Outcomes Prediction via Time Intervals Related Patterns

Robert Moskovitch[1], Colin Walsh[2], Fei Wang[3], George Hripcsak[1], Nicholas Tatonetti[1]
[1]Department of Biomedical Informatics, Columbia University
[2]Department of Biomedical Informatics, Vanderbilt University
[3]Department of Computer Science and Engineering, University of Connecticut
rm3198@columbia.edu, colin.walsh@vanderbilt.edu, fei_wang@uconn.edu, gh13@columbia.edu, npt2105@columbia.edu

***Abstract*—The increasing availability of multivariate temporal data in many domains, such as biomedical [7], security [13] and more, provides exceptional opportunities for temporal knowledge discovery, classification and prediction, but also challenges. Temporal variables are often sparse and in many domains, such as in biomedical data, they have huge number of variables. In recent decades in the biomedical domain events, such as conditions, drugs and procedures, are stored as time intervals, which enables to discover Time Intervals Related Patterns (TIRPs) and use for classification or prediction.In this study we present a framework for outcome events prediction, called Maitreya, which includes an algorithm for TIRPs discovery called KarmaLego^D, designed to handle huge number of symbols. Three indexing strategies for pairs of symbolic time intervals are proposed and compared, showing that the use of FullyHashed indexing is only slightly slower but consumes minimal memory. We evaluated Maitreya on eight real datasets for the prediction of clinical procedures as outcome events. The use of TIRPs outperform the use of symbols, especially with horizontal support (number of instances) as TIRPs feature representation.**

***Keywords*-Prediction, Time Intervals Mining**

I. Introduction

Temporal data is increasingly becoming available in various domains, including biomedical informatics. Analysis of these data promises a better understanding of how events evolve along time. A major challenge in multivariate temporal data analytics in many domains is the large number of variables, the heterogeneity of their types and appearances [7], [8]. In addition to time series, in which a variable is represented as a series of repeated measures, variables can be represented as instantaneous events or an event that has time duration – having start, end time and a symbol, as in our study. Typically in many domains, as in the biomedical domain for example, the data may be described by thousands of symbolic concepts. These symbolic time intervals can be later mined for frequent patterns for the purpose of temporal knowledge discovery and prediction.

Mining time intervals series is becoming increasingly popular in the data mining literature along the past decade [1], [7], [9], [10], [14]. These symbolic time intervals are often mined to discover frequent temporal patterns that can be used directly for knowledge discovery, or as features for classification [1], [6], [8], [10] and prediction, as we show in this study.

In this paper we improve the KarmaLego algorithm [7] for the discovery of frequent TIRPs to handle thousands of symbol types. In previous studies mining temporal data and discovery of TIRPs was performed on datasets with dozens of symbols [6], [8], [10]. However, in many domains, as for example in the biomedical domain, there are thousands of possible symbols. We introduce an index in KarmaLego that is called DharmaIndex that enables mining such data.
Our main contribution is the introduction of Maitreya, a framework for the prediction of outcome events based on TIRPs, which were discovered at the outcome event class. We demonstrate its utility on several Electronic Health Records (EHRs) datasets from Columbia University Medical Center.

The main contributions of this paper are the following:

- KarmaLego^D – an enhanced KarmaLego [7] for time intervals mining algorithm for large number of symbols, including a runtime evaluation.
- Maitreya – a framework for outcome events prediction in multivariate temporal data that learns a model from a single class.

II. Background

The availability of multivariate temporal data in various domains, characterized often by sparse sampling, varying measurements frequencies and types of events, necessitates the use of Symbolic Time Intervals representation, the development of fast and efficient Time Intervals Related Patterns mining algorithms, and their use for classification and prediction in multivariate time series.

A. Symbolic Time Intervals Mining

Representing multivariate temporal data using symbolic time intervals is becoming increasingly common in the data mining literature [1], [6], [8], [10] in which some of the variables, or concepts, are represented by time intervals in their raw form, or after some transformation process. Methods of transformation from time point series into meaningful symbolic time intervals data, a process often called Temporal Abstraction, were proposed in the past two decades by several researchers [4], [5], [6].

