

RESEARCH

Open Access



Hierarchical structuring of bilaterally expanding subtrace patterns for efficient tree-based activity suffix prediction

Simon Rauch^{1,2,5*}, Christian M. M. Frey³, Andrea Maldonado^{4,5}, Daniel Schuster^{1,5}, Gabriel Tavares^{1,5} and Thomas Seidl^{1,5}

*Correspondence:

Simon Rauch

rauch@dbis.fim.uni.de

¹Data Base Systems and Data Mining AI Lab, Ludwig-Maximilians University Munich, Oettingenstr. 67, Munich 80538, Bavaria, Germany

²Division Supply Chain Services, Fraunhofer IIS, Nordostpark 84, Nuremberg 90411, Bavaria, Germany

³Department Computer Science & Artificial Intelligence - Machine Learning Lab, University of Technology, Dr.-Luise-Herzberg-Straße 4, Nuremberg 90461, Bavaria, Germany

⁴School of Engineering and Design, Technical University of Munich, Lise-Meitner-Str. 9, Ottobrunn 85521, Bavaria, Germany

⁵Munich Center for Machine Learning, Ludwig-Maximilians University Munich, Oettingenstr. 67, Munich 80538, Bavaria, Germany

Abstract

Business processes of different domains are potentially prone to high variability, leading to high amounts of uncertainty with respect to future case execution. The field of predictive process monitoring has recognized this fate and developed handling this uncertainty as one of its core concerns for building reliable predictive or prescriptive methods. Over the last decade, deep learning methods have increasingly emerged as a product of this research field and are considered as the preferred approach when it comes to the prediction of next activities or activity suffixes. However, it remains an open question whether deep learning models finally surpass traditional data mining techniques for these tasks. We address this question in our paper by proposing a framework for process event sequence prediction framework which is based on a hierarchical structuring of bilaterally expanding subtraces mined from activity traces which takes their structural relationship and inter-pattern distances into account. The resulting tree structure serves as an efficient alternative approach for currently established deep learning techniques due to its drastically lower model complexity. The hierarchical arrangement can directly be leveraged for forecasting the most probable future activities given the recent trace history. We achieve competitive forecasting results for remaining trace prediction, even surpassing state-of-the-art deep learning approaches on the majority of the analyzed real-world benchmark process event logs while only relying on the available control-flow information.

Keywords Sequential pattern mining, Tree mining, Predictive process monitoring

Introduction

In the field of Predictive Process Monitoring (PPM) one of the main challenges lies in the development of data-aware predictive models and inference tools that are able to deliver meaningful insights (Ceravolo et al. 2024) into the main drivers of future process execution as well as key performance indicators (KPIs) like the scrap rate or throughput times, mainly by employing either model-based approaches or by usage of machine-learning based predictive models (Di Francescomarino and Ghidini 2022). Moreover,

© The Author(s) 2026, modified publication 2026. **Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

it is highly valuable if knowledge about the future execution of a process can be uncovered with a reasonable amount of foresight. Due to the highly complex nature of many business processes, we are required to manage the prevailing uncertainty in these processes to deliver robust recommendations. As a key factor of uncertainty, the occurrence or non-occurrence of certain activities and their order of occurrence inside a process have a high influence and serve as the most basic building block of a business process. Resulting from that, it is undeniably important to incorporate accurate estimations of the future process execution in order to further proceed with inference tasks that are heavily influenced by the manifesting activity sequence like specific positive or negative process outcomes, throughput time prediction or identifying meaningful process treatments. Especially regarding final results of process instances, the need for an as-early-as-possible prediction of the future process and not only for the next few activities gets even clearer.

In the last decade, the research field of PPM experienced an increased amount of contributions from different data analytics domains – especially for the tasks of *Next Activity Prediction* and *Remaining Trace Prediction* as one of the main areas of uncertainty in business process management. Especially in recent years, a large share of these contributions leveraged state-of-the-art techniques from the area of deep learning due to their high popularity and the increased and successful application of its different architectures on relevant prediction tasks from other fields of research.

Contrary to the ongoing trend in PPM of applying and enhancing different deep learning tasks for tasks like remaining trace or runtime prediction, we propose that there is still existing potential of more traditional techniques in PPM which has not yet been fully leveraged. To test this claim, we introduce a novel framework based on a hierarchically structureable bilateral expansion of activity subtraces along with a forecasting algorithm that is based on local and global pattern distances in terms of a newly introduced distance measure. The forecasting framework is tailor-made to the task of Remaining Trace Prediction – predictions are generated dynamically along the execution of ongoing cases and are able to flexibly grasp the case history, and also incorporate the current state of the process by incorporating the past sequence length into the decision-making process. The main contributions of this work are as follows:

- We propose a hierarchical tree structure consisting of bilaterally expanding activity subtrace patterns of different sizes
- Our framework leverages a tailor-made prediction algorithm for discrete sequential data that predicts future activities based on conditional pattern occurrence probabilities and acts dynamically with respect to the current sequence length while considering the maximum possible case history
- We achieve promising results on common benchmark event logs and surpass state-of-the-art deep learning models in prediction performance for Remaining Trace Prediction in many cases while only relying on control-flow information

