find patterns with low frequency. To alleviate these issues, algorithms that can find a constrained subset of sequential patterns from propositional data have been developed. Examples include CloSpan [28] that returns a set of closed sequential patterns, VMSP [12] that finds maximal sequential patterns and cSPADE [29] that finds a constrained set of sequential patterns. By using cSPADE we can find, among other, sequential patterns predictive of one or more classes. In [18] it is claimed that by using highly predictive or contrasting sequences as input to a propositional classifier we can get generalization accuracy gains that range from 10% to 50%.

A different approach, that is known to be successful, is to use post-processors, *filters*, to select interesting patterns. Some use sequential ad-hoc selection, that are model unrelated, whereas others use wrapper filters, that select features based on the induced models.

In [17] the authors introduce MineSeqLog algorithm to find first-order logic (FOL) sequences. The algorithm combines ideas from version spaces [25, 4, 15] and lev-

Table 1: Relational database used to illustrate the encoding technique

Patient Info
(Target Table)

Id	*Sex*	*BornDate*	*Class*
1	m	19520109	**c**
2	f	19750123	**b**

Blood Analysis

Id	*Date*	*RBC*	*WBC*
1	19750102	high	normal
1	19780203	high	high
2	19770107	high	low

Urinalysis

Id	*Date*	*Exam*	*Value*
1	19741201	plt	high
1	19750102	alb	normal
1	19750102	ttp	normal
1	19760204	alb	normal
2	19800403	alb	normal

Algorithm 1: *MuSer* pseudo-code

input : a multi-relational dataset **r**; three thresholds λ, the support value of the sequence miner and k, the number of top patterns to map
output: a classification model

Pre-Processing Encoding
  $\mathbf{s} \leftarrow$ `conversion` (**r**);

$\mathbf{s}_1, \ldots, \mathbf{s}_m \leftarrow$ `partition` (**s**);
Finding Frequent Patterns
  for $i = 1$ *to* m **do**
    $sf_i \leftarrow$ `SequenceMiner` ($\mathbf{s}_i$,λ);

Filtering
  $S_{disc} \leftarrow$ `discriminate` ($sf_1, \ldots, sf_m$);
  $S_{inter} \leftarrow$ `sort` (S_{disc},*chi-square*);

Mapping
  $r_{EnlargedTarget} \leftarrow$ `Mapping` (S_{inter}, k, $\mathbf{r}_{target}$);

Classifier Induction
  `ClassifierIndution` ($r_{EnlargedTarget}$);

elwise search to find the complete set of frequent patterns. This algorithm suffers from two major problems. When fed with large databases, that accumulate data over a long period of time, the algorithm is unable to find long patterns.

ILP approaches have an enormous representational power but are often criticized for lacking scalability [5]: ILP algorithms may not be very effective for the large search spaces induced by sequence databases.

One approach to solve the scalability issue of ILP systems is to use propositionalization [16, 30, 8]. The idea is to augment the descriptive power of the target table by mapping clauses (new attributes) on the target table.

Even with recent progress on scalability there exists Prolog data which remain almost impossible to explore effectively by using only ILP based approaches. As an example, intra-table and inter-table temporal patterns remain hard to explore. One approach, followed by *WARMR* [7] is to use aggregation methodologies, unfortunately losing relevant time information. One example of this approach is the RUSE-WARMR [10] framework. In the first phase the WARMR finds frequent datalog queries [7]. Next, these findings are pruned and the most interesting patterns are mapped into the target table. In the last phase, a classification model is learned using the augmented target table. In View Learning [6] we can define and use alternative *views* of the database, i.e., we can define new fields or tables. Such new fields or tables can also be highly useful in learning, but still require searching a very large search space.

The work presented in [11] introduces a framework that runs a propositional sequence miner to find frequent, closed or maximal patterns available in the full dataset. In the final phase a FOL theory is learned using a ILP algorithm. The main issue with this approach is that by taking the full dataset as input the algorithm cannot find valuable discriminative sequence patterns.

3 The *MuSer* Framework

In this section we present the *MuSer* framework. This new architecture explores temporal and logico-relational information to build descriptive and high accurate classifier models. More precisely, the framework explores sequences of events registered in one or more tables of a Prolog database. In the first phase we convert the Prolog data into a sequence database. Then, in a second phase, we run a sequence miner in each class partition to find all sequential patterns. Next, considering the huge number of findings, we introduce two filters to select the discriminative and class correlated patterns. In the fourth phase, the most interesting fre-

Table 2: Sequence Database

ID	Sequence	Class
1	(9) (3 5 7 8) (7) (3 6)	c
2	(3 4) (7)	b

Table 3: Enlarged target table

MID	subtype	sex	bornDate	S_1	S_2
1	c	m	19520109	1	1
2	b	f	19750123	0	1

quent sequences are mapped into the target table, generating an enlarged target table. In the last phase, we learn a classification model by taking the enlarged target table as input to a propositional classifier.

To best explain the *MuSer* architecture we follow the pseudo-code presented in Algorithm 1.

