autoCEP: Automatic Learning of Predictive Rules for Complex Event Processing

Raef Mousheimish(✉), Yehia Taher, and Karine Zeitouni

DAVID Laboratory, University of Versailles, 78000 Versailles, France
{raef.mousheimish,yehia.taher,karine.zeitouni}@uvsq.fr

Abstract. Complex Event Processing (CEP) is becoming more and more popular in service-oriented practices, especially to monitor the behaviour of continuous tasks within manual business processes, such as in logistics. The inference mechanisms of CEP engines are completely guided by rules, which are specified manually by domain experts. We argue that this user-based rule specification is a limiting factor that complicates the integration of CEP within the realm of Business Process Management (BPM) in a seamless way. Therefore, we present autoCEP as a two-phase data mining-based approach that automatically learns CEP rules from historical traces. In the first phase, complex temporal patterns are learned using early classification on time series techniques, then these patterns are algorithmically transformed into CEP rules in the second phase. Satisfactory results from evaluations on real data demonstrate the effectiveness of our framework.

Keywords: Complex event processing · Rule learning · Time series data mining · Violation prediction

1 Introduction

Manual processes are challenging to support as they usually contain continuous and dynamic tasks such as a Trucking activity in a logistics process. These tasks require an event-based processing with fine granularity, the thing that is beyond the reach of current activity-based BPMS. To cope with this challenge, researchers in the domain have found no solution better than exploiting CEP techniques to extend BPMS capabilities. Therefore this topic is storming the research in the area of BPM/CEP recently, and so many approaches [1–3,5,7] and a European project[1] are held on the subject.

Despite the noticeable amount of proposals, they have all disregarded the fact that the current standard way to define CEP rules is by writing them manually, and human users are in charge of this specification. Depending on the situation to detect (or predict) rules may become easily complicated, and this will add extra burden while managing business processes.

[1] http://getservice-project.eu/.

DOI: 10.1007/978-3-319-46295-0_38

We deem the ultimate fact that experts are in charge of writing rules as a limiting factor for the prosperity and diffusion of CEP, especially that it holds the seamless integration within BPMS, and it restrains the jump towards the next phase of event-driven systems, i.e., proactive complex event processing. To turn around this limitation and instead of manually defining rules, we stress the need to step further, where rules could be extracted, learned from histories, and deployed into engines in an automatic manner with the minimum intervention of humans.

This paper proposes a novel two-phase framework that relies on data mining techniques, more specifically **early classification on time series**. The framework learns historical trends and patterns at the first phase, and then algorithmically transforms them into CEP rules at the second one. Thus addressing the problem of automatic rules generation. In general, the paper makes the following contributions: (1) It is the first approach to integrate time series data mining techniques within the domain of CEP. (2) Automatic learning of **predictive** CEP rules. (3) Any user can now employ an out-of-the-box configured CEP engine without the requirement of being a technical expert in the domain. (4) Since no expertise is required to use autoCEP, it could be seamlessly integrated within BPMS.

2 Background

Univariate Time Series: A univariate time series T is a sequence of real values for one attribute, $T = \{t_1, t_2, ..., t_N\}$. It is attributed a length and a class. The Euclidean distance is used to measure the similarity between two time series of the same length, denoted as $||T_1, T_2||$. In order to calculate the similarity between two time series of different lengths, e.g., s and T where $|s| = n < |T| = N$, one searches for the minimum distance between s and all subsequences q_i of T that has the same length as s, $|q_i| = |s|$. This distance is called the Best Matching Distance (BMD).

Shapelets: A shapelet is a new primitive for data mining that emerged recently [14], it is a temporal pattern that characterizes the time series of the same class. A shapelet is defined as a triple $\hat{s} = (s, \delta, c_s)$, where s is the subsequence that constitutes the shapelet $\hat{s}$, δ is the distance threshold that is going to be used for the run-time classification, c_s is the class of the shapelet. Taking this definition into account, new unclassified instances of time series T are labeled as early as possible with the same class as a specific shapelet $\hat{s}$, if the similarity between them $||\hat{s}, T||$ is less or equal to the distance threshold δ.