1550-4786/15 $31.00 © 2015 IEEE
DOI 10.1109/ICDM.2015.143

Höppner [4] introduced a naive method inspired by association rules mining to mine rules in symbolic time interval sequences. The major contribution from Papapetrou et al. [9] was to index first the 2-sized TIRPs, which are the components of the extended TIRPs, inspired by Sequential Mining. ARMADA [14] is a projection-based time intervals mining algorithm without any preliminary indexing of the data. Wu et al. [15] proposed TPrefixSpan, which is a modification of the PrefixSpan sequential mining algorithm. That principle was later expanded by [2]. Patel [10] introduced IEMiner that improves Papapetrou's method, by extending the patterns during the discovery process directly. Moskovitch and Shahar [7] introduced KarmaLego that introduces an efficient data structure and exploits the transitivity property of the temporal relations for efficient candidates generation. KarmaLego was faster than IEMiner [10], ARMADA [14] , and H-DFS [9] methods. Unlike other methods, KarmaLego discovers the complete set of TIRPs [8] including all their instances.

B. Classification and Prediction with Temporal Patterns

Recently, there has been a growing interest in using sequential patterns, including TIRPs, for classification [3]. In fact, this approach overcomes the requirements listed by Ratanamahatana and Keogh [11] when using patterns for classification of time series, such as the duration of the temporal pattern that is not defined in TIRPs. Quite simultaneously several groups had proposed using TIRPs, as features for classifying multivariate time series [1], [10], [3], [6], [8]. Patel et al. [10] proposed IEClassifier to classify data using TIRPs. Batal et al. [1] performed knowledge based temporal abstraction, but used only two relations: before and co-occur, which is an Apriori sequential mining algorithm called STF-Mine. They compared recent patterns and older to predict outcome events. Moskovitch and Shahar [8] presented a framework for classification of multivariate time series using several discretizations, such as Equal Width Discretization (EWD), and SAX [5], and later proposed a supervised Temporal Discretization for Classification (TD4C) method that increases the accuracy by learning the cutoffs to increase the differences in the states according to the distribution in the classes [6]. TD4C outperformed the unsupervised EWD and SAX methods. Several studies had shown the advantages of using TIRPs over atemproal representation in classifying multivariate temporal data [1], [10]. Recent studies introduced several heuristics to decrease the number of discovered patterns that still maintain the same level of accuracy [3].

III. METHODS

A. Definitions

Definition 1. *To define a flexible framework of Allen's temporal relations with KarmaLego, two relations are defined on time-stamped (point-based) data, given an epsilon value. Given two time-points t_1 and t_2:*

$$t_1 =^{\varepsilon} t_2 \text{ iff } |t_2 - t_1| \leqslant \varepsilon \text{ and } t_1 <^{\varepsilon} t_2 \text{ iff } t_2 - t_1 > \varepsilon \quad (1)$$

Based on the two relations $= \varepsilon$ and $< \varepsilon$ and the epsilon value, a flexible version of Allen's seven relations is defined, as shown in figure 1. In this study we used a set of three

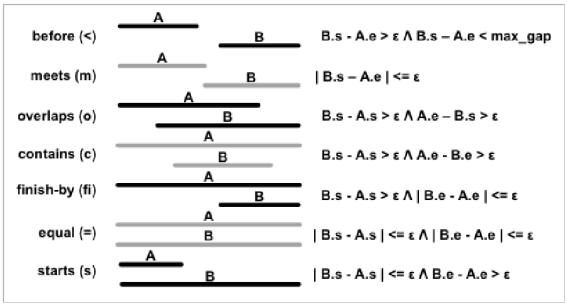

Figure 1: A flexible extension of Allen's temporal relations, using the same epsilon for all of the seven basic relations.

more abstract temporal relations, that performed slightly better than seven relations. Thus, we define BEFORE = {before || meet} that is the disjunction of before or meet; OVERLAP = {overlap}; and CONTAIN = {contains || finish-by || equal || start};

Definition 2. *A symbolic time interval, $I = \langle s, e, sym \rangle$, is an ordered pair of time points, start-time (s) and end-time (e), and a symbol that represents one of the domain's temporal concepts. When referring to these properties we will use the following $I.s$ for its start-time, $I.e$ for its end-time, and $I.sym$ for its symbol.*

