Machine Learning Applications in Process Mining (SE) - WS 2022/2023

Assessment and Recapitulation of "An Alignment Cost-Based Classification of Log Traces Using Machine-Learning"

Tanjila Islam (414659)

Department of Computer Science, RWTH Aachen, Aachen, Germany
tanjila.islam@rwth-aachen.de

Abstract. Identifying the discrepancies between the behaviors indicated in a log and those represented by an associated process model is known as conformance checking, which is a core part of process mining. Sequence analysis is a particularly strong area for machine learning and deep learning techniques. In order to determine whether the alignment cost of a log trace to a process model is below a given threshold, we successfully applied both a Recurrent Neural Network and a Random Forest classifier. We also propose a lower bound for the accuracy of the process model based on the classification.

Keywords: Conformance Checking · Long Short Term Memory · Random Forest.

1 Introduction

The vast majority of enterprises have process-aware information systems for logging system insights. To demonstrate how the process will be carried out, they have a process model. The main goal of process mining is to take information from the event log and use it for analysis, validation, process improvement, and process redesign. We can provide the difference between modeled behavior and actual execution behavior using conformance checking approaches [11]. One of the main criteria in conformance checking is fitness which describes how well the recorded behavior has been modeled. We use alignments to compute fitness. An alignment refers to finding a run of the process model which is the closest to a trace that appears in the event log and we count the number of differences between the run and the trace and it is called alignment cost. We can say that a trace is fitting if its optimal alignment cost is zero. But exact alignments have a high computational cost [2]. For that reason, we might need to deal with the thought of getting a high-quality conformance checking using a less complex process. Researchers were encouraged to explore model compliance from a classification-oriented approach as part of the 2016 Process Discovery Contest, which served as a catalyst for this kind of study [12].

In this context, pure data mining techniques were used to divide the event logs into two categories (compliant and deviant). Classification is a family of

techniques which consists of identifying to which category, among a given set, a new observation belongs. This is done based on a training set of data containing observations whose category membership is known. The main goal is to apply an RNN and a Random Forest (RF) classifier to the problem of categorizing traces according to their alignment costs to a reference process model. The field of recurrent networks is incredibly intriguing. The fundamental design of the RNN changed throughout time, and many outstanding contributions were made. For instance, the LSTM improves several aspects of the fundamental RNN. This branch of artificial intelligence is fascinating and full of potential. RNNs have recently captured the interest of the PM community, especially when it comes to predictive business process monitoring [10, 14, 19, 23, 27, 28]. On the other hand, Random Forest is one of the greatest and highest-performing strategies that is employed in many different sectors because of its effectiveness. It can handle categorical, continuous, and binary data. If someone wants to create a model quickly and effectively, random forest is an excellent option. In this paper, we concentrate on the effectiveness of Deep Learning (DL) and traditional Machine Learning (ML) approaches in situations involving conformance checking.

2 Background

2.1 Process Mining

Process models, often known as human-readable representations of processes, are the core concept behind process mining. Business process management makes extensive use of process models as a tool for defining, recording, and managing business processes within organizations. Each process step that occurs during the execution of a digital business process is recorded in a database. This comprises details about the process step's execution date (timestamp), process step type (activity), and business case it belongs to (case identifier). Every process mining method is built on these three essential pieces of event data, which are typically compiled into a single data structure called an event log.

Each case in a log contains events that were carried out throughout a procedure and certain properties related to that case (case attributes). Each event is described by the name of the activity and its attributes (e.g., a user who executed the event).

Table 1: Example of an event log taken from [24].

Case Id	Activity	Time
1	create purchase requisition	23.11.1994 00.00
2	create purchase requisition	23.11.1994 00.00
3	create purchase requisition	29.11.1994 00.00
4	create purchase requisition	09.12.1994 00.00
5	create purchase requisition	09.12.1994 00.00
6	create purchase requisition	09.12.1994 00.00
4963	enter goods receipt	24.02.2003 15.30
4963	enter invoice receipt	24.02.2003 15.30
4963	enter goods receipt	24.02.2003 15.30
4567	create purchase order	24.02.2003 16.49
4568	create purchase order	24.02.2003 16.53
4569	create purchase order	24.02.2003 16.56
4570	create purchase order	24.02.2003 17.00
7814	change incoming invoice	27.09.2000 10.38
7814	change incoming invoice	27.09.2000 10.38

Event logs can be used to generate a process model automatically. Event logs are examined by discovery algorithms for process patterns with the goal of creating a process model that can be understood by humans. Multiple algorithms for process discovery exist, including the Heuristics Miner [30] and the Inductive Visual Miner [18].

2.2 Conformance Checking

Conformance checking is a technique used to check process compliance by comparing event logs for a discovered process with the existing reference model (target model) of the same process. This technique is used to determine whether the target process corresponds to the actual process, highlighting deviations between the two.

$$\begin{array}{|c|c|c|c|c|c|} \hline a & b & c & x & \gg & e \\ \hline a & b & c & \gg & d & e \\ \hline \end{array} \quad \text{and} \quad \begin{array}{|c|c|c|c|c|c|} \hline a & b & c & \gg & x & e \\ \hline a & b & c & d & \gg & e \\ \hline \end{array}$$

Fig. 1: The Alignment of the log traces $l = \langle a,\text{b,c,x,e} \rangle$ and $m = \langle a,\text{b,c,d,e} \rangle$, where the top row corresponds to l and the bottom row corresponds to m, mapping moves in the log to moves in the model and vice versa [21].

