Inter-item time intervals in sequential patterns

Thomas Kastner, Hubert Cardot, Dominique H Li

Inter-item time intervals in sequential patterns

Thomas Kastner[1,2], Hubert Cardot[1], and Dominique H. Li[1]

[1] Université de Tours, LIFAT, EA 6300, Tours, France
[2] Apivia MACIF Mutuelle, France

Abstract. In sequential pattern mining, the time intervals between each pair of items are usually ignored or mined using additional constraints such as gaps. However, a large part of time-related knowledge is lost during the original proposition of sequential pattern mining. In many applications, time-related information is valuable and its preservation is necessary. In this paper, we present two methods for exploiting time intervals between items and evaluate them in a classification task using a real-world customer intent dataset.

Keywords: Pattern mining · Time intervals · Intention prediction

1 Introduction

Creating realistic representations of real-world applications is an essential aspect of data processing. In data mining, sequential pattern is a widely used pattern structure in many domains, such as finance, medicine or e-commerce. Unlike itemset pattern, sequential pattern mining consists in finding patterns from a *sequence database* with temporal order within a list of items. During nearly 30 years of research, two directions of sequential pattern mining have emerged: (1) Mining patterns faster, with models such as PrefixSpan [6] or BIDE [12]; (2) mining patterns with more information. This approach is often based on algorithms like those mentioned above. On the other hand, timestamp is often difficult to process in sequential pattern mining tasks. In literature, we often consider the timestamp of punctual events (items) as the period between two elements, or as the duration of an interval over which the events spreads. The classical sequential pattern mining model does not include timestamps, but several solutions have been developed to solve this problem. In most approaches, constraints are applied during frequent pattern mining, or as a post-processing step. For instance, the *duration constraint* specifies the minimum or maximum time that the patterns must not exceed, the *gap constraint* specifies the number of items between two items in the resulting patterns and the *length constraint* sets the minimum and maximum number of items in the sequence [10].

In this paper, we present a new sequence structure with itemized time intervals instead of timestamps, with which the sequences can be enriched, and for which, the most important, any existing sequential pattern mining algorithms can be applied. Our classification based experimental evaluation shows that the

features generated by our proposed sequential pattern structure allow to train more accurate models than original sequences.

The rest of this paper is organized as follows. Section 2 introduces the related work. We describe in Section 3 our methods for integrating time intervals within sequences. Section 4 presents an application of our methods on a real customer intent dataset. Finally, we conclude in Section 5.

2 Related Work

In sequential pattern mining, the use of timestamps values, when available, has been rather under-exploited in the literature, especially for representing the time intervals between items. Kitakami et al. [8] and Chen et al. [2] extend PrefixSpan pattern mining algorithm by allowing it to use gaps and time intervals. Hirate et al. [7] generalize sequential pattern mining with item interval, gap and time interval handling with constraints. Tran et al. [3] propose to use time interval constraints and utility value to mine high utility sequential patterns. More elaborate methods have been added to the time constraints: MG-PrefixSpan [9] is a derivative from PrefixSpan that enables to deal with the temporal relationships between successive items in the sequential pattern, based on multi-granularities. Guyet et al. [5] adapted the classical Apriori [1] paradigm to propose an efficient algorithm based on a hyper-cube representation of temporal sequences. The algorithm of Gianotti et al. [4] adopts a prefix-projection approach to mine candidate sequences.

Our approach is the first attempt to provide a way to exploit time in patterns without modifying any mining algorithm and without having to apply constraints.

3 Representing Time in Sequences

In this section we present our contributions. We first introduce preliminary notations, then we propose two efficient methods to enrich sequences upstream of classical sequential pattern mining methods.

3.1 Preliminaries

Let $I = i_1, ..., i_n$ be the set of *all items*. A *sequence* is defined as $s = \langle X_1, ..., X_m \rangle$ which corresponds to an ordered list of items from I. A sequence can be associated with a vector of timestamps, denoted as $v_{time} = \langle t_{i_1}, ..., t_{i_m} \rangle$ which corresponds to the time each event of the sequence occurs. A *sequence database* is a large set of sequences, denoted by $\mathcal{S}$.

A sequence s_a of length p is contained in sequence s_b of length $q > p$ if and only if $X_{a_i} \subset X_{b_i} \forall \{i \mid 1 \leq i \leq p\}$: in this case, s_a is a *subsequence* of s_b, denoted as $s_a \sqsubseteq s_b$. Note that a sequence containing p items can have up to $2^q - 1$ distinct subsequences.

For the following methods we consider to compute the interval between each consecutive items of sequences. In this paper, we define the *vector of sequence intervals* as Definition 1.

Definition 1. *Let the* interval *between item i_x with timestamp t_{i_x} and item i_{x+1} with timestamp $t_{i_{x+1}}$ be $\delta_{(i_x,i_{x+1})} = t_{i_{x+1}} - t_{i_x}$, the vector of sequence intervals is defined as $v_{inter} = \langle \delta_{(i_1,i_2)}, ..., \delta_{(i_{m-1},i_m)} \rangle$.* □

3.2 Integrating Intervals in Sequences

We consider a sequence $s_a = \langle X_1, ..., X_m \rangle$ composed of m items and the vector

$$v_{inter}^{s_a} = \langle \delta_{i_1,i_2}, ..., \delta_{i_{m-1},i_m} \rangle$$

representing time intervals between items of sequence s_a.

