Probability based Heuristic for Predictive Business Process Monitoring

Kristof Böhmer and Stefanie Rinderle-Ma

University of Vienna, Faculty of Computer Science
{kristof.boehmer,stefanie.rinderle-ma}@univie.ac.at

Abstract. Predictive business process monitoring concerns the unfolding of ongoing process instance executions. Recent work in this area frequently applies "blackbox" like methods which, despite delivering high quality prediction results, fail to implement a transparent and understandable prediction generation process, likely, limiting the trust users put into the results. This work tackles this limitation by basing prediction and the related prediction models on well known probability based histogram like approaches. Those enable to quickly grasp, and potentially visualise the prediction results, various alternative futures, and the overall prediction process. Furthermore, the proposed heuristic prediction approach outperforms state-of-the-art approaches with respect to prediction accuracy. This conclusion is drawn based on a publicly available prototypical implementation, real life logs from multiple sources and domains, along with a comparison with multiple alternative approaches.

Keywords: Business Process, Predictive Monitoring, Probability

1 Introduction

Predictive process monitoring enables to predict the *unfolding* of ongoing process executions based on behaviour extracted from historic executions. This includes the prediction of, e.g., the activity to be executed next. Hereby, the planning and prioritisation of instances and their resource utilisation can be supported, e.g., to *prevent* the *violation* of Service-Level Agreements (SLAs), cf. [27, 24].

We found that predictive monitoring work, especially if it strives to predict upcoming activity executions and their timestamps, can largely be categorised into two main groups based on the applied approach, cf. [18]: At first, probability based works which transform historic logged execution behaviour into probability based prediction models to predict, e.g., the most probable future execution behaviour (e.g., the next activity), cf. [26]. We found those models to be small, easy to understand, follow, and interpret. Secondly, neural networks are gaining interest, cf. [18]. Especially, as it was found that they *outperform* probability based approaches, for example, with regards to the prediction accuracy, cf. [27].

Unfortunately, the latter *struggle* with regards to prediction result transparency and understandability – as neural networks are generally assumed as "black boxes", cf. [3]. For example, fully understanding and grasping the relation and inner organisation between hundreds of neurons, which today's increasingly *larger* and *complex* neural networks (prediction models, resp.) are composed of, cf. [27], is extremely challenging. Further, given today's network complexity, it is hardly possible to fully grasp why/how such a complex neural network generated a specific outcome or how changing the network/neurons would affect it, cf. [21]. So, novel techniques are required which: *a*) combine the advantages of probability based techniques (model/result traceability and understandability); and *b*) neural network based techniques (high prediction result accuracy).

Hence, inspired by this observation this paper proposes a probability based prediction technique to predict the next execution event (activity, timestamp). Formally: Let p be an execution trace of process P for which the next execution event should be predicted. Further let L hold all historic execution traces t of P. The key idea is to: first, identify the *relevant* traces in L for this task, and secondly to create a probability based prediction model M from them. Here, trace relevance is measured based on the similarity of the execution events in p and t. Finally, the most probable next event for p is determined based on M.

So, instead of complex almost unfathomable neural network based prediction models this work generates and applies simplistic histogram based probability distributions. Its feasibility is analysed by comparing a prototypical implementation of the proposed approach with state-of-the-art neural network and probability based approaches using real-life execution logs from multiple domains.

This paper is organised as follows: Prerequisites and the proposed approach are introduced in Section 2. The proposed prediction approach (i.e., prediction model generation and its application) is, in detail, described in Section 2 and 3. Related work is discussed in Section 5 while Section 4 holds the evaluation. Finally, conclusions, discussions, and future work is given in Section 6.

2 Prerequisites and General Approach

The presented approach enables to predict, based on a given (sub) trace p, the next execution event (i.e., activity and timestamp). For this a prediction model M is generated from historic traces L (log, resp.), those are: *a*) automatically generated by process engines; *b*) representing real behaviour (including noise and ad hoc changes); and *c*) independent from outdated documentation, cf. [20]. This section, for the sake of understanding, focuses on next event activity prediction to outline the general proposed prediction approach. The more complex prediction of next event timestamps builds on and extends this approach in Sect. 3.

Definition 1 (Execution log). *Let L be a set of* `execution traces` *$t \in L$; $t := \langle e_1, \cdots, e_n \rangle$ holds a non-empty ordered list of execution* `events` *$e_i := (ea, et)$; e_i represents the execution of activity $e_i.ea$ at timestamp $e_i.et \in \mathbb{R}_{>0}$; t's order is given by $e_i.et$, i.e., the events' timestamp, cf. [5]. Based on a given event e_i and trace, $\bullet e_i$ determines its preceding event, i.e., $\bullet e_i = e_{i-1}$ if $i > 1$.*

This notion represents information provided by process execution log formats, such as, the eXtensible Event Stream[1], but also holds the necessary information (activities and timestamps) for the prediction of execution events. Accordingly, the first event e_1 for trace t_1 of the running example, cf. Table 1, is $t_1.e_1 = (\texttt{A}, 23)$. *Auxiliary* functions: $\langle\cdot\rangle_i$ and $\langle\cdot\rangle_{[f,k]}$ retrieve the element with index i (former) or a range of indexes (latter) where $f \leq i \leq k$, while $\langle\cdot\rangle^l$ retrieves the last element. $|T|$ determines the length and T^0 retrieves a random element from a list/set T.

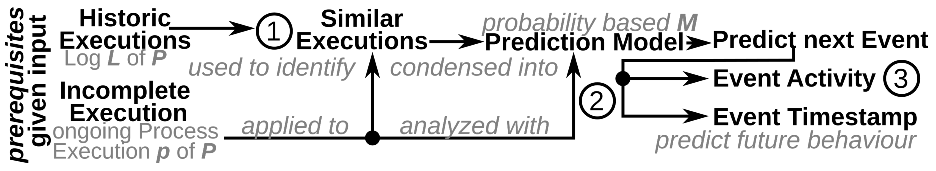

Fig. 1. Proposed probability based predictive monitoring approach – overview

Fig. 1 gives an overview on the proposed three staged prediction heuristic: Sect. 2 (activity) and 3 (timestamp). The core component is a probability based prediction model M which is generated based on a selection of given historic execution traces $t \in L$ and an incomplete executions trace $p \notin L$ – for which the next event should be predicted. Both, p and L are assumed as given input (prerequisites). Not each trace $t \in L$ is relevant for the prediction model M.

This is because some traces $t \in L$ are too dissimilar from $p \notin L$ to provide a glimpse on p's future behaviour ①. For example, because t and p follow dissimilar control flow execution paths in P so that behaviour in t does not allow to draw reliable conclusions for p's upcoming events. Compare, for example, trace t_4 and t_1 in Table 1, both represent an execution of P_1 with vastly different activity orders and occurrences which could stem, e.g., from different control flow decision node evaluations. Accordingly, we propose to utilise the dissimilarity/distance between the given traces when deciding which traces in L are used to build M.