This work is an extension of previous work ‘BEST: Bilaterally Expanding Subtrace Tree for Event Sequence Prediction’ (Rauch et al. 2025) published at the *International Conference on Business Process Management (BPM) 2025* in Seville, Spain. We build on the core concepts and definitions introduced in the original contribution and extend it by:

- A more flexible selection procedure in the main prediction loop that allows for fine-tuning of the approach to specific criteria of individual datasets and extends the potential of BEST for full-scale hyperparameter optimization
- An extended evaluation, including a training and inference duration comparison of our approach with existing deep learning techniques
- Effects of the application of pruning techniques for the tree generation procedure
- A larger body of related work as well as a direct comparison with landmark papers (van der Aalst et al. [2011](#)).

Related work

We embed our approach into existing research on the described topic of activity prediction and analyze works from the domains of deep learning as well as pattern-based approaches. Eventually, we list and shortly discuss differences from a landmark paper for temporal predictions by van der Aalst et al. ([2011](#)).

Deep learning

In the last decade, a surge of contributions leveraging deep learning approaches for the tasks of Next Activity and Remaining Trace / Suffix prediction was apparent. For a longer part of that period, recurrent neural networks (RNNs), specifically long short-term memory (LSTM) networks were the method of choice for activity and temporal predictions with more seldom applications of gated recurrent units (GRU). Among others, the works of Tax et al. ([2017](#)), Camargo et al. ([2019](#)), Evermann et al. ([2017](#)), and Hinkka et al. ([2019](#)) presented and evaluated their deep learning techniques which were tuned for the common PPM tasks of Next Activity Prediction, Remaining Trace Prediction and Next Timestamp as well as Remaining Time Prediction. Key differences that are apparent between those works are mostly bounded to the usage of different architectures and embedding strategies. Through presenting a holistic overview and a comparative benchmark of those contributions, Rama-Maneiro et al. ([2022](#)) thoroughly evaluate their performance for different benchmark datasets and uncover strong suits and weaknesses for the different forecasting tasks. They report a superiority of the approaches of Camargo et al. ([2019](#)) and Evermann et al. ([2017](#)) for the case of predicting the complete remaining activity sequence. Moreover, the approaches of Tax et al. ([2017](#)) and Hinkka et al. ([2019](#)) perform very well for the case of Next Activity Prediction.

Tackling the popularity of LSTM networks in PPM, Wuyts et al. ([2024](#)) recently proposed an encoder-decoder transformer-based neural network structure for suffix prediction. Their sequence-to-sequence trained deep learning architecture is specifically tailored to this task, leading to advantages over the previously discussed approaches. The authors attribute this performance to the novel model architecture which incorporates all available context information along the execution of a process instance. However, their model architecture shows an increased complexity over the also already complex LSTM networks, as the size of the sequence vectors also increases with overall trace length. This comes with an increase in computational cost, making it a resource-intensive approach for complex event logs. In contrast to that, we leverage subtrace patterns up to specified sizes that are matched dynamically with the running case sequence, independent of the overall trace length and are, therefore, not constrained by the maximum trace length of a business process.

Rama-Maneiro et al. (2022) also propose their own suffix prediction approach based on deep reinforcement learning that achieves intelligent learning of sampling strategies. They leverage the powerful proximal policy optimization algorithm and achieve superior results for suffix prediction in comparison to the previous approaches evaluated in their earlier work. However, as the complexity of the deep reinforcement learning as well as the needed training is expected to be even higher than that of all previously discussed approaches, the argument of increased computational cost holds here as well.

Sequential patterns

Regarding other approaches based on frequent patterns, we find existing research regarding various tasks in the broader picture of PPM, Process Mining in general, as well as approaches from areas of research that are not considered to be primarily aimed at developing solutions for the area of Business Process Management (BPM).

Starting from the latter, Yap et al. (2012) build a recommender system for next items of users. They do so by taking their individual user characteristics as well as the recorded item selection behavior. They choose sequential pattern mining as the method to gather these recommendations and mine for important patterns in the historical and user-specific item data. To arrive at the most fitting next item recommendation, they implement frequency-based choice metrics. Due to their focus on user-specific datasets, the general applicability of their approach is not guaranteed for the current state of the domains of BPM or PPM as those disciplines are usually more process-centered and less concerned with user-specific criteria. Shifting our view towards techniques developed in and for BPM, Ceci et al. (2014) present an approach based on tree structures built from different sequential patterns discovered from a process event log. At the core of their model, they leverage the tree structure to train individual C4.5 decision trees and are able to produce iterative forecasts of next activities up until a complete predicted suffix. Key differences to our work are that we leverage the activity and transition frequencies directly in the procedure of mining for the bilaterally expanding subtraces, rather than to rely on the C4.5 entropy-based training procedure of C4.5. However, since the decision trees are also able to represent the temporal perspective, they are able to design a tree-based process duration predictor. Although our and their approach shares similarities in terms of hierarchical tree generation, the fundamental difference lies in the format of the mined patterns, as we introduce bilaterally expanding subtrace patterns and process those patterns differently for our predictions. Their model design exhibits inherent robustness to noisy event logs due to the application of different augmentations of the FAST algorithm for the mining of sequential pattern trees. However, their approach shows drawbacks in terms of complete suffix prediction performance and tends to be memory-intensive due to a potentially large number of nested prediction models when the sequence trees show a higher depth and complexity.