3.1 Encoding Methodology - Phase 0

Here we introduce the strategy that converts the Prolog temporal data into a convenient sequence database. We use the relational database presented in Table 1 to best explain this encoding methodology. This illustrative database is inspired in the PKDD 2005 Hepatitis dataset and has three tables recording the follow-up of two patients. One of these tables is the target table, named *Patient Info*. This table records, for each patient, a masked ID, the sex, the date of birth and the class label. The other two tables record sequences of blood analysis and urinalysis examinations. The *Blood Analysis* table records, for each patient, the examination date and the patient's *RBC* and *WBC* parameters. Table *Urinalysis* registers for each patient the value of three urine test parameters: alb, plt and ttp.

First, for each example available in the Prolog target table we find all the associated relations that record temporal information. Next, we sort all *temporal* tuples using the order of occurrence of the events. After this process, for each example we obtain a chronological sequence of multiple events. In Figure 1 we present the event sequence of patient number one. In this case, the sequence includes a sequence of blood and urinalysis events.

Next, we built an attribute-value sequence for each example. As an example, for patient one we get $(plt = high)(RBC = high, WBC = normal, alb = normal, ttp = normal)(alb = normal)(RBC = high, WBC = high)$. In this new sequence each itemset corresponds to all records registered at a given date/time. Then, we define a one-to-one encoding map $f : Attributes \times Values \longrightarrow \mathbb{N}$. This mapping associates a *unique* number to each attribute-value pair. We define this map according to the type of the attributes. For each discrete attribute we find the range of attribute values and map each attribute-value pair onto a unique integer value. This map is a three element tuple having the attribute name, the attribute value and the unique integer number that identifies the attribute-value. When dealing with continuous attributes, first we apply a discretization strategy and then proceed in the same way we do for discrete attributes. The attribute discretization approach is specific of the problem and the domain we are dealing.

Table 2 shows the sequence database that we obtain when we apply this encoding technique to our example database. In this table each sequence corresponds to an example in the target table and each subsequence corresponds to all events that occur at the same time. In this example we define the one-to-one map $f(rbc, low) = 1$, $f(rbc, normal) = 2$, $f(rbc, high) = 3$, $f(wbc, low) = 4$, $f(wbc, normal) = 5$, $f(wbc, high) = 6$, $f(alb, normal) = 7$, $f(ttp, normal) = 8$, $f(plt, high) = 9$. Using this map the sequence of events registered for patient one is encoded into the sequence (9) (3 5 7 8) (7) (3 6).

3.2 Finding Frequent Sequence Patterns - Phase 1

Having the sequence database built in the pre-processing phase, we now need to use a methodology to find new and interesting descriptors. Thus, we run a sequence miner in each class partition of the sequence database (line 7 of Algorithm 1). This way we get all in-class frequent patterns, the sf_i set. Each one of these i sets contains patterns that have at least a minimum support equal to λ, a user-defined threshold. Regarding the nature of the sequence miner, the type and the dimension of the problem being addressed we usually get an overall huge number of patterns. Among these patterns there are high discriminative and class correlated patterns that we must keep and uninteresting and redundant patterns that must be eliminated.

3.3 Feature Selection - Phase 2

To select discriminative sequences we introduce the *discriminative filter* (line 10). Using this filter we select a set of discriminative sequences, the S_{disc} set, a set of patterns that were found in only one of the partitions. This pruning operation is accomplished with a simple matching operation that keeps non-agreeing patterns.

Consider that when we run the sequence miner in the sequence database of partition **b** we find the two

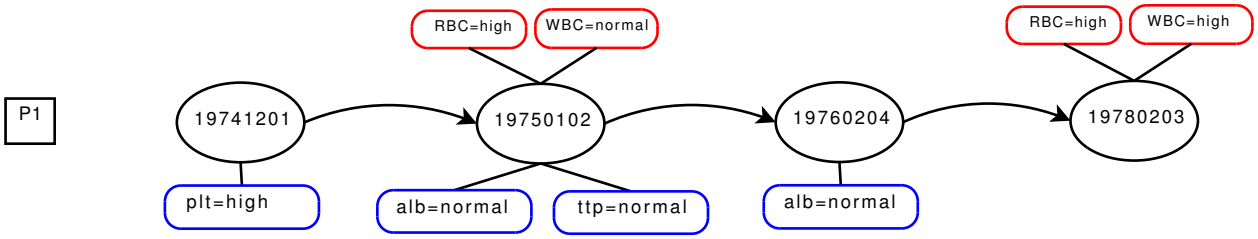

Figure 1: Patient one sequence of events

sequences: S_{b1} : (9) (5 8) (3 6) and S_{b2} :(10) (7). Moreover, by running the same algorithm in the sequence database of partition **c** we obtain sequences S_{c1} : (3) (7) and S_{c2} : (10) (7). After applying the discriminative filter we eliminate S_{b2} and S_{c2} and keep sequences S_1 : (9) (5 8) (3 6) found in class partition **b** and S_2 : (3) (7) from class partition **c**.