Table 1. Realistic running example log L, cf. Helpdesk-Logs in Sect. 4

		Event $e_i := (ea, et)$ where ea=activity, et=timestamp										
Process P	Trace t	e_1		e_2		e_3		e_4		e_5		e_6
P_1	t_1	(A,23)	$\rightarrow$	(E,32)	$\rightarrow$	(E,37)	$\rightarrow$	(F,40)	$\rightarrow$	(E,47)	$\rightarrow$	(D,53)
P_1	t_2	(A,49)	$\rightarrow$	(E,54)	$\rightarrow$	(F,61)	$\rightarrow$	(E,68)	$\rightarrow$	(B,69)	$\rightarrow$	(D,78)
P_1	t_3	(A,40)	$\rightarrow$	(F,45)	$\rightarrow$	(E,49)	$\rightarrow$	(F,51)	$\rightarrow$	(E,57)	$\rightarrow$	(D,63)
P_1	t_4	(C,17)	$\rightarrow$	(A,21)	$\rightarrow$	(A,22)	$\rightarrow$	(A,25)	$\rightarrow$	(F,30)	$\rightarrow$	(E,37)

The applied distance measurement is inspired by the *Damerau-Levenstein distance* (DL for short) [10] – a common algorithm to measure the dissimilarity

[1] `http://www.xes-standard.org` – IEEE 1849-2016 XES Standard

between two sequences (traces, resp.). Here, this metric was chosen, over other approaches, such as, the Levenstein distance [23], which is frequently applied by existing prediction work, such as, [12]. This is because DL explicitly supports the transposition of a sequences' elements, cf. Def 2 – enabling to support parallel executions with varying activity orders but still comparable behaviour, cf. [27].

Definition 2 (Damerau-Levenstein activity based trace dissimilarity). *Let t and t' be two traces. Their dissimilarity is measured by determining the most* `cost efficient sequence` *of* `ins`*ert,* `del`*ete,* `sub`*stitution, and* `tran`*spositions* `operations` *required so that the order of event activity labels (given by $e.ea$) in t, t' becomes* `equal`*. Accordingly, each edit operation gets assigned an individual cost:* $\texttt{ins}, \texttt{del}, \texttt{sub}, \texttt{tran} \in \mathbb{N}_{>0}$*. Finally, the dissimilarity of t and t' is* `recursively` *calculated by applying $\Delta(t,t') \mapsto \mathbb{N}$, i.e., comparable to [10]:*

$$\Delta(t,t') := \begin{cases} \max(|t|,|t'|) & \textit{if } \min(|t|,|t'|) = 0 \\ \min \begin{cases} \Delta(t_{[1,|t|-1]}, t') + \texttt{ins} \\ \Delta(t, t'_{[1,|t'|-1]}) + \texttt{del} \\ \Delta(t_{[1,|t|-1]}, t'_{[1,|t'|-1]}) + \begin{cases} 0 & \textit{if } t^l.ea = t'^l.ea \\ \texttt{sub} & \textit{otherwise} \end{cases} \\ \Delta(t_{[1,|t|-1]}, t'_{[1,|t'|-1]}) + \texttt{tran} \\ \qquad \textit{if } t^l.ea = t'_{|t|-1}.ea \wedge t_{|t|-1}.ea = t'^l.ea \end{cases} \end{cases}$$

In Sect. 3 this definition is extended to analyse activity and temporal behaviour at once, cf. Def. 7. Auxiliary functions $\max(a,b)$ *and* $\min(a,b)$ *determine and return the maximum (minimum, resp.) value in $\{a,b\}$.*

Assume that for the event $e_3 = (\texttt{E}, 37)$ in trace t_1 (i.e., $t_1.e_3$) the directly successive next event should be predicted, cf. Table 1. For this task we assume, e.g., that the subtraces $t_{2[1,3]}$ and $t_{3[1,3]}$ are relevant information sources while $t_{4[1,3]}$ is not. This is because we assume that $t_{4[1,3]}$'s execution behaviour (e.g., given by the order and occurrence of the respective event activities) is too different from $t_{1[1,3]}$ to draw, based on $t_{4[1,3]}$, conclusions on $t_1.e_3$'s next event. The DL distance reflects this assumption, i.e., the DL distance of $t_{1[1,3]}$ and $t_{2[1,3]} = 1$, $t_{3[1,3]} = 1$, and $t_{4[1,3]} = 3$; assuming a general edit cost of one, cf. Def 2 and [10].

Further, the DL can naturally be applied on discrete values, such as, event activity labels, cf. Def. 2 and [12]. However, this work also takes values into account which origins from a continuous data range, such as, the timestamp $et \in \mathbb{R}_{>0}$ of each event. For this the original DL approach is extended into a two step approach. Hereby, the first step follows the original idea of exact equality between event activity labels (i.e., $e.ea$) while the second step factors in the partial similarity of an events' temporal behaviour, cf. Def. 7. Without this extension very similar traces could be classified as completely dissimilar, solely because of minimal temporal fluctuations, which we found to be likely, cf. [5].

Subsequently, the $ms \in \mathbb{N}_{\geq 1}$ *most similar* traces $MS \subseteq L$ are transformed into a probability based prediction model M ②. Hereby, the key idea is that the most probable behaviour, based on the most relevant traces, should become the predicted behaviour. To implement this key idea the proposed approach:

1) searches $\forall t \in MS$ the events $er \in t$ which are most representative for the last known (i.e., most recently occurred) event in p (i.e., p^l), cf. Def. 2; to
2) extract the directly successive event of each event er as a potential representation of probable successive behaviour for p^l, cf. Def. 3; and
3) stores extracted information in M, which is inspired by weighted histograms, cf. Def. 4. The weights represent the relevance of extracted behaviour based on the similarity between p and the traces the behaviour was extracted from.

Definition 3 (Prediction behaviour extraction). *Let $p \notin L$ be a trace for which the* `next event` *should be predicted. Let further $MS \subseteq L$ hold historic traces ($t \in MS$, resp.) which were found to be* `similar` *to p. Finally, for each trace t the events with an index in the range of $[|p|-|p|\cdot s, |p|+|p|\cdot s]$ are analysed. Hereby, $s \in \mathbb{R}$ controls which indexes "*`around`*" $|p|$ are taken into consideration. Behaviour extraction function $ext(p, MS, s) \mapsto L$ extracts (sub) traces by:*

$$ext(p, MS, s) := \{t_{[1,i+1]} \in MS | i \geq (|p|-|p|\cdot s) \wedge i \leq (|p|+|p|\cdot s) \wedge t_i.ea = p^l.ea\}$$

Taking the indexes into account enables to represent our assumption that there is a correlation between the position of an event in a trace and its successive events. For example, it was observed that for a given process typically a correlation between the number of already traversed loop iterations and the likelihood that another iteration occurs (or not) can be found. Accordingly, the number of iterations (roughly represented by the event index) also has an impact on the to be predicted successive events. We factor this observation in by focusing on events which have a similar *index* than the last event in p (i.e., $|p|$).