While not concerned directly with frequent sequential pattern mining directly, van der Spoel et al. (2013) showcase different pathfinding algorithms based on activity sequences in an application of clinical pathways to forecast final care products and paths between a patient's current state and probable care products, showing similarities to the prediction of remaining activity traces. Additionally, their approach incorporates an simple estimation of remaining time by querying the average duration between the start and end of the predicted path.

Regarding the mining of hierarchical subtraces, there are already contributions by Diamantini et al. (2016) and Tax et al. (2017), which leverage the hierarchical patterns to discover frequent subprocesses. Despite the shared terminology, we would assign our work to a different field as we are focussing on process prediction rather than using the patterns for discovery tasks as they do. Nonetheless, we borrow their notion of subprocesses or subtraces as a core building block of our approach.

Time and duration prediction

The authors of (van der Aalst et al. 2011) provide a seminal approach that involves discovering a process model and converting it into an *Annotated Transition System* (ATS). This ATS includes time-related annotations such as average sojourn times and estimated remaining times learned from historical process instances. Hence, this work addresses the question of when a case will finish for different paths a process instance might take bound by the discovered process model. Furthermore, the ATS serves as a model-based alternative to modern deep learning models (Tax et al. 2017; Hinkka et al. 2019), which integrate time prediction. Although there are similarities with respect to possible insights into which paths a process might take, the approach contrasts with our method based on subtrace patterns, as our work focuses on predicting future activity sequences rather than timing. We touch upon additional similarities between the approach in the following section [Preliminaries](#).

The ATS from (van der Aalst et al. 2011) was taken up and extended by Polato et al. (2018) to include also additional attributes in a Data-aware Transition System (DATS). While they focus on duration prediction, they also show capabilities of their approach in terms of predicting future activity suffixes under the usage of different sequence abstraction methods. They also incorporate an estimation of the closest state within their sequential predictions. However, while this benefits the approach in terms of robustness to noisy event data, it can potentially lead to looping behavior during inference which can heavily increase runtime until termination as well as decrease the resulting accuracy of the predicted future paths to the prediction target.

Preliminaries

For preliminary concepts, we repeat the introduced key concepts from the original contribution (Rauch et al. 2025). We start with basic notations for business processes (van der Aalst 2022).

Definition 1 (Event/Trace/Event Log) An *event* is defined as a tuple $\epsilon := (c, a, t)$ describing the *case id* c out of a set of case identifiers $\mathcal{C}$, the executed *activity* a from an activity set $\mathcal{A}$ and the recorded *timestamp* t from the temporal domain $\mathcal{T}$. A *trace* σ is an ordered, finite sequence of events $\sigma := \langle \epsilon_1, \dots, \epsilon_{|\sigma|} \rangle$ that belong to the same process instance, i.e., case id. An *Event Log* L is a multiset of traces $\{\sigma_1, \dots, \sigma_{|L|}\}$ that are recorded during the execution of a given business process. The activity sequence S of trace σ is defined as $S(\sigma) := \langle a_1, \dots, a_{|\sigma|} \rangle$. We refer to unique sequences of activities inside an event log as *variants*.

As mentioned in the introduction, we base our approach upon the core building block of subtrace patterns of process traces. We define them like in our original contribution

(Rauch et al. 2025) and follow up with a definition of event prefixes and suffixes of process events, similar to (Rama-Maneiro et al. 2022):

Definition 2 (Subtrace Pattern) Given an activity sequence S , we define a sequential pattern $P \subseteq S$ as an activity subtrace pattern of length l with $0 \leq l \leq |S|$ that preserves the sequential order of the original sequence. The total number of possible subtrace patterns within a sequence is given by $1 + (|S| \cdot (|S| + 1)/2)$.

Definition 3 (l -Prefix/ l -Suffix) An l -prefix ρ_k^l of an activity $a_k^{(i)} \in S(\sigma_i)$ is an activity subsequence of $S(\sigma_i)$ with activities $\langle a_{k-l}^{(i)}, \dots, a_{k-1}^{(i)} \rangle$ that precede $a_k^{(i)}$. An l -suffix ϕ_k^l of an activity $a_k^{(i)} \in S(\sigma_i)$ is an activity subsequence of $S(\sigma_i)$ with activities $\langle a_{k+1}^{(i)}, \dots, a_{k+l}^{(i)} \rangle$ that succeed $a_k^{(i)}$. We refer to the full prefix ρ_k or full suffix ϕ_k of an activity $a_k^{(i)}$ if $l = k - 1$ or $l = |\sigma_i| - k$, respectively.

