Constrained Sequential Pattern Knowledge in Multi-Relational Learning

Carlos Abreu Ferreira[1], João Gama[2] and Vítor Santos Costa[3]

[1] LIAAD-INESC and ISEP - Polytechnic Institute of Porto, Porto, Portugal
[2] LIAAD-INESC and Faculty of Economics - University of Porto, Porto, Portugal
[3] CRACS-INESC and Faculty of Sciences - University of Porto, Porto, Portugal

Abstract. In this work we present `XMuSer`, a multi-relational framework suitable to explore temporal patterns available in multi-relational databases. `XMuSer`'s main idea consists of exploiting frequent sequence mining, using an efficient and direct method to learn temporal patterns in the form of sequences. Grounded on a coding methodology and on the efficiency of sequential miners, we find the most interesting sequential patterns available and then map these findings into a new table, which encodes the multi-relational timed data using sequential patterns. In the last step of our framework, we use an ILP algorithm to learn a theory on the enlarged relational database that consists on the original multi-relational database and the new sequence relation.
We evaluate our framework addressing three classification problems and running three different sequence miners. Using each one of the sequence miners we can find: all the frequent sequences, all the closed sequences or all the maximal sequences.

1 Introduction

Multi-relational databases are widely used to represent and store data. A multi-relational database is often composed by a *target* table and by a number of *fact* tables. The target table will represent the main objects of interest (say, patients in a medical domain); fact tables will represent the information being accumulated about the entities in the target table (say, medical visits or drug usage in the medical domain). We expect target tables to be relatively stable or to grow slowly over time; in contrast, fact tables may grow quickly. Moreover, quite often the information stored in fact tables is time-based and consists of *sequences* that reflect the evolution of a phenomenon of interest. Referring back to the medical domain, a patient is subject to a sequence of examinations, where a set of measurements, corresponding to results of different analysis, are taken.

In this work, we start from the hypothesis that the evolution of these measurements, as encoded in the fact tables, may hold relevant information for the diagnosis. The problem we address here is *how best to explore such information?* More precisely, we focus on how to learn highly descriptive and accurate decision models given multi-relational data with sequences.

The main goal of this work thus consists of exploiting heterogeneous temporal information stored in the multi-relational sequences of events. To do so, we propose a framework that encodes *timed data*, stored in one or several fact tables, into a separate sequence relation, uses an optimized sequence learner to find the most interesting such sequences, maps back the sequences to the relational database, and then learns a theory on the extended multi-relational database. The extended database thus contains all primitive relations and a sequential relation that stores time events for each example.

We name our framework `XMuSer`(eXtended MUlti-relational SEquential patteRn knowledge learning). It executes in five steps. First, we encode the multi-relational timed data into a sequence database. In this new database each example is a heterogeneous sequence that was built regarding both intra-table and inter-table relations within the temporal data. In a second phase, we use a sequence miner to find frequent sequences in the sequence database. In a third phase, we select interesting frequent sequences. By interesting we mean discriminative ones, that is, those that appear in only one class and do not appear in the others. We further use a filter to select the top-k class related sequential patterns. The fourth phase maps back the newly found sequential patterns by building a new relation where each target example is characterized by the presence or absence of one of the top-k patterns. In a last phase, we apply an Inductive Logic Programming (ILP) algorithm to learn a theory from the enlarged database, that is from the union of the original multi-relational database with the sequence relation.

This methodology allows us to explore multi-relational datasets that have different types of timed data, either sequence data or time-series data. On the one hand, we can benefit from computationally efficient sequential miners such as *PrefixSpan* [10] or *CloSpan* [14] to find the most remarkable sequential patterns. On the other hand, we still have access to the original data and can take advantage of the flexibility of ILP to learn in the extended multi-relational dataset. Indeed, we argue that the first step provides a good insight into the search space, and may enable `XMuSer` to perform better than classical ILP based algorithms in large search spaces. We should observe that the sequence miner and ILP learning algorithm are decoupled: we can use other sequential miners such as *SPIRIT* [9], that constrains the search space using regular expressions.

We experimentally evaluate our methodology in two datasets, addressing three prediction problems: discriminating between two hepatitis subtypes and discriminating between two sets of hepatitis fibrosis degrees, in the Hepatitis dataset, and discriminating between successful and unsuccessful loans, in the Financial dataset. The two datasets, Hepatitis and Financial, were originally introduced at PKDD Discovery Challenges.