In process mining, business processes are visualized using event logs. This visualization is also referred to as the discovered model. All logged process ac-

tivities, as well as their durations and sequences, are represented in this process model. Conformance checking compares the discovered model with a reference or target model, which can be visualized as a superimposition of the two models.

While it is ideal to see a high degree of similarity between the basic activities in both models, this isn't always the case. In reality, process workflows can include extra, unnecessary steps, and steps can be missed or performed in the wrong order. Conformance checking is designed to identify these cases.

Conformance checking helps ensure that all process deviations from the target process or reference model are identified, minimizing risk of audit problems or legal violations. Process deviations often require more resources than the target process or may have a negative impact on product or service quality. As a result, most process deviations lead to financial loss, so companies have a great interest in identifying unplanned process sequences and introducing appropriate preventive measures.

2.3 RNN and LSTM Networks

The design and application of neural networks made up of numerous interconnected layers of neurons that conduct non-linear data transformations which is the focus of the machine learning subfield known as "deep learning". These changes are primarily intended to train the network to "learn" the patterns and behaviors found in the data. The composition of complex functions, which occurs as the number of layers of neurons increases, should theoretically make it easier to recognize higher-level patterns in the data [17].

Recurrent Neural Networks are a generalization of feedforward neural networks that has an internal memory. An RNN is recurrent in nature as it performs the same function for every input of data while the output of the current input depends on the past one computation. RNNs can use their internal state (memory) to process sequences of inputs. But standard Recurrent Neural Networks (RNNs) suffer from short-term memory due to a vanishing gradient problem that emerges when working with longer data sequences. Therefore, now we have more advanced versions of RNNs that can preserve important information from earlier parts of the sequence and carry it forward and it is called LSTM. LSTM employs various gates to decide what information to keep or discard. Also, it adds a cell state, which is like a long-term memory of LSTM. LSTM recurrent unit is much more complex than that of RNN, which improves learning but requires more computational resources [15]. In figure 2 a basic RNN structure is given:

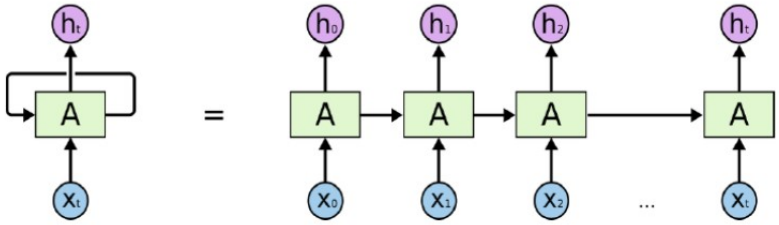


Fig. 2: Basic RNN Structure. The left is the typical RNN structure. The right part is the unfolding version where the previous information is transformed to the later time step [26].

2.4 Random Forest Classifier

An ensemble supervised machine learning technique is random forest. It is quite approachable for high dimensional data modeling because it can handle missing values and can deal with continuous, categorical and binary data. Because of its performance being equivalent to group strategies like bagging and boosting, Random Forest has a great deal of potential to become a common methodology for classifiers in the future [16].

Random forest is a supervised learning algorithm. It is also the most flexible and easy-to-use algorithm. The "forest" refers to a set of uncorrelated decision trees that are combined to lower variation and produce more precise data predictions. Random forests create decision trees on randomly selected data samples, get predictions from each tree and select the best solution by means of voting. Once the forest is trained or built to classify a new instance, it is run across all the trees grown in the forest. Each tree gives a classification for the new instance which is recorded as a vote. The votes from all trees are combined and the class for which maximum votes are counted (majority voting) is declared as the classification of the new instance. It also provides a pretty good indicator of the feature's importance.

3 Related work

The work of Nolle et al. [21], whose results are based on RNN-based alignments, is perhaps most comparable to the present paper. Although they do anomaly detection rather than log trace classification, their results are quite promising.

Sun et al. and Bose et al. looked at labeled traces and association rules mining methods that may be used to extract human-readable results from them [7, 25]. System deviation analysis has been used to study the classification of log traces. These studies often take into account two categories of processes—normal and deviant—and seek to explain why discrepancies exist and deviant processes emerge. Nguyen et al. [20] constructed trace classes from data attributes and

explored the challenge of classification using decision trees, the k-Nearest Neighbors algorithm, and neural networks. Similarly to this, Bellodi et al. [5] offered a technique for categorizing log traces using Markov Logic formulae. In this work they present an approach for the automatic discovery of knowledge-based process models expressed by means of a probabilistic logic, starting from a set of process execution traces.

Numerous studies have previously looked into the use of Long Short-Term Memory (LSTM) networks to solve the problem of forecasting the next step in a business process. Evermann et al. [14] proposed an approach to predicting the next event in a business process, and the duration of activities using recurrent neural networks with LSTM. The implementation of this approach is done using Tensorflow following that it provides RNN-specific functionality, and can be used on high-performance parallel, cluster, and GPU computing platforms [6]. Using LSTM networks, Tax et al. [27] in their research anticipate the subsequent activity of a running case and its timestamps. Learning an activity prediction function and a time prediction function is the first step in the procedure. The feature vector is produced using a single hot encoding, and it is then utilized as the input for LSTM networks. The feature vectors receive three time-based features. This has the benefit of teaching long dependencies, which is necessary to produce quality outputs. For the same goal, Pasquadibisceglie et al. looked at Convolutional Neural Networks instead of RNNs [23]. Taymouri et al. took on the issue by building a Generative Adversarial Network on top of all these methods, with encouraging outcomes [28].