We aim for bilaterally expanding patterns around a designated source activity $a_{c(P)}$. Identical to the original contribution (Rauch et al. 2025), we refer to it as the *center activity* of a subtrace pattern P , i.e.:

Definition 4 (Center Activity) Given a pattern P , we refer to the activity at index $c(P) = \left\lceil \frac{|P|}{2} \right\rceil$ as the center activity of a pattern P .

Our definition of l -Prefix shows similarities to the *maximal horizon* abstraction function in the work of van der Aalst et al. (2011). However, the original work applies this concept only to the prefix. We extend the application to the suffix as well and will build further upon the concept of scanning the current neighborhood in both directions when describing our approach in detail in the following section.

BEST: Bilaterally Expanding Subtrace Tree

In this section, we introduce the building procedure of the *Bilaterally Expanding Subtrace Tree* (BEST), which we developed for predictions of activity suffixes in our original work (Rauch et al. 2025). A repository containing a reference to an implementation of the method as well as all datasets used and results gathered is available online¹.

We provide the needed theory of our approach in the following sections and reuse definitions that were developed for the corresponding conference paper (Rauch et al. 2025). The proposed framework of BEST is visualized in total in Fig. 1. We touch upon each of its components in the following sections. Specifically, we describe in detail how the tree structure of BEST is generated from event data in sections [Hierarchical structuring of bilaterally expanded subtrace patterns](#) and [Subtrace pattern mining and hierarchical matching](#). Subsequently, the mechanism for the prediction of future activities and complete remaining traces based on the tree generated structure is outlined in section [Sequence prediction based on hierarchical bilateral subtraces](#). The definitions and algorithmic procedures listed in the following sections are mostly kept from our original contribution (Rauch et al. 2025). We add Definition 8 and Eq 3 and 4 as concepts needed for our extension regarding a weighted probability metric for pattern selection. The prediction procedure in Algorithms 2 and 3 is slightly adjusted to account for the added filtering metric.

¹<https://doi.org/10.5281/zenodo.15547095>.

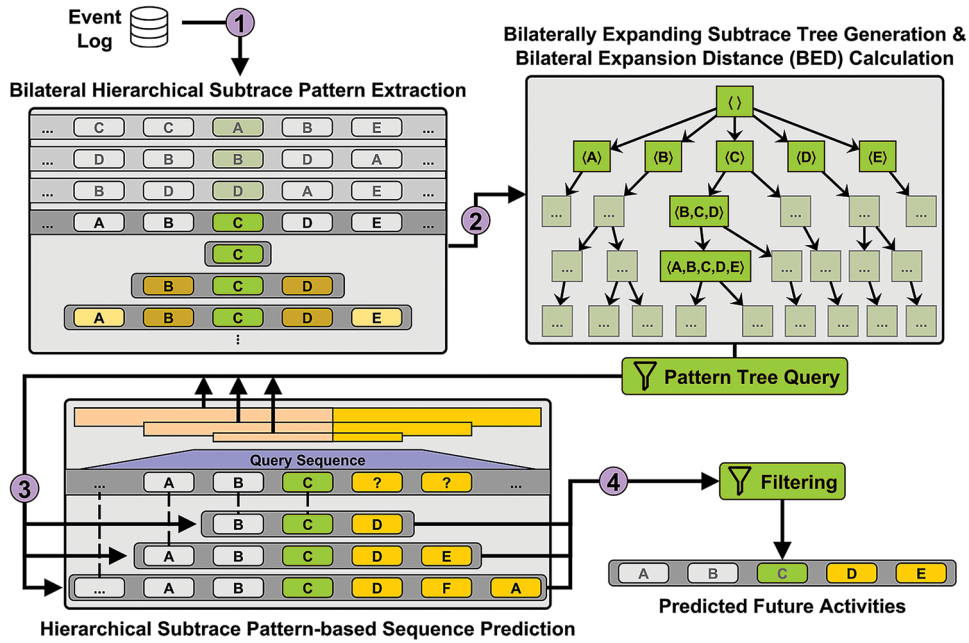


Fig. 1 Hierarchical Sequential Pattern Prediction Framework including the pattern extraction from event data (1) and tree structure generation (2). The remaining trace of a query activity sequence can be predicted by extracting matching patterns from the tree based on the given activity prefix (3) and selecting the fittest pattern through filtering for different criteria (4)

Hierarchical structuring of bilaterally expanded subtrace patterns

We introduce BEST by describing how the mechanism for the pattern expansion around a given center activity works in Definition 5. After gathering all possible extended patterns, the patterns are organized in the tree structure we introduce in Definition 6.

