Analysis of Language Inspired Trace Representation for Anomaly Detection

Gabriel Marques Tavares[1(✉)] and Sylvio Barbon Jr.[2]

[1] Università degli Studi di Milano (UNIMI), Milan, Italy
gabriel.tavares@unimi.it
[2] Londrina State University (UEL), Londrina, Brazil
barbon@uel.br

Abstract. A great concern for organizations is to detect anomalous process instances within their business processes. For that, conformance checking performs model-aware analysis by comparing process logs to business models for the detection of anomalous process executions. However, in several scenarios, a model is either unavailable or its generation is costly, which requires the employment of alternative methods to allow a confident representation of traces. This work supports the analysis of language inspired process analysis grounded in the word2vec encoding algorithm. We argue that natural language encodings correctly model the behavior of business processes, supporting a proper distinction between common and anomalous behavior. In the experiments, we compared accuracy and time cost among different word2vec setups and classic encoding methods (token-based replay and alignment features), addressing seven different anomaly scenarios. Feature importance values and the impact of different anomalies in seven event logs were also evaluated to bring insights on the trace representation subject. Results show the proposed encoding overcomes representational capability of traditional conformance metrics for the anomaly detection task.

Keywords: Anomaly detection · Encoding · Business process · Natural language processing

1 Introduction

The assessment of anomalous business process executions is a real concern for organizations. Naturally, by detecting and mitigating wrongly executed processes, enterprises avoid frauds, save resources, and refine their methods. For that, companies rely on process-aware information systems, which is the group of software systems that support and control business processes [2]. Counting with a process notion, the operation of such software generates a log of events, i.e., the recording of activities within a process. From that, process analysis can be performed on such logs. Process Mining (PM) is the area aimed at extracting information and producing analysis starting from process data. In the PM realm,

the *event log* is the set of events executed under a business process. Furthermore, an event records the execution of an *activity* at a given *time*. Finally, each event relates to a single process instance, referred to as a *case*.

Traditionally, conformance checking is the PM task aimed at evaluating the behavior quantitatively. Conformance methods compare the process model to the event log, checking for deviations and further identifying anomalous instances [19]. Contrasting a process model with event log data is a common way of detecting anomalies. However, this is not feasible in scenarios without a process model. Thus, there is a need for algorithms that infer data patterns without any prior domain knowledge.

Given the necessity of detecting anomalous traces and the limitations of traditional approaches, we propose the usage of a language inspired trace representation. The method's core is based on Natural Language Processing (NLP) encoding of textual data. For that, we map activities and traces as words and sentences, respectively, before applying the word2vec encoding algorithm [13]. Word2vec captures contextual information, i.e., it models activities surroundings, such that each activity is represented by a vector. Traces with uncommon encoding are potential anomalies. We took advantage of the Random Forest machine learning algorithm to induce models for trace classification. Then, Random Forest's importance of features was used to support the discussion about encoding descriptive patterns.

The following sections are organized as follows. Section 2 presents anomaly detection works and encoding attempts in PM research. Section 3 reports (i) event log generation and anomaly injection, (ii) word2vec encoding and classical conformance metrics extraction, and (iii) the experimental setup for anomaly detection in business processes. Section 4 shows the obtained results, compares the encoding strategies, and evaluates how different anomaly types are perceived. The analysis is supported by accuracy, time and features importance metrics. Section 5 concludes the paper and leaves the final remarks.

2 Related Work

Anomaly detection in business process data has been extensively explored in recent years [3,7,14–17]. From the traditional PM pillars, conformance checking methods are the most used for anomaly detection. Conformance techniques rely on the comparison between a process model and an event log [2]. It follows that non-complying business cases can be interpreted as anomalous. This compliance is measured by the use of constraint satisfaction or transition marks, being employed either to control-flow or data-flow perspectives.

One of the earliest works for anomaly detection in PM uses a conformance checking pipeline [6]. The method filters the log based on domain-dependent knowledge and applies process discovery techniques to the filtered log. The most appropriate model is chosen as the process model. Thus, traces are classified depending on model fitting; that is, a non-fitting trace is classified as anomalous. However, this approach depends on a clean event log for model creation and

assumes that process discovery techniques might generate an ideal model, which is not necessarily true. Moreover, domain knowledge costs resources and is not always available.