One obvious distinction between previous efforts and this literature is that we have an oracle at our disposal to categorize our traces, i.e. a process model. The main difference between previous work and this paper is that a model as a base of the classification has been used in this paper. So it can be said that it is a conformance checking like classification. There had also been some process discovery contest where participants were being asked to provide a classification of the process instances instead of applying an algorithm or a process model. So, this is related to what is done in the paper.

4 Theoretical Framework

4.1 Alignment Cost and Fitness

The holy grail in process mining is a process discovery algorithm that, given an event log, produces fitting, precise, properly generalizing, and simple process models. Event logs are generally considered to be accurate representations of the behavior of a system in such a way that each event refers to an activity that was executed in the context of a case. By deriving a process model from such an event log, process discovery algorithms give insights into the underlying system. Figure 3 illustrates a sample event log and related process model:

$\langle open, read, wait, wait, close \rangle$
$\langle open, read, close \rangle$
$\langle write, wait, close \rangle$
$\langle open, wait, write, close \rangle$

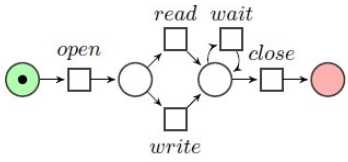


Fig. 3: A log L and an associated process model M from [6].

We compute the fitness of a process model with regard to a trace as follows:

$$fitness(L, M) = 1 - \frac{minCost(\sigma, select(\sigma, M))}{|\sigma| + min_{u\prime \in Runs(M)}|u\prime|} \quad (1)$$

where select(σ, M) returns a run $u \in$ Runs (M) such that the set of alignments of σ with M using u contains an optimal alignment, and minCost(σ, u) returns the minimum cost of aligning σ with M using a run u. When a trace has a fitness of 1, or when its optimal alignment costs 0, it is said to be fitting. The average of the fitness of a process model M with respect to each log trace of L is what we refer to as the fitness of M with respect to a log L.

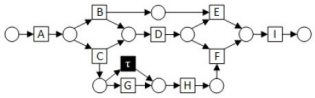
trace	*open*	*wait*	*write*	*close*
run	*open*	$\gg$	*write*	*close*

Fig. 4: An optimal alignment of the log trace $\langle o$pen, $w$ait, $w$rite, $c$lose$\rangle$. The alignment cost is 1 because there is just one non-synchronous move in it.

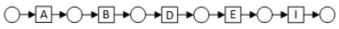
A model fits a log if all traces in the log can be replayed by the model. A fitness metric quantifies the fit of a log in a model. Several different measures exist for this quality dimension. The most recent and robust technique uses a cost-based alignment between the traces in the event log and the most optimal execution of the process model.

An alignment is a sequence of pairs that refer to an event from a trace and a transition in the model, or $\gg$ elements indicating deviations. The trace from the log is produced by projecting these pairs onto the first element, and the firing sequence in the model is produced by projecting these pairs onto the second element. The word “move” refers to each pair in an alignment. A pair is considered to be synchronous if both components of the pair have the same labeling.

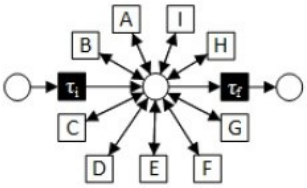
Typically, a cost function is used to compute so-called optimal alignments, such that the number of model moves, i.e. pairs $(a, \gg)$, and log moves, i.e. pairs $(\gg, t)$ in the alignment is minimized. Then, using an optimal alignment, i.e. an alignment with minimal cost, for each trace fitness is calculated by adding all penalties for log and model moves and dividing that by the worst-case costs, i.e. the costs of an alignment with only log and model moves. More details on fitness are discussed in these papers [1, 2, 9].



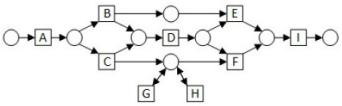
(a) The ideal model. Fitting, fairly precise, and properly generalizing.



(b) Most frequent trace. Precise, but not fitting or generalizing.



(c) The Flower model. Fitting and generalizing, but very imprecise.



(d) A model with G and H in self-loops.

Fig. 5: Fitness, Precision and Generalization for different types of process models from [29].

4.2 Training a Classification Model

Supervised learning is a subcategory of machine learning. It is distinguished by the way it trains computers to accurately classify data or predict outcomes using labeled datasets. The model modifies its weights as input data is fed into it until the model has been properly fitted, which takes place as part of the cross-validation process. Organizations can find scalable solutions to a range of real-world issues with the assistance of supervised learning.

