Prescriptive Business Process Monitoring for Recommending Next Best Actions

Sven Weinzierl, Sebastian Dunzer, Sandra Zilker, and Martin Matzner

Friedrich-Alexander-Universität Erlangen-Nürnberg, Fürther Straße 248, Nürnberg, Germany {sven.weinzierl, sebastian.dunzer, sandra.zilker, martin.matzner}@fau.de

Abstract. Predictive business process monitoring (PBPM) techniques predict future process behaviour based on historical event log data to improve operational business processes. Concerning the next activity prediction, recent PBPM techniques use state-of-the-art deep neural networks (DNNs) to learn predictive models for producing more accurate predictions in running process instances. Even though organisations measure process performance by key performance indicators (KPIs), the DNN's learning procedure is not directly affected by them. Therefore, the resulting next most likely activity predictions can be less beneficial in practice. Prescriptive business process monitoring (PrBPM) approaches assess predictions regarding their impact on the process performance (typically measured by KPIs) to prevent undesired process activities by raising alarms or recommending actions. However, none of these approaches recommends actual process activities as actions that are optimised according to a given KPI. We present a PrBPM technique that transforms the next most likely activities into the next best actions regarding a given KPI. Thereby, our technique uses business process simulation to ensure the control-flow conformance of the recommended actions. Based on our evaluation with two real-life event logs, we show that our technique's next best actions can outperform next activity predictions regarding the optimisation of a KPI and the distance from the actual process instances.

Keywords: Prescriptive business process monitoring, predictive business process monitoring, business process management.

1 Introduction

Predictive business process monitoring (PBPM) techniques predict future process behaviour to improve operational business processes [9]. A PBPM technique constructs predictive models from historical event log data [10] to tackle different prediction tasks like predicting next activities, process outcomes or remaining time [4]. Concerning the next activity prediction, recent PBPM techniques use

state-of-the-art deep neural networks (DNNs) to learn predictive models for producing more accurate predictions in running process instances [25]. DNNs belong to the class of deep-learning (DL) algorithms. DL is a subarea of machine learning (ML) that identifies intricate structures in high-dimensional data through multi-representation learning [8]. After learning, models can predict the next most likely activity of running process instances.

However, providing the next most likely activity does not necessarily support process stakeholders in process executions [21]. Organisations measure the performance of processes through key performance indicators (KPIs) in regard to three dimensions: time, cost and quality [23]. Recent PBPM techniques rely on state-of-the-art DNNs that can only learn predictive models from event log data. Even though an event log can include KPI information, it does not directly affect such an algorithm's learning procedure unless a KPI is the (single) learning target itself. As a consequence, the learned models can output next activity predictions, which are less beneficial for process stakeholders.

Some works tackled this problem with prescriptive business process monitoring (PrBPM) approaches. PrBPM approaches assess predictions regarding their impact on the process performance – typically measured by KPIs – to prevent undesired activities [21]. To achieve that, existing approaches generate alarms [21,5,11,13] or recommend actions [3,6]. However, none of these approaches recommends next best actions in the form of process activities that are optimised regarding a given KPI for running processes. In our case, *best* refers to a KPI's optimal value regarding the future course of a process instance. Additionally, the next best actions, which depend on next activity predictions and prediction of a particular KPI, might obscure the actual business process. Therefore, transforming methods should check whether a recommended action is conform regarding a process description.

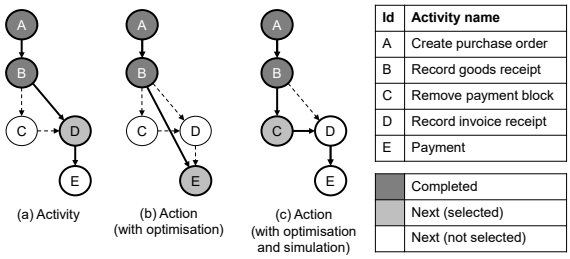

Fig. 1: A next activity prediction vs. a next best action recommendation.

Given a running process instance of a purchase order handling process after finishing the first two activities, a DNN model predicts the next most likely activity D (cf. (a) in Fig. 1). Additionally, the KPI *time* is of interest and the the first two activities A and B take each 1 hour, the predicted activity D takes 2 hours and the last activity E takes 2 hours. In sum, the complete process instance takes 6 hours and a deadline of 5 hours – that exists due to a general agreement – is exceeded. In contrast, the recommended action with optimisation can be the activity E (cf. (b) in Fig. 1). Even though the complete process instance takes 4

hours, the mandatory activity D is skipped. With an additional simulation, the activity "Remove payment block" can be recommended (cf. (c) in Fig. 1) taking 1 hour. Afterwards, the activities D and E are followed with a duration of 2 hours and 1 hour. Here, the activity E takes 1 hour instead of 2 hours since the payment block is already removed. Thus, the complete process instance takes 5 hours, and the deadline is met.

In this paper, we provide a PrBPM technique for recommending the next best actions depending on a KPI. Thereby, it conducts a business process simulation (BPS) to remain within the allowed control-flow. To reach our research objective, we develop a PrBPM technique and evaluate its actions regarding the optimisation of a KPI and the distance from ground truth process instances with two real-life event logs.

This paper is an extended and revised version of a research-and-progress paper [26]. Additionally, it includes a BPS and an evaluation with two real-life logs. The paper is structured as follows: Sec. 2 presents the required background for our PrBPM technique. Sec. 3 introduces the design of our PrBPM technique for recommending next best actions. Further, we evaluate our technique in Sec. 4. Sec. 5 provides a discussion. The paper concludes with a summary and an outlook on future work in Sec. 7.