The bilateral pattern extension around the given center activity serves as a means to stepwise gather more contextual control-flow information of the respective trace. We define it as follows:

Definition 5 (Bilateral Pattern Expansion) Given a pattern $P := \langle a_i^{(S)}, \dots, a_{c(P)}^{(S)}, \dots, a_j^{(S)} \rangle \subseteq S$ of a sequence $S := \langle a_1^{(S)}, \dots, a_{|S|}^{(S)} \rangle$, we define an extension by the activities of the l^p -prefix of $a_i^{(S)}$, respectively, the l^s -suffix of $a_j^{(S)}$, where the expanded pattern is defined by $\tilde{P} := \langle a_{i-l^p}^{(S)}, \dots, a_i^{(S)}, \dots, a_{c(P)}^{(S)}, \dots, a_j^{(S)}, \dots, a_{j+l^s}^{(S)} \rangle$. For an equally bilateral expansion, we set $l = l^p = l^s$. In case the expansion exceeds the sequence length boundaries, i.e., $i < l^p$ or $l^s > |S| - i$, we fill the expansion with dummy activity tokens 'START' and 'END', respectively.

For the remainder of our paper, we perform equally bilateral pattern expansions by their respective l -prefix and l -suffix for $l = 1$, extending a pattern in both directions by one additional activity per extension step.

The gathered subtrace patterns of increasing sizes are subsequently organized in the hierarchical tree structure of BEST to allow for efficient Next Activity and Remaining Trace Prediction. We define the *Bilaterally Expanding Subtrace Tree* (BEST) as follows:

Definition 6 Let $BEST := (V, E, \omega)$ be a weighted tree with $\omega : \mathbb{R} \rightarrow E$ denoting a weighting function for edges and $\nu : \mathcal{P} \rightarrow V$ for mapping patterns to the nodes V within the tree. The tree is fully described by the following:

1. The root node on level 0 of a BEST denotes an empty pattern, i.e., $\nu(\emptyset) =: \text{root}(BEST)$.
2. The root node is connected to singleton patterns on level 1, i.e., to all patterns holding a single activity over all sequences in the event log defining the central activities for respective branches in the tree.
3. Given a node $\nu(P_{pa})$ on the i -level, we apply a bilateral pattern expansion according to Def. 5 over all sequences in the event log yielding a set of child nodes $\{P^{S_1}, \dots, P^{S_\varepsilon}\}$ for the $(i + 1)$ -th level, where for a parent-child relationship in the tree, i.e., for the edge $(\nu(P_{pa}), \nu(P_{ch}))$, it holds by construction that $P_{pa} \subset P_{ch}$ with $a_{c(P_{pa})}^{(P_{pa})} = a_{c(P_{ch})}^{(P_{ch})}$.
4. By construction, each level d within the tree contains patterns of length $|P^d| = (d - 1) \cdot l^p + 1 + (d - 1) \cdot l^s$.
5. We define the weighting function ω as the bilateral expansion distance between a pattern P_{pa} and its child pattern P_{ch} incorporating conditional subtrace pattern probabilities and differences in pattern sizes (see Defs. 7 and 9).

Simply put, a parent-child relationship of two subtrace patterns in our tree captures the identical activity at their respective centers. Tree nodes at deeper levels contain incrementally expanding subtraces that emerge from the same context in control-flow that is expanded into the respective activity's history and future the deeper we traverse into the tree.

Definition 7 (Conditional Subtrace Pattern Probability) We define the conditional probability $Pr(P_{ch}|P_{pa})$ as the probability of the child pattern P_{ch} emerging from the parent pattern P_{pa} given the set of sequences $\mathcal{S}$. The probability is calculated by dividing the overall pattern occurrence count of the child pattern by the overall pattern occurrence count of the parent pattern

$$Pr(P_{ch}|P_{pa}) = \frac{|\{P|P = P_{ch} \forall P \in \mathcal{S}\}|}{|\{P|P = P_{pa} \forall P \in \mathcal{S}\}|}.$$

Definition 8 (Global/Unconditional Subtrace Pattern Probability) We define the global, i.e. unconditional, probability $Pr(P_i)$ as the probability of a subtrace pattern P_i occurring in the set of sequences $\mathcal{S}$. The probability is calculated by simply dividing the overall pattern occurrence count of the pattern by the overall count of all patterns mined from the set of sequences $\mathcal{S}$

$$Pr(P_i) = \frac{|\{P|P = P_i \forall P \in \mathcal{S}\}|}{|\{P|P \in \mathcal{S}\}|}.$$

With the tree generation described from the structural point of view, we need to assign importances to the respective subtrace patterns contained in the tree's nodes to transform the tree into an inference tool about highly probable future activities given a certain activity prefix. Therefore, we introduce a probabilistic distance metric that serves as

a dynamic indicator of the current importance of all possible different subtrace patterns given their hierarchical relationships to other subtrace patterns. The distance measure combines conditional occurrence probabilities of patterns (see Def. 7) with their differences in pattern size and maps both to a single numeric value.

