Conformance checking using activity and trace embeddings

Jari Peeperkorn[1] (✉), Seppe vanden Broucke[1,2], and Jochen De Weerdt[1]

[1] Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Leuven, Belgium
[2] Department of Business Informatics and Operations Management, Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium
{jari.peeperkorn, seppe.vandenbroucke, jochen.deweerdt}@kuleuven.be

Abstract. Conformance checking describes process mining techniques used to compare an event log and a corresponding process model. In this paper, we propose an entirely new approach to conformance checking based on neural network-based embeddings. These embeddings are vector representations of every activity/task present in the model and log, obtained via act2vec, a Word2vec based model. Our novel conformance checking approach applies the Word Mover's Distance to the activity embeddings of traces in order to measure fitness and precision. In addition, we investigate a more efficiently calculated lower bound of the former metric, i.e. the Iterative Constrained Transfers measure. An alternative method using trace2vec, a Doc2vec based model, to train and compare vector representations of the process instances themselves is also introduced. These methods are tested in different settings and compared to other conformance checking techniques, showing promising results.

Keywords: Process mining · Conformance checking · Representation learning · Word embedding

1 Introduction

Conformance checking is a set of process mining techniques capable of comparing event logs and corresponding process models. It can be used to compare the actual execution of a process (log) to the should-be execution (a normative model) or an automatically discovered model. Usually the degree of conformance is described over four quality dimensions: fitness, precision, simplicity and generalisation. Generally two different approaches for obtaining the fitness and precision are distinguished: log replay algorithms and trace alignment algorithms [1]. In this work, we propose an entirely new perspective on conformance checking, moving away from classical approaches relying on replay or alignments, but instead performing a fully data driven conformance analysis and subsequent global conformance measure development. Currently, our novel conformance checking technique relies on generating an event log of a model and comparing this with the actual event log using different representation learning techniques inspired

by work in Natural Language Processing (NLP). As such, the presented fitness and precision metrics are computed as log-to-log metrics.

The remainder of this paper is structured as follows. First, we introduce the notions of activity and trace embeddings in Section 2. In Section 3, the embedding-based conformance checking technique is outlined, before we perform several experimental assessments in Section 4. The paper is concluded with a section discussing related work (Section 5) and our conclusions in Section 6.

2 Activity and Trace Embeddings

Representation or feature learning describes a set of techniques that are capable of extracting (useful) representations of objects for different types of input. This can help machine learning models perform better, using an input more suited or by reducing the dimensions of the input. The techniques proposed in this paper are based on Word2vec [20,21] and Doc2vec [17], two widely used and popular representation learning techniques from NLP. The Word2vec algorithm uses a two layer neural network and a large corpus of words in order to train a vector space in which each different word gets a specific vector value called a representation or embedding. The network tries to predict a certain word in a text by using the window of surrounding words (continuous bag-of-words or CBOW) or tries to predict the surrounding window of a certain word (skip-gram). The input and output are one-hot encoded vectors of the words (with the vocabulary size as dimension). The weight used for the sum in the hidden layer (and the weight used for the output layer) is a matrix with each word's embedding stored in its columns (rows). By updating the weights while training to predict the words better, the embeddings get more optimal/meaningful vector values in such a way that similar words (words used in a similar context) should get similar vector values. In Doc2vec each specific document (or sentence) containing the word also gets a vector representation that is used as input in the neural network. By training the network, these document embeddings are updated as well. During training the representation of the document (sentence) is determined by the information from its content. These representations can then be used to compare different documents (sentences) for e.g. classification purposes.

In process mining the use of representation learning applied to activities, process instances (traces), logs and models has been introduced by [9]. In this work the authors propose among others the algorithms *act2vec* and *trace2vec*. Act2vec works similarly to Word2vec using activity labels as words. During training, activities are predicted based on the activities occurring before and after it within a process instance. In this way (meaningful) vector representations for each type of activity are learned. Trace2vec works similarly to Doc2vec using activity labels as words and process instances as sentences. Therefore every trace ID gets an embedding, next to the activity embeddings. These trace embeddings can then be used for e.g. trace clustering.

3 Conformance Checking Techniques

The basic structure of the new techniques we introduce is illustrated in Figure 1. The required input is, as usual in conformance checking, a (real) log and a model. The first step is playing out a so called "model log" from the model. The model can therefore adopt any usual structure or notation for which execution semantics can be defined (Petri net, Process trees, BPMN etc.). Using both of these logs (real and model), embeddings for the activities (and possibly for each unique trace) can be trained, as described above. Next a decision is to be made on a (dis)similarity function that measures the difference between two traces. Depending on whether one is using activity embeddings or trace embeddings different functions exist, which allows one to obtain a dissimilarity matrix. This matrix gives the function value of each trace from the real log (columns) with each trace from the model log (rows). We can now take the minimum of each column, meaning we find for each trace in the real log its best matching trace in the model log (allowing equal matches). If we take the average of these minima it gives us a measurement for the fitness of the model regarding the real log. The same can be done with finding the minimum in each row, i.e. finding for each trace in the model log its best matching trace in the real log. Again averaging these minima presents us with a measurement for the precision.