2 Background

2.1 Preliminaries

PBPM or PrBPM techniques require event log data. We adapt definitions by Polato et al. [15] to formally describe the terms *event*, *trace*, *event log*, *prefix* and *suffix*. In the following, $\mathcal{A}$ is the set of process activities, $\mathcal{C}$ is the set of process instances (cases), and C is the set of case ids with the bijective projection $id : C \to \mathcal{C}$, and $\mathcal{T}$ is the set of timestamps. To address time, a process instance $c \in \mathcal{C}$ contains all past and future events, while events in a trace σ_c of c contain all events up to the currently available time instant. $\mathcal{E} = \mathcal{A} \times C \times \mathcal{T}$ is the event universe.

Definition 1 (Event). *An event $e \in \mathcal{E}$ is a tuple $e = (a, c, t)$, where $a \in \mathcal{A}$ is the process activity, $c \in C$ is the case id, and $t \in \mathcal{T}$ is its timestamp. Given an event e, we define the projection functions $F_p = \{f_a, f_c, f_t\}$: $f_a : e \to a, f_c : e \to c$, and $f_t : e \to t$.*

Definition 2 (Trace). *A trace is a sequence $\sigma_c = \langle e_1, \ldots, e_{|\sigma_c|} \rangle \in \mathcal{E}^*$ of events, such that $f_c(e_i) = f_c(e_j) \wedge f_t(e_i) \leq f_t(e_j)$ for $1 \leq i < j \leq |\sigma_c|$. Note a trace σ_c of process instance c can be considered as a process instance σ_c.*

Definition 3 (Event log). *An event log $\mathcal{L}_\tau$ for a time instant τ is a set of traces, such that $\forall \sigma_c \in \mathcal{L}_\tau \,.\, \exists c \in \mathcal{C} \,.\, (\forall e \in \sigma_c \,.\, id(f_c(e)) = c) \wedge (\forall e \in \sigma_c \,.\, f_t(e) \leq \tau)$, i. e. all events of the observed cases that already happened.*

Definition 4 (Prefix, suffix of a trace). *Given a trace $\sigma_c = \langle e_1, .., e_k, .., e_n \rangle$, the prefix of length k is $hd^k(\sigma_c) = \langle e_1, .., e_k \rangle$, and the suffix of length k is $tl^k(\sigma_c) = \langle e_{k+1}, .., e_n \rangle$, with $1 \leq k < n$.*

2.2 Long short-term memory neural networks

Our PrBPM technique transforms next activity predictions into the next best actions. Thus, next activity predictions are the basis for our PrBPM technique. To predict next activities, we use a "vanilla", i.e. basic, long short-term memory network (LSTM) [7] because most of the PBPM techniques for predicting next activities rely on this DNN architecture [24]. LSTMs belong to the class of recurrent neural networks (RNNs) [8] and are designed to handle temporal dependencies in sequential prediction problems [1]. In general, consists of three layers: an input layer (receiving data input), a hidden layer (i.e. an LSTM layer with an LSTM cell) and an output layer (providing predictions).

An LSTM cell uses four gates to manage its memory over time to avoid the problem of gradient exploding/vanishing in the case of longer sequences [1]. First, a forget gate that determines how much of the previous memory is kept. Second, an input gate controls how much new information is stored into memory. Third, a gate gate or candidate memory that defines how much information is stored into memory. Fourth, an output gate that determines how much information is read out of the memory.

To learn an LSTM's parameters, a loss function (e.g. the cross-entropy loss for classification) is defined on a data point (i.e. prediction and label) and measures the penalty. Additionally, a cost function in its basic form calculates the sum of loss functions over the training set. The LSTM's parameters are updated iteratively via a gradient descent algorithm (e.g. stochastic gradient descent), in that, the gradient of the cost function is computed by backpropagation through time [19]. After learning the parameters, an LSTM model with adjusted parameter values exists.

2.3 Business process simulation

Actions optimised according to a KPI can be not conform to the process control-flow. Thus, suggesting process-conform actions to process stakeholders is essential. Consequently, we add control-flow knowledge to our PrBPM technique with formal process models.

A well-known approach to assess the quality of process executions is business process simulation (BPS). Several approaches examine processes and their variants regarding compliance or performance with BPS [2,16]. We refer to discrete-event-driven BPS [22]. Here, simulation models formally contain discrete events which are interrelated via process semantics.

BPS usually delivers its insights to users [17]. Unlike existing approaches, such as [27,18], we use the simulation results to process the predictions of an LSTM. Thus, we use discrete-event-driven [22] short-term simulation [18] as a boundary measure to ensure that the DNN-based next best action makes sense from a control-flow perspective. Alike Rozinat et al. [18], our simulation starts from a non-empty process state to aid in recommending the next best action from the current state on.

3 A PrBPM technique for recommending next best actions

Our PrBPM technique transforms next activity predictions into the next best actions. The technique consists of an *offline* and an *online component*. In the offline component, it learns a DNN for predicting next activities and values of a KPI. In the online component, the next best actions are recommended based on the next activity and KPI value predictions.