Definition 9 (Bilateral Expansion Distance (BED)) The BED between a pattern P_i^d of hierarchy level d and a child pattern P_j^{d+1} is given by

$$BED(P_i^d, P_j^{d+1}) = \frac{\exp(\eta(|P_j^{d+1}| - |P_i^d|))}{Pr(P_j^{d+1}|P_i^d)}, \text{ with } \eta = \begin{cases} 1 & d = 0 \\ 0.5 & d > 0 \end{cases}. \quad (1)$$

The factor η controls for varying length differences between patterns of different levels as we only have a length difference of one between levels $d = 0$ (empty pattern) and $d = 1$ (singleton activity patterns).

Taking the logarithm of the BED leads to

$$\begin{aligned} \log(BED(P_i^d, P_j^{d+1})) &= \log\left(\frac{\exp(\eta(|P_j^{d+1}| - |P_i^d|))}{Pr(P_j^{d+1}|P_i^d)}\right) \\ &= \eta(|P_j^{d+1}| - |P_i^d|) - \log(Pr(P_j^{d+1}|P_i^d)). \end{aligned} \quad (2)$$

This ensures that distances between levels are additive so that the following property holds for multiple hierarchically connected patterns $P_i, \dots, P_n$ of corresponding sequentially adjacent levels $D_i, \dots, D_n$:

$$\log(BED(P_i^{D_i}, P_n^{D_n})) = \sum_i^{n-1} \log(BED(P_i^{D_i}, P_{i+1}^{D_{i+1}}))$$

Having defined this probabilistic relationship, we can assign local pattern distances between child patterns and their respective parent pattern and can calculate distances of adjacent levels as well as distances ranging over multiple levels in a more global sense. We can observe from the formulation of the distance metric, that local pattern distances calculated between adjacent levels are mainly influenced by the log conditional probability of the child pattern (see Definition 7) as the first term of Eq. 2 accounting for the length differences always equals one. For calculating the global distance, i.e., the distance of a pattern to the empty pattern, we can leverage the logarithmic formulation of the BED. The global distance can simply be calculated as the cumulative sum of all

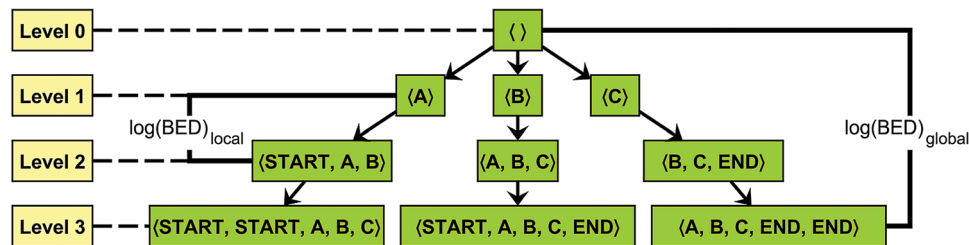


Fig. 2 Hierarchical bilaterally expanding subtrace tree for a single trace event $\log L = \{\sigma_{ABC}\}$ with $S(\sigma_{ABC}) = \langle A, B, C \rangle$. Also schematically depicted are the local edge weights in terms of the logarithmic BED and the global logarithmic BED for subtrace pattern $\langle A, B, C, END, END \rangle$

logarithmic local distances of the nodes along the path of traversal between the tree's root node and the pattern in question (see Fig. 2).

Subtrace pattern mining and hierarchical matching

The procedure for mining of the bilaterally expanding subtrace patterns from individual activity sequences S_i from a set of sequences S is performed by a one-by-one traversal of the activities inside the sequences, taking each activity of the sequence as the respective center activity. For each center activity, we can extract patterns of different sizes. The total amount of patterns contained in a set of sequences S for a given pattern size is then given by

$$1 + \sum_{i=0}^{|S|} |S_i| \cdot D_{max}$$

where D_{max} is the maximum tree depth. Having all patterns of the desired sizes extracted from S , we extract the unique subtrace patterns to generate the hierarchical tree structure outlined in the previous section. We arrange the unique nodes hierarchically by honoring their parent-child relationships (see Def. 6).

Since the transition probabilities between activities and, therefore, the hierarchical relationships between subtrace patterns as well as their occurrence and relevance can change across different phases of the process, we also want to account for those dynamic changes in control-flow of the given event log. To recognize those dynamics in control-flow, we introduce the notion of process stages into the model training and prediction procedures. We place the beginning and end of a process stage by the amount of already executed activities for a case. With a simple slicing operation, we separate all process traces into slices of a predefined process stage width (see Fig. 3). The dynamics mentioned previously can then be introduced into the training of BEST by performing one procedure of pattern mining and hierarchical matching for each of the process stages extracted. This results in n BESTs where n is the number of process stages defined with respect to the maximum trace length of the log. The recursive procedure for the tree generation is detailed in Algorithm 1.

