MuTIny: A MULTI-TIME INTERVAL PATTERN DISCOVERY APPROACH TO PRESERVE THE TEMPORAL INFORMATION IN BETWEEN

Alessio Bertone, Tim Lammarsch, Thomas Turic, Wolfgang Aigner, Silvia Miksch
Danube University Krems, Dr.-Karl-Dorrek-Strasse 30, 3500 Krems, Austria

Johannes Gärtner
XIMES Gmbh, Hollandstrasse 12/12, 1020 Vienna, Austria

ABSTRACT

Finding trends, patterns, and relationships among patterns are very relevant tasks when dealing with time-oriented data and information. However, most of the proposed methods have a sequence of events as outcome, lacking either any knowledge about the intervals between them or about after how much time a particular pattern will reoccur. We present MuTIny, a novel approach extending the I-Apriori algorithm, which is able to discover so called multi-time interval patterns and we describe how it can be customized according to users' needs. Moreover, a real world example illustrates its usefulness.

KEYWORDS

Temporal Data Mining, Interval Mining, Pattern Finding, Time-oriented Data

1. INTRODUCTION

Data Mining is usually considered a step of the KDD process, the aim of which is to extract implicit, previously unknown, and potentially useful information from databases (Fayyad et al. 1996, Frawley et al. 1991). Because of its relevance, many approaches have been proposed to find patterns occurring frequently in a database. Nevertheless, when dealing with time-oriented data and information, exploring trends, patterns, and relationships among patterns are particularly complicated tasks, since in contrast to other quantitative data dimensions (usually "flat"), time has an inherent semantic structure which increases its complexity dramatically (Aigner et al. 2007).

In order to analyze time related data many Temporal Data Mining methods exist in literature. They usually define an event as either a certain value for a given attribute occurring at a certain time (e.g., 30 employees were at work at 4 p.m.) or a fact or sequence of facts following the time line (e.g., first fact: customer X buys a book on 18th April; second fact: customer X buys a DVD one week later). However, except for some exceptions (e.g., Magnusson 2000, Yoshida et al. 2000, Hu et al. 2009), a sequence of events in a certain order is discovered, but neither with any information about the intervals between successive events, nor about when this sequence will reoccur. That is, given an ordered list of item sets and their transaction times, they search for the most frequently occurring patterns, but the result is a sequence of items, without any information about the elapsed time between two items and about the time before a certain sequence will reoccur.

An example illustrated in Figure 1(a) explains our problem definition. A customer decides to buy a Play Station 3 (PS3) gaming console (event A). The next day, s/he buys a racing game (event B). After five days, s/he buys a steering wheel controller (event C). Another racing game is bought after 3 months (event D). If we take into account only the sequence or the order of these events, ABCD, we cannot know after how much time the next item will be purchased. Moreover, we cannot even know after how much time a similar sequence will occur. On the contrary, if also the time intervals are considered, we can not only profile the users according to their interests, habits and requirements, but we can also improve the selling strategies according to the timing of their shopping habits. As a matter of fact, the webshop can vary its offers and catalogues according to the users. For instance, it is possible to send e-mails or letters describing discounts on games for PS3 two months after the first purchase, or make special offers dedicated to those who bought a PS3 in the previous two/three months.

To address this problem, we propose the exploitation of so called *multi-time interval patterns*, which may lead to successful application not only in retailing, but also in many other fields. Therefore, the goal of this paper is to

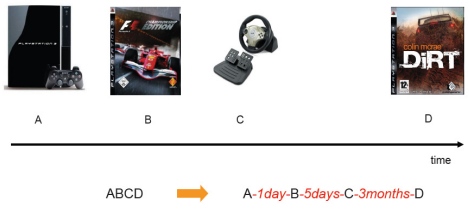
illustrate MuTIny (MUlti-Time INterval pattern discoverY approach), a new approach which is able to find these *multi-time interval patterns* and to show how it can be adopted in real world applications.

2. DESCRIPTION OF THE APPROACH

The idea we propose in this paper extends in particular the I-Apriori algorithm proposed in Chen et al. (2003) to non transactional databases and provide a more general and more customizable approach to find multi-time interval patterns in time-oriented data.