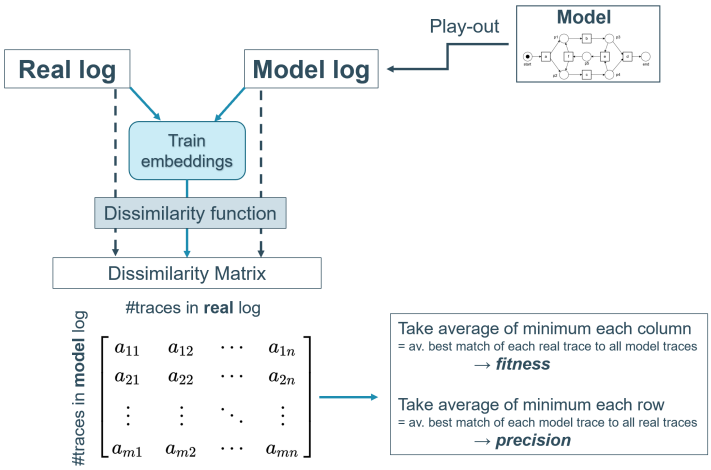

Fig. 1. The general structure of the proposed techniques.

The exact implementation of the algorithm can be decided upon by changing how embeddings are derived, changing the dissimilarity function or even by changing the way the model plays out the log. In this work it was opted to use the act2vec and trace2vec settings described above, in a Continuous Bag of Words (CBOW) and distributed bag of words (PV-DBOW) way respectively. For the time being, the window around the activity being trained is taken to be 3 and bi-directional, and the dimension of the embeddings is set to 16. Obviously, all

of these settings could easily be changed if deemed appropriate. In this work, we develop conformance metrics by implementing two distance functions at the level of activity embeddings: the Words Mover's Distance and the Iterative Constrained Transfers method. A third algorithm, relying on trace2vec, applies the cosine distance between two trace embeddings as the dissimilarity function. In the current research we have built implementations on top of the Gensim-library for unsupervised semantic modelling from plain text [27].

3.1 Words Mover's Distance

A first dissimilarity function put forward is the Words Mover's Distance (or WMD) [16], which is a commonly used function in NLP, for instance for sentence similarity. It compares two sentences (of word embeddings) similarly to the Earth Mover's Distance (or EMD) [29]. The EMD is a distance measure between two distributions in a certain region. In plain words it describes how much work has to be done to go from one vector of distributions to another. Each can be looked at as a certain configuration of "piles of earth", with each element of the vector representing a certain amount of earth in a certain location. The effort to move earth from one location to another is determined by the amount of earth transported and the distance between the locations. The optimization to go from one configuration to another can be seen as a linear program minimizing the transportation cost while satisfying two constraints. The first (outflow) constraint is that from each pile (distribution) in the first configuration there cannot be more weight transported than present. The second (inflow) constraint requires not more weight being transported than allowed to each pile (distribution) in the second configuration. In the WMD the locations of the piles are the different words and the distribution/weight of each word is its normalized word count. The distance between the words is calculated by using the Euclidean distance between its embeddings. Applied to the act2vec environment, this function allows us to calculate the effort to go from one trace to another using the embeddings of the activities within. The method is described in Algorithm 1.

The main attraction of EMD-based approaches is their high accuracy in different applications like e.g. classification [4]. The downside is the inefficiency, as according to the authors the average time complexity of solving the WMD optimization problem scales $O(p^3 \log p)$, with p the size of the vocabulary (amount of unique words/activities). Therefore an alternative method, relaxing one of the constraints is also presented here. It has to be noted that the WMD does not take the order of the activities into account, but only the count of an activity within a trace. Order can have an influence during the training of the embeddings, but is not explicitly taken into account in the WMD (nor in its approximations).

3.2 Iterative Constrained Transfers

A faster alternative for WMD is Relaxed Word Mover's Distance (RWMD), which drops one of the two constraints of the WMD completely. This allows for

Algorithm 1: Method using WMD [16].

Result: Calculates the distance between two traces t_1 and t_2.
n = vocabulary size
d_i = normalized count of activity i within its trace
$c(i, j)$ = Euclidean distance between embeddings word i and word j.
Function $F_{WMD}(t_1, t_2, c)$

$$\text{distance} = \min_{\mathbf{T} \geq 0} \sum_{i,j=1}^{n} T_{ij} c(i, j)$$

subject to:

$$\sum_{j=1}^{n} T_{ij} = d_i \ \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} T_{ij} = d_j \ \forall j = 1, \ldots, n$$