The contributions of our work are therefore: **(i)** We introduce a framework to explore heterogeneous sources of time data, either sequence data or time-series data, stored in multi-relational database using propositional sequence miners. Our ILP based framework is highly efficient and gains both from the descriptive power of the ILP algorithms and the efficiency of the sequential miners. More-

over, we do not use classical aggregation strategies, like time windows, neither neglect valuable logic-relational information. **(ii)** We develop a new methodology to translate any multi-relational timed database into a sequence database.

Next we present the main ideas of our work and related work. In Section 3 we define concepts that will be useful to Section 4 where we discuss in detail our framework. Next, Section 5 describes the experimental setup and presents and discusses the results. We then present our conclusions and future research directions.

2 Methods and Related Work

In this section we present an overview of related work that inspired us and contributed to the overall `XMuSer` framework. First, we present sequential miners, a major component of our architecture. We then present ILP based algorithms that inspired us to design a framework that can benefit from using the logic-relational information without using ILP search mechanism to find sequential patterns. Last, we present different approaches to explore temporal patterns occurring in multi-relational datasets.

There exists a wide range of algorithms that can explore sequential data in an efficient way. To the best of our knowledge, Agrawal and Srikant introduced the sequential data mining problem in [2]. In [13] the same authors introduce the *GSP* algorithm, an algorithm that generalizes the original sequential pattern mining problem and that is also inspired in search procedure of the well known *APRIORI* algorithm [1]. *GSP* uses a candidate-generation strategy to find all frequent sequences, and uses a lattice to generate all candidate sequences. A more efficient approach to find the set of all frequent sequences is *PrefixSpan* [10]. This algorithm is inspired by pattern-growth strategies to efficiently find the complete set of frequent sequences. In real problems these algorithms usually find a huge number of uninteresting sequence patterns. To solve this issue of sequential miners, algorithms such as *CloSpan* [14], that returns a set of closed sequential patterns, and *SPIRIT* [9] that constrains the search space using regular expressions were developed.

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.

Algorithms that use Inductive Logic Programming (ILP) were the first ones to explore successfully the richness of multi-relational data. ILP approaches have an enormous representational power but are often criticized for lacking scalability [4]: ILP algorithms may not be very effective for the large search spaces induced by sequence databases.

One approach to solve the above mentioned issue is to use propositionalization with ILP [16]. The idea is to augment the descriptive power of the target table by projecting clauses (new attributes) on the target table.

Even with recent progress on scalability there exists multi-relational data which remains 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* [6] is to use aggregation methodologies, unfortunately losing relevant time information. In View Learning [5] 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.

Other ILP based approaches develop specific techniques aimed at exploring the space and time information available in multi-relational datasets. The works of [11, 3, 7] introduce special purpose formalisms to find first-order sequential patterns in multi-relational datasets but, by using ILP based search, they suffer from traditional ILP limitations. To explore large spaces they must constrain the search space or use heuristics, otherwise the problems will be intractable. Thus, they may fail to find interesting patterns.

3 Preliminaries

We will start by remembering some key concepts of sequence mining and define the sequence mining problem.

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 $\beta = b_1 b_2 \ldots b_t$, a 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, α) where *sid* is the sequence identification and α is a sequence. The *count* of a sequence α in a sequence database D, denoted $count(\alpha, D)$, is the number of sequences in D containing 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 at least a support value equal to λ. Each one of the obtained sequences is also known as a *frequent sequence*.

When searching for all sequential patterns in a sequence database we face one of the major problems of sequential miners, the huge number of redundant and non-interesting findings. To alleviate this we can use other sequential miners that find a constrained set of sequential patterns. A frequent sequence α is a *closed sequential pattern* if there exists no proper supersequence β having the same support as α. A frequent sequence α is *maximal sequential pattern* if it is not a subsequence of any other frequent sequence. The set of maximal sequential patterns is a subset of all closed sequential patterns.

Moreover, in a classification problem, some sequential patterns have low discriminative power and could be eliminated. Thus, we introduce the concept of Discriminative Sequential Patterns to select inter-class discriminative sequential patterns.

Definition 1. (Discriminative Sequential Patterns) *Consider two sequence database partitions D_1 and D_2, and a support threshold λ. We define the set of discriminative sequential patterns using the xor ($\dot{\vee}$) operator, $S_{disc} = \{\alpha \mid support(\alpha, D_1) \geq \lambda \ \dot{\vee} \ support(\alpha, D_2) \geq \lambda\}$.*

Such discriminative patterns and other interesting sequential patterns represent valuable information that can be extracted from a sequence database and that could be useful at theory learning time. To do so, we introduce the concept of Sequence Relation and Enlarged Database.