More recent approaches explore the use of likelihood graphs to model process behavior and detect anomalous instances [7]. The encoding models both control and data-flow perspectives, and cases deviating from observed probability are classified as anomalous. However, the method introduces bias by connecting attributes to the graph, while the probabilities for activities execution are not connected to their attributes, which is inconsistent with real scenarios. The efficiency of this approach highly depends on the discovered process model quality, which is subjective in most cases. There is no consensus on whether the discovered model best represents log behavior, as the process of discovering a model comes from a trade-off between precision and generality.

Another family of approaches emerges from Machine Learning (ML) methods applied in business process contexts. In [14,16], the authors use an autoencoder to model process behavior. The technique encodes the event log using one-hot encoding and trains the autoencoder using the log as both the input and the output. The mean squared error between the input and output is measured and given a threshold, anomalous instances are highlighted. The main drawback of the approach is that vector sizes increase linearly with the number of activities, which is costly resource-wise. Moreover, the one-hot encoding technique produces very sparse vectors, further increasing computational overhead. To overcome this issue, in [15,17], the authors proposed a deep learning method considering both control and data-flow perspectives. The technique uses a deep neural network trained to predict the next event. Given the network probability score, an activity or attribute with a low execution probability is interpreted as an anomaly. However, the computational cost of deep learning methods is very high, which hinders its application in many scenarios.

Several techniques explore encoding, given that trace context is a determinant factor for anomaly detection. However, advanced deep learning methods demand high resource consumption and have limited interpretability. At the same time, traditional conformance approaches depend on model discovery, which is a challenging task. Our work bridges the gap between these two types of approaches by using a light-weight encoding technique based on NLP, taking advantage of activities context within a trace. The computational burden is considerably inferior when compared to deep learning methods, and without the need of a process model. Thus, combining the best aspects from conformance checking and deep learning approaches.

3 Methodology

3.1 Event Logs

One of the main goals of the experiments is to compare traditional trace metrics with trace encodings based on natural language models. Thus, a controlled scenario with known labels is the best way to evaluate the different modeling

approaches. Using the proposed framework for event log generation in [15], we created several business process logs based on the provided guidelines.

The PLG2 tool [9] was used to generated six random process models exploring various complexities, such as the number of activities, breadth and width. Moreover, a handmade procurement process model (P2P [17]) was added to the model pool. A likelihood graph [7] was adopted to introduce long-term control-flow dependencies, a common characteristic of real-world process logs. Such dependencies regard event to event transitions but also include event to attribute relations. This way, the probability distributions are constrained. For instance, an activity A has a determined probability of being followed by activity B. This can be extended to event attributes, such as an attribute C has a probability of being logged when activity A is executed in specific conditions. The combinations within a likelihood graph are extensive, thus providing a complex graph, mimicking real-world conditions and scenarios.

This way, models created in PLG2 are extended to likelihood graphs. From that, random walks in the graph generate the event log. Note that the random walks comply with transition probabilities, both for control-flow and data perspectives. The next step is to inject anomalous traces in the event log, a traditional practice in the literature [6,7]. As in the reference work [15], six elaborate anomaly types were applied: 1) Skip: a sequence of 3 or less necessary events is skipped; 2) Insert: 3 or less random activities inserted in the case; 3) Rework: a sequence of 3 or less necessary events is executed twice; 4) Early: a sequence of 2 or fewer events executed too early, which is then skipped later in the case; 5) Late: a sequence of 2 or fewer events executed too late, which is then skipped later in the case; 6) Attribute: an incorrect attribute value is set in 3 or fewer events.

The artificial anomalies are applied to 30% of the cases from previously generated event logs. The ground truth label is on the event level, however, it can be easily converted to the case level. Whenever a case has an anomalous event, the respective case is labeled as an anomaly. Table 1 reports the detailed event log statistics. Finally, note that the recreation of these event logs and their anomalies is replicable by following the steps reported in the original work [15].