After eliminating the non-discriminative patterns some class uncorrelated patterns remain. To eliminate some of these patterns we use the *Chi-square filter* that computes the chi-squared test statistic [3]. To compute the chi-square statistic, for each discriminative sequence pattern found, first we compute a contingency table that summarizes the information about the number of sequences in each class partition that contain (and do not contain) the given discriminative sequence pattern. Then, we take each contingency table and compute the chi-squared test statistic, whereas the null hypothesis is that the joint distribution of the cell counts in the 2-dimensional contingency table is the product of the row and column margins. The chi-squared test statistic value that we obtain for each sequence pattern is then used to sort descending the set of discriminative sequence patterns. The lower the chi-square statistic value the closest we are to reject the null hypothesis, i.e, to reject that the two categorical variables (class and pattern) are independent.

By applying sequentially these two filters we get a sorted set of patterns. To include the descriptive power of these patterns in the learning process we need to map each of the new sequences/features into the target table.

3.4 Mapping Features - Phase 3

In this section we explain how we map each one of the k (a user-defined parameter) most interesting patterns. For each pattern we add a new Boolean attribute to the target table. The value of each new attribute is computed using the following rule. If the sequence pattern associated with the new attribute is a subsequence of the example sequence, the new attribute takes value one. Otherwise the attribute takes value zero. This way we get an enlarged target table that includes the primitive attributes and the new Boolean attributes.

In Table 3 we present the augmented target table of the example database presented in Table 1. In this table we have two new attributes, corresponding to the two sequence patterns, S_1 and S_2, introduced above. For each patient and sequence pattern the value of the associated attribute is computed according to the existence of the sequence pattern in the patient related records. This way each patient is described by the primitive attributes in the target table, the sex, the date of birth and the two sequence patterns.

3.5 Classifier Induction - Phase 4

In the last step (line 17) we learn a classification model. We take the enlarged target table as input to a propositional classifier and learn a classification model. This model can include either primitive or new descriptors. In figures 2 and 3 bellow we present two examples of decision tree models that we can learn.

4 Experimental Evaluation

In this section we experimentally evaluate the *MuSer* framework. First we describe the configuration of the experiments. Then we present and discuss the obtained results.

4.1 Datasets

In this section we present the datasets that we used to evaluate the *MuSer* framework. We present the Hepatitis[1] and the Protein Prolog datasets [14, 20].

The Hepatitis dataset records long term monitoring, from 1982 to 1990, of patients having hepatitis B and C subtypes. One table provides personal data about patients. The other tables record blood, urinalysis and

[1] `http://lisp.vse.cz/challenge/`

biopsy examination data. The task that we address in this work is to discriminate between patients having hepatitis subtype B and C. In these experiments we select a subset of tables and performed limited feature selection. We explore table *pte030704*, that records basic information about the patients; table *bioe030704*, that records the results of biopsy made by each patient; table *ilabe030704* that records information about measurements in in-hospital examinations; table *labne030704*, that records information about measurements in in-hospital examinations and table *hemate030704*, that contains results on hematological analysis. All the frameworks and algorithms presented in this work explore temporal data from tables *ilabe030704* and *hemate030704*. Regarding the feature selection, we explore all information available in tables *pte030704* and *hemate030704* and select [22] only the GOT, GPT, TTT, ZTT, T-CHO, CHE, ALB, TP, T-BIL, D-BIL, I-BIL, ICG-15, PLT, WBC and HGB features of the *ilabe030704* table. We discretize each numerical feature using medical knowledge available in table *labne030704*. We use three bins: low, normal and high. The results registered in table *bioe030704* were used to label each patient. The label of each patient is the result obtained in the first biopsy registered for each patient. After preprocessing, in the subtype problem we end up with 206 hepatitis-B patients and 297 hepatitis-C patients.

The second dataset is concerned with protein secondary structure classification. The dataset consists of logical sequences describing the secondary structure of protein domains. This task is a multi-class (5 classes) classification problem and the goal is to predict one of the five most populated SCOP folds of alpha and beta proteins (a/b): TIM beta/alpha-barrel (721 examples), NAD(P)-binding Rossmann-fold domains (360 examples), Ribosomal protein L4 (274 examples), Cysteine hydrolase (441 examples), and Phosphotyrosine protein phosphatases I-like (290 examples). Inside round brackets we present the class distribution of the dataset.

4.2 Configuration of the Experiments

In this section we present the configuration of the experiments that we define to evaluate the performance of the *MuSer* framework.

The *MuSer* framework is a general architecture that can combine a large set of propositional sequence miners and classifiers. In this work we run the PrefixSpan sequence miner [23] to find all the frequent patterns and the C4.5 decision tree algorithm. In fact, we run the J48 algorithm, a C4.5 clone algorithm available in WEKA [27]. Moreover, we define another instance of *MuSer* that runs a linear kernel [24] in the last step of the framework. The SVM algorithm used is also part of the WEKA collection.