Regarding the term pattern, in order to better address our issues, we propose to define a pattern as "*a nonempty set of time intervals (with properties) or time-points, their relationship (as well as to corresponding data), meeting a certain degree of interest*", where the *degree of interest* is a measure of interestingness, which may be context or user dependent (e.g., in case a user is interested in knowing when a higher amount of employees is needed during the week, s/he will probably search for the most frequent patterns; on the contrary, for surveillance systems data, the less frequent ones will likely be of more interest). Then, as the name suggests, a *multi-time interval pattern* is a pattern whose composing parts are divided by different time intervals.

Given such definitions, we can apply them to different kinds of data. We focus on data which contain temporal and non temporal attributes, which may have both ordinal and categorical values, and are expressed as collection of non overlapping sets. Figure 1(b) shows an example of dataset: in this case the data are about "*Turnover*" and the number of employees ("*#E*") present in a certain retail store. They are expressed at a daily level, that is, with a granularity *Day*. The labels "*Begin*" and "*End*" represent the two endpoints of each interval (i.e., each day), and the labels "*DoW*" and "*DoM*" represent "Day of the Week" and "Day of the Month" respectively. Moreover, we can have nominal attributes, such as "*Weather*", with values e.g., "*sunny*", "*rainy*", etc.



(a)

Begin	End	DoW	DoM	#E	Turnover (€)	Event
01.06.2006 00:00	02.06.2006 00:00	Thu	1	24	3.537,00	e_0
02.06.2006 00:00	03.06.2006 00:00	Fri	2	24	3.493,00	e_0
03.06.2006 00:00	04.06.2006 00:00	Sat	3	10	535,30	e_1
04.06.2006 00:00	05.06.2006 00:00	Sun	4	0	0,00	e_2
05.06.2006 00:00	06.06.2006 00:00	Mon	5	0	0,00	e_2
06.06.2006 00:00	07.06.2006 00:00	Tue	6	24	3.483,50	e_0
07.06.2006 00:00	08.06.2006 00:00	Wed	7	25	3.808,70	e_3
08.06.2006 00:00	09.06.2006 00:00	Thu	8	24	3.870,70	e_0
09.06.2006 00:00	10.06.2006 00:00	Fri	9	23	3.396,40	e_4
…	…	…	…	…	…	…

(b)

Figure 1: (a) Traditional approach (order only) vs. Multi-time interval pattern (intervals in between). (b)An example of dataset: data are about turnover and number of employees in a certain shopping mall expressed day by day, with information about day of week and day of month. The last column shows the events (#E is chosen).

Before starting with the description of the approach, we need to define some concepts. In order to find interesting multi-time interval patterns, we have to reprise the above definition and describe more precisely what the "composing parts" of a multi-time interval pattern are and what intervals are taken into account.

Firstly, we follow the definition given by Mannila et al. (1997) that define the event as a pair (event type, time of occurrence), where the event type can actually contain several attributes. Therefore, we can state that each "composing part" of a multi-time interval pattern is an event, whereas an event is the occurrence of a certain configuration of values of one or more attributes. For instance, if the attribute chosen is "*#E*" (as in Figure 1(b)), then event e_0 is 24, event e1 is 10, event e_2 is 0 and so on (note that on June 8th, 24 Employees were at work, this is another occurrence of event e_0). Similarly, if the attributes chosen are "*#E*" and "*Turnover*", then event e_0 is (#E=24 AND T=3537), event e_1 is (#E=24 AND T=3493) and so on.

Secondly, in order to search for the most frequent multi-time interval patterns, we need to define the intervals within which to look for the patterns. Since it would be too time consuming to calculate all the possible intervals between any pair of events, some intervals are given as default[1]. We defined six different not overlapping and consecutive intervals (or ranges), taking into account the most meaningful granularities of social time, according to our experience and to the suggestions coming from experts collaborating in a common project (see Figure 2). Therefore, we can now revise the definition of multi-time interval pattern in a generalized form as follows:

[1] The amount of interval and their end points can be customized as it will be explained later on. However, for simplicity we consider the default case without any loss of validity.