return *distance*
end

more transportation from or to certain words than actually correct (but with lower distance). The RWMD therefore gives a lower boundary to the WMD [16]. The Iterative Constrained Transfers (ICT) proposes the addition of an edge capacity constraint when relaxing one of the two constraints [4]. This additional constraint entails that when moving weight between two words along an edge, the total transportation always has to be smaller than the minimum of the distribution on both sides of the edge. In other words it cannot be bigger than the total distribution of the outgoing word (from the first sentence) and of the ingoing word (from the second sentence). The ICT replaces the second (inflow) constraint of the WMD by this new edge capacity constraint. The new problem can be solved optimally by considering the first sentence word by word. Each time sorting the edges leaving from it to the different words in the second sentence, in increasing order of transportation cost. Then iteratively transferring weights from the word in the first sentence to the words in the second sentence, under the edge constraints, until the outflow constraint is met. The ICT can be approximated by limiting the number of iterations than can be performed, called the Approximate Computation of ICT (or ACT). In this work it was opted to use this constraint with the maximum number of iterations being 3. Only if the outgoing trace has certain activities which occur significantly more than others, this number should be taken higher. The inflow constraint of the WMD is not necessarily met anymore, therefore the ICT (or ACT) provides a lower bound to the WMD, but due to the extra edge constraint more tight than the RWMD. The algorithm used to calculate the ICT between two traces (with activity embeddings) can be found in Algorithm 2.

3.3 Trace embeddings

The third approach uses trace embeddings. In this work it was opted to train one embedding for each unique trace. This means that traces with the same activity

Algorithm 2: Method using ICT, referred to as ACT in [4].

Result: Calculates the distance between two traces p and q.
k = number of edges considered per activity (in this work 3)
$c(i, j)$ = Euclidean distance between embeddings word i and word j
h_p, h_q = amount of different activities in trace p and q
p_i = normalized weights of each activity in trace p with $i = 1, \ldots, h_p$
q_i = normalized weights of each activity in trace q with $i = 1, \ldots, h_q$
Function $F_{ICT}(p, q, c, k)$
  Initialize transportation cost $t = 0$
  for $i = 1 \ldots h_p$ **do**
    Find k smallest: $s = \arg\min_k(c(i, [1, \ldots, h_q]))$
    Initialize l = 1
    while $l < k$ **do**
      Edge constraint: $r = \min(p_i, q_{s(l)})$
      Transport weight: $p_i = p_i - r$
      Update cost: $t = t + r \cdot c(i, j)$
      $l = l + 1$
    end
    Solve for possible excess weight
    if $p_i \neq 0$ **then**
      Move rest to $q_{s(k)}$:
      $t = t + p_i \cdot c(i, s(k))$
    end
  end
  return t
end

sequence, have the same embedding. Once these embeddings are obtained one could use any distance metric in vector spaces deemed appropriate. In this work it was opted to use the cosine similarity. Using the cosine similarity means that the output of this algorithm will always be a number between 0 and 1, where 0 means a perfect value in both the fitness and the precision calculation. The algorithm to calculate the cosine distance between two trace embeddings is displayed in Algorithm 3.

Algorithm 3: Method using trace2vec.

Result: Calculates the distance between two traces embeddings p and q.
n = dimensions trace embeddings
Function $F_{t2v}(p, q)$
  $$\cos = \frac{\sum_{i=1}^{n} p_i q_i}{\sqrt{\sum_{i=1}^{n} p_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$
  return *cos*
end

4 Experimental Evaluation

The goal of the empirical evaluation in this work is twofold. On the one hand, we want to investigate the computational complexity of the proposed techniques. On the other hand, we want to provide evidence that the measures are indeed capable to reveal when models and logs become more discrepant.

4.1 Experimental setup

In order to test the methods proposed, different experiments are performed[3]. A first experiment focuses solely on the scalability of our proposed methods, varying log size and dictionary size (number of activities). Next, in order to assess the conceptual appropriateness, another set of experiments was conducted. For these experiments different process trees, depicting different types of processes, are generated randomly using the implementation of the Python library PM4Py [6,15]. The different settings used for each tree can be found in Table 1. These trees vary in size (depicted by the minimum, mode and maximum number of visible activities) and the probabilities of adding sequence, parallel, choice and loop operators to the tree. Other parameters are left default.

Table 1. The different settings used to generate the process trees used in the experiments.

	Size (min - mode - max)	Sequence	Parallel	Choice	Loop
Tree 1	5-10-15	0.75	0.25	0	0
Tree 2	5-10-15	0.75	0	0.25	0
Tree 3	5-10-15	0.5	0.25	0.25	0
Tree 4	5-10-15	0.25	0.25	0.25	0.25
Tree 5	10-20-30	0.75	0.25	0	0
Tree 6	10-20-30	0.75	0	0.25	0
Tree 7	10-20-30	0.5	0.25	0.25	0
Tree 8	10-20-30	0.25	0.25	0.25	0.25
Tree 9	15-30-45	0.75	0.25	0	0
Tree 10	15-30-45	0.75	0	0.25	0
Tree 11	15-30-45	0.5	0.25	0.25	0
Tree 12	15-30-45	0.25	0.25	0.25	0.25

First, the ability of the methods to measure differences between two logs is assessed in a noise experiment depicted in Figure 2. For each process tree, a ground truth log is played out. Then different levels of noise are introduced to this ground truth log to obtain a noisy log. The different methods are tested

[3] The implementations of the algorithm, the tests and most of the synthetic data used can be found on `https://github.com/jaripeeperkorn/Conformance-checking-using-activity-and-trace-embeddings`.

on these two logs, checking whether more noise actually equals less optimal values. We define three types of noise: replacing a random activity with another random activity label (form the vocabulary), swapping two random activities and removing a random activity. In a first noise experiment each of these noise types are added to only one activity (or two in the case of swapping). But the percentage of traces on which we apply each of these noise functions (one after the other) is varied from 10 to 50%. A second small noise experiment sets the percentage of traces on which we apply noise to 40%, but then varies the amount of activities within these traces we apply noise on. For this we are only using the noise function that randomly replaces activities, omitting the other two.