To best evaluate our contribution we set the parameters of the *MuSer* framework to values that can highlight the strengths of the architecture and facilitate a comparison against the RUSE-WARMR systems [10], that finds first-order logic itemsets and aggregates temporal data using a set of statistics. We set the PrefixSpan minimum support value, the λ parameter of the *MuSer* framework, equal to 90% and 80% and map either the top ten or the top twenty ranked patterns, the k parameter of the framework.

For each instance of the *MuSer* framework we run a stratified ten-fold cross validation procedure and compute the number of patterns, the run-time and the peak memory usage of each step. In the experiments that have been conducted in the multi-class problem, the protein fold classication problem, we use the round-robin evaluation strategy [13]. Thus, we build 20 binary models (this is five class problem) in each one of the ten folds and use the majority vote among all pairwise classification problems to classify each example available in the test set. In case of ties we decide in favor of the majority class. Moreover, for this dataset we report the mean number of patterns, the peak memory usage and the run-time of each algorithm/framework of the first 10 binary problems (f1 vs f2; f1 vs f23; f1 vs f37; f1 vs f55; f2 vs f23; f2 vs f37; f2 vs f55; f23 vs f37; f23 vs f55; f37 vs f55). We use this strategy to clarify the advantage of mining each database partition individually. Different from this strategy, we compute the mean generalization accuracy of each system using all the results obtained by the 20 binary classifiers.

Moreover we compute the Wilcoxon signed-rank hypothesis test [3] to assess if there is statistically significant accuracy improvements over the RUSE-WARMR framework. The null hypothesis of the test is that the median of the differences is zero. Moreover, we define the confidence level to be 95%.

Furthermore, we run the YAP[2] prolog compiler to: convert the Prolog data into a sequence database, eliminate non-discriminative patterns and project the PrefixSpan findings into the target table.

4.3 Results

In this section we present the results that we obtain in the experiments we run to solve the problems available in the Hepatitis and in the Protein dataset.

In tables 4, 6, 7 we present, respectively, the number of patterns found, the peak memory usage and

[1] http://www.dcc.fc.up.pt/∼vsc/Yap/

Table 4: Number of patterns found in each step of the *MuSer* framework

			Sequence Miner		Filters		C4.5
Datasets	Support	Top-K	Part A	Part B	Discriminative	Chi-square	TreeSize
Hepatitis	0.9	10	0.1 (0.3)	67.6 (29.6)	67.5 (29.7)	10.0 (0.0)	7.4 (1.8)
		20	0.1 (0.3)	67.6 (29.6)	67.5 (29.7)	20.0 (0.0)	10.2 (3.4)
	0.8	10	122.1 (59.1)	56010.6 (19036.9)	55888.5 (19060.9)	10.0 (0.0)	3.0 (0.0)
		20	122.1 (59.1)	56010.6 (19036.9)	55888.5 (19060.9)	20.0 (0.0)	6.0 (0.0)
Protein	0.9	10	10006.2 (651.8)	356.0 (19.8)	9923.3 (651.4)	10.0 (0.0)	7.5 (2.1)
		20	10006.2 (651.8)	356.0 (19.8)	9923.3 (651.4)	20.0 (0.0)	11.2 (2.6)
	0.8	10	80537.2 (4301.2)	3230.3 (364.2)	81779.5 (4355.3)	10.0 (0.0)	7.6 (1.8)
		20	80537.2 (4301.2)	3230.3 (364.2)	81779.5 (4355.3)	20.0 (0.0)	10.5 (3.9)

Table 5: Generalization accuracy of the *MuSer* framework

			C4.5		SVM	
Datasets	Support	Top-K	Accuracy	Wilcoxon p-value	Accuracy	Wilcoxon p-value
Hepatitis	0.9	10	64.10 (6.97)	0.33	65.50 (5.32)	0.18
		20	65.50 (5.72)	0.21	65.50 (6.06)	0.23
	0.8	10	79.90 (5.92)	0.00	80.10 (6.31)	0.00
		20	79.90 (5.92)	0.00	80.10 (6.31)	0.00
	RUSE	–	63.42 (6.92)			
Protein	0.9	10	76.86 (0.03)	0.00	76.14 (0.02)	0.00
		20	78.24 (0.03)	0.00	78.20 (0.03)	0.00
	0.8	10	79.11 (0.02)	0.00	76.86 (0.03)	0.00
		20	79.97 (0.03)	0.00	77.87 (0.03)	0.00
	RUSE	–	63.18 (1.18)			

the run-time required by each step of our framework. In each table we identify the dataset used as input, present the value of λ, the support value of the sequence miner, and k, the number of top ranked patterns we map, respectively, in columns one, two and three. Thus, each table row corresponds to an instance of our framework.

In Table 4 we present the sequence patterns found in each partition of the dataset. We use *Part A* and *Part B* column headers to identify the column where we present the patterns found in, respectively, the minority class and the majority class of the Hepatitis dataset. In columns six and seven we present the number of patterns that we get after running the discriminative and the chi-square filters. Moreover, in the last column we present the number of nodes of the decision tree learned. We present the size of the induced tree to highlight the relation between the support value of the sequence miner and the complexity of the generated decision tree models.

