Case2vec: Advances in Representation Learning for Business Processes

Stefan Luettgen, Alexander Seeliger(✉), Timo Nolle, and Max Mühlhäuser

Telecooperation Lab, Technical University of Darmstadt, Darmstadt, Germany
{luettgen,seeliger,nolle,max}@tk.tu-darmstadt.de

Abstract. The execution of a business process is often determined by the surrounding context, e.g., department, product, or other attributes an event provides. Process discovery mainly focuses on the executed activities, although the context of a case may be needed to accurately represent a process instance, e.g., for clustering, prediction, or anomaly detection. Hence, in this paper, we present a representation learning technique (Case2vec) using word embeddings for business process data to better encode process instances. Our work extends Trace2vec and incorporates an additional semantic level by using not only the activity name but also the attributes and thereby incorporating the context. We evaluate our approach in the context of trace clustering. Additionally, we show that Case2vec can be used to abstract events which are semantically similar but syntactically different. We also show that word embeddings allow for interpretability when employing vector space arithmetic.

Keywords: Representation learning · Word embeddings · Process context

1 Introduction

In recent years, process mining has become an important technology for organizations analyzing their business processes. Event logs recorded by process-aware information systems can be analyzed with process mining to obtain valuable insights about how a business process is executed in reality. However, process mining techniques primarily focus on the control-flow of a process without considering the context a case is executed in, e.g., department, product, customer, or other attributes an event provides. This additional process context may help to further reveal patterns within the event log, which are not visible in the control-flow perspective, to enhance process mining techniques. Our goal is to learn vector representations of process cases that include this context information that can be used in various process mining techniques.

Vector representations of cases are required by many techniques in process mining such as trace clustering [4,11,12], prediction [3], and anomaly detection [10,13]. Trace clustering aims to improve the discovery of process models

S. Leemans and H. Leopold (Eds.): ICPM 2020 Workshops, LNBIP 406, pp. 162–174, 2021.
https://doi.org/10.1007/978-3-030-72693-5_13

by grouping similar cases. Clusters of cases that are executed in similar contexts can be generated, allowing the user to compare process models of different contexts. Improved prediction models can be learned that also consider the process contexts. Furthermore, anomaly detection methods based on extended vector representations can provide more reliable results. These are just a few examples for potential use cases of context-including vector representations.

Our work is based on a technique proposed in the area of natural language processing (NLP) for learning vector representations of words and sentences. Similar to a sentence with words, a case of a business process consists of a sequence of activities. Activities are also not executed in random order, but according to a predefined grammar, the underlying process model. The core idea is to model similarities and intentionally avoid comparing by words only, because we know that different words or sentences can bear the same meaning.

A previous work, Trace2vec [4], showed that the representation learning approach Word2vec [8], which constructs a vector space of the words of a corpus to capture similarities, can also be used on process data. To model such similarities, a large event log is crawled to order activities which occur together within this vector space. However, Trace2vec also showed some difficulties in the experiments with the BPIC15 event log: First, the vocabulary of event logs is much smaller compared to the vocabulary of natural language. Second, the context of a case is not taken into account, which can provide further details about the dependencies between activities and attributes. For instance, if the BPIC15 event log is clustered into the municipalities without considering the process context, it is assumed that the control-flow alone clearly determines the municipality. In highly standardized processes like governmental processes, the control-flow is the very part that does not separate one trace from another, but rather its context, e.g., an officer working exclusively in one or a few municipalities.

In this paper, we present an extended approach based on Trace2vec that can indeed lead to sensible results when evaluating these representations for a trace clustering task. We name this extension Case2vec, because it uses event and case attributes to capture the process context. Our extension increases the vocabulary that allows to better exploit case relationships. Besides our extension, we examine a proper hyperparameter strategy that can better deal with the sparse vocabulary in business process data. We revisit the original approach using the BPIC15 event log and show how parameter tuning and especially incorporating attributes improves results. We also show a wider range of results on the BPIC19 event log, which holds not only more traces, but also more attributes.