Assume that for $t_1.e_3 = (\mathtt{E}, 37)$, i.e., $p = t_{1[1,3]}$, cf. Table 1, the successive activity should be predicted while $MS = \{t_2, t_3\}$. For this, a set of all possible subtraces $PS := ext(p, MS, s)$ is extracted from the traces in MS for which the same activity, as given in $t_1.e_3.ea = \mathtt{E}$, occurs roughly at the *same index* as the last event in p ($p^l.ea = \mathtt{E}$) so $MS = \{\langle(\mathtt{A}, 49), (\mathtt{E}, 54), (\mathtt{F}, 61)\rangle, \langle(\mathtt{A}, 49), (\mathtt{E}, 54), (\mathtt{F}, 61), (\mathtt{E}, 68), (\mathtt{B}, 69)\rangle, \langle(\mathtt{A}, 40), (\mathtt{F}, 45), (\mathtt{E}, 49), (\mathtt{F}, 51)\rangle\}$ when $s = 0.\dot{3}$, cf. Def. 3.

From the subtraces given in PS the prediction model M is formed, cf. Def. 4. For this, the last two events (t^l and $t_{|t|-1}$) of each (sub) trace $t \in PS$ are extracted and its `weight` is determined by its reciprocal DL distance to p, so:

Definition 4 (Weighted prediction model). *Let PS hold subtraces from L which were identified as* `relevant behaviour` *sources because of their similarity to p to form the prediction model $M := \{(t^l, t_{|t|-1}, 1/(\Delta(t,p)+1)) | t \in PS\}$ Hereby, for each $m \in M$; $m := (e_1, e_2, w)$ holds* `two events` *e_1 and e_2 and a* `weight` *$w \in \mathbb{R}_{>0}$ representing the subtrace similarity based* `relevance` *of m for the current prediction task at hand, such as, activity or timestamp prediction.*

Subsequently, M is used to predict event activities, cf. Def. 5, and timestamps, cf. Def. 8 ③. For the sake of understanding solely the prediction of activities is described here while the timestamp prediction is given in Sect. 3.4.

Definition 5 (Predicting activities). *Let M be extracted relevant behaviour (i.e., the prediction model), cf. Def. 4. Prediction function $pa(M)$ predicts the most probable activity to be executed* **next** *for p (after p^l resp.) by $pa(M) \mapsto ea$:*

$$pa(M) := \{v|(v, \cdot, \cdot) \in M, \forall(v', \cdot, \cdot) \in M; sa(M, v) \geq sa(M, v')\}^0.ea$$

hereby $sa(M, v) := \sum_{m \in M} m.w$ where $m.e_1.ea = v.ea$, i.e., $sa(M, v)$ sums up the weights in M for a given event v based on the events' activity $v.ea$. This enables the identification of the `most probable` activity to be executed next.

For example, when predicting the successive event for $t_1.e_3 = (\mathtt{E}, 37)$ then $M = \{(\mathtt{F}, \cdot, \mathtt{0.5}), (\mathtt{B}, \cdot, \mathtt{0.\dot{3}}), (\mathtt{F}, \cdot, \mathtt{0.\dot{3}})\}$; this prediction model is visualised in Fig. 2. Based on that model M, $pa(M) = \mathtt{F}$ as the summed up weight for F is $\mathtt{0.8\dot{3}}$. In comparison the second most probable activity B only achieves a summed up weight of $\mathtt{0.\dot{3}}$ – cf. running example in Table 1. Here, in Section 2, we have outlined the proposed event activity prediction approach; in Section 3 it is extended to predict temporal behaviour (event timestamps).

3 Probability based Predictive Temporal Monitoring

This section gives additional details on the approach set out in Fig. 1. It focuses on the prediction of *temporal behaviour* (i.e., p's next event timestamp). Note, that the prediction of the next events' activity was already outlined in Sect. 2.

3.1 Applying Intervals to Analyse Continuous Variables

The similarity calculation and prediction approach proposed in Sect. 2 can naturally be applied to *discrete values*, such as, activities (labels, resp.). However, to apply them to values which origins from a *continuous data* range, such as, timestamps or timespans, they must be extended to prevent the generation of largely incorrect prediction results: similar temporal behaviour would be recognised as dissimilar due to minor temporal *fluctuations*. For this, we propose to represent *continuous values* as *intervals*, cf. [4]. This increases the flexibility as slightly varying temporal business process execution behaviour is still recognised as similar as, for example, $t_1.e_1.et = 1, t_2.e_1.et = 3$ both fit in the interval $[0, 4]$.

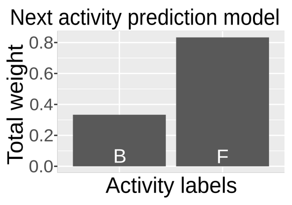

Fig. 2. Exemplary, weighted histogram based visualisation of the prediction model M

In the following the temporal process execution behaviour in L (to define intervals and perform predictions) is represented as *timespans*. Here, such timespans refer to the time which has passed between two directly successive execution event observations. This enables to predict the most probable timespan between the last known event (i.e. p^l) and the time of execution of the to be predicted next/successive process execution event activity

– which can subsequently be mapped on p's next event execution timestamp while not being affected by fluctuations in the specific event execution times. For example, the timespan between $t_1.e_5 \to t_1.e_6$ and $t_3.e_5 \to t_3.e_6$ is equal (i.e., $\texttt{6}$) while the individual event timestamps are different (e.g., $t_1.e_5.et = \texttt{47}$, $t_3.e_5.et = \texttt{57}$), cf. Table 1.

Definition 6 (Timespan extraction). *Let L be a set of execution traces and a, a' two activities for which all timespans should be extracted from L. Extraction function $ate(a, a', L) \mapsto \{d_1, \cdots, d_n\}$ extracts $\forall t \in L$ the timespans ($d \in \mathbb{R}_{\geq 0}$) between directly successive executions of the activities a, a' as a multiset:*

$$ate(a, a', L) := \{|e.et - e'.et||t \in L; e, e' \in t; e.ea = a \wedge e'.ea = a' \wedge e = \bullet e'\}$$

The timespan extraction starts by selecting a pair of activities (i.e., a and a'). Subsequently all traces in L are searched for directly successive events (i.e., e_i, e'_{i+1}) where $e.ea = a$ and $e'.ea = a'$. Finally the timespan d between e, e' is calculated by executing $d = |e.et - e'.et|$, cf. Def. 6. Accordingly, for the running example in Table 1: $ate(\texttt{A}, \texttt{E}, L) = \{\texttt{6}, \texttt{5}\}$ (from t_1 and t_2) when $a = \texttt{A}$ and $a' = \texttt{E}$.

To determine the size and amount of *intervals* required to represent the extracted timespans the *Freedman-Diaconis rule* [4] is utilised. It determines, for a given list of timespans X, i.e., $X = ate(a, a', L)$, a suitable interval size: $int(X) := 2 \cdot (IQR(X)/\sqrt[3]{|X|})$ where $IQR(X)$ is the interquartile range for X.