Complex Event Processing: CEP rules are defined using different CEP operators like windowing, selection, sequence, etc. These operators are considered the main enabler to define complex patterns. Regardless of the various concrete models, we will keep an abstract representation for rules that could be expressed in any description language. A CEP rule is divided into three blocks. First the timeframe (or window) of the rule, which is defined using the **within** construct.

Second the filter block, which contains the events that are relevant for the rule, they will be written between two curly brackets **{}**. Finally the conditions that need to be met on the captured sequence of events in order for the rule to be fired, this block is defined using the **where** construct. In general:

$$\textbf{within}[\mathit{window}]\ \boldsymbol{\{}\mathit{relevant\ events}\boldsymbol{\}}\ \textbf{where}[\mathit{conditions}] \qquad (1)$$

3 autoCEP: From History Records to CEP Rules

3.1 High-Level Framework

Figure 1 sheds some light on the proposed two-phase framework from a high-level perspective.

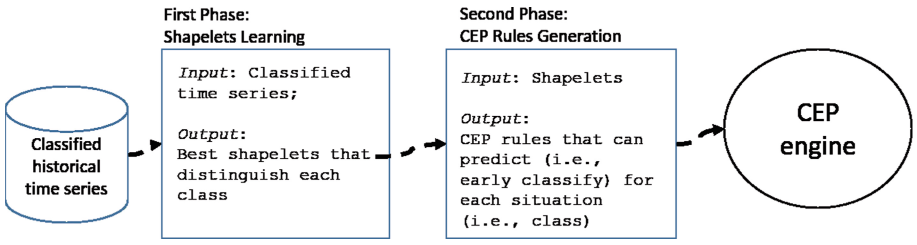

First Phase:
Shapelets Learning
Second Phase:
CEP Rules Generation
Classified historical time series
Input: Classified time series;
Output: Best shapelets that distinguish each class
Input: Shapelets
Output: CEP rules that can predict (i.e., early classify) for each situation (i.e., class)
CEP engine

Fig. 1. Two phase high level framework

4 First Phase: Shapelets Learning

This stage contains our learning algorithm that can extract shapelets with the highest utility scores by learning them from historical time series. This algorithm is the outcome of surveying recent state-of-the-art approaches [4,6,12–14] regarding this kind of classification problems.

The algorithm that we have implemented to learn the shapelets is a brute force extraction algorithm [14] because it yields the most accurate results. In addition to the classified history, this algorithm requires two other input parameters, the minimum and the maximum length of the shapelets, i.e., to specify that the learned shapelets need to be between these two lengths. This is the place where domain expert knowledge could be exploited in order to guide the learning process, i.e., if experts have any prior knowledge about the lengths of the patterns to learn. However, autoCEP offers the capability for users without knowledge to fill these inputs with default values (the minimum and the maximum possible). In the evaluation section, we show the effect of these parameters on the performance of autoCEP.

5 Second Phase: CEP Rules Generation

At this point, the learned shapelets serve as inputs, where they will be automatically transformed into CEP rules to be used later for predictions.