As additional tasks we investigate two useful applications of the neural network architecture presented: (1) *Event abstraction* allows to show that syntactically different activities are semantically similar, given enough traces in a similar context. (2) Arithmetic operations within the vector space keep semantic meaning which we show in an *interpretability task*. This is done on an artificial paper writing process to show the task more clearly because we know how the activities in this process depend on each other.

2 Related Work

Process case representations are used by various process mining techniques such as trace clustering, anomaly detection, and prediction. Different representations have been proposed in the related work. A simple representation technique is the bag-of-words model which is used to compute the similarity of sentences based on the co-occurrences. Song et al. [12] encode sequences of activities as one hot vectors, in which each component corresponds to an activity. Transitions between activities are used instead by Bose et al. [1] to compare cases.

Besides manually defined case representations, automatically generated representation vectors can be learned. For instance, a word embedding is a feature learning technique in which words are mapped to a vector space. Words appearing together frequently within a text corpus will be mapped close together within a vector space to capture their semantic relationship. Word embeddings do not rely on syntactical features and, therefore, can compute a similarity value of two sentences, even if none of the words of each sentence is the same. De Koninck et al. [4] transferred the idea of Word2vec [8] and Doc2vec [7] to process data. An LSTM and CBOW-based approach was introduced by Bui et al. [2]. A supervised representation learning approach based on conditional random fields for event abstraction was introduced by Tax et al. [14].

Representation learning has been used for different analysis methods. Trace2vec representations were used to cluster traces into similar groups in [4]. Tavares et al. [13] use the same representations to identify anomalous cases.

A drawback of most related work in this field is the limitation to the pure control-flow, namely the sequence of activities to learn case representations. Thus, the process context of the cases is not considered.

3 Case2vec

In NLP, word embeddings use the context of the words in a document to exploit semantic similarities of words by mapping them to a vector space. The closer these words appear together in the document the closer they are mapped together in the vector space. Thereby, semantic similarity of different statements can be confirmed as long as they are mapped close together.

As already mentioned, a popular technique for modeling word embeddings is Word2vec [8]. The task is to model what is in the neighborhood of a word. This can be done using two different approaches. We can predict a word given its surrounding words (continuous bag of words, short CBOW), or the other way round, predict the surrounding words given one word (skip-gram). For example, in the sentence *I like ... process mining*, continuous bag of words would insert words of similar representation to fill the gap, e.g., *business*. Vice versa, skip-gram would take the word *business* and amend it with preceding and succeeding words given by example sentences in the training data.

Word2vec learns a CBOW or skip-gram model using a neural network and implicitly constructs an abstract representation of the vocabulary and its

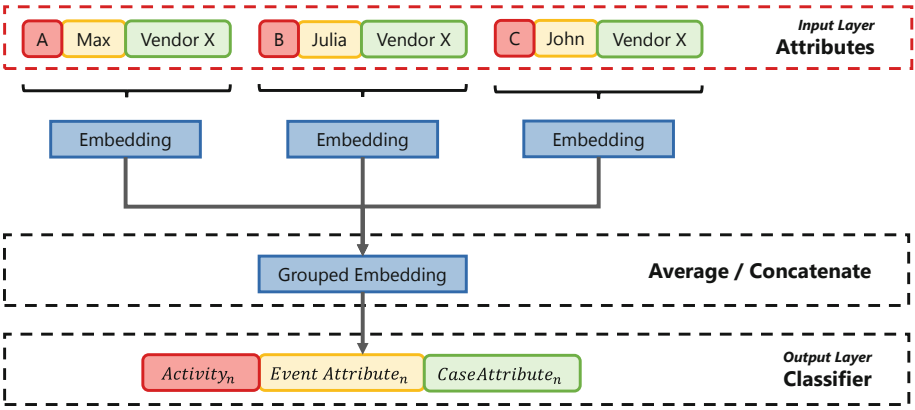

Fig. 1. Architectural overview of Case2vec where each trace's activity names, event, and case attributes are concatenated as single words.

relationships between each other. Similarly, activities within a sequence of a business process are also dependent on the preceding and succeeding activities which form the context. We can employ the idea of word embeddings and map activities to a vector space such that activities in similar regions are related to each other according to their function in the underlying process. Doc2vec serves as a representation of a collection of words, namely a document. Analogous to the Word2vec model, in Doc2vec a word is a document and we want to predict the surrounding documents. The structure of a trace from an event log is of similar form when considering activity names as words in a trace sequence. The resulting embedding space is a representation where activities and traces, given enough sample traces, are projected according to their role in the overall process model.