In tables 6 and 7 we use the same column descriptors used in Table 4 to present the peak memory usage (in kilobytes) and time (in seconds) needed by each step of the *MuSer* architecture.

In Table 5 we present the generalization accuracy (and standard deviation) of the models that we learn using either the C4.5 algorithm or the linear kernel SVM algorithm in the last step of the *MuSer* system. Moreover, we present the generalization accuracy that we get when we run the baseline algorithm. The RUSE-WARMR results were obtained when we run the C4.5 algorithm in the last step, and with the support value of the WARMR algorithm equal to 80% and by mapping the twenty most interesting patterns. Furthermore, we present the p-value of the Wilcoxon test that we obtained when we compare the generalization accuracy of each *MuSer* instance against the baseline, the RUSE-WARMR framework. We show two columns heading the (wilcoxon) *p-value* text. The first one shows the p-value of each instance of the *MuSer* algorithm that uses the C4.5 algorithm in the last step of the *MuSer* framework. The second column shows the same information for the experiments that we run the SVM to learn the classification model.

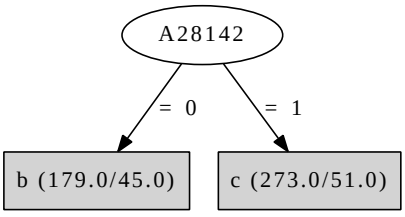

Figure 2: *MuSer*'s best induced decision tree for the Hepatitis subtype problem

In Figure 2 we present the tree model induced in the best run of the experiments we made with the Hepatitis dataset. This results were obtained in the first fold of the 10-cv evaluation. In this run we obtained

Table 6: Peak memory usage in each step of the *MuSer* framework

			Sequence Miner			Filter Chi-Square	Classifier	
Datasets	Support	Top-K	Part A	Part B	Mapping		C4.5	SVM
Hepatitis	0.9	10	613.2 (39.1)	1203.2 (37.1)	196930.4 (54680.0)	42759.6 (2435.1)	35544.4 (611.5)	55045.2 (213.0)
		20	612.4 (39.4)	1204.8 (37.6)	196312.0 (53586.2)	44692.4 (2312.5)	36026.4 (1174.2)	56818.4 (268.4)
	0.8	10	875.6 (22.4)	2006.0 (93.9)	3773773.2 (3341.2)	1263896.8 (495713.9)	36351.6 (713.5)	55445.2 (298.3)
		20	874.4 (23.8)	2006.0 (93.2)	3774758.4 (3256.2)	1270796.4 (502124.9)	35174.0 (788.1)	57100.8 (654.0)
Protein	0.9	10	916.4 (4.9)	752.7 (3.3)	636752.9 (12406.4)	442126.1 (48448.6)	36515.4 (681.5)	58082.6 (848.3)
		20	916.2 (4.7)	752.6 (3.4)	636936.5 (12417.7)	443263.8 (47970.7)	37865.6 (570.9)	60600.5 (908.3)
	0.8	10	1010.9 (7.0)	830.6 (6.9)	787832.9 (1047.4)	3347801.2 (288293.0)	36362.2 (561.2)	58195.1 (871.9)
		20	1011.1 (7.5)	830.6 (7.2)	787873.7 (879.3)	3202965.5 (367953.5)	37245.2 (1202.4)	59632.5 (1715.7)

Table 7: Run-time of each step of the *MuSer* framework

			Sequence Miner			Filter Chi-Square	Classifier	
Datasets	Support	Top-K	Part A	Part B	Mapping		C4.5	SVM
Hepatitis	0.9	10	0.0 (0.0)	0.0 (0.0)	3.2 (0.8)	0.3 (0.5)	0.2 (0.4)	0.3 (0.5)
		20	0.0 (0.0)	0.0 (0.0)	3.0 (0.5)	0.3 (0.5)	0.2 (0.4)	0.2 (0.4)
	0.8	10	0.0 (0.0)	18.6 (6.4)	911.9 (316.2)	10.4 (4.0)	0.1 (0.3)	0.3 (0.5)
		20	0.0 (0.0)	18.4 (6.4)	900.7 (306.8)	10.5 (4.2)	0.2 (0.4)	0.4 (0.5)
Protein	0.9	10	1.8 (0.3)	0.0 (0.1)	142.2 (9.4)	4.8 (1.1)	0.2 (0.4)	0.3 (0.5)
		20	1.8 (0.2)	0.0 (0.1)	141.2 (9.0)	4.1 (0.7)	0.2 (0.3)	0.4 (0.5)
	0.8	10	11.2 (0.8)	0.2 (0.3)	1186.3 (62.0)	39.4 (4.2)	0.2 (0.4)	0.3 (0.4)
		20	11.2 (0.9)	0.2 (0.2)	1190.8 (62.8)	41.5 (5.0)	0.2 (0.4)	0.5 (0.5)