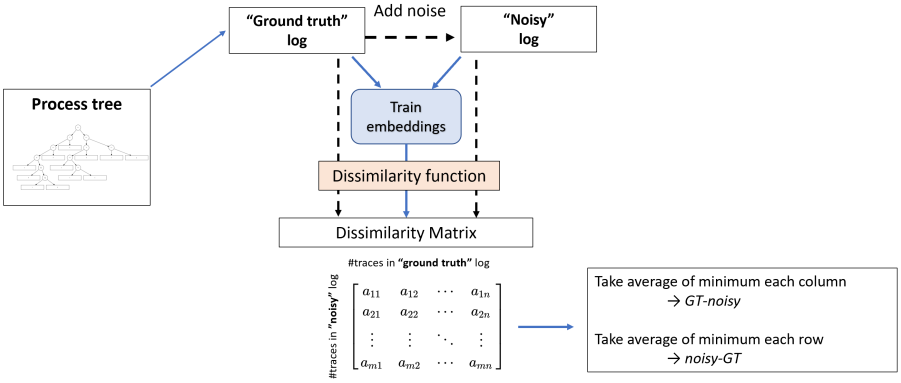

Fig. 2. The noise experiment.

Then, in a second experiment, the conceptual appropriateness is further investigated, again relying on synthetic logs. However, we now include different process discovery algorithms to obtain models with different fitness/precision and compare with well-established conformance metrics. The experiment is shown in Figure 3. Instead of adding noise, we now discover a model from the ground truth logs using different discovery techniques. Once discovered models are obtained, the different proposed methods can be applied by first playing out the discovered model. Regarding the process discovery algorithm selection, the goal was to obtain a varied set of discovered models in terms of fitness and precision. Moreover, we wanted to restrict the number of discovery techniques in order to prevent blowing up the analysis. Therefore, we opted for the following algorithms: Alpha miner [31], Inductive Miner infrequent (IMi) with the noise parameter once set to 0 and once to 1 [19] and the the ILP miner [33]. For the ILP miner settings the alpha parameter was used with the zero value and concurrency ratio left 0. We have used the implementation within the ProM framework [11] for each of the discovery techniques. The conformance checking techniques selected to compare are the behavioral negative event recall [14], alignment based fitness and precision [2] and the ETC precision [22]. All of

these conformance checking algorithms were used as they were implemented in the CoBeFra framework [7].

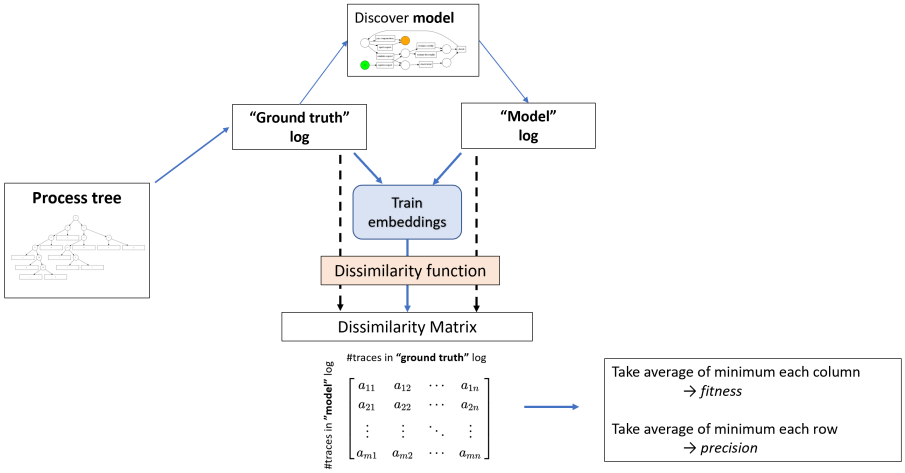

Fig. 3. The discovery experiment.

4.2 Results and discussion

Scalability The experiment each time compares two randomly generated logs, varying in size. The average trace length is left to 20, but the vocabulary size (amount of different activities) is also altered. The results of this experiment, as performed on an Intel(R) Core(TM) i7-9850h CPU @ 2.60ghz, can be found in Table 2 (performed three times, taking the average time). What is important to note is that the methods are, for now written in python. The WMD (EMD) implementation uses however pyemd [24,25], a wrapper that allows the use of numpy (C efficiency). The (own) ICT implementation used in this work is not C optimized, and could therefore still be enhanced significantly. From the results it can be seen that the run times depend on the log size significantly in all three methods. The dictionary size has a big influence on the WMD, smaller influence in the ICT and no influence on the run time of the trace2vec based method. If optimized the ICT could definitely be a computationally less demanding alternative to the WMD. The trace2vec based method is the fastest, as it only has to calculate the cosine distance of the trace embeddings. Training the embeddings does not take a long time.