Definition 3. *A lexicographic symbolic time-interval series TIS is a series of symbolic time intervals, sorted in the lexicographical order of the start-time, end-time using the relations $< \varepsilon$, $= \varepsilon$ and the symbols, $TIS = \{I^1, I^2, \cdots, I^n\}$, such that*

$$\forall I^i, I^j \in TIS(I < J) \wedge ((I^i_S <^{\varepsilon} I^j_S) \vee (I^i_S =^{\varepsilon} I^j_S \wedge I^i_e <^{\varepsilon} I^j_e) \vee (I^i_S <^{\varepsilon} I^j_S \wedge I^i_e =^{\varepsilon} I^j_e \wedge I^i_t <^{\varepsilon} I^j_t)) \quad (2)$$

Definition 4. *A non-ambiguous lexicographic Time Intervals Related Pattern (TIRP) P is defined as $P = \{I, R\}$, where $I = \{I_1, I_2, \cdots, I_k\}$ is a set of k symbolic time intervals and*

$$R = \cap_{i=1}^{k-1} \cap_{j=i+1}^{k} r(I^i, I^j) = \{r_{1,2}(I^1, I^2), \cdots, r_{1,k}(I^1, I^k), \cdots, r_{k-1,k}(I^{k-1,k})\} \quad (3)$$

defines all the temporal relations among each of the $(k_2 - k)/2$ pairs of symbolic time intervals in I.

Figure 2 presents a typical TIRP, represented as a half-matrix of temporal relations. We will usually assume such a representation through the description of KarmaLego^D.

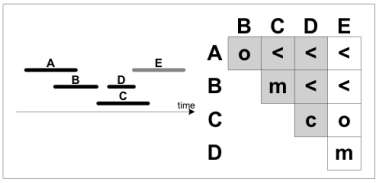

Figure 2: A Time-Interval Related Pattern (TIRP) of five symbolic time intervals and all of their pair-wise temporal relations defined in the half matrix on the right.

Definition 5. *Given a database of $|E|$ distinct entities (e.g., different patients), the vertical support of a TIRP P is denoted by the cardinality of the set E^P of distinct entities for which P holds, divided by the total number of entities (e.g., patients) $|E|$: $ver\_sup(P) = |E^P|/|E|$. The vertical support is actually what is commonly used as support in association rules, itemset and sequential mining. A TIRP having above minimal vertical support threshold is referred to as frequent.*

Definition 6. *The horizontal support of a TIRP P for an entity e_i (e.g., a single patient's record), $hor\_sup(P, e_i)$ is the number of instances of the TIRP P found in e_i. For example, the number of times a particular temporal pattern P was found in a particular patient's record.*

Definition 7. *The mean duration of the n supporting instances of the same k-sized TIRP P within an entity e (e.g., within a single patient's record; note that, per definitions 6 and 7, an entity may have several supporting instances of a TIRP) is defined by:*

$$MeanDuration(P, e) = \frac{\sum_{i=1}^{n} \left(\max_{j=1}^{k} I_e^{i,j} - I_s^{i,1}\right)}{n} \quad (4)$$

We will use these metrics for the TIRP representation as features.

Definition 8. **The time-intervals mining task***: We redefine the task of time intervals mining for the sake of completeness [8]. Given a set of entities E, described by a set of lexicographically ordered symbolic time-interval series, the goal of the time-intervals mining task is to find all the TIRPs whose vertical support is above a predefined minimal vertical support threshold, and all their horizontally supporting instances within any given entity.*

B. The KarmaLego^D Algorithm

The KarmaLego algorithm is a fast time intervals mining algorithm [7], but is limited in its capability to handle more than dozens of symbols. The KarmaLego^D was designed to handle thousands of symbols in the mining process, unlike most of the time intervals mining studies presented, in which dozens of symbols were used at most [2], [7], [9], [10], [15].

KarmaLego^D (Algorithm 1) consists on two main steps: Karma, in which the entire set of E entities (in the database)

Algorithm 1 KarmaLego^D

Input: db – database of $|E|$ entities' symbolic time intervals; min_ver_sup – the minimal vertical support threshold

Output: T – an enumerated tree of all frequent TIRPs

```
1: T ← ∅
2: T^2 ← Karma(db, min_ver_sup, epsilon)
3: for t ∈ T^2 do
4:    if ver_sup(t) > min_ver_sup then
5:       Lego(T, t, min_ver_sup)
6:    end if
7: end for
8: return T
```

are scanned and the pairs of intervals are indexed and Lego, in which they are extended recursively to longer TIRPs using an efficient TIRPs candidate generation method that exploits the transitivity of temporal relations [7].