Definition 2. (Sequence Relation and Enlarged Database) *Consider a multi-relational database* $\mathbf{r}$, *a sequence database D coded from* $\mathbf{r}$, *where each sequence id equals the primary key of* $\mathbf{r}$ *target table. Also consider the set of sequential patterns S obtained from solving the sequential mining problem. We define the sequence relation,* $\mathbf{r}_{sr}$, *to be the set of tuples $(sid, \alpha_1^B, \alpha_2^B, \ldots, \alpha_n^B)$ where sid is the sequence id in D and α_i^B is a binary attribute whose value is obtained according to the projection of the sequential pattern α_i in the sequence database. The enlarged database is the database resulting from the union of the multi-relational database and the sequence relation. Formally we define the enlarged database as being* $\mathbf{E_r} = \mathbf{r} \cup \mathbf{r_{sr}}$.

Our algorithm is suitable to explore timed data present in multi-relational databases, mainly sequence data. Thus, we introduce a strategy to code time-series data into a sequence database.

Definition 3. (Sequence coding) *Given a multi-relational database* $\mathbf{r}$, *having fact tables that register time events. We define sequence coding as a procedure that for all, or a subset, of database fact tables,* $\mathbf{r_i} \in \mathbf{r}$, *translates all, or a subset, of attributes* $\mathbf{r_i.A_j}$ *into a sequence database.*

4 The XMuSer Algorithm

In this section we present XMuSer. This new methodology explores multi-relational information, mainly heterogeneous sequential data widespread across a multi-relational dataset, to learn a theory. The main idea of the algorithm is to explore work developed in the sequential pattern mining field to include time information in the ILP learning process.

The framework has five main steps. In the first phase we code the temporal data into a sequence database. In a second phase, we run a sequence miner to find all sequential patterns in each class partition. In a third phase, we apply two filters to select the most discriminative or class related patterns. In a fourth phase, for each example in the target table, we built a relation where the example is characterized by presence or absence of the most interesting sequential patterns. Last, we learn a theory on the enlarged database, where enlarged database is the union of the original database with the new sequence relation.

Algorithm 1 XMuSer pseudo-code

input a multi-relational database $\mathbf{r}$; two thresholds λ, the sequence miner support value, and k, the number of most interesting patterns to retain
output a classifier model

Sequence Coding
$\mathbf{s} \leftarrow$ SequenceCoding($\mathbf{r}$)

Frequent Sequential Patterns
$\mathbf{s}_1, \ldots, \mathbf{s}_l \leftarrow$ partition($\mathbf{s}$)
for $i = 1$ **to** l **do**
  $sf_i \leftarrow$ SequenceMiner($\mathbf{s}_i$,λ)
end for

Filtering
$S_{disc} \leftarrow$ discriminate($sf_1, \ldots, sf_l$)
$S_{interesting} \leftarrow$ **SU**(S_{disc}, k)

Mapping
$\mathbf{r}_{sr} \leftarrow$ Mapping($S_{interesting}$, $\mathbf{r}_{targetExamples}$)
$\mathbf{E}_\mathbf{r} \leftarrow \mathbf{r} \cup \mathbf{r}_{sr}$

Learn a Theory with an ILP algorithm
ILP Algorithm($\mathbf{E}_\mathbf{r}$)

Blood Analysis

ID	Date	RBC	WBC
1	19750102	high	normal
1	19780203	high	high
2	19770107	high	low

Target table
Patient

ID	Sex	BornDate	Class
1	M	19520109	a
2	F	19750123	b

Urinalysis

ID	Date	Exam	Result
1	19741201	plt	high
1	19750102	alb	normal
1	19750102	ttp	normal
1	19760204	alb	normal
2	19800403	alb	normal

Fig. 1. Database Relations. ID is the patient ID, Date is in numeric format, RBC and WBC are blood parameters, and we show alb, plt and ttp urine exams.

Algorithm 1 presents the pseudo-code for `XMuSer`. Next, we explain each one of the major components in self-contained subsections. Throughout, we follow an illustrative example, a classification problem, at Figure 1. This example is inspired on the relational Hepatitis dataset. The example has three tables registering the follow-up of two patients. One of the tables is the target table, named *Patient*, where each record describes each patient, identified by a masked ID, and registers the class of each patient. The other two tables are fact tables registering timed blood analysis and urinalysis examinations.