Interval	
I_o	$0 \leq T < 1$ hour
I_1	1hour $\leq T < 1$ day
I_2	1 day $\leq T < 1$ week
I_3	1 week$\leq T < 1$ month
I_4	1 month $\leq T < 1$ quarter
I_5	$T \geq 1$ quarter

Figure 2: The chosen (default) time intervals (left) and how they graphically appear (right).

Be D a dataset of dimension *p x t* with attributes $A_1,...,A_t$, and values $V_{A1,1}$, $V_{A1,2}$,...$V_{A1,p}$, $V_{A2,1}$,... $V_{At,p}$, E the set of possible events, having the form $A_h = V_{Ah,1} \wedge A_{h+1} = V_{Ah+1,1} \wedge ...$, $1 \leq h \leq t$, and TI the set of all the possible time intervals between any pair of events, then $M = e_0\ I_0\ e_1\ I_1\ e_z\ I_2 \ldots\ e_m\ In$ is a multi-time interval pattern if

- $e_j \in E$ for $0 \leq j \leq m$
- $I_k \in TI$ for $0 \leq k \leq n$
- $k \leq w \Rightarrow I_k$ (consecutive) precedes I_w
- $e_j\ I_b\ e_q\ I_c\ e_s\ \neg \Rightarrow\ I_b$ (consecutive) precedes I_c

In other words, a multi-time interval pattern can be defined as a pattern whose composing parts are events, divided by non overlapping and consecutive time intervals. Moreover, any time interval in between can occur and repeat independently of its consecutive order. That is, if we compare the indices of the intervals in between, I_2 (consecutive) precedes I_3, but the order the intervals appear does not imply the former precedes the latter (e.g., $e_1\ I_2\ e_2\ I_1\ e_2$ is a possible multi-time interval pattern and then does not imply that I_2 precedes I_1).

2.1 The approach

The inputs for the approach are: support threshold; the attributes chosen (to define the events), and the intervals of interest (as in Figure 2). Moreover, as it will be explained below, if the values of one or more columns spread out too diffusely, they can be discretized into a limited number of classes. Given such inputs, we proceed as follows:

Step 0 (selection of single events):
- Select all single events with support $>, \geq$ the support given by the user

Step 1 (patterns composed of two events):
- Starting from the selected single events, search for all possible patterns according to given I_o, I_1, I_2…
- Count all the patterns at this step and select those patterns whose support is $>, \geq$ the support given by the user

…

Step *i* (patterns composed of *i+1 events)*:
- Starting from the selected patterns composed of *i* events, search for all possible patterns according to given ranges I_o, I_1, I_2…
- Count all the patterns at this step and select those patterns whose support is $> , \geq$ user's support

Stop Condition: no more new patterns are found

The output is a list of multi-time interval patterns having the form "Event Interval Event…" as in the given definition. Moreover, according to the above definition of pattern and degree of interest, we are able to find the most frequent patterns as well as the least frequent ones. However, due to the anti-monotonocity property[2], the less frequent patterns are "local" to each step and no further patterns can be found starting from them.

Beside that, as it was previously outlined, our approach allows to customize the intervals according to users' or tasks requirements, both in terms of their amount and in terms of their end points. To this aim, different granularities are currently allowed: milliseconds, seconds, minutes, hours, days, weeks, months, quarters, and years. In this way we allow a more flexible use of human time and give the user the possibility to discover more interesting patterns.

Finally, note that the values of a certain attribute may spread out too diffusely (as for the attribute "*Turnover*" in Figure 1(b)). This can lead to the creation of too many different events, or to consider events like (#E=24 AND

[2] The anti-monotonocity property states that if a multi-time-interval pattern is frequent, so are all of its composing patterns (sub patterns). Accordingly, if a multi-time-interval pattern is not frequent, all the patterns starting from it (super patterns) will not be either.

T=3537) and (#E=24 AND T=3493) as different, whereby for most of the analysis they could be considered the same. Therefore, a *discretization step* (or classification step) helps to reduce the amount of different events.

2.2 Application to real world data

The **dataset** analyzed concerns data coming from a flight company. Along with the actual number of passengers of economy and business class, the data contain information about departure time, flight number, flight destination (as airport code) and the number of seats on the flight (both economy and business). Moreover, the data also provide information about the number of expected passengers for any flight and day (economy and business), according to a not given forecast algorithm, whose forecast are provided about two months in advance and then further released six, four, and two weeks before the scheduled departure. The aim of the company is to find interesting patterns in the data and to possibly improve their forecast.