Based on the prerequisites given in Def. 6 and the Freedman-Diaconis rule multiple auxiliary functions for temporal behaviour are defined. These functions are applied, in the following, when predicting the most probable timespan which has to pass after the timestamp $p^l.et$ till the next execution event can be observed, cf. Sect. 3.3 (i.e., enabling the prediction of the next events' timestamp).

First, $bc(a, a', L) := \lceil (max(X) - min(X))/int(X) \rceil$ where $X := ate(a, a', L)$. It determines the number of intervals the timespans between two successive activities a, a' can be divided in – based on the behaviour in L. Secondly, $bi(e, e', L) := \{i | i = 0, \cdots, bc(e.ea, e'.ea, L); mi(X, i) < d; mx(X, i) \geq d\}^0$ where $d = |e.et - e'.et|, X := ate(e.ea, e'.ea, L), mi(X, i) = min(X) + int(X) \cdot i$ and $mx(X, i) = mi(X, i) + int(X)$. It determines how many intervals i must be summed up to cover the timespan d between the directly successive events e, e'.

Finally, $bt(e, e', L) := \lceil min(X) + int(X) \cdot bi(e, e', L) + int(X)/2 \rceil$ maps timespans/intervals which are related to the events e and e' (based on L) on a single value based on the interval size identified by $int(X)$. This is utilised in Section 3.3, for example, to determine if given pairs of events have equal/similar temporal behaviour. Further, $bt(e, e', L)$ is applied when determining the most propable timespan between p^l an the to be predicted next execution event occurrence.

Given the example traces in Table 1 the auxiliary functions would behave as follows: $bc(\texttt{F}, \texttt{E}, L) = 2$, i.e., the Freedman-Diaconis rule determines that two intervals (here: $[4, 6.4]$ and $[6.4, 8.8]$) are required to cover the timespan between the activities $\texttt{F} \to \texttt{E}$ for L's traces. Hereby, $X = \{7, 7, 4, 6, 7\}$ and $int(X)$ becomes 2.4. Accordingly, the first interval always starts at $min(X)$ and has a size of $int(X)$. Subsequent intervals always start at the end of the previous one. Additional intervals, if necessary, are generated till all timespans in X are covered.

In this example $bi(t_1.e_4, t_1.e_5, L) = 1$, i.e., the timespan between $t_1.e_4 \rightarrow t_1.e_5$ is covered by the second interval (which is $[6.4, 8.8]$) as the timespan between both events is 7. Finally, $bt(t_1.e_4, t_1.e_5, L) = \lceil 4 + 2.4 \cdot 1 + 2.4/2 \rceil = \lceil 7.6 \rceil$. Based on these auxiliary functions the Damerau-Levenshtein distance metric (DL), cf. Def. 2, is extended to incorporate temporal process execution behaviour.

3.2 Temporal Behaviour based Trace Similarity

Sect. 2 argues that the relevance of historical execution traces $t \in L$, for the prediction of future events, for a given incomplete trace p, is related to the similarity between p and L's traces. So, Sect. 2 applies the DL distance to measure activity focused trace similarities. However, the *unaltered* DL algorithm is too *sensible* to be applied on continuous data, such as, timestamps or timespans. This is because temporal behaviour is frequently fluctuating, e.g., the timespan between two activity executions is sometimes a bit shorter or longer. Such fluctuations would result in determining similar execution behaviour (traces) as completely dissimilar. So, we propose to extend the DL algorithm to address this limitation.

Definition 7 (Extended Damerau-Levenstein operation cost). *Let e, e' be `two events` in L's traces. Further, let $c \in \mathbb{N}_{>0}$ be the `cost` assigned to a chosen DL `edit operation`, such as, `ins`. The individual operation cost, taking the temporal differences into account for e, e', is calculated by $tc(c, e, e') \mapsto \mathbb{R}$:*

$$tc(c,e,e') := \begin{cases} c & \text{if } eqa(e,e') = \texttt{false} \\ co(c,e,e',L) & \text{if } eqa(e,e') = \texttt{true} \wedge eqt(e,e',L) = \texttt{false} \\ 0 & \text{if } eqa(e,e') = \texttt{true} \wedge eqt(e,e',L) = \texttt{true} \end{cases}$$

where $eqa(e,e') := e.ea = e'.ea \wedge \bullet e.ea = \bullet e'.ea$ and $eqt(e,e',L) := bi(e, \bullet e, L) = bi(e', \bullet e', L)$ determine if the activity ($eqa(e,e')$) or timespan interval ($eqt(e,e',L)$) of e, e' and their directly preceding events are equal. Further $co(c,e,e',L) := c \cdot (1 - (|bi(e, \bullet e, L) - bi(e', \bullet e', L)|)/bc(e.ea, e'.ea, L))$ calculates the relative edit cost if $eqa(e,e') =$ `true` and $eqt(e,e',L) =$ `false`. The latter is the case if the activities represented by both events are equal but the temporal behaviour is not.

The proposed extension of the DL algorithm, cf. Def. 2 and Def. 7, replaces the cost variables `ins`, `del`, `sub`, `tran` with a cost function $tc(c,e,e')$; where $c \in \{\texttt{ins}, \texttt{del}, \texttt{sub}, \texttt{tran}\}$ represent the configured costs and e, e' represent two events to be compared. Hereby three scenarios can emerge: 1) unequal event activity: full cost (i.e., c); 2) equal activity, dissimilar temporal behaviour: fraction of c, relative to the temporal dissimilarity; 3) equal activity and timespan interval: cost = 0. The following examples are based on the running example given in Table 1 and cover all three scenarios given above with exemplary event pairs:

1) **Unequal event activity**: e.g., $t_1.e_1 = (\texttt{A}, 23)$ and $t_2.e_2 = (\texttt{E}, 54)$, cost $= c$;
2) **Equal activity, dissimilar temporal behaviour**: $t_1.e_5 = (\texttt{E}, 47)$ and $t_3.e_3 = (\texttt{E}, 49)$ both cover the same activity (i.e., equal activity label, $t_1.e_5.ea =$

$t_3.e_3.ea$). Accordingly, the timespan d between these two events ($t_1.e_5$ and $t_3.e_3$) and their relative directly preceding event (i.e., $t_1.e_4$ and $t_3.e_2$) is analysed to take temporal differences into account: for the activity transition $F \to E$ five timespans can be extracted from L, such that, $ate(\mathtt{F}, \mathtt{E}, L) = \{7, 7, 4, 6, 7\} = X$. For this $int(X) = 2.4$, such that, two timespan intervals become relevant, first, $[4, 6.4]$ representing $t_3.e_3$ and, secondly, $[6.4, 8.8]$ representing $t_1.e_5$. So both events are represented by different adjacent intervals, such that, the cost becomes $c \cdot (1 - (|0 - 1|)/2) = c \cdot 0.5$; and finally

3) **Equal activity and temporal behaviour**: e.g., $t_1.e_6 = (\mathtt{D}, \mathtt{53})$ and $t_3.e_6 = (\mathtt{D}, \mathtt{63})$. For these events the activity and the transition timespan (i.e., 6) interval from the preceding events is equal, i.e., the dissimilarity/edit cost=0.