Noise experiment In a first noise experiment the percentage of traces on which we apply each of the three noise functions is varied between 10 and 50%.

Table 2. Table showing the run times of the different variations of the algorithm.

Log Size	Dictionary size	wmd	ict	t2v
100	10	1s	1s	1s
	20	2s	2s	1s
	30	3s	2s	1s
500	10	20s	26s	12s
	20	46s	39s	12s
	30	1m15s	45s	12s
1000	10	1m14s	1m43s	42s
	20	2m57s	2m30s	43s
	30	4m50s	2m51s	43s
5000	10	30m38s	37m2s	15m20s
	20	1h10m4s	55m23s	15m24s
	30	1h57m12s	1h3m44s	15m20s

Again: we apply it in each of these traces only once. For each setting in Table 1, 5 different process trees are generated and the results over these 5 trees are averaged. The generated ground truth consists each time of 1000 traces. Because this number is usually very high as compared to the amount of different variants of traces, the noisy log often still contains at least once each (original) variant. This means that the ground truth - noisy value as shown in Figure 2 is almost always 0. The values of the noisy - ground truth values of this noise experiment can be found in Table 3. We can see that for each setting and each method, adding noise adds to the distance. Note that you should not directly compare the activity embedding based methods and the trace2vec model, as they have a different scale. Figure 4 shows the average over all the different process tree generations of each noise level.

As mentioned earlier another small noise experiment was performed as well in order to show how the methods handle different levels of noise within the noisy traces. In this setting we are only using the noise of randomly replacing and activity. The process trees are generated using the 10^{th} setting described in Table 1. The noise is each time applied on 40 percent of the traces in the log but on different amount of activities (1-15). The results shown in Table 4 and Figure 5 show that for each of the three methods an increase in noise corresponds to a higher distance.

Discovery experiment For the time being the generated logs for both the original tree and the discovered models are generated randomly and are each time limited to 1000 traces. This also means that the discovery algorithms use a log of (only) 1000 traces. The embeddings are being trained each time again and thus are not used over the multiple discovered models (nor over the three different techniques). In the real life situation of comparing multiple models to select the optimal one, it might be beneficial to train embeddings only once for all of the models together. The discovery experiment was performed for Trees

Table 3. Results of the noise experiment using three different types of noise, each on one (two for swapping) activity in different amounts of traces.

	Tree 1			Tree 2			Tree 3			Tree 4		
Noise	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v
10%	0.135	0.126	0.044	0.101	0.095	0.023	0.150	0.142	0.030	0.155	0.148	0.038
20%	0.244	0.227	0.123	0.192	0.179	0.089	0.282	0.264	0.090	0.282	0.268	0.091
30%	0.349	0.327	0.194	0.276	0.260	0.143	0.387	0.365	0.142	0.397	0.378	0.147
40%	0.439	0.4121	0.249	0.349	0.328	0.190	0.492	0.465	0.194	0.499	0.472	0.199
50%	0.517	0.482	0.290	0.417	0.391	0.228	0.570	0.535	0.232	0.579	0.546	0.238
	Tree 5			Tree 6			Tree 7			Tree 8		
Noise	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v
10%	0.151	0.128	0.027	0.083	0.074	0.010	0.119	0.110	0.014	0.121	0.106	0.016
20%	0.285	0.247	0.090	0.162	0.145	0.041	0.228	0.210	0.036	0.225	0.200	0.051
30%	0.407	0.344	0.149	0.237	0.212	0.067	0.321	0.295	0.067	0.314	0.282	0.088
40%	0.506	0.435	0.197	0.305	0.273	0.096	0.410	0.377	0.098	0.390	0.348	0.115
50%	0.594	0.513	0.224	0.371	0.335	0.117	0.484	0.445	0.118	0.476	0.421	0.144
	Tree 9			Tree 10			Tree 11			Tree 12		
Noise	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v	wmd	ict	t2v
10%	0.089	0.080	0.020	0.075	0.062	0.006	0.122	0.111	0.007	0.093	0.084	0.005
20%	0.173	0.157	0.074	0.147	0.121	0.021	0.231	0.213	0.018	0.182	0.164	0.013
30%	0.250	0.228	0.130	0.215	0.179	0.046	0.327	0.301	0.031	0.257	0.231	0.023
40%	0.322	0.293	0.174	0.281	0.232	0.067	0.420	0.386	0.048	0.331	0.300	0.038
50%	0.390	0.354	0.240	0.341	0.285	0.096	0.506	0.467	0.064	0.400	0.361	0.054

Table 4. Results of the noise experiment, varying only the amount of activities affected.