The embedding on the control-flow level is constructed by using the activity name as a single word. The set of different activity names forms the vocabulary of the embedding, and a Doc2vec representation is constructed by treating a trace as a document. The control-flow level (Fig. 1 without event and case attributes) has been introduced as Trace2vec [4]. One drawback of this approach is the focus on the control-flow. Therefore, we introduce Case2vec, which incorporates the different kinds of attributes by concatenating them with the corresponding activity name. The key idea is to incorporate attributes in addition to activity names to enlarge the vocabulary and induce a better separation of cases. If attributes are taken into account, the concatenation of the activity and its respective attributes becomes an additional word and, therefore, includes the process context.

Figure 1 shows the architecture with the attribute extension, where the words of the vocabulary are constructed by concatenating `Activity`, `Resource`, and `Vendor`. We also evaluate the approach either using event or case attributes.

4 Experimental Evaluation

We implemented[1] the described representation learning techniques using *gensim*, *scikit-learn* and *fastcluster* in Python to evaluate their performance. We use two Business Process Intelligence Challenge (BPIC) event logs, an amended version of them, and a fully synthetic paper writing process to evaluate on the following objectives: Trace clustering, event abstraction, and interpretability through vector arithmetic operations in the vector space.

In the following, we describe the event logs, the experimental setup and report the results.

4.1 Datasets

We use real-life and artificial event logs to evaluate the different objectives.

Real-Life Event Logs. We use the BPIC15 [5] and BPIC19 [6] event logs to compare the applicability of the different approaches. We select a case attribute for both event logs that can be considered as the ground truth label for clustering. Although we do not know in advance if this process provides features that will lead to good clustering results with this label, we are not necessarily interested in the best clustering result, but rather how incorporating different attributes can influence the clustering performance.

For BPIC15, event logs are already split into five different municipalities. In BPIC19, the case attribute `Item Type` is used as the cluster label without the `Standard` cases to obtain evenly distributed clusters.

During the experiments for event abstraction we amended the real-life event logs with noise or additional attributes. For the event abstraction task, we amended activity names with random numbers in a certain amount of traces to show that the method is robust to small changes in activity names.

Artificial Event Log: Paper Writing Process. The artificial example event log is based on a synthetically generated process depicted in Fig. 2. It describes the main steps in a scientific paper writing process from identifying a problem to the submission of the paper. The activities are dependent on each other according to their sequential order. This event log is more comprehensible for interpreting the results of the vector arithmetic experiment. For the experiments we sampled 5,000 traces of this process according to [9].

4.2 Real-Life Event Logs: Trace Clustering

In the first experiment, we use the case representations for clustering cases into their classes to show applicability for process context separation.

[1] Source code publicly available at: https://github.com/alexsee/case2vec.

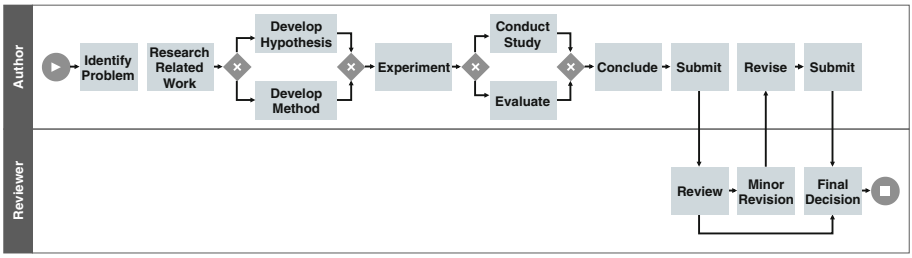

Fig. 2. Overview of the paper writing process [9].

Experimental Setup. Each event log is used individually to train the network according to the description in Sect. 3. For training, activities and attributes are used and none of the sequences are trimmed. Afterwards, we obtain the internal representation of each case and use the feature vectors as input for clustering.