Classification uses an algorithm to accurately assign test data into specific categories. It identifies particular entities in the dataset and makes an effort to determine how those entities should be defined or labeled. To validate a classification model, here we acknowledge a few criteria for evaluating classification models and a well-known validation method:

Definition 1 (*k*-fold cross validation). *In k-fold cross-validation, the initial data are randomly partitioned into k mutually exclusive subsets or "folds",*

$D_1, D_2, \ldots, D_k$, each of approximately equal size. Training and testing is performed k times. In iteration i, partition D_i is reserved as the test set, and the remaining partitions are collectively used to train the model. That is, in the first iteration, subsets $D_2, \ldots, D_k$ collectively serve as the training set to obtain a first model, which is tested on D_1; the second iteration is trained on subsets $D_1, D_3, \ldots, D_k$ and tested on D_2; and so on. Unlike the holdout and random subsampling methods, here each sample is used the same number of times for training and once for testing. For classification, the accuracy estimate is the overall number of correct classifications from the k iterations, divided by the total number of tuples in the initial data.

In the following, given a classification model C and a given input i, we write (y_C, i) the actual class of the input and $(\hat{y}_C, i)$ its predicted class by C.

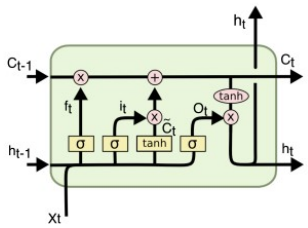
Definition 2 (Accuracy). *For a given classification model C and an input i, we say that the classification is accurate when $(y_C, i) = (\hat{y}_C, i)$. For a set of inputs S, we define $E_S, C = \{i : i \in S, (\hat{y}_C, i) = (y_C, i)\}$. The accuracy $acc_C(S)$ of the classification of S by C is given as $acc_C(S) = |\frac{E_{S,c}}{S}|$.*

Definition 3 (Cross-Entropy loss). *For a given binary classification model C and a given set of inputs S, there exists an error function called cross-entropy loss, $loss_C(S)$ defined by $loss_C(S) = \frac{1}{S} \sum_{i \in S} -\log(P((y_C, i) = (\hat{y}_C, i)))$.*

4.3 Bag of Words Encoding

Bag of words (BoW) is a Natural Language Processing (NLP) technique of text modelling. In technical terms, we can say that it is a method of feature extraction with text data. This approach is a simple and flexible way of extracting features from documents. Orders, also known as sequential information, are thought to be crucial for processing a sequence using convolutional or recurrent neural network-based encoders. Humans may still be able to decipher the meaning of a collection of words from a disorganized sentence by rearranging or reconstructing the words.

Definition 4. *For an alphabet A and a sequence $\sigma \in A^*$, a Bag-of-Words encoding maps σ to a multiset of words of A.*



$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c)$$
$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$h_t = o_t * \tanh(C_t)$$

where b_g is the bias added at gate g, W_g is the weight vector for gate g, x_t is the current input, C_{t-1} the memory of last hidden unit, and h_{t-1} the output of last hidden unit.

Fig. 6: BoW encoding in an LSTM structure (adapted from [22]).

In its simplest version, the multiset is encoded as a vector of integers X_σ, and the element at index i in X_σ gives the count of the word at index i in O_A in the sequence σ, where O_A is a vector containing exactly all the elements of A in some arbitrary order. For instance, the respective BoW encodings of the log traces in Figure 3 are $\langle 1, 1, 2, 1, 0\rangle$, $\langle 1, 1, 0, 1, 0\rangle$, $\langle 0, 0, 1, 1, 1\rangle$, and $\langle 1, 0, 1, 1, 1\rangle$ for $O_A = \langle o\text{pen}, r\text{ead}, w\text{ait}, w\text{ait}, c\text{lose}\rangle$.

One of the biggest problems with text is that it is messy and unstructured, and machine learning algorithms prefer structured, well-defined fixed-length inputs and by using the Bag-of-Words technique we can convert variable-length texts into a fixed-length vector.

5 Approach

5.1 Supervised Learning and Building Model

There are numerous methods for supervised training of classification models using sequential data. They all share the requirement to encode variable-length sequences as fixed-size vectors; the classifier then uses these vectors as training samples to create a classification model. Based on the accuracy of the model in classifying fresh inputs, its quality is subsequently evaluated using a variety of metrics and techniques. Different methods can be used to create the vectors mentioned above, such as ignoring the order of the sequences (Bag-of-Words) and hoping that understanding the frequency of each word in the sequences will be sufficient to train a classifier (such as an RF classifier) or training Deep Neural Networks that can encode the ordering of the sequences in the vectors (e.g. a LSTM network).

The sequences (represented as sequences of integers) are often sent through an embedding layer before being transmitted through the LSTM layer; the prediction is then the output of a dense layer when using LSTM networks for sequence

classification. To minimize overfitting during training, one can add dropout layers to the network that randomly reject a portion of the units. This design is unique in that the entire sequence is supplied into the network as input, and the embedding is discovered during training. This enables the creation of a representation of the sequence that incorporates its sequential features.

In case of classification in random forests, it employs an ensemble methodology to attain the outcome. The training data is fed to train various decision trees. This dataset consists of observations and features that will be selected randomly during the splitting of nodes. A rain forest system relies on various decision trees. Every decision tree consists of decision nodes, leaf nodes, and a root node. The leaf node of each tree is the final output produced by that specific decision tree. The selection of the final output follows the majority-voting system [8]. In this case, the output chosen by the majority of the decision trees becomes the final output of the rain forest system.