3.3 Temporal Behaviour: Predicting Event Timestamps

Predicting *timestamps* of upcoming/next events follows the *same* key idea as the prediction of upcoming event *activities*, cf. Def. 5. However, instead of directly predicting the most probable next event execution timestamp an indirect approach is applied. So, the proposed approach predicts the most probable timespan between the last known activity execution event in p (i.e., p^l) and the most probable occurrence of the "to be predicted" next event. For this, initially, comparable to the activity prediction, the model M, which holds the most relevant behaviour for the current prediction task, is formed, cf. Sect. 2 and Def. 8.

Definition 8 (Predicting timespans). *Let M be extracted relevant behaviour in L where $m \in M$; $m := (e_1, e_2, w)$, cf. Def. 5. Here the weight $w \in \mathbb{R}_{>0}$ is calculated by using the DL algorithm given in Def. 2 and its extension given in Def. 7, i.e., the proposed cost function; which enables to take temporal dissimilarity into account. Further, let $mwp \in \mathbb{R}_{>0}$ control the minimum relevant relative weight to handle highly fluctuating temporal behaviour. Finally, function $pt(M, mwp) \mapsto \mathbb{R}_{>0}$ predicts the most probable timespan, which is expected to pass after $p^l.et$, till p's next execution event (activity execution) will be observed:*

$$pt(M, mwp) := \left\lceil \frac{\sum_{i=0,(at,\cdot,o)\in SF}^{o} at}{\sum_{(\cdot,\cdot,o)\in SF} o} \right\rceil$$

assuming $MF := \{(e_1, e_2, w) \in M | e_2.ea = pa(M)\}$ filters M based on the activity prediced by $pa(M)$ and $gt(m) = bt(m.e_1, m.e_2, L)$ determines an average timespan for m; $MFF(at) := \{m \in MF | gt(m) = at\}$ filters MF based on the average transition timespan. Further, $S := \{(at, g, o) | m \in MF; at := gt(m), o := |MFF(at)|, g := \sum_{m \in MFF(at)} m.w\}$ maps triplets in MF onto average interval driven timespans along with the relevant metadata. Finally, $minW(S, mwp) := \{s.g | s \in S; \forall s' \in S; s.g \geq s.g'\}^0 \cdot mwp$ determines the minimal relevant weight and $SF := (s \in S | s.g \geq minW(S, mwp))$ filters S accordingly.

We found that, the behaviour hold by M can further be focused/filtered. For this the next activity $a = pa(M)$, cf. Def. 5, can be predicted and utilised to remove all triples from $m \in M$ where $m.e_2.ea \neq a$ (such that, M becomes MF). Hereby, the data is further condensed to only hold the most relevant behaviour which is related to the most probable next activity execution (optional step).

Subsequently, all $m \in MF$ are condensed to form triplets $(at, g, o) \in S$. Hereby a single triplet in S can represent one or more entries in M. In each triplet at identifies the relevant average timespan given by $bt(m.e_1, m.e_2, L)$, $g \in \mathbb{R}_{>0}$ represents the summed up weights (cf., $m.w$), and $o \in \mathbb{N}_{>0}$ represents the number of entries $m \in MF$ which were condensed to form this triplet in S. Note, all entries in $m \in M$ which result in the same $at := bt(m.e_1, m.e_2, L)$ are mapped on the same triplet in S. Finally, the triplets in S are utilised to predict the timespan till p's next execution event will most probably be observed.

For this, all $s \in S$ are filtered to identify the ones which have a $s.g \geq minW(S, mwp)$ where $mwp \in \mathbb{R}_{>0}$. Here, the user chosen mwp enables to handle situations were multiple timespans have an equal or close probability. For example, it was found that when timespans are heavily fluctuating (e.g., from minutes to days) between successive events in L then the Freedman-Diaconis rule does not choose the interval sizes perfectly. In such cases a large amount of observations is assigned, for example, in two adjacent intervals which would, if only the most probable timespan is determined, result in large prediction errors.

The use of mwp enables to take this into account. So, *not* the single most probable timespan is used but the average timespan of the mwp "most probable" ones. For this, the average timespans (i.e., at) of the filtered ($SF := (s \in S | s.g \geq minW(S, mwp))$) triplets $(at, g, o) \in SF$ are multiplied by o and summed up. Finally, the resulting summed up timespan is divided by the summed up number of condensed entries (cf. o) of all relevant triples to determine the most suitable average timespan between the last known and the, to be predicted, next event.

Assume that, based on the running example in Table 1, the timespan between $t_1.e_4$ and $t_1.e_5$ ($\texttt{F} \rightarrow \texttt{E}$) should be predicted (i.e., $p = t_{1[1,4]}$). Assuming that $ms = 4$; all traces t_2, t_3, t_4 are relevant. Accordingly, $M = \{((\texttt{F}, \texttt{61}), (\texttt{E}, \texttt{68}), 0.289), ((\texttt{F}, \texttt{45}), (\texttt{E}, \texttt{49}), 0.227), ((\texttt{F}, \texttt{51}), (\texttt{E}, \texttt{57}), 0.217), ((\texttt{F}, \texttt{30}), (\texttt{E}, \texttt{37}), 0.2)\}$ when $s = 1$ and a general DL edit cost of 1 is assumed (following related raw DL distances were calculated: $2.4\dot{6}, 3.4, 3.59, 4$). Hereby, the first triplet, is motivated by t_2, the last triplet from t_4 and the remaining ones represent behaviour from t_3.

In this example, $MF = M$, i.e, as $pa(M) = \texttt{E}$ filtering M has no effect. Subsequently, $S = \{(5.2595, 0.444, 2), (7.7785, 0.489, 2)\}$ as two intervals ($[4, 6.519]$ and $[6.519, 9.038]$) need to be created when mapping the triplets from MF; $int(\{7, 4, 6, 7\}) = 2.519$. When assuming $mwp = 0.9$ then $minW(S, mwp) = 0.4401$ (i.e., $0.489 \cdot 0.9$) so that both triplets in S are identified as being relevant, such that $S = SF$. Accordingly $pt(M, mwp) = \lceil (5.2595 + 5.2595 + 7.7785 + 7.7785)/(2 + 2) \rceil = 7$ which matches to the observed timespan for $t_1.e_4 \rightarrow t_1.e_5$.

3.4 Predicting future Execution Events

Predicting the next execution event $en := (ea, et)$, cf. Def. 1, for an incomplete execution trace p requires to predict the events' activity $en.ea$ and timestamp $en.et$. [27] found that both are interdependent. Accordingly, this work initially predicted, based on Def. 5, the next activity $en.ea$. Subsequently $en.ea$ can be applied when prediction the timespan $d \in \mathbb{R}_{\geq 0}$ between the last known event p^l and $en.et$. Hence, for predicting $en.et$ the most probable timespan d between $p^l.et$ and $en.et$ is predicted, cf. Def. 8, and added to $p^l.et$ to get $en.et := p^l.et + d$.