3.1 Offline component

The offline component receives as input an event log $\mathcal{L}_\tau$, and outputs the two ML models m_{pp} and m_{cs}. While m_{pp} (process prediction model) predicts next activities and a KPI value related to next activities, m_{cs} (candidate selection model) selects a fix set of suffix candidates. The technique learns both models from individually pre-processed versions of $\mathcal{L}_\tau$. Fig. 2 visualises the steps of the offline component. In the following, we describe the steps of the offline

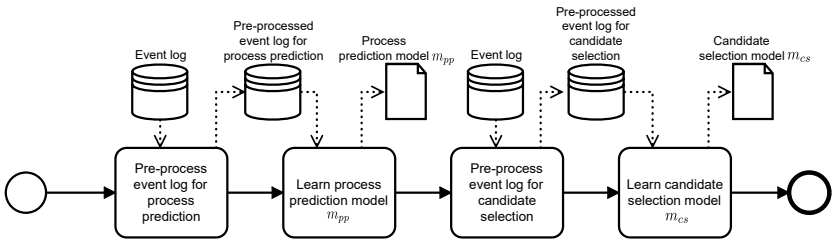

Fig. 2: Four-step offline component scheme with the two models m_{pp} and m_{cs}.

component based on the exemplary finished process instance σ_1^f, as represented in (1). The last attribute per event is the KPI; here the defined costs for executing an activity.

$$\begin{aligned}\sigma_1^f = \langle &(1, \text{“Create Application”}, \text{2011-09-30 16:20:00}, 0),\\ &(1, \text{“Concept”}, \text{2011-09-30 17:30:00}, 10),\\ &(1, \text{“Accepted”}, \text{2011-09-30 18:50:00}, 20),\\ &(1, \text{“Validating”}, \text{2011-09-30 19:10:00}, 40)\rangle.\end{aligned} \tag{1}$$

Pre-process event log for process prediction. m_{pp} is a multi-task DNN for predicting next activities and a KPI value at each time step of a running process instance. For m_{pp}, the pre-processing of $\mathcal{L}_\tau$ comprises four steps. First, to determine the end of each process instance in $\mathcal{L}_\tau$, it adds a termination event to the end of each process instance. So, for σ_1^f, as represented in (1), we add the event (1, “End”, 2011-09-30 19:10:00, 0) after the fourth event with the activity name “Validating”. Additionally, for termination events, we always overtake the timestamp value of the previous event and set the value of the KPI to 0. Second,

it onehot-encodes all activity names in the process instances as numeric values (cf. (2) for σ_1^f including the termination event's activity).

$$\sigma_1^f = \langle (0,0,0,0,1),(0,0,0,1,0),(\dots),(0,1,0,0,0),(1,0,0,0,0)\rangle. \tag{2}$$

This step is necessary since LSTMs, as used in this paper, use a gradient descent optimisation algorithm to learn the network's parameters. Third, it crops prefixes out of process instances by using the function $hd^k()$. For instance, a prefix with size three of σ_1^f is:

$$hk^3(\sigma_1^f) = \langle (0,\ 0,\ 0,\ 0,\ 1),(0,\ 0,\ 0,\ 1,\ 0),(0,\ 0,\ 1,\ 0,\ 0)\rangle. \tag{3}$$

Lastly, it transforms the cropped input data into a three-order tensor (prefixes, time steps and attributes). Additionally, m_{pp} needs two label structures for parameter learning. First, for the onehot-encoded next activities, a two-dimensional label matrix is required. Second, if the KPI values related to the next activities are scaled numerically, a one-dimensional label vector is required. If the values are scaled categorically, a two-dimensional label matrix is needed.

Create process prediction model. m_{pp} is a multi-task DNN. The model's architecture follows the work of Tax et al. [20]. The input layer of m_{pp} receives the data and transfers it to the first hidden layer. The first hidden layer is followed by two branches. Each branch refers to a prediction task and consists of two layers, a hidden layer and an output layer. The output layer of the upper branch realises next activity predictions, whereas the lower creates KPI value predictions. Depending on the KPI value's scaling (i.e. numerical or categorical), the lower branch solves either a regression or classification problem. Each hidden layer of m_{pp} is an LSTM layer with an LSTM cell.

Pre-process event log for candidate selection. m_{cs} is a nearest-neighbour-based ML algorithm for finding suffixes "similar" to predicted suffixes. For m_{cs}, the pre-processing of $\mathcal{L}_\tau$ consists of three steps. First, it ordinal-encodes all activity names in numerical values. For example, the ordinal-encoded representation of σ_1^f, as depicted in (1) including the termination event's activity, is $\langle (1),(2),(3),(4),(5)\rangle$. Second, it crops suffixes out of process instances through the function $tl^k(\cdot)$. For instance, the suffix with size three of σ_1^f $(lt^3(\sigma_1^f))$ is $\langle (4),(5)\rangle$. Lastly, the cropped input data is transformed into a two-dimensional matrix (suffixes and attributes).

Create candidate selection model. m_{cs} is a nearest-neighbour-based ML algorithm. It retrieves k suffixes "nearest" to a suffix predicted for a given prefix (i.e. a running process instance at a certain time step). The technique learns the model m_{cs} based on all suffixes cropped out of $\mathcal{L}_\tau$.