1) Karma and DharmaIndex: As shown in algorithm 2 the Karma algorithm goes over all the entities in the database and for each pair of symbolic time intervals e.I$^i{}_{.sym}$ and e.I$^j{}_{.sym}$ ordered lexicographically, calculates the temporal relation r among them (using the definitions in figure 1), and indexes this instance in DharmaIndex. In each index there is a hash-table that stores all the instances of the 2-sized TIRPs in the DharmaIndex. At the end of the Karma method the first and second level, the DharmaIndex, of the enumeration tree (shown in [7]) are constructed.

Thus, DharmaIndex indexes all the 2-sized TIRPs based on their *first symbol, second symbol and relation* among them. Each entry is a 2-sized TIRP containing a hash-table that holds all its horizontally supporting instances. This is done using a hash-table, in which the Key is the entity_id and the first symbolic time interval instance (symbol, start and end time), and the Value is a List of the second symbolic time interval instances (the complete list). In previous papers, since only few dozens of symbols were used an efficient data structure of the index was used with a three dimensional array [7], [9], [10] that enables to access in $O(1)$ but with thousands of symbols this data structure consumes too much memory that becomes a limitation.

In this paper we propose and evaluate three strategies to index the 2-sized TIRPs (DharmaIndex) having thousands of symbols efficiently, while maintaining fast access and reasonable memory consumption. We examine three data structure strategies for the indexing of the triple $\langle sym, r, sym \rangle$ that contain the hashtables of the complete set of the horizontally supporting instances.

- **FullyHashed**: each element in the triple is implemented as a hash-table. Thus, the first symbol is a Key of the first hash-table, whose value is a hash-table that its Key

Algorithm 2 Karma

Input: db – database of $|E|$ entities' symbolic time intervals; min_ver_sup – the minimal vertical support threshold
Output: T – an enumerated tree of all frequent TIRPs

1: **for** $e \in E$ **do**
2: **for** $I^i, I^j \in e.I \wedge i < j$ **do**
3: $S \leftarrow e.I^i_{sym}; S \leftarrow e.I^j_{sym}$
4: $r \leftarrow$ the temporal relation among I^i, I^j given ε
5: DharmaIndex $\langle e\_id, e.I^i.sym, r, e.I^j.sym \rangle$
6: **end for**
7: **end for**
8: **return** T

is the temporal relation r. That hash-table Value is the second symbol, whose Value is the hash-table of the supporting instances.

- **RelArraySymArray**: the temporal relation and the first symbol are indexed in a two dimensional array. Each entry contains a hash-table, in which the Key is the second symbol and the Value is the hash-table of the supporting instances.
- **RelArray**: the temporal relation is indexed using an array, and the first and second symbols are indexed using hash-tables that eventually contain the supporting instances hash-table.

After the DharmaIndex is created, each frequent 2-sized TIRP in DharmaIndex (or T^2) is extended by Lego as shown in Algorithm 1 in line 5. Lego is a highly efficient TIRPs candidate generation algorithm that exploits the transitivity of temporal relations, as described in details in [7].

C. Maitreya - Prediction of Outcome Events

We propose a methodology for the prediction of outcome events using time intervals mining shown in figure 3. Our methodology consists of discovering TIRPs only in the set of entities (i.e., patients) having the outcome event, from which a prediction model is learnt. A control set of entities (i.e., patients) is defined that are selected randomly from the rest of the patients who do not have the outcome. Since in TIRPs mining in each fold a different set of frequent TIRPs can be discovered, and to avoid overfitting, we perform a rigorous evaluation strategy, in which the cohort and control datasets are divided into three folds. In each iteration TIRPs are discovered from 1/3 of the cohort (one fold) and these TIRPs are detected using SingleKarmaLego [8] at the other 2/3 (two folds) of the cohort and the control.
Once the TIRPs' instances were detected a matrix of TIRP-features is created using the most frequent TIRPs. The matrix of values describes patients (rows) and the detected TIRPs features (columns). A default matrix for that would be a boolean representation, indicating the presence or absence of the TIRP for a specific patient, or more advanced as horizontal support and mean duration that we will use here.