Taking into account that process data in comparison to natural language has shorter sentence length and substantially smaller vocabulary, we employ a hyperparameter strategy to overfit the dataset for the clustering task. This is done using the ground truth label as an attribute with the goal to reach a Normalized Mutual Information (NMI) measure of 1.0 to ensure that the trained vector space has the capacity to model the underlying processes. This step is important before running the actual experiments to exclude weak results because of an impaired modeling capability of the underlying neural network. After a set of parameters is found that can overfit the dataset, the same optimization strategy can be employed during the actual experiments to maximize the NMI without the ground truth label.

In our parameter optimization strategy, we first optimize the vector size. We vary the vector size of the hidden and the embedding layer (2, 3, 4, 8, 16, 32, 64, 128, 256), and the number of epochs $(10, 25, 50)$. Next, we optimize the window size of the embedding which determines how many activities before and after the current activity are considered. A value 5 or 7 seems optimal, and similar to the vector size, larger values do not improve the result and only run the risk of overfitting. Training epochs are varied between 10 and 50. The other parameters were standard parameters according to [4]. We trained the embedding with $sg = 0$ for the CBOW model, a learning rate set constant with $lr = 0.025$ for both Trace2vec and Case2vec and the decay factor alpha to 0.002. The number of inference epochs is set to 50. For clustering we opted to use hierarchical clustering. As a distance metric we use cosine distance to avoid a bias when dealing with traces of different length.

As an evaluation metric, we measure the NMI. We analyze the results using the non-parametric Friedman test. The Bonferroni corrected pairwise Wilcoxon signed-rank test is used for post-hoc analysis. We further report Kendall's W effect size.

Table 1. Best clustering performance grouped by approach, configuration, and event log.

Log	Approach	Vector size	Epochs	NMI
BPIC15	Trace2vec (original)	64	40	0.080
	Trace2vec (optimized)	4	50	0.132
	Case2vec (org:resource)	8	25	**0.980**
	Case2vec (Case Type)	3	25	0.010
	Case2vec (org:resource + Case Type)	8	50	**0.983**
	Case2vec (Responsible Actor)	128	25	0.398
	Case2vec (org:resource + Responsible Actor)	4	50	0.424
BPIC19	Trace2vec	32	10	0.560
	Case2vec (org:resource)	128	50	0.657
	Case2vec (org:resource + Document Type)	16	50	0.566
	Case2vec (Document Type)	128	25	0.591
	Case2vec (org:resource + Item Category)	16	25	0.626
	Case2vec (Item Category)	256	50	**0.805**
	Case2vec (org:resource + Vendor)	128	25	0.330
	Case2vec (Vendor)	2	50	0.296

Results. As a first step, we recreated the results by De Koninck et al. using the BPIC15 event log. As depicted in Table 1, Trace2vec reaches an NMI of 0.080 and increases to 0.132 after hyperparameter optimization. Using Case2vec with the case attribute `Responsible Actor` leads to a significant performance increase up to 0.398. The event attribute `org:resource`, which refers to the executing user, shows a performance of 0.980. Combining `org:resource` with one of the case attributes `Case Type` or `Responsible Actor` decreased the performance.

For the BPIC19 event log, Trace2vec reaches a performance up to 0.560. The case attribute `Item Category` reached the highest results with 0.805. However, using the `Vendor` results in a lower NMI than the control-flow only. Also, combining attributes also does not guarantee better results. Used separately, `org:resource` results in an NMI of 0.657 and `Document Type` in an NMI of 0.591. Combining the two leads to an NMI of 0.566, which results in a lower NMI than used separately.

Detailed results regarding the vector size are depicted in Fig. 3. The analysis of the results confirmed significant differences ($\chi^2(2) = 108$, $p < .001$, $W = 1$) between the approaches with a large effect. Post-hoc tests confirmed differences ($p < .001$) between all approaches with Case2vec performing better than Trace2vec. Incorporating `org:resource` lead to a significant better performance ($p < .001$) for BPIC15. For BPIC19, we discovered consistent results across the different parameter configurations, still there are significant differences ($\chi^2(2) = 24.111$, $p < .001$, $W = .223$) between the approaches. Similar to BPIC15, post-hoc tests confirm significant ($p < .001$) differences between all approaches.