3.2 Online component

The online component receives as input a new process instance, and the two trained predictive models m_{pp} and m_{cs}. It consists of five steps (see Fig. 3), and

outputs next best actions. After pre-processing (first step) of the running process instance, a suffix of next activities and its KPI values are predicted (second step) by applying m_{pp}. The second step is followed by the condition, whether the sum of the KPI values of the suffix and the respective prefix exceeds a threshold or not. If the threshold is exceeded, the predicted suffix of activities is transferred from the second to the third step (i.e. find candidates) and the procedure for generating next best actions starts. Otherwise, it provides the next most likely activity. To find a set of suffix candidates, the technique loads m_{cs} from the offline component. Subsequently, it selects the best candidate from this set depending on the KPI and concerning BPS. Finally, the first activity of the selected suffix represents the best action and is concatenated to the prefix of activities (i.e. running process instance at a certain time step). If the best action is the end of the process instance, the procedure ends. Otherwise, the procedure continues and predicts the suffix of the new prefix. In the following, we detail the online

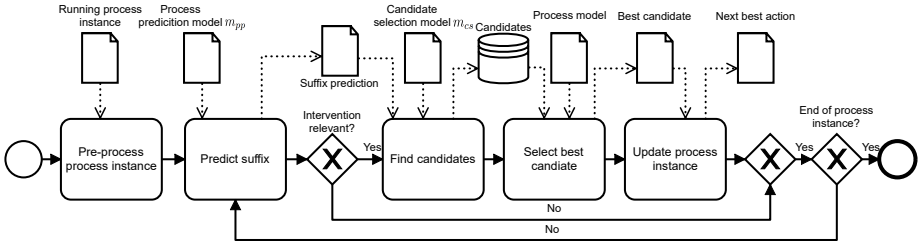

Fig. 3: Five-step online component scheme with the two models m_{pp} and m_{cs}.

component's five steps. Thereby, we refer to the running process instance σ_2^r, for that the second event has just finished.

$$\begin{aligned}\sigma_2^r = \langle &(2,\ \text{“Create Application”},\ \text{2012-09-30 18:00:00},\ 20),\\ &(2,\ \text{“Concept”},\ \text{2012-09-30 18:30:00},\ 20)\rangle.\end{aligned} \tag{4}$$

Pre-process process instance. To predict the suffix of the running process instance with m_{pp}, we onehot-encode all activity names in numerical values and transform the output into a third-order tensor.

Predict suffix. Based on a running process instance (prefix), m_{pp} predicts the next sequence of activities and the KPI values. To get the complete suffix, we apply m_{pp} repeatedly. Afterwards, the technique calculates the sum of the KPI values over the activities of the complete process instance consisting of the prefix and its predicted suffix. For instance, if the prefix of a process instance is σ_2^r, one potential suffix is:

$$s_{\sigma_2^r} = \langle(\text{“Accepted”},\ 20),(\text{“Validating”},\ 40),(\text{“End”},\ 10)\rangle. \tag{5}$$

For a better intuition, we omit the suffixes' encoding in the online component. The values 20, 40 and 10 assigned to the events are KPI values (e.g. cost

values) predicted by m_{pp}. In line with Tax et al. [20], we do not perform the suffix prediction for prefixes with size ≤ 1 since the amount of activity values is insufficient. After predicting the suffix, the total costs of σ_2^r are 110. To start the procedure for recommending the next best actions, the total KPI value of an instance has to exceed a threshold value t. The value of t can be defined by domain experts or derived from the event log (e.g. average costs of process instances). Regarding σ_2^r, the procedure starts because we assume $t = 100$ ($110 > t$).

Find candidates. Second, for the predicted suffix, m_{cs} from the offline component reveals a set of alternatives with a meaningful control-flow. For example, m_{cs} ($k = 3$) selects based on $s_{\sigma_2^r}$ the following three suffix alternatives:

$$\begin{aligned} m_{cs}(s_{\sigma_2^r}) = [&\langle(\text{“End”, 10})\rangle, \\ &\langle(\text{“Accepted”, 20}),(\text{“Validating”, 40}),(\text{“End”, 10})\rangle, \\ &\langle(\text{“Validating”, 20}),(\text{“Accepted”, 10}),(\text{“Validating”, 10}), \\ &(\text{“End”, 10})\rangle]. \end{aligned} \tag{6}$$

In (6), the first and the third suffix result in total costs (50 and 90) falling below t.

Select the best candidate. In the third step, we select the next best action from the set of possible suffix candidates. We sort the suffixes by the KPI value. Thus, the first suffix is the best one in regard to the KPI. To incorporate control-flow knowledge, a simulation model checks the resulting instance. Thereby, we reduce the risk of prescribing nonsensical actions. The simulation uses a formal process model to retrieve specific process semantics. The simulation produces the current process state from the prefix and the process model. If the prefix does not comply with the process model, the simulation aborts the suffix selection for the prefix and immediately recommends an intervention. Otherwise, we check the k selected suffixes whether they comply with the process model in the simulation from the current process state on. If a candidate suffix fails the simulation, our technique omits it in the selection. However, when all suffix candidates infringe the simulation, the technique assumes the predicted next activity as the best action candidate. Concerning the candidate set from (6), the best candidate is suffix three since it does not infringe the simulation model.

Update process instance. To evaluate our technique, we assume that a process stakeholder performs in each case the recommended action. Thus, if the best suffix candidate exists, the activity (representing the next best action) and the KPI value of the first event are concatenated to the running process instance (i.e. prefix). After the update, σ_2^r comprises three events, as depicted in (7).

$$\begin{aligned} \sigma_2^r = \langle&(2, \text{“Create Application”, 2012-09-30 18:00:00, 20}), \\ &(2, \text{“Concept”, 2012-09-30 18:30:00, 20}), \\ &(2, \text{“}\mathbf{Validating}\text{”, –, } \mathbf{20})\rangle. \end{aligned} \tag{7}$$

The technique repeats the complete procedure until the termination event is reached.

4 Evaluation

We provide an evaluation regarding our PrPBM technique's optimisation of a KPI and the distance from ground truth process instances. For that, we developed a prototype that recommends next best actions depending on the KPI *throughput time* and concerning a process simulation realised with DCR graphs. We compare our results to a representative baseline [20] for two event logs.