Data Coding The algorithm that we present next takes a multi-relational dataset as input, usually represented as a database of Prolog facts. To explore the richness of this representation, and namely temporal patterns, we introduce a strategy that converts multi-relational timed data into an amenable sequence

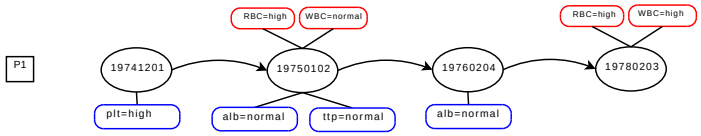

Fig. 2. Temporal Patient Events for Patient One

Table 1. Transformed Event Database. We introduce three new patients that will be useful in the following steps.

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

database. First, we find all relations that have temporal records. Second, we sort the records in these relations by time order. We thus obtain a chronological sequence of multiple events for each example. Figure 2 shows an example event sequence for patient one. The sequence includes a sequence of blood and urine analysis. Third, we built a temporal attribute-value sequence for each example. In this new sequence each item corresponds to all records registered at a given date/time. We have *(plt=high) (RBC=high, WBC=normal, alb=normal, ttp=normal) (alb=normal) (RBC=high, WBC=high)* for patient one. We then define a one-to-one coding map $f : Attributes \times Values \longrightarrow \mathbb{N}$. This mapping associates an *unique* number to each attribute-value pair. In the example, we use the map to code the attribute-value sequence into an integer number sequence. The definition of this map is done according to the type of attributes in each database relation. The mapping assumes discrete attribute (continuous attributes will be discretized beforehand).

Table 1 shows the transformed sequence database: each sequence tuple corresponds to an example in the target table and each subsequence corresponds

Table 2. Sequence Database Partition with Classes

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

Table 3. Sequential patterns found in each class partition

Class **a** Sequential Patterns	Class **b** Sequential Patterns
(3): 2	**(3)**: 2
(3) (7): 2	**(3) (7)**: 2
(3 5): 2	(3 4) (7): 2
(3 5) (7): 2	(4): 2
(5): 2	(4) (7): 2
(5) (7): 2	**(7)** :2
(7): 3	
(8): 2	
(8) (7): 2	

to all one-time events. In the example, if we define the one-to-one map to be *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,* then, patient one sequence of events is thus coded as (9) (3 5 7 8) (7) (3 6).

This stage obtains a sequence suitable to be processed by a flat sequential miner. In Table 1 we present a sequence database registering the coded sequence of patients one and two, and three new patients that will be useful in the following steps.

Finding Frequent Sequential Patterns We run a sequence miner in each partition of the sequence database (Table 2) in order to find *frequent* sequences, that is, having a support equal or higher than a user defined threshold. Thus, for each partition and each class we obtain all frequent patterns sf_i. In Table 3 we present the patterns that we found by running *PrefixSpan* algorithm (setting support value equal to 60%) in the two class partitions, the class **a** and class **b** partitions. Notice that several patterns occur in both classes.

Filtering The previous step usually obtains a large number of findings. We would like to retain highly discriminative, class correlated, patterns and drop uninter-

Table 4. Inter-Class Discriminative Sequential Patterns

	Discriminative Sequential Patterns
1	(3 4) (7)
2	(3 5)
3	(3 5) (7)
4	(4)
5	(4) (7)
6	(5)
7	(5) (7)
8	(8)
9	(8) (7)

Table 5. Sequence Relation

ID	S_1	S_6
1	0	1
2	1	0

esting and/or redundant patterns. To do so, we introduce a *Discriminative Filter* that selects the set S_{disc} of sequences that have support above some parameter λ in one and only one partition. The filter is implemented by using matching. The set of discriminative sequential patterns, S_{disc}, is formally defined in Section 3. In the example above, Table 4 presents the nine discriminative patterns obtained by applying this filter.

Note that even after eliminating non-discriminative patterns, non-interesting patterns may remain. We use *Symmetrical Uncertainty(SU)* [15] to sort and select the top k class-related features. The SU evaluates the worth of an attribute by measuring the symmetrical uncertainty with respect to the class. For instance, if we set $k = 2$ the algorithm would select S_1, or (3 4) (7), and S_6, or (5), patterns. The sequential composition of the two filters thus results in two interesting features/sequences that we will use to build the final classification model.

Mapping Back Interesting Sequences We have coded the timed data as a sequence database. We now want to build a relation where each example is characterized by the most interesting sequences. Table 5 shows the key idea in our approach. We construct a new table, the *sequential relation*, with an entry per example and with $k + 1$ attributes: the top-k most interesting features and the example ID. If the sequence associated with the new attribute is a subsequence of the example sequence at the sequence database, the new attribute takes value one. Otherwise, it takes the value zero.