First we map each item of $v_{inter}^{s_a}$ to a discrete partition of $z = \langle I_1, I_2, ..., I_{|z|} \rangle$. Using $v_{inter}^{s_a}$ and an ordered set of thresholds parameters $\alpha = \langle th_1, th_2, ..., th_{|z|-1} \rangle$ with $th_1 < th_2 < ... < th_{|\alpha|}$, each continuous interval $\delta_{(i_{x-1},i_x)} \in v_{inter}^{s_a}$ is mapped to a partition $I_y \in z$ if it validates the following relation:

$$\delta_{(i_{x-1},i_x)} \in I_y \Leftrightarrow th_{y-1} \leqslant \delta_{(i_{x-1},i_x)} < th_y \tag{1}$$

The function $Z : v_{inter}^{s_a} \rightarrow z$, where $z = \{I_1, ..., I_n\}$ discretizes each value of vector $v_{inter}^{s_a}$ into the matching partition of z according to the relation presented above.

Integration as Items (Method 1) We construct a new sequence $s_a{}'$ from the original sequence s_a by inserting intervals partitions from $Z(v_{inter}^{s_a})$ between each observations of s_a as follows:

$$s_a{}' = \langle X_1, Z(i_1), X_2, Z(i_2), ..., X_{m-1}, Z(i_{m-1}), X_m, Z(i_m) \rangle \tag{2}$$

The resulting sequence contains twice as many items as the original sequence and the last interval element $Z(i_m)$ inserted equals zero.

Integration as Items suffixes (Method 2) This method is derived from the method proposed in the previous section. As for the first method we consider the sequence s_a and the mapping function Z. Instead of adding new items to represent time intervals in the sequence, we add a suffix to each item of s_a that indicates which time interval partition $Z(i)$ has been associated to it. Each item can thus take $|I|^{|z|}$ distinct values.

$$s_a{}'' = \langle X_1 Z(i_1), X_2 Z(i_2), ..., X_{m-1} Z(i_{m-1}), X_m Z(i_m) \rangle \tag{3}$$

4 Experiments

We report in this section that sequences with added time intervals have a better predictive performance than normal sequences. We compare the results of models trained on features created from frequent patterns in the raw and modified sequence databases.

4.1 Datasets and models

We use a clickstream dataset provided by an e-commerce website, published and described in [11] which has recently been used in the work of [13]. Authors present *seq2pat*, a new research library for sequence-to-pattern generation that allows to turn patterns into features by one hot encoding the presence of each pattern for each sequence. The dataset contains 203,084 click sequences including 8,329 sequences with a purchase (called positive sequences). The sequences are composed of six different types of events (page view, add to cart, purchase,...), and the biggest sequence has 155 items.

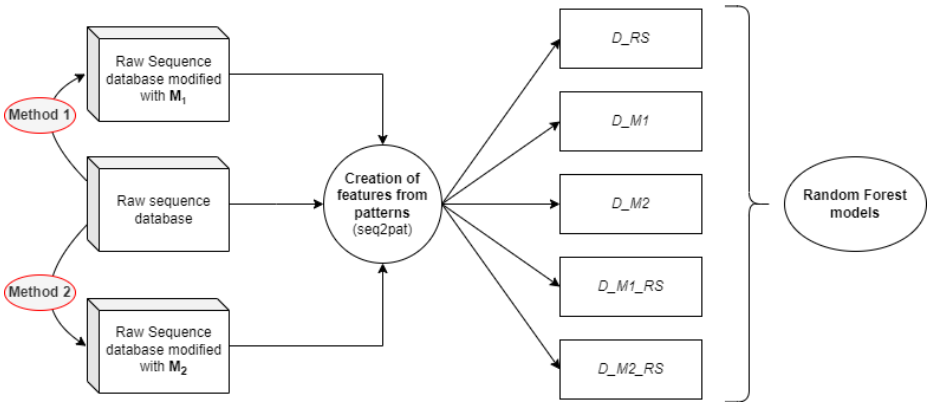

Fig. 1. Overall view of experiments

We conduct our experiments using the dataset presented previously and *seq2pat* python library. We clean and prepare the dataset using the code provided by [13]. Figure 1 shows the different steps of our analysis. First we apply the two methods presented in the last section to the sequence database using the timestamp vector provided. We choose to model inter-items intervals with three discrete partitions and set the two corresponding thresholds parameters (10 seconds and 30 seconds) using the quantiles of the list of all inter-items intervals of the database. It ensures that there is a consistent number of intervals in each partition. This results in three different sequential databases created using *set2pat* to mine frequent patterns. From those three sequential databases we generate features from patterns with *set2pat* (one-hot encoding). Resulting datasets

are described as follow: dataset *D_RS* includes features generated from frequent patterns in the raw sequence dataset. *D_M1* includes features generated from frequent patterns in the raw sequence dataset enhanced with method 1. *D_M2* includes features generated from frequent patterns in the raw sequence dataset enhanced with method 2. *D_M1_RS* and *D_M2_RS* include respectively features from *D_RS* + *D_M1* and *D_RS* + *D_M2*. For each dataset we train on AUC a random forest classifier composed of 100 trees, using a ten fold cross validation with *scikit-learn* python library [1].