3.5 Fostering Understandability and Trust

Recent predictive monitoring approaches are commonly applying *blackbox* like prediction techniques, such as, neural networks, see Sect. 5. For those approaches typically an outstanding prediction performance (e.g., a high activity prediction accuracy) is reported. However, simultaneously they fall short when required to:

1) explain *alternative* futures which were not classified as being most probable;
2) explain the aspects which *motivated* the specific prediction results; or
3) understand how and why a prediction result would most probably *change* when adapting the prediction logic, configuration, or model in certain ways.

We assume, that these drawbacks limit the acceptance and applicability in the respective predictive monitoring target group (e.g., management or production planning staff). For example, when only providing the most probable futures but not providing any alternatives expert knowledge can hardly be incorporated into the prediction. However, based on expert knowledge a user can decide, e.g., that not the most probable, but the second or third most probable activity will most probably be observed next, potentially, because of some external factors which cannot (or not yet) be grasped by the prediction algorithms, cf. [19].

Further, providing information also about less probable futures enables to perform random walks, cf. [14]. Those enable, for example, to depict and clarify how multiple potential futures will unfold – enabling users to build trust into the results and quickly grasp the related uncertainty and potential developments. For this, we assume, that the low number of simple configuration parameters, applied here, is advantageous, i.e., it gives the users a simple "knob" to play with and explore different configurations, predictions, and futures quickly, cf. [19].

The proposed approach can support users during result and prediction process interpretation based on weighted histogram like visualisations which can directly be derived from the generated and utilised weighted prediction models M and MF, cf. Fig. 2. We assume that this enables users to grasp, but also learn, for example, how a (potential) configuration change has affected the performed prediction steps and results – fostering trust and understandability, cf. [28]. However, the question remains if the listed limitations of blackbox like prediction work may not be acceptable given the overall prediction performance of such neural network based techniques. To address this question, the following section will evaluate the feasibility of the proposed approach and compare it with multiple state-of-the-art predictive monitoring approaches.

4 Evaluation

The evaluation utilises real life process execution logs from multiple domains in order to assess the prediction quality and feasibility of the proposed approach,

namely: BPI Challenge 2012[2] (BPIC) and Helpdesk[3]. Both logs are also utilised by existing state-of-the-art approaches, such as, [27] – enabling to compare the proposed approach with multiple alternative approaches: [1, 16, 6, 27]. Hereby, we especially focus on incorporating neural network based prediction approaches as those are generally assumed as outperforming probability based ones, such as, the one proposed in this work, cf. [27]. Overall the evaluation, execution logs, and how the evaluation is carried out is *similar* to [27] to ensure comparability.
BPIC 2012 log: The BPIC 2012 log[2] is provided by the Business Process Intelligence Challenge (BPIC) 2012 in conjunction with a large financial institution. It contains traces generated by the execution of a finance product application process. This process consists out of one manually and two automatically executed subprocesses: 1) application state tracking; 2) handling of application related *work items*; and 3) offer state tracking. The comparison approaches, such as, [27] are only interested in the prediction of manually performed events. Accordingly, this and the comparison work narrow down the events to the work items subprocess to ensure comparability. The same motivation resulted in filtering this log to only contain events whose type is defined as *complete*. Overall 9,657 traces with 72,410 execution events were retained for the work items subprocess.
Helpdesk log: This log[3] is provided by an Italian software company and contains execution traces generated by a support-ticket management process. Each trace starts by generating a novel ticket and ends with closing the ticket when the related issues were resolved. Overall the utilised process consists of 9 activities while the log holds 3,804 traces which consist of 13,710 execution events. We assume the helpdesk log as being more *challenging* than the BPIC log. This is because the temporal fluctuation is larger and the number of activities is higher while the amount of traces, from which behaviour can be learned, is lower.
Comparison approaches: The proposed approach is compared with seven alternative prediction approaches described in [1, 16, 6, 27]. Four of those can be categorised as probability based techniques which apply finite state automata or transition systems. The latter can further be subdivided into set, bag, or sequence abstraction – depending on the applied transition system building and abstraction techniques. Three approaches apply neural networks. Either in the form of Recurrent Neural Networks (RNN) – which incorporate feedback channels between the neurons a network is composed of – or Long Short-Term Memory (LSTM) based neural networks. The latter (LSTM) were found to deliver consistent high quality results in a wide range of domains by coupling neural networks with the capabilities to "remember" previous internal states, cf. [27].

4.1 Metrics and Evaluation

This section analyses the feasibility of the presented prediction approach. For this, two metrics are applied to determine and compare the prediction result quality of multiple prediction approaches. First, *Mean Absolute Error* (MAE)

[2] DOI: 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[3] DOI: 10.17632/39bp3vv62t.1

is utilised to analyse the prediction quality for temporal behaviour, i.e., the difference between the observed real event timestamps in the given log traces and the predicted event timestamps. Hereby, MAE was chosen as it is less affected by outliers, where the timespan between two events is unusually large, cf. [27], than other approaches, such as, the Root Mean Square Error. Secondly, for activity prediction the *accuracy* is measured. For this, the percentage of predicted events for which the predicted and the observed event activities are equal is determined.

While performing the evaluation the first 2/3 of the chronologically ordered traces hold by the logs are utilised as training data, e.g., to form prediction models. The remaining 1/3 of the traces are utilised to evaluate the activity and event timestamp prediction performance of the proposed and comparison approaches (testing data). For this, basically, all possible subtraces with $t_{[1,n]}$ where $2 \leq n < |t|$ are generated from the testing data and the $n+1$ event (activity and timestamp) is predicted. Note, that only subtraces with a size of ≥ 2 are used so that sufficient behaviour is known to base the prediction on, cf. [27]. Further, the evaluation results were only calculated once as the proposed approach contains no random aspects, i.e., deterministic results were observed.

4.2 Evaluation Results

The results were generated based on the BPIC 2012 and Helpdesk process execution logs. The implementation was found to execute the required preparatory (e.g., similar trace extraction) and next event prediction steps fast enough to output the predicted events almost instantaneously (i.e., below one second).

It was found that this performance could only be achieved because the proposed approach filters and separates the given traces into traces which are relevant or non-relevant for the prediction task at hand. This significantly reduces the amount of data which must be processed during the main prediction steps. Further, the initial similarity based relevance calculations were found to be executed quickly. Computationally intense temporal behaviour focused calculations and their results can be stored and reused for multiple predictions in a row. This suggests an applicability onto even larger process repositories and execution logs.