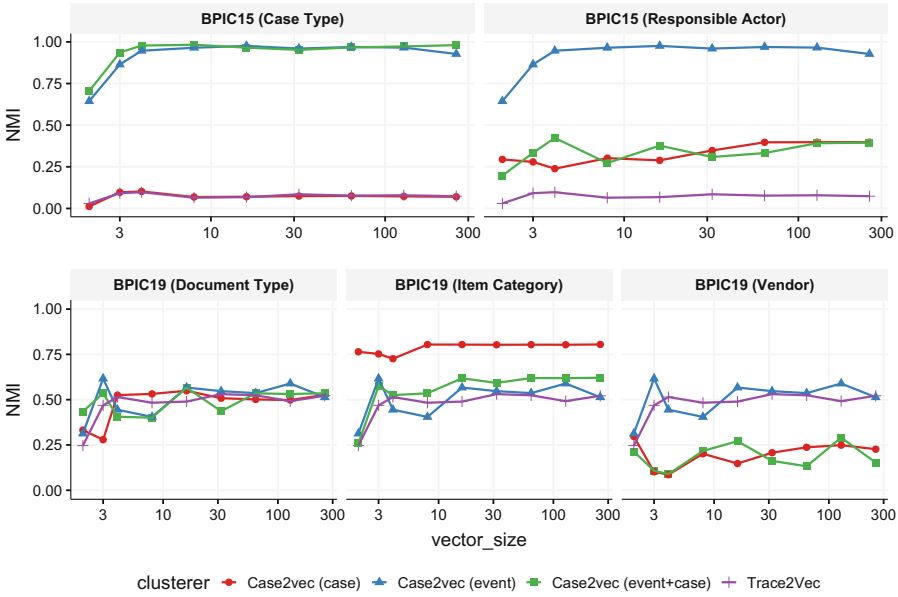

Fig. 3. Clustering results grouped by vector size, approach, configuration, and event log with 50 epochs.

4.3 Amended Real-Life Event Logs: Event Abstraction

The goal of event abstraction is to identify similar traces although activity names are slightly different. Eventually, these activity names can be adjusted to clean the event log. An example would be the activity name `PR_created` and `Create_PR`, which describes the same action with just a different name. The idea is to identify activities with similar function within the process so that a vector representation will allocate both activities close together in the vector space despite their different names.

The level of distorted activity names ranges from 0%, which is the normal case, up to 50%. The number of variations indicates the number of noise which is added to the activity name, e.g., letters or numerals. For example, if there are 2 variations and a 20% distortion level, the same random number is added to 20% of the traces, and the remaining 80% describe the unmodified variation. In case of 6 variations, besides the undistorted traces, there are traces distorted with 5 different random numbers to further increase uncertainty. Figure 4 shows results for different levels for different variations of the activity names. Case2vec with event attribute shows consistent results with deviations of $\leq$0.1 in NMI for distortion levels from 0% to 40%. A larger deviation of $\sim$0.1 can only be seen with Trace2vec for 20% distortion.

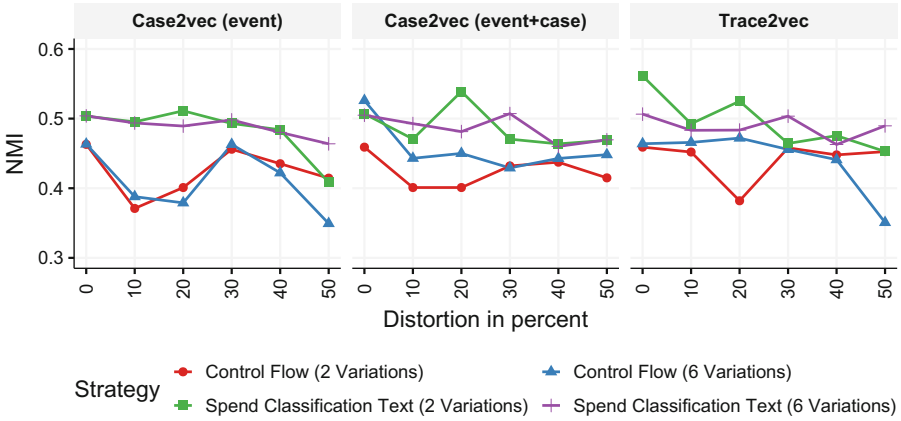

Fig. 4. Overview of the results of the event abstraction task.