Table 1. Event log statistics: each log contains different levels of complexity

Name	#Logs	#Activities	#Cases	#Events	#Attributes	#Attribute values
P2P	4	27	5k	48k–53k	1–4	13–386
Small	4	41	5k	53k–57k	1–4	13–360
Medium	4	65	5k	39k–42k	1–4	13–398
Large	4	85	5k	61k–68k	1–4	13–398
Huge	4	109	5k	47k–53k	1–4	13–420
Gigantic	4	154–157	5k	38k–42k	1–4	13–409
Wide	4	68–69	5k	39k–42k	1–4	13–382

ML techniques operate at the instance level, i.e., an event is a complete instance representation, while PM methods operate at the business case level, i.e., the group of events from the same case composes an instance. Due to this mismatch at the representation level, traditional ML algorithms can not be directly applied to business process logs [20,21]. This way, an encoding layer extracting case features is necessary to overcome this issue, thus, merging the gap between process science and ML methods.

Log encoding was already explored in the literature [14,16]. In [14], the authors use one-hot encoding whereas in [15,17], the authors use integer encoding. Both approaches transform the event log, and consequently traces, into a numerical representation, which is then fed to ML algorithms. However, one-hot and integer encodings generate sparse data. In [10], the authors represent a trace using the NLP *doc2vec* encoding [11]. Though the technique is innovative, its activity and trace representation lacks context as the doc2vec encoding is designed for paragraphs, which generally contain more data than a trace.

Inspired by NLP research, which has solid literature in representational learning, we propose the use of the *word2vec* encoding method to capture business process behavior [13]. Word2vec produces word encodings using a two-layer neural network aimed at reconstructing the linguistic context for each word in the corpus. The produced vectors model semantic and synthetic characteristics using the weights produced by the neural networks. Namely, words with similar contexts have analogous vectors. This way, our approach interprets each activity as being a word. By consequence, the set of unique activities is the corpus used by the word2vec model. Within a model, each activity is represented by a numerical vector describing its context. To retrieve trace-level encoding, we aggregate the word vectors that compose a respective trace. For that, we use element-wise mean, i.e., the trace representation is the mean of its activities representations.

3.2 Traditional Feature Engineering

A classical approach to evaluate business cases uses conformance checking techniques. Conformance aims to compare a process model to the event log and measure their differences [2]. This process must account for the alignment between trace and model elements. Here, we employ two of the most traditional conformance approaches: token-based replay and alignment. Token replay matches a trace to a process model in the Petri net format. The resulting value is usually a fitness score produced by firing tokens and accounting the mismatch between trace and model-allowed transitions. Further, following a comparison approach, trace alignment links trace activities into Petri net transitions [1]. The alignment is measured by comparing a trace to log moves, accounting moves that can or not be mimicked in the log, e.g., skipped and silent activities.

Both conformance techniques were implemented using the PM4Py Python package [5], following standard hyperparameters[1]. Table 2 shows the extracted

[1] https://pm4py.fit.fraunhofer.de/documentation.

features for each trace. These features are used for model creation and posterior anomaly detection. Moreover, both techniques require a Petri net model to compute conformance measures. This way, before applying feature extraction, we induct a process discovery algorithm using the Inductive Miner Directly Follows (IMDF) algorithm [12] (employing PM4Py library). IMDF was chosen since its goal is to construct a sound model with good fitness values, leveraging the quality of extracted features.

Table 2. Extracted trace features to describe trace behavior. There are two types of features: token replay and alignment

Feature type	Feature	Meaning
Token replay	*trace_is_fit*	Indicates if the trace fits the model
Token replay	*trace_fitness*	Trace fitness value
Token replay	*missing_tokens*	Number of missing tokens
Token replay	*consumed_tokens*	Number of consumed tokens
Token replay	*remaining_tokens*	Number of remaining tokens
Token replay	*produced_tokens*	Total number of tokens produced
Alignment	*cost*	Cost of the alignment
Alignment	*visited_states*	Number of visited states
Alignment	*queued_states*	Number of queued states
Alignment	*traversed_arcs*	Number of traversed arcs
Alignment	*fitness*	Trace fitness value

3.3 Experimental Setup

The experiments aim at measuring two main aspects: (i) how different anomalies behave and how much they affect classification performance, and (ii) to which extent traditional and natural language inspired encodings can represent log behavior, including anomalies. For that, we designed two sets of experiments. The first is a binary classification using only the normal class and one anomaly (this is replicated for all anomalies). The second experiment is a multi-class detection task, thus, using all the available classes. The latter experiment is more challenging as the encodings need to represent all different behaviors at the same time, considering that anomalies tend to harm encoding quality.