Primary tests were applied to identify appropriate configuration values for each log and prediction task – which are summarised in Table 2. This is $ms \in \mathbb{N}_{>0}$, i.e., the amount of most similar traces hold by MS, $s \in \mathbb{R}$, i.e., the index spreading control variable for the prediction behaviour extraction, cf. Def. 3, and, $mwp \in \mathbb{R}$ which enables to configure how less probable temporal behaviour is incorporated to compensate less than ideal interval definitions which can stem from heavily fluctuating temporal execution behaviour, Def. 8. Finally, $ps \in \mathbb{N}_{>0}$ controls the maximum number of events which are taken into account during the similarity calculation by creating and using a subtrace with the length of at most ps events (e.g., $t_{[|p|-ps,|p|]}$) for the similarity based trace relevance analysis.

Different configurations were used for different prediction tasks (activity vs. timestamp) and logs to reflect the unique characteristics of each task and log. Given the low amount of simple numeric configuration values those can, likely, also be automatically optimised and defined based on computer algorithms. In, general, it was observed that choosing an overly high value for the configuration variables ms, s, and mwp could result in being affected by irrelevant behaviour and noise while too low values could result in not extracting sufficient behaviour for the prediction task at hand. In comparison, the value ps seems to mainly affect the amount of computational effort which must be invested (higher=more).

Finally, following edit costs are utilised for the original and the extended DL algorithm: `del` = 2, `sub` = 3, `tran` = 2, `ins` = 1. Hereby, each cost was chosen based on our assumption how strongly the related edit operation (e.g., to delete an event) would affect the effective trace behaviour. For example, *sub* was defined as three as it combines a delete (cost 2) and insert (cost 1) operation at once. In comparison `tran` "solely" moves events towards a new trace index.

Table 2. Evaluation configuration for each execution log and prediction task

Configuration	ms	s	mwp	ps
BPIC Event Activity Prediction	200	0.2	0.05	10
Helpdesk Event Activity Prediction	10	0.1	0.05	10
BPIC Event Timestamp Prediction	50	0.2	0.2	20
Helpdesk Event Timestamp Prediction	200	0.2	0.2	6

The achieved evaluation results are summarised in Table 3 (event timestamp prediction) and 4 (event activity prediction) for all compared approaches (bold = best). As can be seen, the proposed probability based approach consistently outperforms the alternative probability *and* neural network based comparison approaches for all logs. Overall, an average improvement of 5% for the event timestamp and 4% for the event activity prediction can be observed over the best performing comparison approach given in [27]. In general, we found that the observed advantage, of the proposed approach, over the compared approaches is even increasing when the analysed process execution behaviour along with the prediction task becomes more challenging. This indicates that the observed advantage would further increase at more challenging prediction tasks/behaviour.

Further it was found that the time, but also computational effort which is required to prepare and execute the predictions is substantially lower for the proposed approach than for the compared neural network based approaches. For example, the authors in [27] utilise an expensive professional high end NVidia Tesla k80 GPU and still need between "*15 and 90 seconds per training iteration*" [27, p. 483] – of which, typically, tens of thousands are required for a single neural network to achieve reasonable results. Moreover, multiple prediction models (neural networks) must be prepared for each process and sub-trace length, i.e., for $t_{[1,2]}, t_{[1,3]}, t_{[1,4]} \cdots t_{[1,n]}$ where $n \in \mathbb{N}_{>0}$ is the longest expected trace length: if traces become longer than n, approaches, such as, [27] are no longer applicable.

In comparison cheap general purpose processors used in today's office PCs are sufficient for the proposed approach to quickly perform predictions. This enables users to dynamically and quickly explore the impact of different configuration values and the related futures – which we assume as helpful when in need to understand the unfolding of complex process executions. Finally, given the computational performance of the proposed approach it is not necessary to execute lengthy prediction model preparations for individual sub-trace lengths, i.e., there is no predetermined upper limit on the trace lengths which are supported.

Table 3. Evaluation Results: Execution Event Timestamp Prediction MAE

	Mean Absolute Error (MAE) in days					
	Proposed Probability	Set abstraction Probability [1]	Bag abstraction Probability [1]	Sequence abstraction Probability [1]	LSTM Neural Network [27]	Recurring Neural Network [27]
Helpdesk	**3.54**	5.83	5.74	5.67	3.75	3.98
BPIC 2012	**1.49**	1.97	1.97	1.91	1.56	N.A.[4]

Table 4. Evaluation Results: Execution Event Activity Prediction Accuracy

	Activity prediction accuracy				
	Proposed Probability	LSTM Neural Network [16]	Finite automaton Probability [6]	LSTM Neural Network [27]	Recurring Neural Network [27]
Helpdesk	**0.77**	N.A.[4]	N.A.[4]	0.71	0.66
BPIC 2012	**0.77**	0.62	0.72	0.76	N.A.[4]

This evaluation shows the feasibility of the proposed approach. However, due to space restrictions the user focused application benefits, compared to related neural network based approaches, are only discussed, but not yet evaluated, cf. Fig. 2 – which will be done, based on user studies, in future work, cf. Section 6.

5 Related Work

Overall, it was found that existing work mainly addresses four areas: *a*) predicting the next event [27]; *b*) estimating remaining execution times [27]; *c*) classifying and predicting instance outcomes [9]; and *d*) predicting risks which could hinder successful instance completions [2]. Here, we assume *a*) as most relevant.

In general it was observed that early related work was mainly focusing on *probability* based approaches using transition networks, state automata, (Hidden) Markov Models, frequent (sub) sequences, and fuzzy logic, cf. [25, 17, 18, 7]. Later work, mainly starting in 2015, heavily focused on *neural networks*, initially,

[4] Results denoted as "N.A." are not available as the compared/related work does not cover the respective log or prediction task during its respective evaluation.

starting with recurring neural networks (RNN) and later extending RNNs with Long Short Term Memory (LSTM) capabilities, cf. [18, 27, 13, 15]. Extending neural networks with LSTM capabilities enables the neurons, a neural network is composed of, to remember historic internal "states" over arbitrary time intervals, cf. [27], resulting in an improved prediction quality. Overall, recent work, such as, [27] has indicated that neural network based approaches significantly outperform alternative approaches, e.g., the probability based ones. However, this should be reconsidered as the probability based approach given in this work was found to outperform both, i.e., RNN and LTSM based neural networks, cf. Sect. 4. Alternative machine learning techniques, such as, Support Vector Regression [8] or (regression) trees [11], seem to be rarely applied – in comparison.

We assume that the reported advantages, cf. [27], of RNN and LTSM based approaches over alternative approaches origin from the *memory* capabilities of RNN (limited capabilities) and LTSM (extensive capabilities). Hence, LTSM based approaches can factor in a wide range of instant dependent historic states and observations throughout the prediction. In comparison alternative approaches, such as, Markov Chains, heavily focus on the most recently observed event (p^l) during the prediction phase without taking previously observed instant behaviour (e.g., the number of loop iterations) sufficiently into account. Further, existing probability based work was found to apply a *global* prediction approach, namely, incorporating all known historic trace behaviour during each prediction task at hand – even if a majority of the historic traces are unsuitable for this task as they are significantly dissimilar from the instance p which is predicted upon, cf. [22]. Accordingly, the behaviour representation of *previous* alternative probability based approaches is assumed by us as *overly generic and abstract* – resulting in the observed *unsatisfying* prediction performance and quality.