5.2 Classification of Log Traces

The important information for conformance checking is represented by how well a log trace fits a process model. Aligning the trace with the model is necessary for computing the fitness, which is an expensive procedure. In this part, the Alignment Cost Threshold-based Classification has been proposed which is a binary classification of log traces based on how closely they resemble a process model (ACTC). While still ensuring a lower bound for the fitness of a process model to a log, this categorization offers ways to extract pertinent information at a considerably cheaper cost than alignments. The definition and related theorem are discussed below:

Definition 5 (Alignment Cost Threshold-based Classification:). *Let M be a process model and L be a log. For a given alignment cost threshold $t_{AC} \in N$, the ACTC maps each log trace $\sigma \in L$ to one of two classes depending on its minimal alignment cost $c_{\sigma,M}$:*

- the positive class L_{pos} if $c_{\sigma,M} < t_{AC}$

- the negative class L_{neg} otherwise

By allowing us to work with arbitrary close traces rather than just fitting ones, the t_{AC} option gives us greater versatility and helps us to manage the balance between our two classes.

Theorem 1. *Given the ACTC of a log L for a model M and a cost threshold t_{AC}, the following holds:*

$$fitness(L, M) \geq \ 1 - \frac{\sum_{\sigma \in L_{pos}} \frac{minCost(\sigma, select(\sigma, M))}{\sigma + min_{u\prime \in Runs(M)}|u\prime|}}{|L|} \tag{2}$$

i.e. fitness(L, M) is bounded from below.

Proof. The fitness of a process model M with regards to a log L is defined as the average of the fitness of M with regards to each log trace of L, i.e.

$$fitness(L, M) = 1 - \frac{\sum_{\sigma \in L} \frac{minCost(\sigma, select(\sigma, M))}{|\sigma| + min_{u\prime \in Runs(M)}|u\prime|}}{|L|} \tag{3}$$

Let there be an ACTC of cost threshold t_{AC}. For every $\sigma \in L$, we have

$$fitness(\sigma, M) \geq \begin{cases} 0 & \text{if } \sigma \in L_{neg} \\ 1 - \frac{t_{AC}}{|\sigma| + min_{u\prime \in Runs(M)}|u\prime|} & \text{if } \sigma \in L_{pos} \end{cases} \tag{4}$$

since t_{AC} is the highest allowed alignment cost for a trace to be classified into L_{pos}. It follows trivially that

$$fitness(L, M) \geq \frac{\sum_{\sigma \in L_{pos}} 1 - \frac{t_{AC}}{|\sigma| + min_{u\prime \in Runs(M)}|u\prime|}}{|L|} \tag{5}$$

In the following, we write $B = \text{fitness}(\sigma, M)$ for any $\sigma \in L_{pos}$.

In contrast, a big t_{AC} will result in a greater cardinality for L_{pos} but a smaller B. Taking a modest number for t_{AC} results in a large B but a potentially lesser cardinality for L_{pos}. In the scenario where L_{pos} is created using a predictive approach, the goal of the following is to compute B using predictions. There is a chance that traces will be incorrectly identified in this situation. In the sections that follow, we demonstrate that classification models are effective enough in real-world applications to ensure a lower constraint on their fitness that is extremely close to the one shown above.

6 Experimental Works

6.1 Datasets for calculating Alignment cost

In order to provide the ACTC with the training set of alignments it needs, alignment datasets have been created that include the trace variants of each dataset (i.e., the distinct sequences in the log) and their minimal alignment costs for a number of process models. This allows us to eliminate noise from duplicate traces from our results. Using models from the work of Augusto et al. [3], the tests have been conducted on the three largest logs from the Business Process Intelligence Challenges that are currently available. The models were found using the Conforti et al. [13] preprocessing approach, followed by the Inductive Miner (IM) [18], Split Miner (SM) [4], or Heuristic Miner (SHM) [30]. The pertinent facts relating to the datasets are summarized as a table in Figure 7.

A collection of 1000 random mock traces with lengths ranging from 1 to the longest trace in the log are also generated for each log. The cost of aligning these traces with the process models is typically very expensive.

Log	Number of Trace Variants	Method of Model discovery	Average Alignment Cost	Median Alignment Cost	Dataset Name
BPIC_2012	4 366	Noise Filter + IM	2.14	2.00	A^{im}_{2012}
		Noise Filter + SM	3.02	3.00	A^{sm}_{2012}
		Noise Filter + SHM	7.60	6.00	A^{shm}_{2012}
BPIC_2017	15 930	Noise Filter + IM	14.90	13.00	A^{im}_{2017}
		Noise Filter + SM	15.03	13.00	A^{sm}_{2017}
		Noise Filter + SHM	16.31	14.00	A^{shm}_{2017}
BPIC_2019	11 973	Noise Filter + IM	24.38	6.00	A^{im}_{2019}

Fig. 7: Event log description and alignment costs taken from [6].

6.2 Methodology

An RF on BoW-encoded sequences and an LSTM network on sequences whose encoding embeds the sequential features of the activities were the two classifiers learned in this study. Figure 8 depicts an overall perspective of the training procedure.