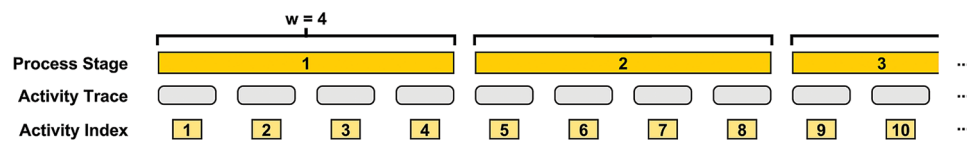


Fig. 3 Process stage slicing of activity traces for process stage width of $w = 4$

Algorithm 1 *buildBEST*(*initP*, $\mathcal{P}$, *d*)

Require: input pattern node *initP* (default: []) to build the tree for, set of occurring subtrace patterns $\mathcal{P}$ inside a process stage *S*, maximum tree depth D_{max}

Ensure: BEST in form of a hierarchical node dictionary

```

1: currentSize = |initP|
2: childPatterns  $\leftarrow$  getChildPatterns(initP,  $\mathcal{P}$ )
3: localBEDs  $\leftarrow$  calcLocalBED(childPatterns, initP)
4: globalBEDs  $\leftarrow$  localBEDs + getLocalBED(initP)
5: globalPrs  $\leftarrow$  getGlobalPr(childPatterns, initP)
6: localPrs  $\leftarrow$  getLocalPrs(childPatterns, initP)
7: children  $\leftarrow$  []
8: if (currentSize - 1)/2 ==  $D_{max}$  then
9:   return {pattern, localBED, globalBED, localPr, globalPr, children}
10: else
11:   for pattern in childPatterns do
12:     children  $\leftarrow$  append(children, buildBEST(pattern,  $\mathcal{P}$ ,  $D_{max} + 1$ ))
13:   end for
14:   return{pattern, localBED, globalBED, localPr, globalPr, children}
15: end if

```

Sequence prediction based on hierarchical bilateral subtraces

In the following, we present the algorithmic procedure to generate future activity predictions for running cases, i.e., a trace where we find at least one ‘START’ and no ‘END’ activity token. For this, we search for fitting patterns inside our generated tree structure like stated in Algorithm 2 aided by a helper function outlined in Algorithm 3.

Algorithm 2 Hierarchical Subtrace Pattern-based Sequence Prediction

Require: input sequence, *S*, maximal process stage *psMax*, filtering method *method* with parameters *params*

Ensure: sequence extended by predicted activities *S*

```

1: while last(S)  $\neq$  END do
2:   psL  $\leftarrow$  psL  $\leftarrow$  psR  $\leftarrow$   $\lfloor |S|/pswidth \rfloor$ 
3:   predP = predictForStage(ps, S, method, params)
4:   while predP = NULL do
5:     predL  $\leftarrow$  predictForStage(psL, S, method, params)
6:     predR  $\leftarrow$  predictForStage(psR, S, method, params)
7:     if getLocalBED(predL) < getLocalBED(predR) then
8:       predP  $\leftarrow$  predL
9:     else
10:      predP  $\leftarrow$  predR
11:    end if
12:    psL  $\leftarrow$  max(psL - 1, 0)
13:    psR  $\leftarrow$  min(psR + 1, psMax)
14:  end while
15:  pred  $\leftarrow$  predP  $\left[ \left( \left\lceil \frac{|predP|}{2} \right\rceil + 1 \right) : |predP| \right]$ 
16:  S = append(S, pred)
17: end while
18: return S

```

Algorithm 3 *predictForStage*(*ps*, *S*, *method*, *params*)

Require: matching patterns from BEST *candidatePatterns* subject to *ps*, *S* and maximum tree traversal depth, filter method *method* and filter params *params*

Ensure: single prediction *predP*

```

1: if |candidatePatterns| = 0 then
2:   return NULL
3: end if
4: if method = "BED" then
5:   predP ← argmin(getLocalBED(candidatePatterns))
6:   if |predP| > 1 then
7:     predP ← argmax(getPatternSize(predP))
8:     if |predP| > 1 then
9:       predP ← argmin(getGlobalBED(predP, ⟨ ⟩))
10:    end if
11:  end if
12: else
13:   if method = "WEIGHTED" then
14:     predP ← argmax(getPrW(candidatePatterns, params))
15:     if |predP| > 1 then
16:       predP ← argmax(getPatternSize(predP))
17:     end if
18:   end if
19: end if
20: if |predP| > 1 then
21:   predP ← randomChoice(predP)
22: end if
23: return predP

```

Algorithm 2 specifies the overall forecasting loop where we are trying to generate a prediction for the current process stage given the current prefix information of the sequence we want to predict for. If no fitting pattern could be found, we iteratively check the trees generated for adjacent process stages for valid predictions. Only if no valid activity prediction could be generated in any of the process stages, we break out of the forecasting loop and return the sequence forecasted until the faulty iteration.