4.1 Event logs

First, we use the *helpdesk*[1] event log containing data from an Italian software company's ticketing management process. It includes 21,348 events, 4,580 process instances, 226 process instance variants and 14 activities. Second, we include the *bpi2019*[2] event log from the BPI challenge 2019, provided by a company for paints and coatings. It depicts a purchase order handling processes. For this event log, we only considered a random 10%-sampling with sequences of 30 events or shorter, due to the high computation effort. It includes 101,714 events, 24,900 process instances, 3,255 process instance variants and 32 activities.

4.2 Process models

We used DCR graphs as models for the BPS in our technique's best candidate selection. In Fig. 4, we present the DCR graph for the *helpdesk* event log. The three most important constraints are the following. First, after "Closed" the other activities should not happen. Second, if "Assign seriousness" occurs, someone must take over the responsibility. Third, before a ticket is closed, "Resolve ticket" must occur.

Fig. 5 shows the DCR graph for the *bpi2019* event log. The three most essential constraints are the following. First, "Create Purchase Order Item" may only happen once per order. Second, After the goods were received, "Change Quantity" and "Change price" should not occur. Third, "Record Goods Receipt", "Record Invoice Receipt" and "Clear Invoice" must eventually follow each other.

4.3 Procedure

We split both event logs in a 2/3 training and 1/3 test set with a random process-instance-based sampling. As a baseline, we use the most cited next event PBPM technique from Tax et al. [20]. We evaluate the technique in two ways. First, we evaluate the optimisation of the KPI *throughput time* (*in-time* value) by the percentage of process instances that could comply with the temporal threshold for different prefix sizes. The temporal threshold is the average throughput time of a process instance in an event log. Second, we evaluate the *distance* from

[1] https://data.mendeley.com/datasets/39bp3vv62t/1.
[2] https://data.4tu.nl/repository/uuid:a7ce5c55-03a7-4583-b855-98b86e1a2b07.

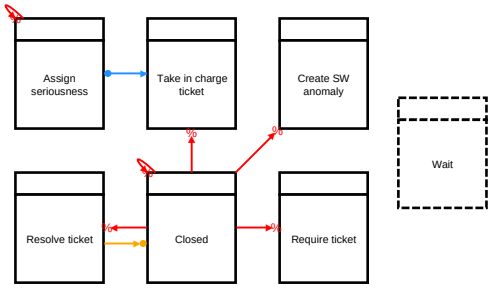

Fig. 4: DCR graph for the *helpdesk* event log.

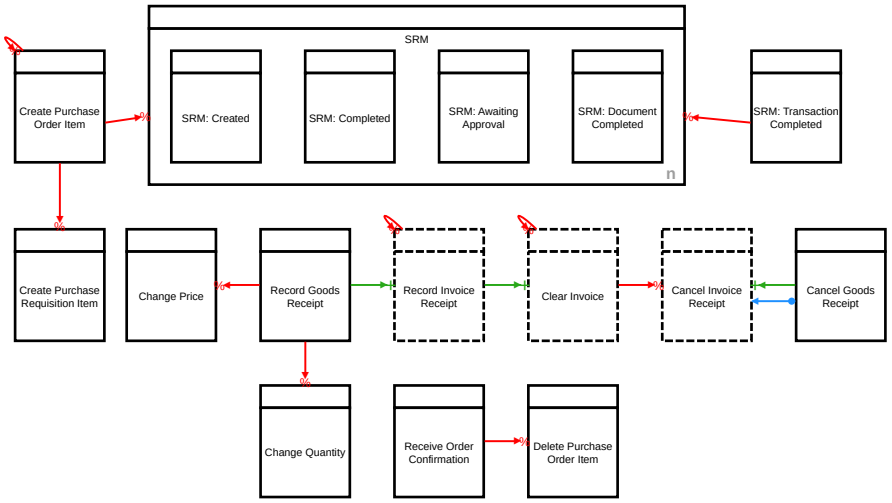

Fig. 5: DCR graph for the *bpi2019* event log.

the ground truth process instance through the average Damerau-Levenshtein distance. This metric determines the distance between two strings or sequences through the minimum number of operations (consisting of insertions, deletions or substitutions of a single character, or transposition of two adjacent characters), i.e. the lower the value, the more similar the strings are.

To train the multi-task LSTM m_{pp}, we apply the *Nadam* optimisation algorithm with a *categorical cross-entropy loss* for next activity predictions and a *mean squared error* for *throughput time* (KPI) predictions. Moreover, we set the batch size to 256, i. e. gradients update after every 256^{th} sample of the training set. We set the default values for the other optimisation parameters. For training the candidate selection model m_{cs}, we apply the nearest-neighbour-based ML algorithm ball tree [14]. Ball tree utilises a binary tree data structure for

maintaining spatial data hierarchically. We choose a spatial-based algorithm to consider the semantic similarity between the suffixes of activities and KPI values. Moreover, we set the hyperparameter k (number of "nearest" neighbours) of m_{cs} to 5, 10 and 15. Thereby, we check different sizes of the suffix candidate set.

Finally, technical details and the source code are available on GitHub[3].