The proposed algorithm in this phase extracts the three building blocks of the rule from the input shapelets and their parameters. For each shapelet a CEP rule is created, and thus we overcome the limitation of assuming just one rule for each composite event, which is the assumption that is made by other approaches [8].
Given a shapelet $\hat{s} = (s, \delta, c_s)$, the window parameter win for the **within** block is derived directly from the length of the shapelet, $win = |s|$. Then the relevant stream of events are of the same type as the elements that constitute the sequence s of the shapelet. Finally, the condition to be met in order to predict if a stream of incoming events correspond to c_s (the same class as the shapelet) is that $\hat{s}$ needs to cover the stream within the window win. This is listed in the following algorithm (Algorithm 1).

```
Input: A set of shapelets $\hat{S}$
Output: A set of CEP rules $rules$
$rules \leftarrow \emptyset$;
for each shapelet $\hat{s}$ in $\hat{S}$ do
    /* create an empty cep rule $cep$ */
    $win \leftarrow |\hat{s}.s|$;
    $cep.setWindowBlock(win)$;
    $E \leftarrow$ Extract event types from $s$;
    $cep.setEventTypes(E)$;
    $cep.setConditionBlock(||\hat{s}, E|| \leq \delta)$;
    $cep.setListener($ this stream is predicted to belong to the class $c_s)$;
    $rules.add(cep)$;
end
return $rules$;
```

Algorithm 1. Transforming Shapelets into CEP Rules

6 Experiments

Interested readers are encouraged to download the programs that we have implemented from GitHub[2].

Two of the important factors that we are really interested in are the accuracy and the earliness of the predictions. To calculate the *Avg.f-score* (accuracy) we employed this formula (where C designates the set of classes):

$$\frac{1}{|C|} \sum_{cl \in C} \frac{2 \times precision(cl) \times recall(cl)}{precision(cl) + recall(cl)} \tag{2}$$

with $precision(cl) = \frac{TP}{TP+FP}$ and $recall(cl) = \frac{TP}{TP+FN}$.

[2] https://github.com/rmgitting/autoCEP.

On the other hand, the earliness is computed from the average percentage of time points needed to make the predictions (i.e., how much in advance regarding the whole length of the tested time series). Given a dataset D, a time series T, and the shapelet $\hat{s}$ that was matched with T, we calculate the earliness percentage as (EMT is the point in time when $\hat{s}$ matched with T; how much data points was read from T):

$$\frac{1}{|D|}\sum_{T\in D}\frac{EMT(\hat{s},T)}{|T|} \tag{3}$$

Artworks Transportation: In this kind of transport processes, the involved parties are interested in analyzing temperature readings and predict them in advance to prevent violations whenever possible (i.e., trespassing a minimum or a maximum threshold). These violations will eventually affect the qualities of the transported piece of arts. Table 1 presents information about the training and the evaluation data sets.

Table 1. Training and evaluation data sets (transport of artworks)

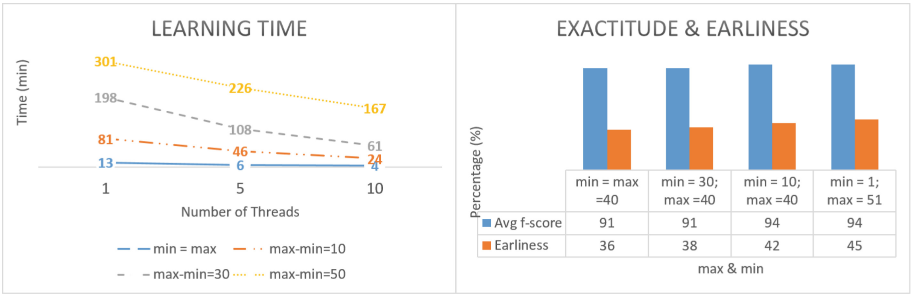
	Violated scenarios	Normal scenarios	Longest series	Shortest series
Train	16	17	451	51
Eval	17	17	460	39

The learning algorithm is implemented following a concurrent computing methodology, therefore the codes that build the shapelets and calculate their attributes are distributed over a pool of threads. From another point of view, the minimum and the maximum lengths of the shapelets that are provided as inputs for the algorithm may have some impacts on the performance of the framework as well. To this end, we ran different experiments to study the effects of the aforementioned factors.

The left side of Fig. 2 illustrates the learning time in minutes regarding two factors: the number of employed threads and the difference between the provided maximum and minimum lengths for the shapelets ($max - min$). On the other hand, the right side of Fig. 2 depicts the average f-score and the earliness of the framework when given larger spaces to build shapelets (different min and max).