Learning a Theory Last, we add the new sequential relation to the initial tables and use an ILP algorithm, such as *Aleph*, to learn a set of clauses. One illustrative example of a found clause is:

```
patient_info(A,B,C,a):-
  blood_analysis(A,D,high,normal),
  urinalysis(A,E,plt,high),
  sequence_relation(A,0,1).
```

This clause calls the predicate *blood_analysis*, the predicate *urinalysis* and the predicate associated with the sequence relation, predicate *sequence_relation*. The clause explains (or covers) patient number one, a patient from class **a**, the majority class.

5 Experimental evaluation

In this section we describe the configurations and results of our experimental evaluation.

5.1 Experimental Configuration

We evaluated our algorithm in two real-life datasets obtained from the PKDD data-mining competitions. We experimented three prediction problems: discriminating between two hepatitis subtypes and discriminating between two sets of hepatitis fibrosis degrees, in the Hepatitis dataset, and discriminating between successful and unsuccessful loans, in the Financial dataset.

Throughout, and as our framework can use any sequence miner, we tested: *PrefixSpan* algorithm to find all frequent sequences; *CloSpan* algorithm to find all closed sequential patterns; and a post-processing procedure, that works over the *CloSpan* findings, to select maximal sequential patterns, a subset of closed sequential patterns. This post-processing technique was named *MaxSpan* to easily present the results. We also use *Aleph* to learn a logical theory. We applied the following predefined parameter configuration: $k = 30$ and test two different λ values, 90% and 80%. For comparison purposes we also run the stand-alone *Aleph* algorithm to solve each one of the four problems.

We evaluate our framework using a ten-fold cross-validation procedure and compute: the mean number of patterns found after each step of `XMuSer`, the mean generalization accuracy and standard deviation of `XMuSer` and the mean time spent to complete each step of `XMuSer`. Using this same procedure, the 10-CV, we also compute: the mean number of rules learned by the *Aleph* algorithm, both as a component of `XMuSer` and as a stand-alone algorithm. We also compute the generalization accuracy and the time spent by the stand-alone *Aleph* algorithm. Concerning the generalization accuracy, we also compute the *Wilcoxon* hypothesis test p-value to measure how significantly our algorithm differs from the reference algorithm, the stand-alone *Aleph* algorithm. We set the level of significance equal to 0.10.

We use the same background knowledge and bias when running *Aleph* algorithm, either as the last component of `XMuSer` or as stand-alone. The major difference is that in `XMuSer` we introduce an extra sequence relation. The bias is relatively simple, relying on the predefined tables in the database. We evaluate our framework performance on unseen data. We further analyze the contribution of each step of `XMuSer`. We analyze the number of patterns found after each step, including the compactness of the theory learned by the *Aleph* algorithm. We analyze the generalization accuracy and run-time needed to complete each step of `XMuSer`.

5.2 Datasets and Tasks

We present below the two datasets that we used to evaluate our ILP based classifier: Hepatitis and Financial datasets. Both these datasets are available to

Table 6. Mean Number of Patterns in each step of `XMuSer` and of the stand-alone *Aleph* algorithm

		`XMuSer` (with Aleph)															Stand-alone Aleph (Rules)
		Seq. Miner						Disc.			**SU**-rank			Aleph(Rules)			
		PrefixSpan		CloSpan		MaxSpan		PrefixSpan	CloSpan	MaxSpan	PrefixSpan	CloSpan	MaxSpan	PrefixSpan	CloSpan	MaxSpan	
	λ	Pos.	Neg.	Pos.	Neg.	Pos.	Neg.										
Hepatitis Subtype	0.9	68	0	51	0	26	2	68	51	24	30	30	30	124	124	124	129
	0.8	56011	122	21058	93	4260	38	55889	21151	4294	30	30	30	115	114	117	
Hepatitis Fibrosys	0.9	166	3	64	3	40	4	164	63	40	30	30	30	47	49	50	67
	0.8	22809	877	2773	610	1441	228	22134	3049	1648	30	30	30	50	45	47	
Financial	0.9	137	30	136	12	63	11	109	125	64	30	30	30	166	160	161	169
	0.8	7622	818	7605	313	2682	238	6900	7341	2865	30	30	30	149	150	142	

Table 7. Mean Generalization Accuracy: `XMuSer` against stand-alone *Aleph*

	λ	`XMuSer` (With Aleph)			Stand-alone Aleph	Wilcoxon p-value		
	λ	PrefixSpan	CloSpan	MaxSpan		PrefixSpan	CloSpan	MaxSpan
Hepatitis Subtype	0.9	0.79(0.10)	0.79(0.10)	0.79(0.10)	0.78(0.11)	0.269	0.201	0.524
	0.8	0.80(0.10)	0.80(0.10)	0.80(0.09)		0.073	0.093	0.308
Hepatitis Fibrosys	0.9	0.64(0.06)	0.65(0.05)	0.65(0.05)	0.58(0.09)	0.097	0.020	0.020
	0.8	0.61(0.10)	0.66(0.06)	0.64(0.13)		0.407	0.029	0.192
Financial	0.9	0.73(0.06)	0.75(0.04)	0.74(0.05)	0.71(0.07)	0.854	0.021	0.027
	0.8	0.74(0.04)	0.74(0.05)	0.73(0.07)		0.138	0.096	0.029

download at the PKDD challenge web page (`http://lisp.vse.cz/challenge/CURRENT/`).