4.4 Results

Fig. 6 shows the results for the *helpdesk* event log. For most of the prefixes in

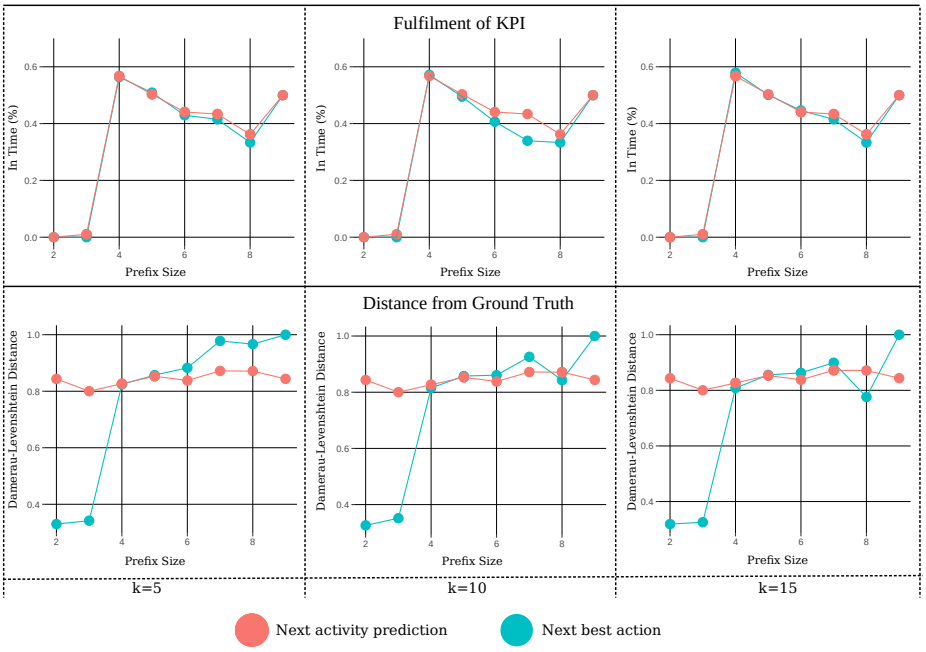

Fig. 6: Results for the *helpdesk* event log.

the *helpdesk* event log, our technique's next best actions are more *in time* than next activity predictions. While for $k = 10$ next best actions have the lowest *in-time* values compared to next activity predictions, *in-time* values of next best actions with $k = 5$ and $k = 15$ are rather similar to each other. Furthermore, the higher the k, the lower is the *distance* of the next best actions from the actual process instances. Up to prefix size 4, the *distance* of the next best actions is lower compared to next activity predictions.

Fig. 7 shows the results for the *bpi2019* event log. For most of the prefixes with a size ≥ 8, the next best actions of our technique are more *in time* than next activity predictions. For $k = 15$ and prefixes ≥ 8, next best actions have

[3]https://github.com/fau-is/next-best-action.

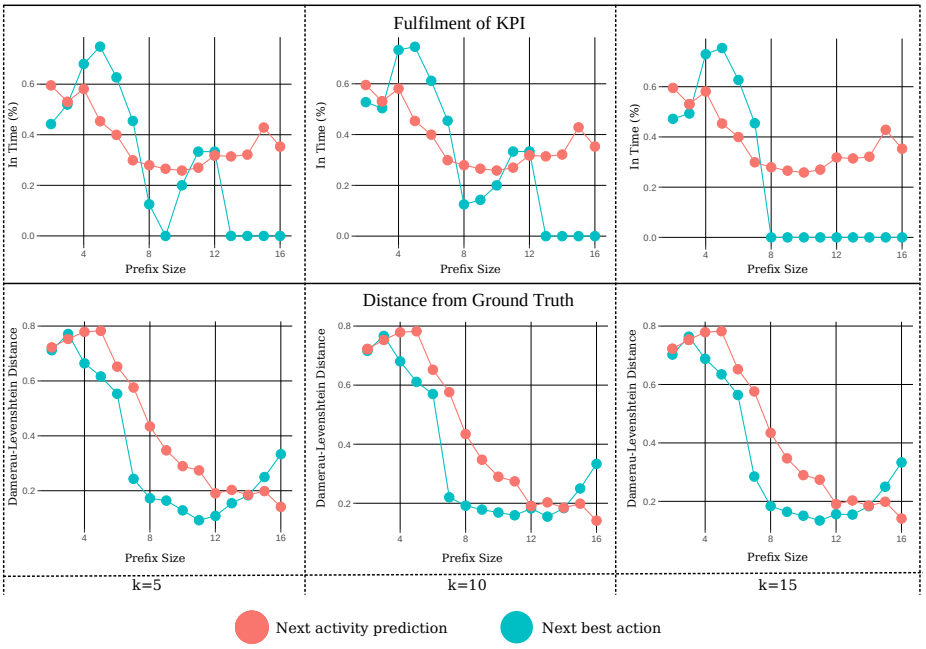

Fig. 7: Results for the bpi2019 event log.

an *in-time* value of 0. With an increasing k, the *in-time* values of the next best actions vary less from prefix size 2 to 12. In contrast to next activity predictions, the *distance* of next best actions is lower for prefixes with a size > 3 and < 15. Over the three k values, the *distance* of next best actions is rather similar.

5 Discussion

Our contribution to academia and practice is a PrBPM technique that recommends the next best actions depending on a KPI while concerning BPS. Moreover, the evaluation presents an instantiation of our PrBPM technique. The KPI is the *throughput time*, and a DCR graph realises BPS via the event log.