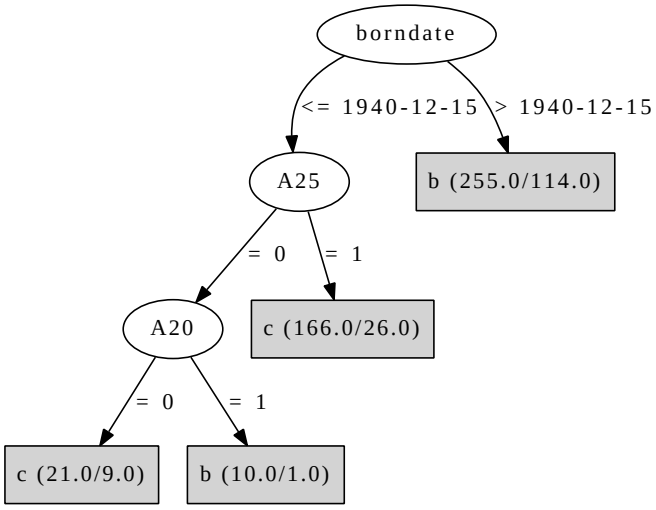

Figure 3: Example of a decision tree induced when the support value of *MuSer*'s sequence miner is set to 90%

an accuracy of 92%. This result was obtained by setting the support value of the sequence miner to 80% and mapping the ten most interesting patterns. In this specific example the split attribute that appears in the inner node of the tree is the sequence pattern A_{28142} : (30 33 37 48) that corresponds to the (che=normal d-bi=normal gpt=high t-bil=normal) itemset. This is a sequence of size one but we found trees and linear kernels that use sequences of larger sizes. In Figure 3 we present a tree that was found when we set the support value of the sequence miner to 90%. Different from the tree presented in Figure 2, this tree uses also primitive attributes, the borndate attribute.

4.4 Analysis of the Results

The results of the experiments show that important gains can be obtained if we explore time information using special purpose methodologies. In particular, we show that the *MuSer* framework obtained significant gains in two classification problems. Even without using all the information available in each dataset and without including background knowledge from experts, we get significant accuracy wins over the baseline in the hepatitis subtype problem and in the protein fold classification problem. Only when we run the *MuSer* framework with support value equal to 90% we do not get significant accuracy gains. In both problems we get accuracy gains of approximately 17 percentage points.

Moreover, the results that we present here are better than the ones obtained by algorithms that also explore the order of the events. In the Hepatitis dataset we get gains of approximately 1 percentage points over the work presented in [22], that was developed with the special purpose to explore the Hepatitis dataset. It is important to stress this result because we do not explore information available in all tables and do not include expert knowledge in the mining process. [22] uses computational expensive sub-graph patterns found by the Graph-Based Induction [19] as input to a decision tree learner.

In the protein problem we obtained generalization accuracy gains of approximately 4 percentage points over the work developed by Kersting [14]. This work introduces Logical Hidden Markov Models, a methodology that can include expert/background knowledge in the mining process.

Moreover, we run the XMuSer framework [11] but replacing the ILP learner with a propositional learner. We test the same C4.5 and the linear kernel SVM algorithms that we run the *MuSer* framework. With this modification we get a system that only differs in the mining step. Using the same configurations we use to evaluate the MuSer framework, we get in XMuSer runs a generalization accuracy losses of at least 15 and 4 percentage points in, respectively, the Hepatitis and Protein datasets. Thus, this *MuSer* gains are explained by the dual pattern mining strategy, that finds a large number of highly discriminative frequent sequences by mining each database partition individually.

Furthermore, we observe that in both problems the generalization accuracy increases with the decrease of the PrefixSpan support value. Moreover, if we increase the number of patterns to map from ten to twenty patterns, typically we get better results. This later behavior has some limitations as we explain in [9]. If we map a larger set of patterns the propositional classifier can overfit the data.

Regarding the analysis of the number of patterns found in each step of the *MuSer* framework, we get an exponential growth in the number of patterns found. The high growth rate is clear from the analysis of columns 4 and 5 of Table 4. The lower the support value the higher the number of patterns. This issue of sequence miners is well known [23] and was the source of many problems we have in this work. We were unable to find all patterns having support equal or less than 70%.

The effectiveness of the filters is clear when we analyze the number of patterns we found when we run the sequence miner with a minimum support value equal to 80%. Typically we get numerous irrelevant and redundant patterns. Without the use of both filters, mainly the chi-square filter that selects class related patterns, we could not obtain the presented results or would get worst generalization ability of the obtained classification models [9]. The number of patterns found by the sequence miner makes almost impossible to map all patterns and/or run any classifier on the augmented target table. We run some experiments with lower support values and we have to wait several weeks to learn a classification model in a single dataset fold.

Moreover, the number of patterns comes at high memory and run-time costs as we can see in Table 6 and Table 7. In these tables we do not present information about the discriminative filter, the one that removes matching patterns and is implemented with a simple matching operation that keeps non-agreeing patterns. We do so because the peak memory usage and run-times of this operation can be several orders of magnitude smaller than the other operations. This is especially clear when we run the sequence miner with low support values. The effectiveness of the discriminative filter can be apparent in datasets where we get many agreeing patterns.