In comparison the proposed approach builds, on the fly, individual prediction models which are tailored specifically for each incoming novel and unique prediction task – exploiting that it requires only a low amount of computational effort to perform the prediction model generation and to execute the required prediction steps. Accordingly, "optimal" prediction models are generated which factoring in the most similar and so most relevant behaviour – enabling so called "local" prediction, cf. [22]. Section 4 shows that this enables to *outperform* existing work in the area of activity and timestamp based predictive monitoring.

The dynamic and fast prediction model generation also enables to react quickly on changes, such as, *concept drift*, as time intense training phases, required by alternative machine learning approaches, such as, neural networks, are no longer necessary. Further, this also enables to apply the proposed approach in dynamic flexible *online* prediction scenarios where the timespan between *a*) when the most recent execution event becomes available; and *b*) the prediction result becomes obsolete (as the process progressed) is small. This becomes relevant, for example, when execution events are constantly streamed by a process execution engine at a steady high pace, e.g., as production facilities and their *ever changing* processes cannot be "halted" till a prediction algorithm has drawn a conclusion.

6 Discussion and Outlook

This paper focuses on two main challenges *a*) to outperform existing state-of-the-art predictive monitoring approaches (next event prediction); while *b*) striving to implement a transparent prediction approach to foster the trust users put into the results. We conclude that this paper was able to meet the first challenge as the conducted evaluation shows that we outperform the second best compared prediction approach (based on LTSM neural networks) by 4 to 5 percent.

With regards to the second challenge we assume that the results generated by predictive monitoring approaches can have a significant impact on an organisation. For example, a prediction result could trigger a reorganisation of staff/ project assignments resulting in the fulfilment or, given incorrect results, the violation of SLA agreements. Accordingly, we argue that prediction processes should be transparent and understandable. One the one hand to foster the trust in the results but also to enable experts to draw informed decisions while factoring in their unique domain knowledge. Accordingly, the proposed approach strives to provide a transparent prediction process which enables users to grasp the prediction model and the possible different prediction results. For this the prediction models are deliberately simple, contain all relevant futures along with their probability, and could, as we assume, be visualised as weighted histograms.

Future work will concentrate on two aspects: *a*) further improving the prediction result quality; and *b*) evaluating if the proposed approach is capable of fostering the trust in the predicting results, e.g., by providing a transparent and understandable prediction result generation process. The first aspect will be tackled by further extending the proposed approach, for example, by combining it with alternative techniques to incorporate their unique advantages. For the latter we will concentrate on expanding and evaluating the proposed approaches' result and prediction process understandability/transparency. Accordingly, visualisation, cf. Fig. 2, and management tools will be created that enables to handle the provided information (e.g., which potential futures are probable) in an interactive manner. Further, user studies will be performed to assess the benefits of the proposed approach on predictive monitoring driven management decisions.
Acknowledgment This work has been funded by the Vienna Science and Technology Fund (WWTF) through project ICT15-072.

References

1. Van der Aalst, W.M., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information systems 36(2), 450–475 (2011)
2. van Beest, N.R., Weber, I.: Behavioral classification of business process executions at runtime. In: Business Process Management. pp. 339–353. Springer (2016)
3. Benítez, J.M., Castro, J.L., Requena, I.: Are artificial neural networks black boxes? Transactions on neural networks 8(5), 1156–1164 (1997)
4. Birgé, L., Rozenholc, Y.: How many bins should be put in a regular histogram. ESAIM: Probability and Statistics 10, 24–45 (2006)

5. Böhmer, K., Rinderle-Ma, S.: Multi instance anomaly detection in business process executions. In: Business Process Management. pp. 77–93. Springer (2017)
6. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Quarterly 40(4), 1009–1034 (2016)
7. Ceci, M., et al.: Completion time and next activity prediction of processes using sequential pattern mining. In: Discovery Science. pp. 49–61. Springer (2014)
8. Cesario, E., Folino, F., Guarascio, M., Pontieri, L.: A cloud-based prediction framework for analyzing business process performances. In: Availability, Reliability, and Security. pp. 63–80. Springer (2016)
9. Conforti, R., et al.: Prism–a predictive risk monitoring approach for business processes. In: Business Process Management. pp. 383–400. Springer (2016)
10. Damerau, F.J.: A technique for computer detection and correction of spelling errors. ACM 7(3), 171–176 (1964)
11. De Leoni, M., et al.: A general framework for correlating business process characteristics. In: Business Process Management. pp. 250–266. Springer (2014)
12. Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I.: Clustering-based predictive process monitoring. Services Computing (2016)
13. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Petrucci, G., Yeshchenko, A.: An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring. In: Business Process Management. pp. 252–268. Springer (2017)
14. Durrett, R.: Probability: theory and examples. Cambridge university press (2010)
15. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems 100, 129–140 (2017)
16. Evermann, J., et al.: A deep learning approach for predicting process behaviour at runtime. In: Business Process Management. pp. 327–338. Springer (2016)
17. Ferilli, S., et al.: Extenaded process models for activity prediction. In: Methodologies for Intelligent Systems. pp. 368–377. Springer (2017)
18. Francescomarino, C.D., et al.: Predictive process monitoring methods: Which one suits me best? In: Business Process Management. pp. 77–93 (2018)
19. Gleicher, M.: Explainers: Expert explorations with crafted projections. Visualization and computer graphics 19(12), 2042–2051 (2013)
20. Greco, G., Guzzo, A., Pontieri, L.: Mining taxonomies of process models. Data & Knowledge Engineering 67(1), 74–102 (2008)
21. Idri, A., Khoshgoftaar, T.M., Abran, A.: Can neural networks be easily interpreted in software cost estimation? In: Fuzzy Systems. vol. 2, pp. 1162–1167. IEEE (2002)
22. Klinkmüller, C., et al.: Towards reliable predictive process monitoring. In: Information Systems in the Big Data Era. pp. 163–181. Springer (2018)
23. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet physics doklady. vol. 10, pp. 707–710 (1966)
24. Mehdiyev, N., et al.: A multi-stage deep learning approach for business process event prediction. In: Business Informatics. vol. 1, pp. 119–128. IEEE (2017)
25. Pandey, S., Nepal, S., Chen, S.: A test-bed for the evaluation of business process prediction techniques. In: Collaborative Computing. pp. 382–391. IEEE (2011)
26. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-markovian stochastic petri nets. Information Systems 54, 1–14 (2015)
27. Tax, N., et al.: Predictive business process monitoring with lstm neural networks. In: Advanced Information Systems Engineering. pp. 477–492. Springer (2017)
28. Verenich, I., Nguyen, H., La Rosa, M., Dumas, M.: White-box prediction of process performance indicators via flow analysis. In: Proceedings of the 2017 International Conference on Software and System Process Pages. pp. 85–94. ACM (2017)