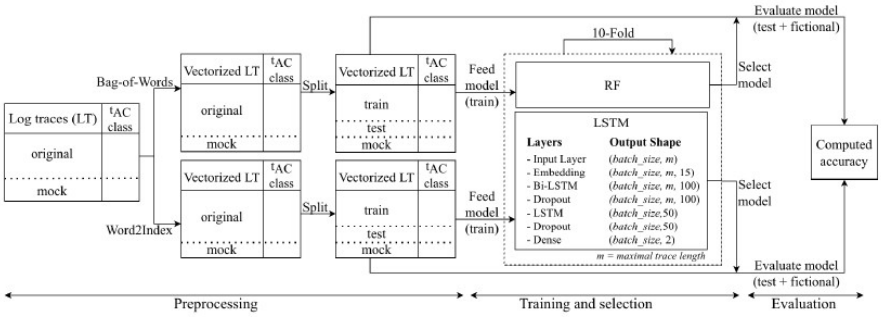


Fig. 8: The experimental setup in general adapted from the original research paper [6].

This model uses constant-length vectors of integer inputs to train an LSTM network, where each integer represents exactly one activity. Traces that fall short of the vectors' anticipated length are padded if necessary. Figure 8 illustrates the model's architecture after training. The input layer accepts a vector with elements from the set of all actions taken in the log traces, with a size of m (equal to the length of the longest trace in the log). An embedding layer is used to encode the vector into a 15-element vector. After being sent to a bi-LSTM layer to ensure that both the left and right contexts of the actions in the input traces are remembered, the resulting vector is then fed to a second, more basic LSTM layer. Overfitting is avoided by including dropout layers with a frequency rate of 0.5. The predicted classes are generated by the dense layer using the softmax

activation function, ensuring their exclusivity. With a batch size of 50 instances, we train the model across 10 epochs. After several preliminary experiments, the settings that produced the best results—the size of the embedding layer, the number of epochs, the batch size, and the stack of LSTM layers—were selected.

The RF classifier uses input vectors that reflect an ensemble of features, in this case activities, and does not take into consideration the order of the events. Thus, vectors obtained from a BoW encoding of the traces are used to train the classifier. Depending on the sequence's lowest alignment cost, the target values, or prediction classes, are either 0 (negative) or 1 (positive). First, we used a 10-fold cross-validation on the training sets to separate the dataset into a training set (67%) and a testing set (33%) in order to identify the most accurate predictive model. The classes of the sequences in the testing sets are then predicted, and the accuracy during training and testing are compared; they ought to be comparable. Finally, we feed the prediction model randomly produced traces with a high alignment cost; they must always be categorized negatively.

6.3 Findings and Analysis

Here, I will be discussing about the evaluation and analysis of the original paper [6] as well as my own observations. Firstly, I am going to present the analysis techniques explained in this literaure. Then I will showcase my findings.

Two different classifiers—one RNN and one RF—have been created during the experiment. Each pair (d,m) represents an ACTC problem, where d is one of the seven datasets in Figure 7 and m is a potential alignment cost for the model. The findings of the research [6] are summarized in Figure 9 as a table, where t_{AC} is the median of the alignment costs listed in the table that is shown in Figure 7. The table compares the running times for computing the alignments with the running times of ProM[1] and includes the accuracies and losses for the testing data.

Align-ments	Fitness	t_{AC}	% of positive	Fitness Lower Bound	RNN				Random Forest				ProM
					Acc	Loss	Predicted Fitness Lower Bound	Avg. Run-time (ms)	Acc	Loss	Predicted Fitness Lower Bound	Avg. Run-time (ms)	Avg. Run-time (ms)
A^{im}_{2012}	0.950	2	73	0.695	0.999	0.011	0.695	12.00	0.988	0.057	0.700	0.06	42.28
A^{sm}_{2012}	0.932	3	73	0.670	0.829	0.377	0.745	19.72	0.820	0.472	0.713	0.08	52.85
A^{shm}_{2012}	0.837	6	56	0.476	0.969	0.104	0.491	23.75	0.972	0.136	0.479	0.06	99.89
A^{im}_{2017}	0.874	13	53	0.463	0.984	0.047	0.473	10.01	0.979	0.056	0.467	0.03	5.12
A^{sm}_{2017}	0.819	13	52	0.415	0.985	0.049	0.420	2.70	0.985	0.053	0.421	0.03	7.72
A^{shm}_{2017}	0.794	14	52	0.400	0.981	0.055	0.410	4.05	0.984	0.055	0.405	0.03	33.23
A^{im}_{2019}	0.561	6	53	0.328	0.973	0.078	0.338	15.11	0.958	0.103	0.344	0.03	1.09

Fig. 9: Alignment Cost Thresold based Classification using RNN and Random Forest classifier, where t_{AC} is the median of the Alignment costs. The resluts have been adapted from [6].

[1] https://www.promtools.org

As shown in Figure 9, after the model has been trained, it is typically far quicker to estimate the class of a trace than to determine its precise alignment. The models must first be trained before they can produce predictions, which is an important drawback of employing predictive techniques. The time required to train a model in this experiment ranged from 3.18 seconds to 8.97 seconds for the RF classifier and from 267.57 seconds to 34837.31 seconds for the LSTM network. Both models entailed 10-fold cross-validation.