Also, we think that the obtained patterns require the analysis of domain technicians. We are not specialists in hepatitis or protein fold structure but we obtain some sequence patterns that, we hope, can be useful to unveil the mechanisms of disease progression or protein structure.

5 Conclusions

In this work we presented *MuSer*, a framework that explores temporal information stored in Prolog databases. The main contributions of the proposed system is a method that explores temporal patterns presented in relational data without losing valuable relational information, the ordering of the events, and a dual mining strategy that explores each database partition individually.

Based on the excellent results of other algorithms that use the propositionalization technique, the architecture of our algorithm consists of four steps. After a pre-processing phase, where we convert the temporal information available in the relational database into a database of sequences, in the first phase, we find all sequence patterns using the efficient PrefixSpan algorithm. Then, in a second phase, considering the huge number of uninteresting and irrelevant patterns we introduce two filters. The discriminative filter that prunes agreeing patterns and a chi-square filter that sorts patterns using the chi-square statistic. Next we map the most interesting patterns into the relational target table. This way we obtain an enlarged target. In the last phase, we induce a decision tree model using the augmented table as input to C4.5 algorithm.

We experimentally evaluate the *MuSer* framework using two datasets and defining a large set of configurations. Our experiments show that the learned models are highly accurate and readable. Moreover, when we compare the results of *MuSer* against the baseline algorithm, the RUSE-WARMR algorithm we get significant better results. Remember that RUSE-WARMR aggregates temporal information, i.e., do not explore the temporal information conveniently. Moreover, in

the hepatitis subtype problem we get the best results we can find in the literature. In the Protein dataset we are aware of better results but we beat the work of Kersting[14]. This is an important work that was a reference during a set of years. These results are even more interesting because we do not use all the information available in the dataset and do not include domain expert knowledge.

The algorithm has some limitations related to the huge number of sequential patterns found and the inclusion of domain knowledge in the learning process. For instance, we try to explore the Financial dataset (`http://lisp.vse.cz/challenge/`), that has too many long sequence patterns with support values higher than 80%, and we were unable to run the original version of the PrefixSpan algorithm to find all patterns. With the original implementation we are unable to constrain the algorithm to find small sequence patterns. Moreover, even if run an algorithm that finds all patterns, we may be unable to map them. If we map all such patterns we get files with dozens of gigabytes that are intractable. Thus, in the future we will introduce strategies to constrain the search space, eliminate expensive filtering and mapping operations. An example of a recent work that considers such strategies is [21].

Acknowledgements

We acknowledge projects NORTE-07-0124-FEDER-000056/59 financed by the North Portugal Regional Operational Programme (ON.2 - O Novo Norte), under the National Strategic Reference Framework (NSRF), through the Development Fund (ERDF), and by national funds, through the Portuguese funding agency, Fundao para a Cincia e a Tecnologia (FCT). Authors also acknowledge the support of the European Commission through the project MAESTRA (Grant Number ICT-750 2013-612944). The first author was also funded by FCT and the Polytechnic Institute of Porto, Portugal, under the PhD grant SFRH/PROTEC/49634/2009.

References

1. Agrawal R, Srikant R (1994) Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases, Morgan Kaufmann, Santiago de Chile, Chile, pp 487–499
2. Agrawal R, Srikant R (1995) Mining sequential patterns. In: Eleventh International Conference on Data Engineering, Taipei, Taiwan, pp 3–14
3. Baron M (2013) Probability and Statistics for Computer Scientists, Second Edition, 2nd edn. Chapman & Hall/CRC
4. Bayardo RJ (1998) Efficiently mining long patterns from databases. SIGMOD Rec 27(2):85–93, DOI 10.1145/276305.276313, URL `http://doi.acm.org/10.1145/276305.276313`
5. Blockeel H, Sebag M (2003) Scalability and efficiency in multi-relational data mining. SIGKDD Explorations 5(1):17–30
6. Davis J, Burnside E, Dutra IC, Page D, Costa VS (2005) An integrated approach to learning bayesian networks of rules. In: Proceedings of the 16th European conference on Machine Learning, Springer-Verlag, Berlin, Heidelberg, ECML'05, pp 84–95, DOI 10.1007/11564096_13, URL `http://dx.doi.org/10.1007/11564096_13`
7. Dehaspe L, Toivonen H (1999) Discovery of frequent datalog patterns. Data Min Knowl Discov 3(1):7–36, DOI 10.1023/A:1009863704807, URL `http://dx.doi.org/10.1023/A:1009863704807`
8. Dolques X, Mondal K, Braud A, Huchard M, Le Ber F (2014) Rca as a data transforming method: A comparison with propositionalisation. In: Glodeanu C, Kaytoue M, Sacarea C (eds) Formal Concept Analysis, Lecture Notes in Computer Science, vol 8478, Springer International Publishing, pp 112–127
9. Ferreira CA, Gama J (2007) Rank ensemble features for constructive induction. In: Proceedings of the Workshop on General Artificial Intelligence, in the 13th Portuguese Conference on Artificial Intelligence (EPIA), Guimarães, Portugal, pp 45–57
10. Ferreira CA, Gama J, Costa VS (2008) RUSE-WARMR: Rule Selection for Classifier Induction in Multi-relational Data-Sets. In: 20th IEEE International Conference on Tools with Artificial Intelligence, IEEE Computer Society, Dayton, Ohio, USA, vol 1, pp 379–386
11. Ferreira CA, Gama J, Costa VS (2012) Predictive sequence miner in ilp learning. In: Proceedings of the 21st Inductive Logic Programming Conference, Springer, Windsor Great Park, United Kingdom, Lecture Notes in Computer Science, pp 130–144
12. Fournier-Viger P, Wu C, Gomariz A, Tseng VS (2014) VMSP: efficient vertical mining of maximal sequential patterns. In: Advances in Artificial Intelligence - 27th Canadian Conference on Artificial Intelligence, Canadian AI 2014, Montréal, QC, Canada, May 6-9, 2014. Proceedings, Springer, pp 83–94