The first step of our analysis was devoted to a short **pre-processing phase**: we added the city name to the destination, the distance between departure and destination, and a measure of how good the forecast is. To this aim, rather than the difference between the actual number of economy and business passengers (for a certain flight and date) and those provided by the forecast, this measure ("how good is the forecast") expresses the absolute error of such a difference with respect to the whole number of passengers. As a matter of fact, having on board 20 persons more or less for a Boeing 767 with about 300 available seats is completely different from the same situation for a small private jet or aircraft such as an ATR 42 (which have between 40 and 50 seats). Moreover, we divided each day in *Morning* (from 7 till 12), *Afternoon* (from 12 till 19), Evening (from 19 till 24) and Night (from 0 till 7) as well as we added the possibility to distinguish between *businessweek* (Monday to Friday) and *weekend* (Saturday and Sunday). As shown in the "*Event Configuration*" (see Figure 3), for the first analysis, the destinations are classified into short distance (0-500 km) and middle-distance destinations (500-1500 km), the part of day is considered as an exact value as well as the day of the week, and the absolute error is divided into small error (from 0 to 0.3) and large error (from 0.3 to 1). Moreover, the three intervals chosen span from 0 to 12 hours, from 12 hours to 1 day and from 1 day to 3 days respectively.

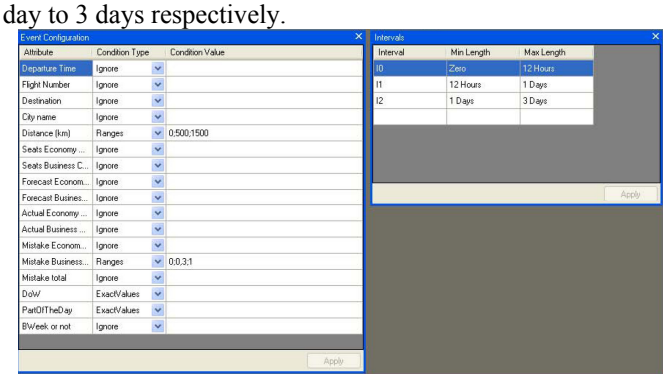


Figure 3: Event Configuration window (left) allows to choose the attributes of interest in order to define the events (e.g., Day of the week, part of the day) and possibly to discretize one or more of them (e.g., Distance, Mistake Business); the Interval window (right) lets the user define the intervals within which to look for patterns

The **results** of the analysis shows that there are clearly some similar behaviours between passengers of economy and business class, that is, the most occurring multi-time interval pattern occurs between Friday and Sunday (then in what in the social time is usually know as "long weekend"). However, firstly they differ in the temporal occurrence of this pattern, that is, Friday morning-Friday afternoon-Sunday afternoon for the economy class and Friday afternoon-Saturday morning-Sunday afternoon for the business class. Secondly and most interestingly, they differ in the absolute error: whereas for the economy class is a small one, for the business class is greater than 0.3.

On top of such results, a further analysis has been conducted. Since it may be relevant to know whether the error between the forecast and the actual number of passenger is negative or positive, that is, whether there were more or less passengers than expected, we added such information to the analysis. Moreover, we divided the day into *Business Morning* (from 6 to 9), *Business Evening* (from 16 to 24), and so called *Tourist time* (from 9 to 16), to distinguish between business trips and leisure trips. In this case, the most occurring patterns related to the economy class occur not only during the weekend (or long weekend), but also during the business week. However, the error is always limited, assuming both negative and positive values. Concerning the most occurring patterns related to the business class, they occur both in the weekend and during the business week, but more interestingly they occur with

a short time interval in between during the same day of the week (i.e. Friday). Moreover, the error is mainly focused on positive value in the small positive range.