4.2 Results

Table 1 presents the average results on several metrics for all previously described datasets. When we use only the sequences modified by our methods the performances are not better than with the initial sequences. *D_M1_RS* and *D_M2_RS* have higher scores for all metrics than *D_RS*. This shows the potential of our method to enhance the task of customer intention prediction.

Table 1. Results of the 10-folds cross-validation on a RF using five datasets.

Dataset	AUC	Recall	Precision	F1-score
D_RS	89.32 (±1.91)	75.50 (±4.48)	83.59 (±3.11)	79.26 (±3.11)
D_M1	88.95 (±1.99)	77.80 (±4.28)	83.43 (±3.35)	80.47 (±3.31)
D_M2	85.96 (±2.24)	65.60 (±4.80)	83.68 (±3.60)	73.43 (±3.53)
D_M1_RS	93.06 (±1.87)	**78.70(±3.77)**	83.78 (±3.20)	**81.12(±2.99)**
D_M2_RS	**93.68(±2.24)**	75.90 (±4.25)	**83.84(±3.07)**	79.61 (±3.05)

When looking at features importance of the model trained with *D_M1*, five of the ten most important features contain interval items that we have constructed. For *D_M2* the notion of time is included in all the features by construction.

5 Conclusion

We presented two methods to include inter-item time intervals into sequences. They have the advantage of being independent from pattern mining algorithms as it is a data preparation step and can be used with almost all existing pattern mining algorithms, including sliding windows based methods. It enhances the sequences using a fast and simple implementation. The resulting sequence database stays interpretable as the mapped time intervals are user defined. Table 2 summarizes the advantages and disadvantages of the presented methods. Experiments show an improvement of performances when using our enhanced sequences on a customer intent classification task. Moreover, our methods do not have a processing time concern as it is a small and simple pre-processing task and it is independent of the pattern mining algorithm used. The integration

Table 2. Advantages and drawbacks of presented methods

	Advantages	Drawbacks
Method 1	- Independant from pattern mining algorithm - Sequence enriched by interval data - Fast and simple implementation	- Double sequence length - Decrease of mean support - Setting parameters
Method 2	- Independent from pattern mining algorithm - Sequence enriched by interval data - Fast and simple implementation - No increase in sequence length	- More distinct items in database - Setting parameters

of temporal knowledge within sequences could improve pattern mining applications, for instance in medicine or in finance. As future work, this method can be adapted to interval items, in other words items that have a duration.

References

1. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
2. Chen, Y.L., Chiang, M.C., Ko, M.T.: Discovering time-interval sequential patterns in sequence databases. Expert Systems with Applications **25**(3), 343–354 (2003)
3. Duong, T.H., Janos, D., Thi, V.D., Thang, N.T., Anh, T.T.: An Algorithm for Mining High Utility Sequential Patterns with Time Interval. Cybernetics and Information Technologies **19**(4), 3–16 (Nov 2019)
4. Giannotti, F., Nanni, M., Pedreschi, D.: Efficient mining of temporally annotated sequences. In: Proceedings of the 2006 SIAM international conference on data mining. pp. 348–359. SIAM (2006)
5. Guyet, T., Quiniou, R.: Mining Temporal Patterns with Quantitative Intervals. In: 2008 IEEE International Conference on Data Mining Workshops. pp. 218–227. IEEE, Pisa, Italy (Dec 2008)
6. Han, J., Pei, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: proceedings of the 17th international conference on data engineering (2001)
7. Hirate, Y., Yamana, H.: Generalized Sequential Pattern Mining with Item Intervals. Journal of Computers **1**(3), 51–60 (Jun 2006)
8. Kitakami, H., Kanbara, T., Mori, Y., Kuroki, S., Yamazaki, Y.: Modified prefixspan method for motif discovery in sequence databases. In: PRICAI 2002 (2002)
9. Li, N., Yao, X., Tian, D.: Mining Temporal Sequential Patterns Based on Multigranularities. IJCCC (2014)
10. Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: Proceedings of the eleventh international conference on Information and knowledge management. pp. 18–25 (2002)
11. Requena, B., Cassani, G., Tagliabue, J., Greco, C., Lacasa, L.: Shopper intent prediction from clickstream e-commerce data with minimal browsing information. Scientific reports **10**(1), 1–23 (2020)
12. Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: Proceedings. 20th international conference on data engineering. pp. 79–90. IEEE (2004)
13. Wang, X., Hosseininasab, A., Colunga, P., Kadıoğlu, S., van Hoeve, W.J.: Seq2pat: sequence-to-pattern generation for constraint-based sequential pattern mining. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 36 (2022)