For the word2vec encoding, we explored several vector sizes (25, 50, 100, 200, 400), this way, evaluating if more complex vectors capture better trace behavior. The Random Forest (RF) [8] algorithm was used for the classification task following the *scikit-learn* Python package [18]. RF was selected due to its extensive use in ML literature. RF is very robust and controls overfitting with its ensemble nature. Moreover, RF requires less computational resources compared to deep

learning methods. To provide a common testbed for the different encodings, we implemented a grid search method for hyperparameter tuning, a standard technique in literature [4]. Grid search trials are formed by assembling all possible hyperparameter values combination. Table 3 lists the explored hyperparameters, their implication and employed values.

Table 3. Random Forest hyperparameters. A grid search method was used to combine all possible hyperparameter values, yelding an optimal performance

Hyperparameter	Meaning	Values
n_estimators	Number of forest trees	50, 100, 250, 500, 750, 1000
max_features	Number of features to consider for the best split	auto, log2
max_depth	Maximum tree depth	4, 8, 16, 32, 64, default
criterion	Function to measure split quality	gini, entropy

4 Results and Discussion

All the discussion in this section was made using a RF model induced using *n_estimators* of 50, *max_features* with log2, *max_depth* with default value and entropy as *criterion*. These values were found after the tuning procedure. The time presented was computed during the model induction period.

4.1 Overall Performance

An overview of predictive performance is exposed in Fig. 1. The accuracy to detect anomalies is sorted from left to right. The lowest performance was reported in the logs with all anomalies concurrently. Late, attribute and early anomalies followed with similar performance. The most accurate classifications were made over the insert, rework and skip anomalies.

It is easy to comprehend the low performance of all anomalies scenario since it is a multi-class problem in which the model needs to deal with seven different outcomes (common behavior and six different anomalies). Aggregating the performance of all models by each encoding, word2vec methods obtained superior performance (average of 84.7%) in comparison to the classic method (76.3%). No specific word2vec length outperformed the others, the obtained accuracies were 84.6%, 84.6%, 84.7%, 84.7% and 84.9% by 100, 200, 25, 50 and 400, respectively.

Late, attribute, and early anomalies were classified with a similar predictive performance by the RF models. Word2vec methods obtained an average accuracy of 93.3% while the classic method achieved a slightly superior performance of 93.5%. The most predictable anomalies were insert, rework and skip, where word2vec encoding obtained 99.8% accuracy. The standard deviation of $1e^{-3}$ within word2vec models shows that different lengths did not affect performance. Contrarily, classic encoding obtained inferior results, an average of

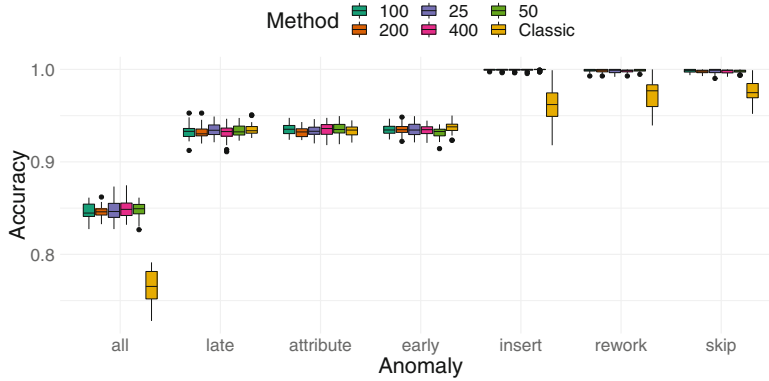


Fig. 1. Comparison of accuracy among word2vec using different lengths (25, 50, 100, 200 and 400) and classic encoding methods when representing seven different anomaly scenarios (all anomalies concurrently, late, attribute, early, insert, rework and skip)

96.9% ($\pm 1e^{-4}$), with insert as the less accurate classification (96.1%). Rework and skip obtained (97.1%) of accuracy. Overall, word2vec encoding had a better performance when compared to classical encoding.