Table 2. More paper process interpretability tasks

Add	Subtract	Top result
Experiment, Develop Method, Submit	Final Decision	Conduct Study
Conclude, Review, Submit	Final Decision	Develop Method
Develop Method, Submit	Final Decision	Conclude
Final Decision	Submit	Review
Experiment, Conclude	Submit	Develop Method

4.4 Synthetic Paper Process: Vector Arithmetic Interpretability

Since representation vectors are spanned within a vector space, arithmetic operations can be performed between vectors. The famous `king - man + woman = queen` example from Word2vec showed that representations can contain important semantic relationships. In this experiment, we investigate if vector arithmetic operations can also be used with process data. For testing the interpretability task, we have to come up with a certain scenario which allows a semantic interpretation. We use the paper writing process (see Fig. 2) because it is not pseudonymized and the activities can be read and understood by an analyst.

The first scenario is that the experiment was done and the paper was submitted, but the final decision has not taken place, because something is still missing that fulfills the criteria for an accepted paper. A possible composition would be to add the experiment and submission but subtract the final decision. When performing `Experiment` + `Submit` − `Final Decision` we would expect that something between `Experiment` and `Submit` is missing so that the `Final Decision` is still pending. The result of this computation returns `Evaluate` as the top result. The second top result is `Conduct Study` and the third top result

is `Review`. Table 2 shows more example interpretability tasks. The result column shows the top result.

5 Discussion

In this section, we elaborate on the results from the experiment section and follow the order presented there.

5.1 Trace Clustering

BPIC15. Our experiments showed that hyperparameter optimization increases the performance of the original approach, but did not lead to useful results. However, it is not known if the separation by municipality solely based on the control-flow is possible. The process of a building permit application may be presumably highly standardized and, therefore, not a useful criterion for separating by municipality.

The results of our experiments show that the user of an activity is an attribute that is able to separate the cases into the five municipalities. This may be an obvious observation because persons may only work for a specific municipality. However, the event log also contains several persons that work across multiple municipalities. Case2vec, which includes the control-flow and the attributes, is able to find case representations for clustering that discriminate between the municipalities.

BPIC19. For the BPIC19 experiments, we found that Trace2vec performed significantly better compared to the BPIC15 event log. The best result was achieved by incorporating the `Item Category` case attribute, which seems to be strongly related to the `Item Type`. Interestingly, not all attributes improve the control-flow performance. For instance, the case attribute `Vendor` decreased the performance down to 0.290. This could be explained by the fact that a vendor is not a good separating attribute when categorizing according to an item type a company purchases. This would be the case if the company acquires most of its items from the same vendor regardless of the category of the item. Hence, even if the control-flow is able to separate by item type to some extent, an attribute, which is identical for most items, like a vendor, can obfuscate the results. This means that we cannot arbitrarily add more attributes for better results.

Case2vec seems to be sensitive to the selection of the attributes. Even though we showed that those methods provide good results after hyperparameter tuning, applying them to real-life event logs can be difficult because the quality of the result can usually not be determined since the ground truth is unknown. Attributes that contain random values or do not contribute to the desired clustering result lead to a significant drop in performance. However, when selecting appropriate attributes, Case2vec can outperform Trace2vec significantly. Still, finding good attributes can be difficult without prior knowledge.

5.2 Event Abstraction

For event abstraction, we ran several experiments with different amounts of traces including random numbers. We also changed the amount of different random numbers. Every attribute including a different random number will increase the vocabulary size. Still, Case2vec was able to identify and group traces according to their function despite them being amended. An even more interesting application than finding functionally similar traces with different names would be finding functionally similar traces with different performance metrics like cost or time. An analyst could study why these traces are similar in their role but differ in cost or time.