Based on our results, our PrBPM technique can provide actions with lower *in-time* values and less *distance* from the ground truth compared to the next most likely activities for both event logs. However, the *in-time* values (i.e. the percentage of process instances that could comply with the temporal threshold) of next best actions differs more from the baseline's next activity prediction for the *bpi2019* event log than for the *helpdesk* event log. The *helpdesk* event log has a lower instance variability than the *bpi2019* event log. Therefore, fewer process paths exist from which our technique can recommend actions with lower *in-time* values. Further, the number of candidates k has an effect on the KPI's optimisation. While we get the actions with the lowest *in-time* values with $k = 10$

for the *helpdesk* event log, the KPI values with $k = 5$ and $k = 15$ are similar to each other. For the *bpi2019* event log, our technique provides actions with the lowest *in-time* value if k is set to 15. The results with $k = 5$ are similar to those of $k = 10$. A higher k value leads to lower *in-time* values in the *bpi2019* event log because of a higher instance variability. On the contrary, the *helpdesk* event log needs a lower k value. Regarding the *distance* from ground truth process instances, most of the next best actions (especially those for the *bpi2019* event log) reach a better result than next activity predictions. A reason for that could be the limited predictive quality of the underlying DNN model for predicting next activities. However, our technique integrates control-flow knowledge and therefore overcomes this deficit to a certain degree. Moreover, for the *bpi2019* event log, our technique provides actions with lower *in-time* values for prefixes with a size ≥ 8. In terms of the *helpdesk* event log, we get actions with lower *in-time* values for shorter prefixes. We suppose that our technique requires a longer prefix for event logs with higher instance variability to recommend next best actions. Finally, even though our technique provides actions with lower *in-time* values, it seems that it does not terminate before the baseline. Our results show the aggregated values over different prefix sizes. Thus, we assume that few sequences, for which the termination can not be determined, distort the results.

Despite all our efforts, our technique bears three shortcomings. First, we did not optimise the hyperparameters of the DNN model m_{pp}, e.g. via random search. Instead, we set the hyperparameters for m_{pp} according to the work of Tax et al. [20]. We used the same setting since we compare our technique's next best actions to their next activity predictions. Second, even though our technique is process-modelling-notation agnostic, we argue that declarative modelling is an appropriate approach for the process simulation. Due to its freedoms, declarative modelling facilitates the partial definition of the control-flow. As a consequence, we have a more flexible definition of a process's control-flow than by using a restricted procedural process model. While our DNN-based technique copes well with rather flexible processes, other techniques using *traditional* ML algorithms (e.g. a decision tree) might handle restricted processes faster and with a higher predictive quality. Third, for our PrBPM technique's design, we neither consider cross-instance nor cross-business-process dependencies. In an organisational environment, additional effects like direct and indirect rebound effects can hinder our technique.

6 Related Work

A variety of PBPM techniques were proposed by researchers as summarised by, e.g. Márquez-Chamorro et al. [10] or Di Francescomarino et al. [4]. Many of these techniques are geared to address the next activity prediction task. For that, most of the recent techniques rely on LSTMs [24] such as Weinzierl et al. [25]. To predict not only the next activities with a single predictive model, Tax et al. [20] suggest a multi-task LSTM-based DNN architecture. With this architecture, they predict the next activities and their timestamps. Metzger et

al. [12] extend their architecture by another LSTM layer to additionally predict the binary process outcome whether a delay occurs in the process or not. These techniques output predictions and do not recommend next best actions.

Furthermore, researchers suggested PrBPM approaches that raise alarms or recommend actions to prevent undesired activities. Metzger et al. [11] investigate the effect of reliability estimates on (1) intervention costs (called adaption cost) and (2) the rate of non-violation of process instances by performing a simulation of a parameterised cost model. Thereby, they determine the reliability estimates based on the predictions of an ensemble of multi-layer perceptron classifiers at a pre-defined point in the process. In a later work [13], reliability estimates were determined based on an ensemble of LSTM classifiers at different points in the process. The recommendation of actions is not part of these works. Teinemaa et al. [21] propose a concept of an alarm-based PrBPM framework. They suggest a cost function for generating alarms that trigger interventions to prevent an undesired outcome or mitigate its effect. In a later work [5], a multi-perspective extension of this framework was presented. In both versions, the framework focuses on alarms. Gröger et al. [6] present a PrBPM technique that provides action recommendations for the next process step during the execution of a business process to avoid a predicted performance deviation. Performance deviation is interpreted as a binary outcome prediction, i.e. exists a deviation or not. In detail, an action recommendation comprises several action items and is represented by a rule extracted from a learned decision tree. An action item consists of the name and value of a process attribute. Even though this approach recommends actions in the form of process attribute values of the next process step which are optimised according to a KPI (e.g. lead time), process steps as next best actions are not recommended. Conforti et al. [3] propose a PrBPM technique that predicts risks depending on the deviation of metrics during process execution. The technique's purpose is to provide decision support for certain actions such as the next process activity which minimises process risks. However, this technique can only recommend actions which are optimised regarding the KPI risk. Thus, with the best of our knowledge, there is no PrBPM approach that transforms next most likely activity predictions into the next best actions (represented by activities) depending on a given KPI.

7 Conclusion

Next activity predictions provided by PBPM techniques can be less beneficial for process stakeholders. Based on our motivation and the identified research gap, we argue that there is a crucial need for a PrBPM technique that recommends the next best actions in running processes. We reached our research goal with the evaluation of our developed PrBPM technique in Sec. 5. Thereby, we show that our technique can outperform the baseline regarding KPI fulfilment and distance from ground truth process instances. Further research might concern different directions. First, we plan to adapt existing loss functions for LSTMs predicting next most likely activities. Such a loss function can enable an LSTM

to directly consider information on KPIs in the learning procedure. Second, future research should further develop existing PrBPM approaches. More advanced multi-tasking DNN architectures can facilitate the recommendation of more sophisticated next best actions. For instance, next best actions that optimise more than one KPI. Finally, we call for PrBPM techniques that are aware of concept evolution. Our technique is not able to recommend an activity as the best action if it was not observed in the training phase of the ML models.