Beyond the predictive performance, we performed a time analysis to deepen our discussion about the suitable method. Figure 2 presents violin plots built using the time to induce a RF model using different encoding methods. We choose this visualization to illustrate the average performance and to clarify some possible distortions of time due to different datasets compromised with the same anomaly.

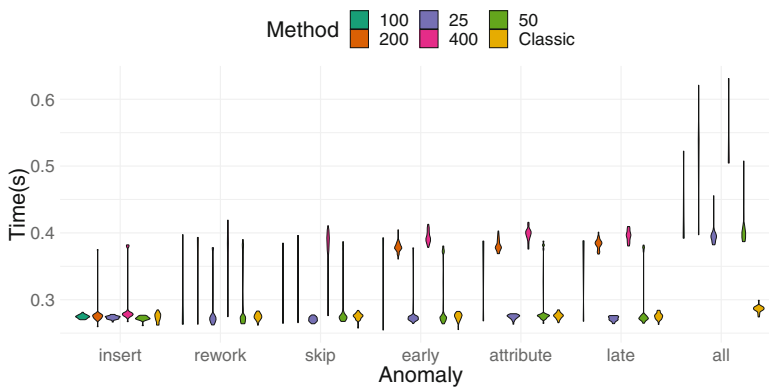


Fig. 2. Comparison of time among word2vec using different lengths (25, 50, 100, 200 and 400) and classic encoding method when representing seven different anomaly scenarios (all anomalies concurrently, late, attribute, early, insert, rework and skip)

Classic encoding takes a similar time for all anomalies, an average of 0.23 s ($\pm 6e^{-3}$), to induce a RF model, being the fastest encoding technique. Word2vec encodings required more time for particular anomalies (e.g., all concurrently, late, attribute and early), an average of 0.35 s (± 0.07). When modeling a classifier to the all anomalies scenario, this difference becomes more evident since classic encoding obtained 0.28 s and word2vecs 0.39 s, 0.40 s, 0.43 s, 0.48 s and 0.55 s for 25, 50, 100, 200 and 400, respectively. Regarding word2vec time performance, it is possible to observe a straightforward relation between time and feature vector length, as expected. An analogous relation was observed between feature vector length and stability of the model. Using the standard deviation of induction time for each word2vec range, we found the same behavior, the larger feature vector, the higher time variability. The most stable encoding was the smallest (25 features).

4.2 Encoding Strategy and Feature Importance

One of the key-points of an encoding method is its capability to support class disjunctions. Taking advantage of the RF models, we observed the importance of each variable grounded on RF importance. Figure 3 presents an overview of RF feature importance for classic encoding. The alignment family of features (*traversed_arcs*, *cost*, *visited_states*, *queued_states*) was the most important group, except by *fitness*. Token replay features did not contribute to the model.

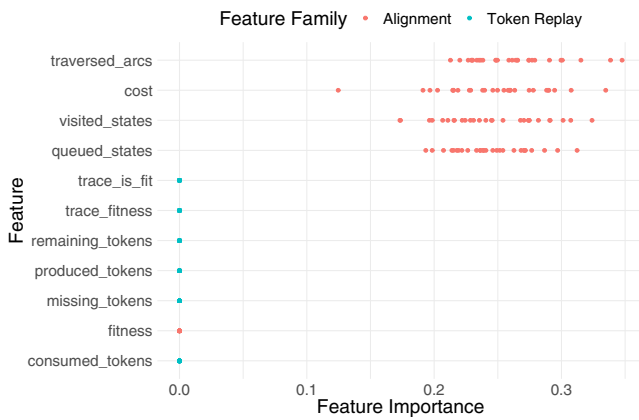
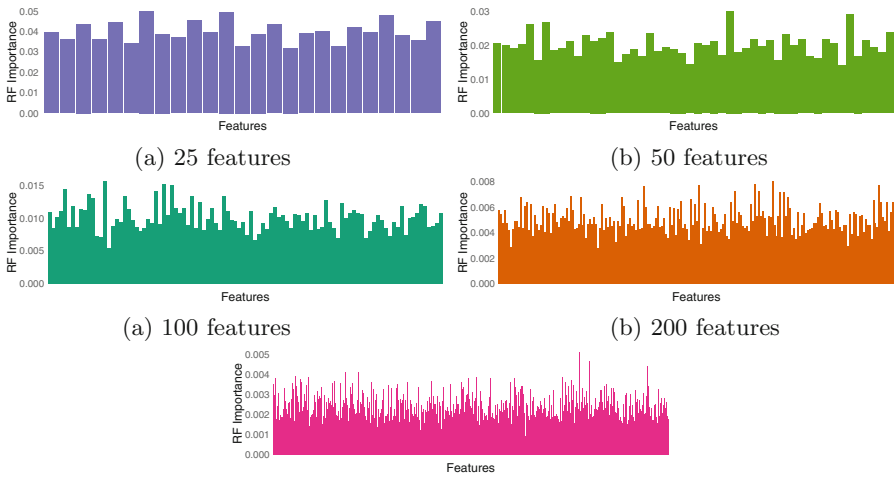