13. Fürnkranz J (2002) Round robin classification. Journal of Machine Learning Research 2:721–747
14. Kersting K, Raedt LD, Raiko T (2006) Logical hidden markov models. J Artif Intell Res (JAIR) 25:425–456
15. Kramer S, Raedt LD, Helma C (2001) Molecular feature mining in hiv data. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, New York, NY, USA, KDD '01, pp 136–143, DOI 10.1145/502512.502533, URL http://doi.acm.org/10.1145/502512.502533
16. Krogel MA, Rawles S, Zelezný F, Flach P, Lavrač N, Wrobel S (2003) Comparative evaluation of approaches to propositionalization. In: Horváth T (ed) Inductive Logic Programming, 13th International Conference on Inductive Logic Programming (ILP–2003), Springer Verlag, Lecture notes in computer science, vol 2835, pp 197–214
17. Lee SD, Raedt LD (2004) Constraint based mining of first order sequences in seqlog. In: Database Support for Data Mining Applications: Discovering Knowledge with Inductive Queries, Lecture Notes in Computer Science, Springer-Verlag Berlin, vol 2682, pp 155–176
18. Lesh N, Zaki MJ, Ogihara M (1998) Mining features for sequence classification. Tech. Rep. TR98-22, MERL - Mitsubishi Electric Research Laboratories, 201 Broadway, Cambridge, MA 02139, URL http://www.merl.com/publications/TR98-22/
19. Matsuda T, Horiuchi T, Motoda H, Washio T, Kumazawa K, Arai N (1999) Graph-based induction for general graph structured data. In: Discovery Science, Lecture Notes in Computer Science, Porto, Portugal, vol 5808, pp 340–342
20. Mauro N, Basile TMA, Ferilli S, Esposito F (2011) Optimizing probabilistic models for relational sequence learning. In: Kryszkiewicz M, Rybinski H, Skowron A, Ra Z (eds) Foundations of Intelligent Systems, Lecture Notes in Computer Science, vol 6804, Springer Berlin Heidelberg, pp 240–249, DOI 10.1007/978-3-642-21916-0_27, URL http://dx.doi.org/10.1007/978-3-642-21916-0_27
21. Mauro ND, Esposito F (2013) Ensemble relational learning based on selective propositionalization. CoRR abs/1311.3735
22. Ohara K, Yoshida T, Geamsakul W, Motoda H, Washio T, Yokoi H, Takabayashi K (2004) Analysis of hepatitis dataset by decision tree graph-based induction. In: Proceedings of Discovery Challenge, pp 173–184
23. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC (2001) PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. ICDE pp 215–224
24. Platt JC (1998) Sequential minimal optimization: A fast algorithm for training support vector machines. Tech. rep., ADVANCES IN KERNEL METHODS - SUPPORT VECTOR LEARNING
25. Raedt LD (2008) Logical and Relational Learning. Cognitive Technologies, Springer, URL http://www.springer.com/computer/artificial/book/978-3-540-20040-6
26. Srikant R, Agrawal R (1996) Mining sequential patterns: Generalizations and performance improvements. In: Apers P, Bouzeghoub M, Gardarin G (eds) Advances in Database Technology EDBT '96, Lecture Notes in Computer Science, vol 1057, Springer Berlin Heidelberg, pp 1–17, DOI 10.1007/BFb0014140, URL http://dx.doi.org/10.1007/BFb0014140
27. Witten IH, Frank E (1999) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann
28. Yan X, Han J, Afshar R (2003) CloSpan: Mining Closed Sequential Patterns in Large Datasets. SDM pp 166–177
29. Zaki MJ (2000) Sequence mining in categorical domains: Incorporating constraints. In: CIKM, pp 422–429
30. Zelezný F, Lavrač N (2006) Propositionalization-based relational subgroup discovery with rsd. Machine Learning 62(1-2):33–63