A comparison of the predictions in Figure 10 from the initial investigation [6] in the form of a table with various t_{AC} levels has been shown in order to more accurately determine the effect of t_{AC} on the outcomes. It can be seen by summing the accuracy, loss, and distribution for the testing data and for randomly produced mock data that the accuracy rapidly decreases as t_{AC} increases for the mock data, which is caused by a similarly rapid decline in the proportion of log traces categorized as negative. However, both classifiers are fairly accurate when given genuine log traces in each of the examples that were taken into consideration. We observe that the anticipated lower bound of the fitness is rather close to the one provided by the precise formula, as was the case in Figure 9. Although the process model's actual fitness with respect to the log is relatively far off at 0.837, this is still a respectable outcome.

t_{AC}	Class	%	Fitness Lower Bound	RNN			Random Forest		
				Acc	Loss	Predicted Fitness Lower Bound	Acc	Loss	Predicted Fitness Lower Bound
2	all	100	0.071	0.992	0.029	0.076	0.998	0.009	0.073
	pos	8		0.982	0.214		1.000	0.043	
	neg	92		0.992	0.013		0.998	0.006	
	mock	100	/	0.961	0.108	/	0.904	0.207	/
4	all	100	0.169	0.991	0.042	0.166	0.999	0.016	0.170
	pos	20		0.968	0.151		1.000	0.021	
	neg	80		0.997	0.016		0.998	0.015	
	mock	100	/	0.937	0.303	/	0.876	0.317	/
6	all	100	0.476	0.971	0.104	0.491	0.972	0.150	0.479
	pos	56		0.990	0.066		0.978	0.063	
	neg	44		0.944	0.156		0.962	0.268	
	mock	100	/	0.871	0.543	/	0.837	0.548	/
8	all	100	0.500	0.976	0.092	0.498	0.984	0.077	0.501
	pos	65		0.980	0.079		0.989	0.031	
	neg	35		0.970	0.116		0.974	0.161	
	mock	100	/	0.818	1.189	/	0.782	0.911	/
10	all	100	0.524	0.937	0.165	0.508	0.971	0.103	0.522
	pos	73		0.943	0.100		0.979	0.055	
	neg	27		0.921	0.336		0.949	0.233	
	mock	100	/	0.364	3.759	/	0.620	1.650	/

Fig. 10: Comparison of the A_{2012}^{shm} testing set's prediction results for various t_{AC} values taken from [6]. The chosen sublog's precise fitness value is 0.837.

In the running study, I have been conducted the experiment with the datasets described in Figure 7. I have compared the prediction results of A_{2017}^{im} testing dataset for various values of t_{AC}. The result of my analysis is very similar to

the main experimental outcomes of [6], which is shown as a form of table in Figure 11. Similar to the study, I have also outlined accuracy, loss, and mock data that was generated at randomly. It demonstrates that when the value of t_{AC} rises, the rate of accuracy and loss, both positive and negative, will be reduced. The predicted lower bound of the fitness is quite close to that provided by our precise calculation (0.473 for RNN and 0.467 for Random Forest (from Figure 9)). Therefore, based on my observations, the model is fairly accurate for both classifiers even though the process model's actual fitness with respect to the log is quite poor at 0.874.

t_{AC}	Class	%	Fitness Lower Bound	RNN			Random Forest		
				Acc	Loss	Predicted Fitness Lower Bound	Acc	Loss	Predicted Fitness Lower Bound
2	all	100		0.984	0.042		0.990	0.033	
	pos	7	0.073	0.972	0.210	0.085	0.962	0.098	0.077
	neg	93		0.985	0.029		0.993	0.028	
	mock	100	/	0.960	0.068	/	0.998	0.032	/
4	all	100		0.973	0.064		0.982	0.045	
	pos	12	0.115	0.926	0.172	0.124	0.941	0.137	0.119
	neg	88		0.980	0.049		0.988	0.032	
	mock	100	/	0.896	0.280	/	0.982	0.080	/
6	all	100		0.983	0.045		0.989	0.037	
	pos	19	0.179	0.961	0.118	0.180	0.980	0.068	0.182
	neg	81		0.989	0.028		0.992	0.030	
	mock	100	/	0.907	0.374	/	0.923	0.207	/
8	all	100		0.958	0.101		0.968	0.083	
	pos	30	0.276	0.944	0.162	0.284	0.964	0.096	0.285
	neg	70		0.964	0.074		0.970	0.078	
	mock	100	/	0.918	0.291	/	0.897	0.297	/
10	all	100		0.971	0.085		0.979	0.080	
	pos	40	0.358	0.973	0.078	0.365	0.981	0.045	0.364
	neg	60		0.970	0.090		0.977	0.103	
	mock	100	/	0.854	0.704	/	0.848	0.370	/

Fig. 11: Comparison of the A_{2017}^{im} testing set's prediction results for various t_{AC} values, as an experimental part of this research work. The exact fitness value is 0.874.

According to the research, both learning models show good accuracy and minimal losses, supporting the potential of predictive methods for the alignment problem. The traces that were identified as positive were used to calculate the anticipated lower bound of the fitness, which is extremely close to the actual lower bound. But there has been evidence of a large discrepancy between the

actual fitness and these lower bounds. This is due to the coarse-grained nature of the fitness function that was utilized, which only provides a binary score indicating whether a log trace is considered either positive or negative. Despite this flaw, it is nonetheless fairly helpful as a heuristic for selecting which of two models matches a trace more closely. A classification like this is likely to be very helpful to decision makers because the binary classification is simple to understand whereas understanding alignments typically requires more expertise.