Amount of noise	**1**	**3**	**5**	**7**	**9**	**11**	**13**	**15**
wmd	0.134	0.326	0.451	0.542	0.611	0.659	0.692	0.702
ict	0.090	0.238	0.344	0.421	0.491	0.538	0.563	0.570
t2v	0.015	0.079	0.118	0.135	0.146	0.151	0.156	0.157

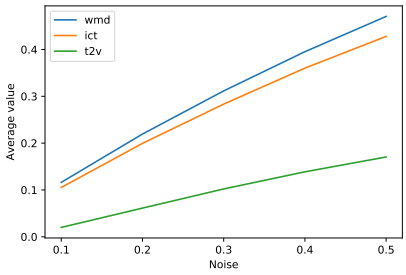

Fig. 4. The average of the first noise experiment.

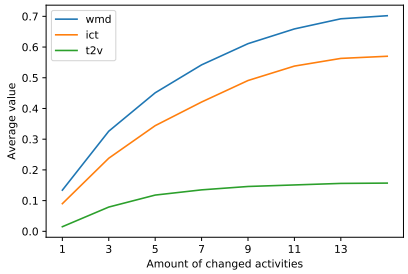

Fig. 5. The second noise experiment.

5-12 from Table 1. The results for each of the fitness and precision measurements can be found in Table 5. Beware that for the newly introduced WMD, ICT and t2v method a more optimal value corresponds to a value closer to 0 but for the other algorithms this corresponds to a value closer to 1. For the Tree setting 8 and 12 the alpha discovery algorithm produced a Petri Net not readable by the log replay algorithm used here due to its unconventional structure. Because the corresponding Petri Nets did not correspond to realistic (useful) models, they were omitted as a whole. The alignment based fitness and precision did also not produce any results (within reasonable time) when considering the net discovered by the infrequent inductive miner with noise parameter set to 0 in tree 12.

From the results in Table 5 it can be seen that when the alignment based fitness and ETC precision agree on perfect fitness or precision the WMD, ICT and t2v usually do as well. For the rest these results are harder to interpret as the methods from the literature do not always agree as well. It can be seen however that when alignment based fitness or ECT precision seem to pick one of the discovered models as having a significantly better (or worse) fitness or precision, the proposed measures seem to agree. Nevertheless, more rigorous testing should be put in place to get more conclusive insights into the algorithms proposed in this work.

5 Related Work

The first literature trying to quantify a relation between process models an event logs can be found in [8]. In later years multiple conformance checking techniques have been proposed in the field of Process Mining. Most of them consider log-model conformance techniques. For fitness some of the first noteworthy algorithms are proper completion and token based sequence replay [28]. Other approaches are e.g. behavioral recall [14], which uses a percentage of correctly classified positive events, or behavioral profile based metrics [32] which is based on different constraints a process model can impose on a log. Most of the recent research has been following the (average) alignment based trace fitness approach introduced in [2]. One of the first proposed precision measurements is advanced behavioral appropriateness [28]. Another approach, called behavioral specificity [14], replays the sequences and takes the percentage of correctly classified negative events. A similar approach using the amount of "false positives" (behavior allowed by model, but labeled a negative event based on the log) was defined by [10]. A different approach using log prefix automatons and the number of "escaping" edges, called the ETC precision, was introduced in [22]. Also alignment based precision methods have been proposed in [2], and later the one align precision and best align precision [3]. Further, in recent years fitness and precision models comparing Markovian abstractions of the models and event logs have been presented [5]. The Earth Mover's Distance has previously been used in conformance checking in the recent work of [18], although their approach requires the Petri Nets to be stochastic (and does not use embeddings). For a more

Table 5. Results of the discovery experiment. For WMD, ICT and t2v a value closer to zero means a better fitness or precision. For the alignment based fitness and precision [2], behavioral fitness [14] and the ETC precision [22] a value closer to 1 means a better value.