Acknowledgments

This project is funded by the German Federal Ministry of Education and Research (BMBF) within the framework programme *Software Campus* under the number 01IS17045.

References

1. Bengio, Y., Simard, P., Frasconi, P., et al.: Learning long-term dependencies with gradient descent is difficult. Transactions on Neural Networks **5**(2), 157–166 (1994)
2. Centobelli, P., Converso, G., Gallo, M., Murino, T., Santillo, L.C.: From process mining to process design: A simulation model to reduce conformance risk. Engineering Letters **23**(3), 145–155 (2015)
3. Conforti, R., De Leoni, M., La Rosa, M., Van Der Aalst, W.M.: Supporting risk-informed decisions during business process execution. In: Proceedings of the 25th International Conference on Advanced Information Systems Engineering. pp. 116–132. Springer (2013)
4. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive process monitoring methods: Which one suits me best? In: Proceedings of the 16th International Conference on Business Process Management (BPM). pp. 462–479. Springer (2018)
5. Fahrenkrog-Petersen, S.A., Tax, N., Teinemaa, I., Dumas, M., de Leoni, M., Maggi, F.M., Weidlich, M.: Fire now, fire later: Alarm-based systems for prescriptive process monitoring. arXiv preprint arXiv:1905.09568 (2019)
6. Gröger, C., Schwarz, H., Mitschang, B.: Prescriptive analytics for recommendation-based business process optimization. In: Proceedings of the 17th International Conference on Business Information Systems. pp. 25–37. Springer (2014)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**(8), 1735–1780 (1997)
8. LeCun, Y., Bengio, Y., Hinton, G.: Deep learning. Nature **521**(7553), 436 (2015)
9. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Proceedings of the 26th International conference on Advanced Information Systems Engineering (CAiSE). pp. 457–472. Springer (2014)
10. Márquez-Chamorro, A., Resinas, M., Ruiz-Cortás, A.: Predictive monitoring of business processes: a survey. IEEE Transactions on Services Computing (TSC) pp. 1–18 (2017)
11. Metzger, A., Föcker, F.: Predictive business process monitoring considering reliability estimates. In: International Conference on Advanced Information Systems Engineering. pp. 445–460. Springer (2017)

12. Metzger, A., Franke, J., Jansen, T.: Data-driven deep learning for proactive terminal process management. In: Proceedings of the 17th International Conference on Business Process Management (BPM). pp. 196–211 (2019)
13. Metzger, A., Neubauer, A., Bohn, P., Pohl, K.: Proactive process adaptation using deep learning ensembles. In: International Conference on Advanced Information Systems Engineering. pp. 547–562. Springer (2019)
14. Omohundro, S.M.: Five balltree construction algorithms. International Computer Science Institute Berkeley (1989)
15. Polato, M., Sperduti, A., Burattin, A., de Leoni, M.: Data-aware remaining time prediction of business process instances. In: Proceeding of the International Joint Conference on Neural Networks (IJCNN). pp. 816–823. IEEE (2014)
16. Redlich, D., Gilani, W.: Event-driven process-centric performance prediction via simulation. In: Lecture Notes in Business Information Processing. vol. 99 LNBIP, pp. 473–478 (2012)
17. Rosenthal, K., Ternes, B., Strecker, S.: Business process simulation: A systematic literature review. In: Proceedings of the 26th European Conference on Information Systems (ECIS) (2018)
18. Rozinat, A., Wynn, M.T., van der Aalst, W.M., ter Hofstede, A.H., Fidge, C.J.: Workflow simulation for operational decision support. Data and Knowledge Engineering **68**(9), 834–850 (2009)
19. Rumelhart, D.E., Hinton, G.E., Williams, R.J.: Learning representations by back-propagating errors. Nature **323**(6088), 533–536 (1986)
20. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Proceedings of the 29th International Conference on Advanced Information Systems Engineering (CAiSE). pp. 477–492. Springer (2017)
21. Teinemaa, I., Tax, N., de Leoni, M., Dumas, M., Maggi, F.M.: Alarm-based prescriptive process monitoring. In: Proceedings of the 16th International Conference on Business Process Management (BPM). pp. 91–107. Springer (2018)
22. Tumay, K.: Business process simulation. In: Proceedings of the Winter Simulation Conference. pp. 93–98. ACM (1996)
23. van der Aalst, W.M.P.: Process Mining: Data Science in Action. Springer, 2 edn. (2016)
24. Weinzierl, S., Zilker, S., Brunk, J., Revoredo, K., Nguyen, A., Matzner, M., Becker, J., Eskofier, B.: An empirical comparison of deep-neural-network architectures for next activity prediction using context-enriched process event logs. arXiv:2005.01194 (2020b)
25. Weinzierl, S., Stierle, M., Zilker, S., Matzner, M.: A next click recommender system for web-based service analytics with context-aware LSTMs. In: Proceedings of the 53rd Hawaii International Conference on System Sciences (HICSS) (2020)
26. Weinzierl, S., Zilker, S., Stierle, M., Park, G., Matzner, M.: From predictive to prescriptive process monitoring: Recommending the next best actions instead of calculating the next most likely events. In: Proceedings of the 15th International Conference on Wirtschaftsinformatik. AISeL (2020c)
27. Wynn, M.T., Dumas, M., Fidge, C.J., ter Hofstede, A.H.M., van der Aalst, W.M.P.: Business process simulation for operational decision support. In: Business Process Management Workshops, Lecture Notes in Computer Science, vol. 4928, pp. 66–77. Springer (2008)