7 Conclusion and Future Works

With the knowledge and inspiration provided by the study, models may now be created using sophisticated deep learning techniques to predict sequences and the resource pools that go along with them. This work presents some preliminary applications of machine learning and deep learning for binary oracle construction for conformance checking. In this case, an LSTM network and an RF classifier have been used to estimate with high accuracy whether the lowest alignment cost of a log trace with respect to a process model is below a threshold point. It has also been established that there is a lower bound on a process model's fitness. We may state that alignment cost-based classification has some advantages over traditional conformance checking. The main advantage is that alignments can be expensive to compute in some situations, such as when a trace is far from the process model or when a process model is complex. Additionally, it is incredibly helpful when we need a quick result. Another intriguing study may build on the work of Nolle et al. [21] to predict optimal alignments of a log trace to a process model. It might entail determining whether the precise optimal alignment cost of a trace with a process model can be predicted using a regression model.

In the future, we can go deeper into the prediction. We can anticipate the trace fitness which is not a binary classification, and we can also model a sequence neural network in order to predict the alignment with model move, log move and synchronous moves.

Bibliography

[1] Van der Aalst, W., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and performance analysis. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery **2**(2), 182–192 (2012)

[2] Adriansyah, A.: Aligning observed and modeled behavior (2014)

[3] Augusto, A., Armas-Cervantes, A., Conforti, R., Dumas, M., Rosa, M.L., Reißner, D.: and-compare: A family of scalable precision measures for automated process discovery. In: International Conference on Business Process Management. pp. 158–175. Springer (2018)

[4] Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split miner: automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems **59**(2), 251–284 (2019)

[5] Bellodi, E., Riguzzi, F., Lamma, E.: Probabilistic declarative process mining. In: International Conference on Knowledge Science, Engineering and Management. pp. 292–303. Springer (2010)

[6] Boltenhagen, M., Chetioui, B., Huber, L.: An alignment cost-based classification of log traces using machine-learning. In: International Conference on Process Mining. pp. 136–148. Springer (2021)

[7] Bose, R.J.C., van der Aalst, W.M.: Discovering signature patterns from event logs. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM). pp. 111–118. IEEE (2013)

[8] Breiman, L.: Random forests. Machine learning **45**(1), 5–32 (2001)

[9] Buijs, J.C., van Dongen, B.F., van der Aalst, W.M.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. International Journal of Cooperative Information Systems **23**(01), 1440001 (2014)

[10] Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: International Conference on Business Process Management. pp. 286–302. Springer (2019)

[11] Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance checking. Switzerland: Springer.[Google Scholar] (2018)

[12] Carmona, J., de Leoni, M., Depaire, B., Jouck, T.: Summary of the process discovery contest 2016. In: Proceedings of the Business Process Management Workshops, Springer (2016)

[13] Conforti, R., La Rosa, M., ter Hofstede, A.H.: Filtering out infrequent behavior from business process event logs. IEEE Transactions on Knowledge and Data Engineering **29**(2), 300–314 (2016)

[14] Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems **100**, 129–140 (2017)

[15] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[16] Kullarni, V., Sinha, P.: Random forest classifier: a survey and future research directions. Int. J. Adv. Comput **36**(1), 1144–1156 (2013)
[17] LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. nature **521**(7553), 436–444 (2015)
[18] Leemans, S.J., Fahland, D., Van Der Aalst, W.M.: Process and deviation exploration with inductive visual miner. BPM (demos) **1295**(8) (2014)
[19] Lin, L., Wen, L., Wang, J.: Mm-pred: A deep predictive model for multi-attribute event sequence. In: Proceedings of the 2019 SIAM international conference on data mining. pp. 118–126. SIAM (2019)
[20] Nguyen, H., Dumas, M., La Rosa, M., Maggi, F.M., Suriadi, S.: Mining business process deviance: a quest for accuracy. In: OTM Confederated International Conferences. pp. 436–445. Springer (2014)
[21] Nolle, T., Seeliger, A., Thoma, N., Mühlhäuser, M.: Deepalign: alignment-based process anomaly correction using recurrent neural networks. In: International Conference on Advanced Information Systems Engineering. pp. 319–333. Springer (2020)
[22] Olah, C.: Understanding lstm networks (2015)
[23] Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: 2019 international conference on process mining (ICPM). pp. 129–136. IEEE (2019)
[24] Selig, H.: Continuous event log extraction for process mining (2017)
[25] Sun, C., Du, J., Chen, N., Khoo, S.C., Yang, Y.: Mining explicit rules for software process evaluation. In: Proceedings of the 2013 International Conference on Software and System Process. pp. 118–125 (2013)
[26] Tai, L., Liu, M.: Deep-learning in mobile robotics-from perception to control systems: A survey on why and why not. arXiv preprint arXiv:1612.07139 **1** (2016)
[27] Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with lstm neural networks. In: International Conference on Advanced Information Systems Engineering. pp. 477–492. Springer (2017)
[28] Taymouri, F., Rosa, M.L., Erfani, S., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In: International Conference on Business Process Management. pp. 237–256. Springer (2020)
[29] Van Dongen, B., Carmona, J., Chatain, T.: Alignment-based metrics in conformance checking (summary). In: Fachgruppentreffen der GI-Fachgruppe Entwicklungsmethoden für Informationssysteme und deren Anwendung. pp. 87–90 (2016)
[30] Weijters, A., Ribeiro, J.T.S.: Flexible heuristics miner (fhm). In: 2011 IEEE symposium on computational intelligence and data mining (CIDM). pp. 310–317. IEEE (2011)