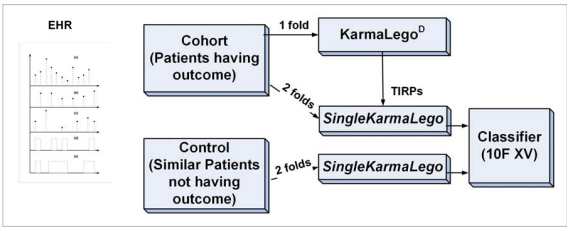

Figure 3: Maiterya, a framework for outcome events prediction.

IV. Evaluation Setting

Our evaluation has two main folds. First, the runtime of KarmaLego^D with the three indexing strategies. Second, is the prediction of outcome events using Maitreya. We describe our experimental objectives in the results section. We extracted clinical data from Columbia University Medical Center clinical data warehouse. In total, the EHR contains medical records data of more than 1.5 million patients going back to 1989. We used only coded data for this analysis, including drug exposures, conditions (billing codes), and procedures. These concepts were mapped to RxNorm, SNOMED-CT, and ICD-9-Procedure, respectively, to conform to OMOP [12]. Medical concepts are then transformed into symbolic time intervals (called "eras") and further concatenated according to definition 12. We used only patients having at least two types of concepts among the conditions, procedures and drug exposers, which left us with about hundred thousands of patients. We selected eight clinical procedures from our dataset presented in table 1. Then we created a prediction time period of two months, and an observation time prior to that month of two years of patients' data.

Definition 12. Abstraction function: *Given two symbolic time intervals having the same symbol I^i and I^j, and I^i is before I^j according to the lexicographical order, and $I^j s - I^i e < max\_time$ holds, it will be abstracted into a new single time interval having the start time of I^i and the end time of I^j. In our case max_time was set to 30 days, according to the OMOP standard* [12].

$$\forall I^i, I^j \in TIS \wedge (i < j) \wedge (I^i_{sym} = I^j_{sym}) \wedge (I^j_s - I^i_e < max\_time_{time}) : TIS \leftarrow TIS - I^i \wedge TIS \leftarrow TIS - I^j \wedge newI.s = I^i_S \wedge newI.e = I^j_e \wedge TIS \leftarrow TIS + newI \quad (5)$$

For the runtime experiments we used seconds as measurements, and for the prediction evaluation we used the Accuracy Under the Curve (AUC) measure.

Table I: The outcome clinical procedures datasets

Procedure Code	Procedure Description	#cases
ven_cath	Venous catheterization	2,704
punct_vein	puncture of vein	3,700
inj_prop	Injection prophylactic substance	3,858
emerg_vst	Emergency department visit	2,858
pro_dia_inj	prophylactic or diagnostic injection	1,012
ent_infus	Enteral infusion of nutritional	1,006
diag_ultra	Diagnostic ultrasound	1,276
comp_ax_tomo	Computerized axial tomography	1,742

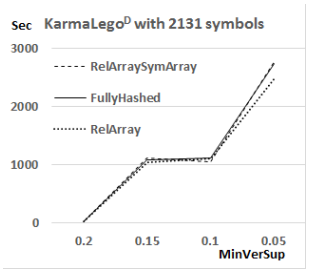

Figure 4: On data with 2131 symbols, RelArray is a bit faster than the others.

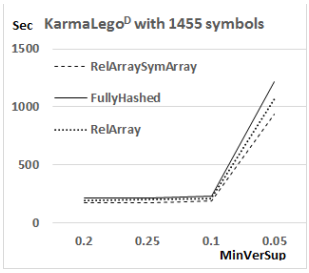

Figure 5: For data with 1455 symbols, RelArraySymArray is the fastest.

V. Results

For the first two experiments we ran the algorithms on a windows machine having 2.7 GHz Intel Core I7 with 16Gb. All the algorithms were implemented in Visual C# and the output of the methods was exactly the same. Thus, all the KarmaLego^D versions discovered all the horizontally supporting TIRPs' instances as defined in definition 8.