Fig. 3. RF feature importance from alignment and token replay encodings

The limited performance of token replay features happens as the approach tend to consistently produce high fitness values. Moreover, a path through the model is not created for non-fitting cases. Therefore, the same values for token replay features are produced to the majority of traces, adding no benefit in the trace encoding. Consequently, their significance for classification is downgraded,

which is represented by the low RF importance. On the other hand, the alignment family of features overcomes these limitations by introducing more robust rules [1], thus, being decisive in anomaly detection.

Figure 4 displays RF feature importance of word2vec encodings. No optimal subset of features can be found. Further, the importance of features spreads as vector length grows, i.e., the spike of importance had its value reduced as vector length enlarged.



(a) 25 features (b) 50 features

(a) 100 features (b) 200 features

(d) 400 features

Fig. 4. RF feature importance from five different word2vec lengths

The distribution of importance values over the feature vector exposed an intrinsic characteristic of word2vec encoding, the capacity of exploring the problem using an inputted granularity. In other words, this encoding method dispersed throughout all features the capacity of trace representation. Complex problems that demand a more precise description can take advantage of large feature vectors, at the cost of time and stability.

4.3 Anomalies Analysis

Figure 5 presents the average accuracy from all word2vec models over each event log. From this, *Gigantic* appears as the most complex event log with lower accuracies, obtaining its lowest classification performance when compromised by all anomalies. This is explained by the log complexity, which can be observed by the high number of activities (Table 1). Higher performances were observed over *Large*, *Small* and *Wide* event logs. Although there exists an accuracy variation on the datasets classification performance, we need to mention the most difference was presented in the anomaly perspective. When comparing the

average accuracy per dataset, a variation of 1% was observed. In contrast, the average accuracy difference within anomalies stood at 16%.

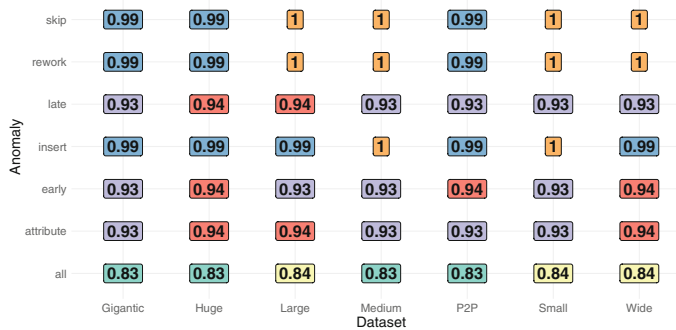


Fig. 5. Average accuracy obtained with word2vec encoding for each combination of dataset and anomaly

According to Fig. 5, late, attribute, and early anomalies are the most difficult of being recognized. Late and early anomalies affect an activity being executed before or after, respectively, its expected execution. This way, the trace context is affected more subtly because the activity is being executed, even if in the wrong position. The attribute anomaly is challenging as it affects the data-flow perspective, so methods that do not consider this flow tend to present lower performances. The skip, rework and insert anomalies affect more profoundly the control-flow perspective. This aggressive behavior is easily detected by the encoding methods, e.g., even the classical method was able to perform better in these anomalies (Fig. 1).