3. RELATED WORKS AND DISCUSSION

Data Mining concerns the search for *patterns* of interest in a particular representational form (Fayyad et al. 1996). However, there is no fully accepted definition of pattern in the scientific community. Definitions vary depending on the field of application or research, e.g., in the genetic field (Sacchi et al., 2007) or in the discovery of events or sequences of events (Bettini et al. 1996). Nevertheless a comparison of such definitions is out of the scope of the paper. Therefore, in this section we provide a brief overview on the previous research in the field of Temporal Data Mining, in particular the part related to interval mining and sequence pattern mining.

Apart from the easiest solution (i.e., ignoring the time intervals completely) which we do not take into account, the time-window approach is well known (e.g., Mannila et al. (1995), Srikant and Agrawal (1996)). However, both methods cannot find time intervals between events and may miss patterns if they exceed the window size.

The approach we propose extends some ideas about how to find sequential patterns with time intervals to non transactional databases and deals explicitly with time-oriented data. Then, we want to outline in particular the differences with methods adopting "Apriori strategies", those focusing on temporal annotations and the method illustrated by Magnusson (2000).

Regarding the methods adopting on Apriori strategies, Chen et al. (2003) and Yang (2003) focus on the time intervals between successive items (*time interval sequential patterns*), whereas Hu et al. (2009) reprise and extend these ideas also adding the possibility to deal with different time-intervals between different pairs of items. More in details, the first main difference concerns the pruning strategies: Chen et al. (2003), Yang (2003) and Hu et al. (2009) propose three pruning strategies, i.e., descending property, the time interval information matrix and the anti-monotonocity property. Dealing with non transactional databases and due to the fact that the datasets we consider are composed by non overlapping intervals (see Figure 2), in our case we do not have to take into account the first two strategies. On the contrary, the third property holds also in our case. Moreover, concerning the definition of the intervals, Chen et al. (2003) and Yang (2003) and Hu et al. (2009) propose a set of fixed time intervals beforehand, so that they automatically obtain five equally deep intervals, containing roughly the same number of data. In our case, we pose our focus not only on the duration of the intervals between the events, but also on the temporal reference as well as on the social aspects of time. As a matter of fact, we defined not only six different default intervals which are relevant to the social time for several fields of applications, but we also allow users to define their own intervals, both in term of their amount and in term of their end points.

Regarding the methods focusing on temporal annotations, Yoshida et al. (2000) propose an algorithm to find so called *delta patterns*. However, they do not consider any solution to find all of them and they are limited to serial phenomena. Vautier et al.(2000) extend the notion of delta patterns with the use of so called *chronicles*, so that interval bounds of chronicles can be both positive and negative, and they can express both serial and parallel phenomena. Similarly, Giannotti et al. (2006) introduce the notion of *Temporally-annotated sequences*: each transition in a sequential pattern is annotated with a typical transition time derived from the source data and then any annotation is used to feed a clustering algorithm in order to determine the representative values over a certain support. However, as the two previous attempts, they neither take into account social time aspects, nor allow any interval choice, neglecting any user influence.

Concerning the method proposed by Magnusson (2000), firstly he defines *T-Patterns* and propose an algorithm to detect particular relationships (*critical intervals*) between pair of events. Though such approach is particularly interesting, since it also provides a visualization (the *Theme*[3] program), there are still some differences. Firstly, Magnusson proposes a bottom up, level-by-level (or breadth-first) detection strategy by which simpler patterns are detected first, whereas more complex patterns are detected as patterns of simpler ones. In our case, we reduce the amount of time windows taking into account either six default different intervals or the user defined ones. Secondly, the temporal information in between is not given explicitly by Magnusson, whereas in our approach the intervals are explicitly provided.

In general, apart from the above differences, our approach gives the user the possibility to be directly involved in the analysis, by choosing the attributes of interest, the intervals in between, or even providing a support

[3] *Theme* (the implemented version of Magnusson's method) shows the observation period as a time line under the plot. Using such line one can indirectly obtain the temporal information in between.

threshold. Furthermore, due to the methodology with which the events are created, we are able to deal not only with numerical but also with categorical attributes and both of them at the same time.

4. CONCLUSION AND FUTURE WORK

In this paper we presented MuTIny, a novel approach which is able to discover multi-time interval patterns and how it can be customized according to users' needs. In particular, we focused on the relevance of preserving temporal information between events and patterns, as this aspect is usually neglected or ignored. Moreover, we illustrated how not only numerical attributes, but also categorical attributes can be mined.