6.1 Discussion

The tests done to predict temperature violations demonstrate that employing more threads can indeed improve the learning time. The graph in the left side of Fig. 2 also shows that this time is directly affected by the maximum and minimum lengths of the shapelets, because big differences between these lengths mean bigger spaces to search for patterns. The experience and the prior knowledge of domain experts should be used to calibrate these lengths and guide the



	min = max =40	min = 30; max =40	min = 10; max =40	min = 1; max = 51
🟦 Avg f-score	91	91	94	94
🟧 Earliness	36	38	42	45

Fig. 2. The results of the experiments

learning phase. Although this is recommended but not required, and the software provides non domain experts the capability to set default values for the maximum and minimum lengths which comes on the expense of slower learning. A particular strength to note is the classification time (at run-time), which is as fast as the events in the window are received. This instantaneous run-time classification is the result of exploiting the processing speed of CEP engines instead of writing ad-hoc classification algorithms.

The experiments also show that the earliness and the accuracy are affected by the values of the maximum and minimum lengths (Fig. 2 right side), as giving the framework more space to search for patterns will eventually help it to detect more useful and smaller shapelets. The autoCEP framework proved to maintain high accuracies regardless of the shapelet lengths but with different earliness percentages.

7 Related Work

Little work exists on the learning of CEP rules, and to the best of our knowledge, only four related approaches [8–11] have suggested to take the path of integrating data mining to this end. However, none of them is capable of dealing with periodic and numeric readings of events or with trends recognition. In addition, all these approaches focus only on the detection of situations of interest, and not on the prediction. So as far as we can tell, this is the only work that is done on: first integrating trend mining techniques with CEP, and second automatically learning **predictive** CEP rules.

Authors in [8] proposed the iCEP framework for the automatic learning of CEP rules. The problem of learning CEP rules in this works is boiled down to the learning of the operators of these rules. Authors followed a flexible modular architecture, where they associated each operator with a module. Therefore in each module, ad-hoc algorithms could be used to learn one specific operator, and build one part of the rule. Although the followed methodology has its strong points regarding the rule expressiveness that it tries to achieve, but the most limiting factor is that it counts on the strict intersection theory that leads to one and only one rule. In other words, the proposals will only work under the assumption that for each situation of interest there is just **one** rule that leads to it. We argue that in real life different rules may indeed lead to the same situation.

Another approach to integrate data mining techniques is proposed in [11]. The authors have suggested an iterative framework (prediction-correction) to address the problem of CEP rules learning. The general idea is to first initiate the rule parameters with some arbitrary values, and then tune them after each iteration depending on human experts' feedback. To give more details, the domain of observed events is divided into time-based intervals. Then at the end of each interval, experts need to highlight the false positives and negatives, so the parameters of the rule could be tuned and used in the next interval. The tuning relies mainly on discrete Kalman filters. First, we noticed that the approach cannot learn the rules completely by itself, but experts need to create templates with placeholders for rules parameters, and then the framework will learn these parameters. Secondly, it is very user centric, and it requires the intervention of experts after each interval or window. Thirdly, the interval is not learned but it needs to be specified by experts, which is not an evident task. In a similar iterative fashion, the approach proposed in [10] helps experts to refine and tune the rules parameters, but it lacks the capability to learn rules completely.

In the work discussed in [9], authors proposed an extension for the hidden Markov models, called noise Hidden Markov Models or nHMM. These extended models could learn sequences of events but they could also discard the noise. More specifically, when a noisy event is received, the Markov model stays on the same state, and does not proceed to the next one. In general the work is more concerned with the exclusion of noisy events rather than learning a complete rule. In addition the approach is demonstrated to work just on sequence patterns, but it cannot take windowing constraints into account.