		Fitness					Precision				
		wmd	ict	t2v	[2]	[14]	wmd	ict	t2v	[2]	[22]
Tree 5	alpha	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ind. 0	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ind. 1	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ilp	0.000	0.000	0.000	0.898	0.898	0.000	0.000	0.000	1.000	1.000
Tree 6	alpha	3.233	2.751	0.766	0.300	0.786	3.327	2.864	0.504	0.000	1.000
	ind. 0	0.000	0.000	0.015	0.933	0.071	0.000	0.000	0.011	0.410	0.274
	ind. 1	0.000	0.000	0.018	0.849	0.727	0.000	0.000	0.011	1.000	0.998
	ilp	0.000	0.000	0.290	0.766	0.804	0.000	0.000	0.000	0.969	1.000
Tree 7	alpha	0.459	0.459	0.178	0.882	0.866	1.718	1.670	0.316	0.923	0.989
	ind. 0	0.000	0.000	0.000	0.879	0.733	0.000	0.000	0.000	0.995	0.994
	ind. 1	0.000	0.000	0.000	0.879	0.733	0.000	0.000	0.000	0.995	0.994
	ilp	0.000	0.000	0.001	0.859	0.711	0.000	0.000	0.000	0.998	0.999
Tree 8	ind. 0	0.000	0.000	0.006	0.832	0.466	0.002	0.002	0.006	0.691	0.526
	ind. 1	0.183	0.172	0.048	0.809	0.542	0.000	0.000	0.007	0.954	0.909
	ilp	0.001	0.001	0.006	0.784	0.710	0.000	0.000	0.004	0.937	0.957
Tree 9	alpha	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ind. 0	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ind. 1	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ilp	0.000	0.000	0.000	1.000	0.920	0.000	0.000	0.000	1.000	0.917
Tree 10	alpha	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.000	1.000	1.000
	ind. 0	0.000	0.000	0.000	0.976	0.950	0.000	0.000	0.000	0.998	1.000
	ind. 1	0.000	0.000	0.000	0.976	0.950	0.000	0.000	0.000	0.998	1.000
	ilp	0.000	0.000	0.002	0.948	0.946	0.000	0.000	0.000	1.000	1.000
Tree 11	alpha	0.114	0.093	0.034	0.315	0.979	0.000	0.000	0.001	0.000	0.936
	ind. 0	0.000	0.000	0.002	0.802	0.721	0.273	0.237	0.041	0.586	0.352
	ind. 1	0.971	0.866	0.265	0.296	0.342	1.542	1.136	0.221	0.987	0.901
	ilp	0.210	0.177	0.061	0.811	0.713	0.000	0.000	0.001	0.901	0.952
Tree 12	ind. 0	0.243	0.149	0.136	/	0.238	0.400	0.232	0.192	/	0.001
	ind. 1	2.440	2.158	0.537	0.640	0.241	0.459	0.459	0.237	0.664	0.646
	ilp	1.441	1.281	0.564	0.640	0.428	0.000	0.000	0.001	0.944	1.000

comprehensive overview of conformance checking techniques, interested readers are referred to [12].

6 Conclusion and Future Work

In this work a novel, fully data driven conformance analysis was introduced. The techniques are inspired by the work in Natural Language Processing (NLP) and rely on training meaningful embeddings for the different activities (and instances) of the process. In a first simple empirical assessment based on exper-

iments with artificial logs and models, we obtain promising results. More specifically, we show that the newly introduced techniques are capable to correctly assess when logs start to differ from each other. In our discovery experiment, we do show that the metrics are intrinsically capable to detect fitness and/or precision problems respectively. Nonetheless, a more elaborate design for the discovery experiment is required in order to better understand and adjust the values obtained. More concretely the WMD and ICT based method could be standardized, in order to obtain a real metric with value between 0 and 1. In the future we are also planning to move away from artificial data only and test with real life logs. From a business point of view, it should be checked to what extend similar traces but one crucial difference (error) are detected as being not conforming. The information about conformance that could be extracted beyond fitness and precision will also be investigated. So far, we have demonstrated that embedding-based conformance checking can provide an interesting alternative to classical conformance checking. Significant opportunities exist to leverage the abstraction that neural networks can obtain, to gain insights into conformance problems and report to end users. While for now, we have solely focused on measurement development,it is important to realise that our methods have strong potential to be extended towards real-life application. Nonetheless, there is still plenty of future work that should be considered. First, an investigation on the application of wrappers on the code seems worthwhile, especially when calculating the ICT, as this should ameliorate its efficiency significantly. More research on the influence of different settings when training the embeddings could improve understanding of how this novel technique could potentially be used in real life. Optimizing the window size or possibly limiting the direction of the window could potentially improve the methods. Other possible improvements can be made in the inclusion of other n-grams in stead of only using 1-grams. Due to the low vocabulary size of business processes as compared to NLP, usual arguments to not use bag of n-grams based methods do not necessarily hold. In this way the order of activities can also be taken into account. Other word embeddings used in NLP like the deep contextualized Embeddings from Language Models (ELMo) [13] or Global Vectors for Word Representation (GloVe) [26] may be interesting to investigate. Also experimenting with the influence of how the model plays out the model log, could potentially prove some valuable insights. For the moment being, it was chosen to use random playing out but one could also use e.g. a fully covering approach. Another key improvement to the algorithm is related to a more explicit incorporation of the order of activities into the calculations. This could either be done by changing the trace embeddings, training them with e.g. a recurrent neural network [23], or, at the level of activity embeddings, e.g. by adding extra dimensions to the activity embeddings (after training) corresponding to the location in the trace. Evaluating the algorithm with the recently proposed Conformance Propositions [30], would also help to evaluate the quality of the technique. The evaluation of the technique regarding these axioms should be performed in the near future.