A. DharmaIndex Runtime Evaluation

KarmaLego^D's main contribution is the indexing of the two-sized-TIRPs in the DharmaIndex. In [8] a detailed general complexity analysis for time intervals mining algorithms is provided. However, to compare specific algorithms and specific data structures in time intervals mining, as well as in sequential mining algorithms, it is required to perform a runtime evaluation. Thus, in this study we evaluate the indexing strategies that were proposed for DharmaIndex in order to compare the overall mining runtime of the indexing and the candidate generation. We implemented the three indexing strategies for the DharmaIndex and ran them on two datasets having 511 and 2147 entities, as well as 1455 and 2131 symbols respectively. We ran them and measured the runtime using 0.2, 0.15, 0.1 and 0.05 minimal vertical support.

All the runs on the four datasets show that the FullyHashed strategy, which has the lowest memory consumption is the slowest, but in most of the cases it is just slightly slower. In figure 4 the FullyHashed behaved very similar to the RelArraySymArray, in figure 5 it was the slowest, while the strategies having array based indexing were slightly faster. However, in general it seems that fully hashing strategy of the DharmaIndex is the best strategy, which is also most efficient memory-wise.

B. Outcome Prediction Evaluation

The second evaluation is of the prediction of outcome events, using Maitreya, on the eight procedure datasets that were described in table 1. We wanted to answer the following questions: (1) whether using TIRPs as features is better than symbols (as features without any temporal representation) that was our baseline; (2) whether the TIRPs representation will be better than the default boolean representation that were used in previous studies [1], [3], [10]. We used 20% minimal vertical support for all the procedures and 730 days maximal gap. All the experiments were performed with three folds mining and ten folds cross validation of the classification experiments using weka's RandomForest. Thus, each result that is reported is based on thirty experimental runs.

We ran Maitreya on the eight clinical procedures prediction tasks datasets in table 1. To avoid over-fitting, we limited the number of features (symbols or TIRPs) to five roots of the size of the cohort, which was about few dozens of the most frequent TIRPs. We used three relations with epsilon zero. Table 2 presents the results of the procedures prediction when using only Symbols, which is the baseline, on the first column, and when using TIRPs with five types of TIRP representations boolean (Bool), Horizontal Support (HS) and mean duration (MeanD). The results in table 2 report the mean Area Under the Curve (AUC) and the 95% confidence interval of each evaluation run. The results show the advantages of using TIRPs over symbols as features for classification. The use of the HS and the MeanD performed very well compared to the Boolean representations. In fact, all the TIRPs metrics were better than the symbols that were without any temporal aspect.

VI. Discussion

In this paper we described a general framework, called Maitreya for the prediction of outcome events and applied it to Electronic Health Records, although it can be applied in any domain. Maitreya was designed to learn a model based on the cohort class of the outcome event, while a control class of entities (i.e. patients) is selected for evaluation purposes. Maitreya discovers TIRPs from the cohort using KarmaLego^D, a fast TIRPs discovery algorithm that was designed to handle large amount of symbols. Later TIRPs are detected, which are used as features for classification, using SingleKarmaLego. For KarmaLego^D and specifically DharmaIndex we proposed three strategies to handle large number of TIRPs, since with thousands of symbols it is impossible to allocate a three dimensional array.

Table II: The prediction results in mean Area Under the Curve and 95% confidence intervals

VerSup	Symbols (baseline)	TIRPs		
		Bool	HS	MeanD
ven_cath	0.683 -/+ 0.068	0.692 -/+ 0.078	0.728 -/+ 0.074	0.692 -/+ 0.075
punct_vein	0.629 -/+ 0.082	0.668 -/+ 0.090	0.701 -/+ 0.083	0.703 -/+ 0.084
inj_prop	0.642 -/+ 0.081	0.679 -/+ 0.091	0.680 -/+ 0.097	0.698 -/+ 0.089
emerg_vst	0.709 -/+ 0.082	0.734 -/+ 0.086	0.753 -/+ 0.080	0.715 -/+ 0.083
pro_dia_inj	0.766 -/+ 0.051	0.796 -/+ 0.065	0.745 -/+ 0.066	0.792 -/+ 0.056
ent_infus	0.661 -/+ 0.064	0.671 -/+ 0.070	0.713 -/+ 0.076	0.731 -/+ 0.081
diag_ultra	0.657 -/+ 0.082	0.702 -/+ 0.074	0.676 -/+ 0.075	0.741 -/+ 0.078
comp_ax_tomo	0.629 -/+ 0.072	0.649 -/+ 0.081	0.727 -/+ 0.061	0.641 -/+ 0.071
mean	0.664 -/+ 0.073	0.700 -/+ 0.079	0.712 -/+ 0.076	0.717 -/+ 0.076