5 Conclusion

Conformance checking is one of the most important tasks of PM in real-life business applications, mainly for anomaly detection. This paper compared classic conformance features (token replay and alignment) with language inspired trace representation (word2vec) over different event logs compromised in seven different scenarios. A RF classification model was combined with the encoding techniques for the anomaly detection task. Word2vec correctly captured trace context and demonstrated a better performance than classic encodings. The most challenging scenario is dealing with all anomalies concomitantly, where lower accuracies were achieved. On the other hand, detecting *insert*, *rework*, and *skip* anomalies leveraged accuracy performance. In both extreme scenarios, i.e., higher and lower accuracies, word2vec overcame classic encoding. The performance results of encoding methods were similar just for medium-range accuracy scenarios (*late*, *attribute*, and *early* anomalies). Regarding different event

logs, *Gigantic* was the most complex. When comparing the importance of the classic encoding techniques, alignment features prevailed token replay features. As future work, we plan to investigate anomaly detection in online settings.

References

1. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Min. Knowl. Disc. **2**(2), 182–192 (2012)
2. van der Aalst, W.M.P.: Process Mining: Data Science in Action, 2nd edn. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
3. Barbon Junior, S., Tavares, G.M., da Costa, V.G.T., Ceravolo, P., Damiani, E.: A framework for human-in-the-loop monitoring of concept-drift detection in event log stream. In: Companion Proceedings of the The Web Conference 2018, pp. 319–326. WWW 2018, International World Wide Web Conferences Steering Committee (2018)
4. Bergstra, J., Bengio, Y.: Random search for hyper-parameter optimization. J. Mach. Learn. Res. **13**(10), 281–305 (2012)
5. Berti, A., van Zelst, S.J., van der Aalst, W.: Process mining for python (pm4py): Bridging the gap between process- and data science (2019)
6. Bezerra, F., Wainer, J.: Algorithms for anomaly detection of traces in logs of process aware information systems. Inf. Syst. **38**(1), 33–44 (2013)
7. Böhmer, K., Rinderle-Ma, S.: Multi-perspective anomaly detection in business process execution events. In: Debruyne, C., et al. (eds.) OTM 2016. LNCS, vol. 10033, pp. 80–98. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48472-3_5
8. Breiman, L.: Random forests. Mach. Learn. **45**(1), 5–32 (2001)
9. Burattin, A.: Plg2: Multiperspective processes randomization and simulation for online and offline settings (2015)
10. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and model2vec: representation learning for business processes. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 305–321. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_18
11. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on International Conference on Machine Learning, vol. 32. p. II-1188-II-1196. ICML 2014, JMLR.org (2014)
12. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery with guarantees. In: Gaaloul, K., Schmidt, R., Nurcan, S., Guerreiro, S., Ma, Q. (eds.) Enterprise, Business-Process and Information Systems Modeling, pp. 85–101. Springer, Cham (2015)
13. Mikolov, T., Chen, K., Corrado, G.S., Dean, J.: Efficient estimation of word representations in vector space. CoRR abs/1301.3781 (2013)
14. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Analyzing business process anomalies using autoencoders. Mach. Learn. **107**(11), 1875–1893 (2018)
15. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: Binet: multi-perspective business process anomaly classification. Inf. Syst. **1**, 101458 (2019)
16. Nolle, T., Seeliger, A., Mühlhäuser, M.: Unsupervised anomaly detection in noisy business process event logs using denoising autoencoders. In: Calders, T., Ceci, M., Malerba, D. (eds.) DS 2016. LNCS (LNAI), vol. 9956, pp. 442–456. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46307-0_28

17. Nolle, T., Seeliger, A., Mühlhäuser, M.: BINet: multivariate business process anomaly detection using deep learning. In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 271–287. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_16
18. Pedregosa, F., et al.: Scikit-learn: machine learning in python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)
19. Rozinat, A., van der Aalst, W.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. **33**(1), 64–95 (2008)
20. Tavares, G.M., Ceravolo, P., Turrisi Da Costa, V.G., Damiani, E., Barbon Junior, S.: Overlapping analytic stages in online process mining. In: 2019 IEEE International Conference on Services Computing (SCC), pp. 167–175, July 2019
21. Tavares, G.M., Turrisi Da Costa, V.G., Martins, V., Ceravolo, P., Barbon Junior, S.: Leveraging anomaly detection in business process with data stream mining. iSys - Revista Brasileira de Sistemas de Informação **12**(1), 54–75 (2019)