The Hepatitis dataset consists of seven tables registering a long term, from 1982 to 1990, monitoring of 771 patients having hepatitis B or C. One table provides personal data about patients. The other tables record blood and urinalysis examinations. We address two classification problems. Our first task is to discriminate between patients having B and C hepatitis. The second task is to determine the degree of liver fibrosis. Following previous work [12], we study fibrosis degree 2 and 3 against fibrosis degree 4. We perform limited feature selection, based on the dataset description [12]. We select GOT, GPT, TTT, ZTT, T-CHO, CHE, ALB, TP, T-BIL, D-BIL, I-BIL, ICG-15, PLT, WBC and HGB features. As these features are numerical, and in fact take a wide range of values, we discretized each feature according to medical knowledge. We use three bin values, low, normal and high. For the sub-type problem we end with 206 hepatitis-B patients and 297 hepatitis-C patients. For the Fibrosis problem, we have 209 patients of {2,3}-class and 67 of the {4}-class.

The Financial dataset includes eight tables with data about clients and their accounts. A number of tables store static information on accounts, clients and regional demographics. The remaining tables register information concerning credit card types, payments, transactions, and loans for each account. The sequence database relies on the *Balance* attribute only. This attribute was then discretized into three levels, low, normal and high, based on the Chebychev inequality. Our target is predicting successful loans: 606 loans were classified as successful and 76 unsuccessful.

5.3 Results

We first present results that show the contribution of the filtering methodology and the number of rules learned by the *Aleph* algorithm (Table 6), both as a component of the `XMuSer` framework or a stand-alone algorithm. Then, we present results concerning the generalization accuracy of `XMuSer` and the stand-alone *Aleph* algorithm (Table 7). Finally, we present the running time of each component of the `XMuSer` and the running time of the stand-alone *Aleph* algorithm (Table 8). Moreover, we present the results that we obtain by using each one of the three sequence miners: *PrefixSpan*, *CloSpan* and *MaxSpan* (*CloSpan* + post-processing procedure).

In Table 6 we present, for each one of the three problems that we address and the two values of the λ parameter (90% and 80%), the mean number of sequential patterns found in each phase of `XMuSer` and the number of rules that were learned by *Aleph*. As we run each one of the sequential miners in each class partition, and for each one we report the number of sequential patterns found in both the positive (the Pos. column) and negative (the Neg. Column) class partitions. The positive partition is the majority class of each problem. We also present the number of discriminative patterns (the Disc. column), the patterns selected after applying the Discriminative filter, and the number of patterns that were selected using the SU filter (the SU-rank column). The maximum number of patterns to select, using SU metric, is an input parameter, the k in the framework pseudo-code above, that we set before the experiments to be 30. We obtain 30 patterns to build each sequence relation.

In Table 7 we present the mean generalization accuracy and standard deviation, inside brackets, of both the `XMuSer` framework and the stand-alone *Aleph* algorithm, the reference algorithm. This table reports results that were obtained by setting $\lambda = 90\%$, $\lambda = 80\%$ and $k = 30$, two input parameters of our framework. We also present the p-value obtained by running the *Wilcoxon* signed-rank test. We present the p-value for each sequence miner and using the stand-alone *Aleph* algorithm as the baseline.

In Table 8 we present the mean run-time of each component of our framework and the mean run-time of the stand-alone *Aleph* algorithm. The run-time for *PrefixSpan* adds the time needed to find sequential patterns in the positive and negative class partitions. These results were obtained by running `XMuSer` with the same settings that were used to measure the mean generalization error.