5.3 Interpretability Task

The example computation `Experiment` + `Submit` − `Final Decision` returns `Evaluate` as the top result and `Conduct Study` as the second top result. Both are sensible choices when we assume that `Evaluate` has already taken place and both are performed before `Submit`. The third top result is `Review`, which takes place directly after `Submit` and also shows a sensible reason assuming `Evaluate` and `Conduct Study` have been taken place and therefore cannot be the reason the submission is still blocked. Further results shown in Table 2 can be interpreted with similar reasoning.

However, in real-life event logs the interpretation of activities is not as clear because often they do not have interpretable names and even if, these names do not necessarily relate to a role in the process its name might suggest. Additionally, the developers of the Word2vec framework remark that vector arithmetic is not guaranteed to always produce sensible results. It is still interesting to see that on a small and well-defined event log the vector representation can deliver these results.

6 Conclusion

In this paper, we presented Case2vec, a representation learning technique based on a vector space model. It is trained using a neural network in an unsupervised fashion by using the sequence of activities including event attributes. It does not rely on any prior knowledge about the process and is able to learn robust and compact representations automatically.

The results of the evaluation in a trace clustering task showed that Case2vec is able to learn a good representation given useful control-flow or case attributes. When selecting appropriate attributes Case2vec can outperform Trace2vec significantly as shown in our real-life evaluation. The experiments on the additional tasks like event abstraction or arithmetic operations in the constructed vector space support that the learned representation is able to capture semantic characteristics of the process. However, Trace2vec and Case2vec seem to be sensitive to the selection of the attributes and finding good attributes can be difficult without prior knowledge. Feature selection methods from machine learning may help

to identify attributes with a high information value, helping analysts to select useful attributes. Another limitation is that Case2vec only supports categorical attributes. Numerical values could be incorporated by grouping them into bins beforehand.

In conclusion, the internal representation of Case2vec is highly useful for trace clustering, finding functionally similar traces or executing vector space arithmetic operations for interpretability tasks.

Acknowledgment. This work is funded by the German Federal Ministry of Education and Research (BMBF) research project KI.RPA [01IS18022D].

References

1. Bose, R.P.J.C., van der Aalst, W.M.P.: Context aware trace clustering: towards improving process mining results. In: International Conference on Data Mining (SIAM) (2009)
2. Bui, H.-N., Vu, T.-S., Nguyen, H.-H., Nguyen, T.-T., Ha, Q.-T.: Exploiting CBOW and LSTM models to generate trace representation for process mining. In: Sitek, P., Pietranik, M., Krótkiewicz, M., Srinilta, C. (eds.) ACIIDS 2020. CCIS, vol. 1178, pp. 35–46. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-3380-8_4
3. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) BPM 2019. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
4. 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
5. van Dongen, B.: BPI Challenge 2015. https://doi.org/10.4121/uuid:31a308ef-c844-48da-948c-305d167a0ec1
6. van Dongen, B.: BPI Challenge 2019. https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1
7. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: International Conference on Machine Learning (2014)
8. Mikolov, T., Sutskever, I., Chen, K., Corrado, G., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems (2013)
9. Nolle, T., Luettgen, S., Seeliger, A., Mühlhäuser, M.: BINet: multi-perspective business process anomaly classification. Inf. Syst. (2019)
10. 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
11. Song, M., Yang, H., Siadat, S.H., Pechenizkiy, M.: A comparative study of dimensionality reduction techniques to enhance trace clustering performances. Expert Syst. Appl. **40**(9), 3722–3737 (2013)

12. Song, M., Günther, C.W., van der Aalst, W.M.P.: Trace clustering in process mining. In: Ardagna, D., Mecella, M., Yang, J. (eds.) BPM 2008. LNBIP, vol. 17, pp. 109–120. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-00328-8_11
13. Tavares, G.M., Barbon, S.: Analysis of language inspired trace representation for anomaly detection. In: Bellatreche, L., et al. (eds.) TPDL/ADBIS -2020. CCIS, vol. 1260, pp. 296–308. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-55814-7_25
14. Tax, N., Sidorova, N., Haakma, R., van der Aalst, W.M.P.: Event abstraction for process mining using supervised learning techniques. In: Bi, Y., Kapoor, S., Bhatia, R. (eds.) IntelliSys 2016. LNNS, vol. 15, pp. 251–269. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-56994-9_18