8 Conclusion and Future Work

The main goal of our work is twofold. First we targeted the problem of automatic CEP rules learning in certain fields, and so sparing domain experts from this tedious task. Second, we tackled the learning of predictive rules and thus adding proactivity to the domain of CEP. Therefore, autoCEP paves the way for non-expert users to easily exploit the predictive capabilities of CEP engines, and it allows for a seamless integration within BPMSs.

We introduced a novel two-phase framework that efficiently tackles the automatic learning of CEP rules. It is well suited to work in application fields where primitive events are observations made periodically over time. The framework exploits the latest advancements in the domain of early classification on time series to learn accurate and predictive rules. Shapelets, which constitute a new primitive in the data mining field are learned at the first phase, and effectively transformed into CEP rules at the second one.

In the near future, we project to adopt our algorithms to favor multivariate time series, and thus support the processing of simultaneous events and the prediction using multidimensional temporal patterns.

References

1. Baumgrass, A., Ciccio, D., Claudio, C., Dijkman, R., Hewelt, M., Mendling, J.J., Meyer, A.A., Pourmirza, S.S., Weske, M.M., Wong, T.: GET controller and UNICORN: event-driven process execution and monitoring in logistics. In: CEUR Workshop Proceedings (2015)
2. Cabanillas, C., Baumgrass, A., Mendling, J., Rogetzer, P., Bellovoda, B.: Towards the enhancement of business process monitoring for complex logistics chains. In: Lohmann, N., Song, M., Wohed, P. (eds.) BPM 2013 Workshops. LNBIP, vol. 171, pp. 305–317. Springer, Heidelberg (2014)
3. Cabanillas, C., Di Ciccio, C., Mendling, J., Baumgrass, A.: Predictive task monitoring for business processes. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 424–432. Springer, Heidelberg (2014)
4. Ghalwash, M.F., Obradovic, Z.: Early classification of multivariate temporal observations by extraction of interpretable shapelets. BMC Bioinf. **13**(1), 1 (2012)
5. Herzberg, N., Meyer, A.: Improving process monitoring and progress prediction with data state transition events. In: ZEUS, pp. 20–23 (2013)
6. Lin, Y.-F., Chen, H.-H., Tseng, V.S., Pei, J.: Reliable early classification on multivariate time series with numerical and categorical attributes. In: Cao, T., Lim, E.-P., Zhou, Z.-H., Ho, T.-B., Cheung, D., Motoda, H. (eds.) PAKDD 2015. LNCS, vol. 9077, pp. 199–211. Springer, Heidelberg (2015)
7. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., Mylopoulos, J., Quix, C., Rolland, C., Manolopoulos, Y., Mouratidis, H., Horkoff, J. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Heidelberg (2014)
8. Margara, A., Cugola, G., Tamburrelli, G.: Learning from the past: automated rule generation for complex event processing. In: Proceedings of the 8th ACM International Conference on Distributed Event-Based Systems, pp. 47–58. ACM (2014)
9. Mutschler, C., Philippsen, M.: Learning event detection rules with noise hidden Markov models. In: 2012 NASA/ESA Conference on Adaptive Hardware and Systems (AHS), pp. 159–166. IEEE (2012)
10. Sen, S., Stojanovic, N., Stojanovic, L.: An approach for iterative event pattern recommendation. In: Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, pp. 196–205. ACM (2010)
11. Turchin, Y., Gal, A., Wasserkrug, S.: Tuning complex event processing rules using the prediction-correction paradigm. In: Proceedings of the Third ACM International Conference on Distributed Event-Based Systems, p. 10. ACM (2009)
12. Xing, Z., Pei, J., Dong, G., Philip, S.Y.: Mining sequence classifiers for early prediction. In: SDM, pp. 644–655. SIAM (2008)
13. Xing, Z., Pei, J., Philip, S.Y., Wang, K.: Extracting interpretable features for early classification on time series. In: SDM, vol. 11, pp. 247–258. SIAM (2011)
14. Ye, L., Keogh, E.: Time series shapelets: a new primitive for data mining. In: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 947–956. ACM (2009)