Table 8. Mean Run-Time, in seconds, of each `XMuSer` phase and the stand-alone *Aleph* algorithm

		`XMuSer` (With Aleph)												Stand-alone Aleph (Rules)
		Seq. Miner			Disc.			*SU*-rank			Aleph(Rules)			
	λ	PrefixSpan	CloSpan	MaxSpan	PrefixSpan	CloSpan	MaxSpan	PrefixSpan	CloSpan	MaxSpan	PrefixSpan	CloSpan	MaxSpan	
Hepatitis Subtype	0.9	3	3	13	0	0	0	3	3	3	8	8	6	129
	0.8	18	16	39	0	0	0	581	258	58	6	7	7	
Hepatitis Fibrosys	0.9	1	1	8	0	0	0	3	2	1	5	5	5	67
	0.8	3	3	17	0	0	0	138	20	11	6	4	5	
Financial	0.9	1	1	12	0	0	0	3	5	2	1	32	28	169
	0.8	3	3	20	0	0	0	151	215	75	2	22	23	

5.4 Analysis

Our goal in this study is to prove that our framework can effectively explore the timed data present in multi-relational databases.

Thus, we must be able to address multi-relational datasets, usually recording a large amount of data, and where each table has a wide number of attributes, discrete or continuous ones. Quite often, datasets are unbalanced. Unfortunately, most previous works that addresses these same problems, performs preprocessing steps, such as balancing the datasets, thus making it impossible to make a fair comparison against other works.

We present experimental results for $\lambda = 90\%$ and $\lambda = 80\%$. Lower values of λ cause a memory explosion. Further experiments with other values of k we do not show significant changes in performance, although larger k might provide more interesting patterns.

Given these constraints, we believe that we obtained very good results. In all three classification problems, `XMuSer` obtained significant wins against the stand-alone *Aleph* algorithm. This is especially clear when we use *CloSpan* algorithm to find sequential patterns. The best accuracy gains were obtained in the Hepatitis Fibrosis task. In this problem our mean accuracy gains range from 3% to 8%. In Subtype we obtained a significant win for a support value $\lambda = 80\%$. These results are among the best that we could find in related work that addresses this problem, and indeed outperform a previous work where the final step is a propositional classifier [8].

We further observed that the theories that we learn in all four tasks use sequences, and that `XMuSer` generates more compact theories. Indeed, one important point suggesting that we are in the right direction is a clear relation between improvements in the obtained classification models and the number of timed data attributes that we use to build the sequence database. In both Hepatitis tasks we eventually use a larger number of time data attributes to construct theories. In the financial task we benefit much less of time attributes: we could only use information from two time attributes in Thrombosis and one time attribute in Financial.

As expected, `XMuSer` execution time takes longer to learn than stand-alone *Aleph* algorithm, with execution time heavily depending on λ. To explain this behavior, consider the information of both Table 6 and Table 8. If we run `XMuSer` by setting a low λ we get a huge increase in the number of sequential patterns found and thus we get a run-time overhead in the initial phases of `XMuSer`. This is especially noticeable in two steps for the *PrefixSpan* and the *SU* filter. This problem is typical of sequential miners, the lower the support the higher the execution time and the number of patterns found. This is clear when we run *CloSpan* and *MaxSpan* technique to find a subset of sequential patterns. We get a small number of patterns and, despite the overhead of the post-processing of *MaxSpan*, we get an overall short run-time. This is clear if we observe the run-time of the *SU* filter for each sequence miner. Moreover, the last component of `XMuSer` uses more time when coupled with either *CloSpan* and *MaxSpan*. This is the result of getting a low number of redundant sequential patterns, sequential

patterns that have the same support. The best results where obtained when we used *CloSpan* algorithm. This indicates that some frequent subsequences patterns (with the same support as a supersequence) that are not found by *MaxSpan* can contain valuable information.

The overhead observed in the *SU* filter does not result from computing the well known metric but from the preprocessing needed to build the contingency table. In order to compute the *SU* metric, for each pattern we need to know if that pattern is a subsequence of each example, which can be expensive.

Notice that the ILP search space of `XMuSer` is larger by a fact of 2^k, where k is the number of patterns/attributes in the sequential relation. On the other hand, our experiments do not show a significant variation in running times between *Aleph* in `XMuSer` and stand-alone. Observation suggests that this is because sequence-based rules can often cover a large number of positive examples, and prune well the search space.

These developments argues for the development of a sequential miner that can find class closed sequential patterns, i.e., that can find sequential patterns using class information to prune the search space and find discriminative patterns, not relying on post-processing strategies and filters. By using it we can go further ahead and explore even more timed data available in relational datasets.

6 Conclusions and Future Work

In this work, we presented `XMuSer`, a multi-relational framework that explores temporal information stored in a database. The methodology is an effective alternative to previous ILP-based approaches that incorporate timed data in the final first-order theory by either using time aggregation strategies, like time window aggregation, or by refining ILP clauses or predicates that explicit explore time. Differently from these approaches, our methodology, can take advantage of the strengths of sequential miners to explore efficiently any kind of timed data. Our approach can be used in conjunction with any classical ILP algorithm.