Concerning future work, in order to overcome the actual limitations of the approach, we plan to provide the possibility not only to define events but to allow events with different length. In the former case, e.g., the event is created from the payday set on the 15th of each month and the turnover, in order to investigate the hypothesis that during the week after the payday, the turnover will be higher and more employees will be required. In the latter case, e.g., an event could be one day long, another one two days and the third one a week long.

Finally, one of the most interesting challenges is to properly visualize our approach. To this aim, we are designing a visual analytics approach, which allows for presenting an overview of available data and results, and permits an easy parameterization. Adopting an intuitive interaction with each visual component and during each step of the approach we are able to encompass in the discovery process both the analytical and the visual methods.

ACKNOWLEDGEMENT

This work was partially supported by the program "FIT-IT Visual Computing", Project number: 813388.

REFERENCES

Aigner W., Bertone A., Miksch S., Schumann H. and Tominski C., Towards a Conceptual Framework for Visual Analytics of Time and Time-Oriented Data, Henderson, S. G.; Biller, B.; Hsieh, M.; Tew, J. D. & Barton, R. R. (eds.), Proceedings of the 2007 Winter Simulation Conference, p. 721-729, Washington, D.C., December, 2007.

Bettini C., Wang X.S. and Jajodia S.,. Testing Complex Temporal Relationships Involving Multiple Granularities and Its Application to Data Mining. Proc. of ACM PODS, pp. 68-78, Montreal, 1996.

Chen Y. L., Chiang M. C. and Kao M. T., Discovering time-interval sequential patterns in sequence databases, Expert Systems with Applications, 25 (3), pp.343-354, 2003.

Fayyad U., Smyth P., Piatetetsky-Shapiro G. and Uthurusamy R., Advances in knowledge discovery and data mining, AAAI Press/The MIT Press, 1996.

Frawley W.J., Piatetsky-Shapiro G. and Matheus C.J., Knowledge Discovery in Databases: An Overview, MIT press, 1991.

Giannotti F., Nanni M., Pedreschi D., Pinelli F., Mining sequences with temporal annotations, Proceedings of the 2006 ACM symposium on Applied computing, , Dijon, France, April 23-27, 2006.

Hu Y.-H., Huang T.C., Yang H.-R ,and Chen Y.-L, On mining multi-time-interval sequential patterns, Data & Knowledge Engineering, Vol. 68, No. 10, (23 October 2009), pp. 1112-1127, 2009.

Magnusson M.S., Discovering Hidden Time Patterns in Behavior: T-Patterns and their Detection. Behavior Research Methods, Instruments and Computers, 32(1): pp. 93-110, 2000.

Mannila H., Toivonen H., Verkamo A.I., Discovery of Frequent Episodes in Event Sequences, Data Mining and Knowledge Discovery, Volume 1, Issue 3, pp. 259 – 289, 1997.

Mannila H., Toivonen H.and Verkamo A.I., Discovering frequent episodes in sequences. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining. Montreal, Canada, pp. 210–215, 1995.

Sacchi L, Larizza C., Combi C. and Bellezzi R., Data mining with Temporal Abstractions: learning rules from time series, Data Mining and Knowledge Discovery archive, Volume 15 , Issue 2, pp.: 217-247, 2007.

Srikant R. and Agrawal. R, Mining Sequential Patterns: Generalizations and Performance Improvements. In Proc. of the Fifth Int'l Conference on Extending Database Technology (EDBT), Avignon, France, 1996.

Vautier, M.-O. Cordier, and R. Quiniou. An inductive database for mining temporal patterns in event sequences. In Workshop on Mining Spatial and Temporal Data, 2000.

Yang H.R., Discovering multi-time-interval sequential patterns in sequence database, http://thesis.lib.ncu.edu.tw/ETD-db/ETD-search-c/view_etd?URN=91423007 (accessed on 18/02/2010).

Yoshida M., Iizuka T., Shiohara H., and M. Ishiguro, Mining sequential patterns including time intervals, In "Data Mining and Knowledge Discovery: Theory, Tools and Technology II (SPIE Conference)", 2000.