Fast access, while efficient memory consumption is crucial for the mining process. Our runtime evaluation showed that the FullyHashed strategy was slightly slower than the other array based strategies, but due to its optimized minimal memory consumption it is recommended to use it. While these results may be intuitive, the magnitude of the differences between the strategies couldn't be estimated or the complexity analyzed ahead due to the nature of the data and the algorithms. We evaluated Maitreya on eight outcome events from the EHR domain for which we created datasets from Columbia University Medical Center EHR. Our results show that using TIRPs with Boolean representation was already better than the symbols, but when using advanced TIRP representation it was significantly better. In this study we consisted our model based on diagnoses, procedures and drug codes in the EHR data, but without lab tests and codes extracted from textual reports. As future work we would like to add the lab tests and transform them into symbolic time intervals using various temporal abstraction methods, especially the TD4C method [6] that is expected to increase the classification performance through determining the cutoffs that mostly differentiate the classes states distributions.

REFERENCES

[1] I. Batal, H. Valizadegan, G. F. Cooper, and M. Hauskrecht. A temporal pattern mining approach for classifying electronic health record data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(4):63, 2013.

[2] Y.-C. Chen, J.-C. Jiang, W.-C. Peng, and S.-Y. Lee. An efficient algorithm for mining time interval-based patterns in large database. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 49–58. ACM, 2010.

[3] D. Fradkin and F. Mörchen. Mining sequential patterns for classification. *Knowledge and Information Systems*, In Press, DOI 10.1007/s10115-014-0817-0.

[4] F. Höppner. Learning temporal rules from state sequences. In *IJCAI Workshop on Learning from Temporal and Spatial Data*, volume 25, 2001.

[5] J. Lin, E. Keogh, S. Lonardi, and B. Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11. ACM, 2003.

[6] R. Moskovitch and Y. Shahar. Classification-driven temporal discretization of multivariate time series. *Data Mining and Knowledge Discovery*, 29(4):871–913, 2015.

[7] R. Moskovitch and Y. Shahar. Fast time intervals mining using transitivity of temporal relations. *Knowledge and Information Systems*, 42(1):21–48, 2015.

[8] R. Moskovitch and Y. Shahar. Classification of multivariate time series via temporal abstraction and time intervals mining. *Knowledge and Information Systems*, In Press.

[9] P. Papapetrou, G. Kollios, S. Sclaroff, and D. Gunopulos. Mining frequent arrangements of temporal intervals. *Knowledge and Information Systems*, 21(2):133–171, 2009.

[10] D. Patel, W. Hsu, and M. L. Lee. Mining relationships among interval-based events for classification. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 393–404. ACM, 2008.

[11] C. A. Ratanamahatana and E. Keogh. Three myths about dynamic time warping data mining. In *SDM*, pages 506–510, 2005.

[12] S. J. Reisinger, P. B. Ryan, D. J. O'Hara, G. E. Powell, J. L. Painter, E. N. Pattishall, and J. A. Morris. Development and evaluation of a common data model enabling active drug safety surveillance using disparate healthcare databases. *Journal of the American Medical Informatics Association*, 17(6):652–662, 2010.

[13] D. Stopel, Z. Boger, R. Moskovitch, Y. Shahar, and Y. Elovici. Improving worm detection with artificial neural networks through feature selection and temporal analysis techniques. In *Proc. Third International Conference on Neural Networks, Barcelona*, 2006.

[14] E. Winarko and J. F. Roddick. Armada–an algorithm for discovering richer relative temporal association rules from interval-based data. *Data & Knowledge Engineering*, 63(1):76–90, 2007.

[15] S.-Y. Wu and Y.-L. Chen. Mining nonambiguous temporal patterns for interval-based events. *Knowledge and Data Engineering, IEEE Transactions on*, 19(6):742–758, 2007.