Moreover, we developed sequence coding, a technique to code timed data into a sequence database. We introduce the sequential relation to encode the patterns found by the sequential miner. Last, we define the enlarged database, a database that is the union of the original database and the sequence relation, and induce a theory over this relation. This methodology allows us to explore the valuable logic-relational information available in multi-relational datasets, especially temporal patterns, with good results as shown by empirical evaluation.

In order to improve our framework we will explore other ILP algorithms and develop a sequence miner that can use class information to prune the search space and find more interesting patterns. One way is to update *CloSpan* algorithm. New sequence miners, say *SPIRIT*, should be considered as they apply well to our algorithm.

Acknowledgments: This work was supported by the Portuguese Foundation for Science and Technology (FCT) under the projects KDUS (PTDC/EIA-EIA/098355/2008) and HORUS (PTDC/EIA-EIA/100897/2008). Carlos Abreu

Ferreira was financially supported by the Portuguese Polytechnic Institute of Porto (ISEP/IPP).

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of the 20th International Conference on Very Large Data Bases. pp. 487–499. Morgan Kaufmann, Santiago de Chile, Chile (1994)
2. Agrawal, R., Srikant, R.: Mining sequential patterns. In: Eleventh International Conference on Data Engineering. pp. 3–14. Taipei, Taiwan (1995)
3. de Amo, S., Furtado, D.: First-order temporal pattern mining with regular expression constraints. Data & Knowledge Engineering 62(3), 401–420 (2007), including special issue: 20th Brazilian Symposium on Databases (SBBD 2005)
4. Blockeel, H., Sebag, M.: Scalability and efficiency in multi-relational data mining. SIGKDD Explorations 5(1), 17–30 (2003)
5. Davis, J., Burnside, E., Ramakrishnan, R., Costa, V., Shavlik, J.: View learning for statistical relational learning: With an application to mammography. In: Proc. of the 19th International Joint Conference on Artificial Intelligence. pp. 677–683. Professional Book Center, Edinburgh, Scotland, UK (2005)
6. Dehaspe, L., Toivonen, H.: Discovery of frequent DATALOG patterns. Data Mining and Knowledge Discovery 3(1), 7–36 (1999)
7. Esposito, F., Di Mauro, N., Basile, T.M.A., Ferilli, S.: Multi-dimensional relational sequence mining. Fundamenta Informaticae 89(1), 23–43 (2009)
8. Ferreira, C.A., Gama, J., Costa, V.S.: Sequential pattern mining in multi-relational datasets. In: Proc. of the 13th Conference of the Spanish Association for Artificial Intelligence, LNCS 5988. pp. 121–130. Springer Heidelberg, Seville, Spain (2010)
9. Garofalakis, M., Rastogi, R., Shim, K.: Mining sequential patterns with regular expression constraints. IEEE Transactions on Knowledge and Data Engineering 14(3), 530–552 (2002)
10. Jian, P., Han, J., Mortazavi-asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. of the 17th International Conference on Data Engineering. pp. 215–224. IEEE Computer Society, Heidelberg, Germany (2001)
11. Lee, S.D., Raedt, L.D.: Constraint based mining of first order sequences in seqlog. In: Database Support for Data Mining Applications: Discovering Knowledge with Inductive Queries, LNCS 2682. pp. 155–176. Springer-Verlag Berlin (2004)
12. Ohara, K., Yoshida, T., Geamsakul, W., Motoda, H., Washio, T., Yokoi, H., Takabayashi, K.: Analysis of Hepatitis Dataset by Decision Tree Graph-Based Induction. In: Proceedings of Discovery Challenge. pp. 173–184 (2004)
13. Srikant, R., Agrawal, R.: Mining Sequential Patterns: Generalizations and Performance Improvements. In: 5th Int. Conf. on Extending Database Technology, LNCS 1057. pp. 3–17. Springer-Verlag, Avignon, France (1996)
14. Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large datasets. In: Proc. of the Third SIAM International Conference on Data Mining. pp. 166–177. SIAM, San Francisco, CA, USA (2003)
15. Yu, L., Liu, H.: Feature selection for high-dimensional data: A fast correlation-based filter solution. In: 20th Int. Conf. on Machine Learning. pp. 856–863 (2003)
16. Zelezny, F., Lavrac, N.: Propositionalization-Based Relational Subgroup Discovery with RSD. Machine Learning 62(1-2), 33–63 (2006)