References

1. van der Aalst, W.: Process Mining: Data Science in Action. 2nd edn. (2016)
2. van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. WIREs Data Mining and Knowledge Discovery **2**(2), 182–192 (2012)
3. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment based precision checking. In: Business Process Management Workshops. pp. 137–149 (2013)
4. Atasu, K., Mittelholzer, T.: Linear-complexity data-parallel earth mover's distance approximations. In: Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 364–373 (09–15 Jun 2019)
5. Augusto, A., Armas-Cervantes, A., Conforti, R., Dumas, M., Rosa, M.L., Reissner, D.: Measuring fitness and precision of automatically discovered process models : A principled and scalable approach (2019)
6. Berti, A., van Zelst, S.J., van der Aalst, W.: Process mining for python (PM4Py): Bridging the gap between process-and data science. In: Proceedings of the ICPM Demo Track 2019, co-located with 1st International Conference on Process Mining (ICPM 2019), Aachen, Germany, June 24-26, 2019. p. 13–16 (2019)
7. vanden Broucke, S., Weerdt, J.D., Vanthienen, J., Baesens, B.: A comprehensive benchmarking framework (cobefra) for conformance analysis between procedural process models and event logs in prom. In: Proceedings of the 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2013), pp. 254–261 (2013)
8. Cook, J.E., Wolf, A.L.: Software process validation: Quantitatively measuring the correspondence of a process to a model. ACM Trans. Softw. Eng. Methodol. **8**(2), 147–176 (Apr 1999)
9. De Koninck, P., vanden Broucke, S., De Weerdt, J.: act2vec, trace2vec, log2vec, and emodel2vec: Representation learning for business processes. In: Business Process Management. pp. 305–321 (2018)
10. De Weerdt, J., De Backer, M., Vanthienen, J., Baesens, B.: A robust f-measure for evaluating discovered process models. In: IEEE Symposium series on computational intelligence. pp. 148–155 (2011)
11. van Dongen, B.F., de Medeiros, A.K.A., Verbeek, H.M.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The prom framework: A new era in process mining tool support. In: Applications and Theory of Petri Nets 2005. pp. 444–454 (2005)
12. Dunzer, S., Stierle, M., Matzner, M., Baier, S.: Conformance checking: A state-of-the-art literature review. In: Proceedings of the 11th International Conference on Subject-Oriented Business Process Management. S-BPM ONE '19 (2019)
13. Gardner, M., Grus, J., Neumann, M., Tafjord, O., Dasigi, P., Liu, N.F., Peters, M., Schmitz, M., Zettlemoyer, L.S.: Allennlp: A deep semantic natural language processing platform (2017)
14. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. Journal of Machine Learning Research **10**, 1305–1340 (2009)
15. Jouck, T., Depaire, B.: Ptandloggenerator: a generator for artificial event data. In: Proceedings of the BPM Demo Track 2016 (BPMD 2016), 1789:23–27. Rio de Janeiro: CEUR workshop proceedings, 2016. (2016)

16. Kusner, M.J., Sun, Y., Kolkin, N.I., Weinberger, K.Q.: From word embeddings to document distances. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. p. 957–966. ICML'15 (2015)
17. Le, Q., Mikolov, T.: Distributed representations of sentences and documents. In: Proceedings of the 31st International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 32, pp. 1188–1196 (22–24 Jun 2014)
18. Leemans, S., Syring, A., Aalst, W.: Earth Movers' Stochastic Conformance Checking, pp. 127–143 (07 2019)
19. Leemans, S., Fahland, D., Aalst, van der, W.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops : BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers. pp. 66–78. Lecture Notes in Business Information Processing (2014)
20. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. In: 1st International Conference on Learning Representations, ICLR 2013 (2013)
21. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems 26, pp. 3111–3119 (2013)
22. Muñoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: Business Process Management. pp. 211–226 (2010)
23. Palangi, H., Deng, L., Shen, Y., Gao, J., He, X., Chen, J., Song, X., Ward, R.K.: Deep sentence embedding using the long short term memory network: Analysis and application to information retrieval. CoRR **abs/1502.06922** (2015)
24. Pele, O., Werman, M.: A linear time histogram metric for improved sift matching. In: Computer Vision–ECCV 2008. pp. 495–508 (October 2008)
25. Pele, O., Werman, M.: Fast and robust earth mover's distances. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 460–467 (September 2009)
26. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP). pp. 1532–1543 (2014)
27. Řehůřek, R., Sojka, P.: Software Framework for Topic Modelling with Large Corpora. In: Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks. pp. 45–50 (May 2010)
28. Rozinat, A., Aalst, van der, W.: Conformance checking of processes based on monitoring real behavior. Information Systems **33**(1), 64–95 (2008)
29. Rubner, Y., Tomasi, C., Guibas, L.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision **40**, 99–121 (01 2000)
30. Syring, A., Tax, N., Aalst, W.: Evaluating Conformance Measures in Process Mining Using Conformance Propositions, pp. 192–221 (11 2019)
31. van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Transactions on Knowledge and Data Engineering **16**(9), 1128–1142 (Sep 2004)
32. Weidlich, M., Polyvyanyy, A., Desai, N., Mendling, J., Weske, M.: Process compliance analysis based on behavioural profiles. Inf. Syst. **36**, 1009–1025 (11 2011)
33. van der Werf, J.M.E.M., van Dongen, B.F., Hurkens, C.A.J., Serebrenik, A.: Process discovery using integer linear programming. In: Applications and Theory of Petri Nets. pp. 368–387 (2008)