The steps to find and decide a valid activity prediction given the current sequence and respective process stage can be followed in Algorithm 3. In this prediction procedure, we first extract a set of matching patterns from our generated tree structure. Here, we leverage the hierarchical structuring of our tree that has separate branches for each unique activity in the original trace data. Since we want to infer from the prefix information of a pattern to the most likely future activity sequence, we consider a pattern *P* to be matching onto our sequence prefix in question if its activities up to and including the center activity are identical to the last $c(P)$ elements of the query sequence². The separate branches of our tree come in handy here as we only need to search for nodes emerging from the respective center activity we see in our activity prefix. Moreover, the hierarchical structure ensures that the query for matching patterns is stopped for all the descendants of a node where we cannot find a match regarding the historical information of our query sequence. This ensures that our tree-based search for matching patterns we can use for predictions stays efficient as the deeper levels of the tree are only entered in case of a pattern match. From the set of matching patterns, we select the desired

²We show this also schematically in the lower left panel of our framework visualization in Fig. 1 and provide a short written example in the following paragraph

pattern according to a pattern selection strategy to arrive at a chosen predicted pattern. We detail these pattern selection strategies in the following paragraph after providing a minimal example:

Example

Considering a query sequence $S := \langle A, B, C \rangle$, we can gather potential candidate patterns from our generated hierarchical tree with different pattern sizes that contain the most recent activity of the query sequence (C) as their center activity. Matching patterns of length 3 are all patterns from the hierarchical subtrace tree that are structured like $\langle B, C, w_1 \rangle$, where w_1 serves as a wildcard item. Possible patterns could be, e.g., $\langle B, C, D \rangle$, $\langle B, C, E \rangle$ depending on which patterns occur in the event log dataset at hand. Increasing pattern size to 5 delivers matching patterns $\langle A, B, C, w_1, w_2 \rangle$, where w_1 and w_2 serve as wildcard items. Possible patterns of length 5 could then be, e.g., $\langle A, B, C, D, E \rangle$, $\langle A, B, C, D, F \rangle$, $\langle A, B, C, E, F \rangle$. With respect to the designated pattern selection strategy, which is detailed in the following paragraphs, we select one single pattern as the predicted pattern. The predicted activities are then the elements from the center activity onwards. Assuming that we choose the exemplary subtrace pattern $\langle A, B, C, D, E \rangle$, the predicted activities would be D followed by E . Subsequently, we perform the next prediction cycle with the extended query sequence $S' := \langle A, B, C, D, E \rangle$.

During pattern selection, we either apply a strict pattern selection procedure using the introduced local and global BED (see Eq. 2) or employ a more flexible pattern filtering with respect to a weighted probability metric. For the former case, we filter for patterns with the smallest local BED. If we find multiple matching patterns showing the minimum local BED, we refine the set of candidate patterns by choosing those patterns that show the maximum match length, i.e., the longest possible subtrace patterns and – if needed – search further for the patterns with the minimal global BED, i.e., their distance to the empty pattern. For the latter case, we introduce the weighted probability metric Pr_W that is tunable with respect to dataset characteristics regarding a weight parameter $w \in [0, 1]$. It is defined as follows:

$$Pr_W(P_i^d, P_j^{d+1}) = Pr(P_n^d + 1 | P_i^d)^w \cdot Pr(P_n^{d+1})^{(1-w)} \quad (3)$$

$$\log(Pr_W(P_i^d, P_j^{d+1})) = w \cdot \log(Pr(P_n^d + 1 | P_i^d)) + (1 - w) \cdot \log(Pr(P_n^{d+1})) \quad (4)$$

We can use Pr_W either in original form (see Eq. 3) or with logarithmized probabilities (see Eq. 4) leading to the same filtering. With Pr_W , we first select the patterns from the set of matching patterns that show the maximum weighted probability. In case we find multiple patterns with that weighted probability, we filter further for the patterns that show the maximum match length. Regardless of which selection method is chosen, if we still have multiple candidate patterns, we choose randomly between the remaining ones to arrive at a valid prediction (see lines 19–21 of Algorithm 3). Then, the remaining activities subsequent to the center activity of the chosen pattern form the prediction of the respective iteration of Algorithm 2.

The weight parameter has the function to put the focus on either the global pattern probability or the conditional pattern probability. Choosing values close to one puts more emphasis on the conditional probability, choosing values close to zero puts more emphasis on the global pattern occurrence probability, while also a more balanced

probabilistic choice metric is possible with values around 0.5. This aids the overall applicability of our approach for various domains and types of process data of business processes with different behaviors in terms of their activity transition profiles. Additionally, it opens up the potential for an enriched hyperparameter optimization strategy that ensures tailor-made training of our approach for the respective event log at hand.

Regarding the strict pattern selection method using the local and global BEDs of the patterns, we can argue that filtering first for the minimum local BED leads to the fact that high-entropy decisions where multiple child subtrace patterns are equally probable are avoided. Moreover, focussing on the local BED initially leads us to decisions that are *easier* given the current historical activity context of our query sequence as we can identify more clear-cut decisions that have fewer competing sibling subtrace patterns to choose from. In contrast to that, only filtering for the minimum global BED mostly leads to the selection of very short subtrace patterns, as we favor the globally most frequent, i.e., in most cases the smallest, subtrace patterns. As those patterns include only a short amount of history as well, this leads to overall lower performance for the prediction of the remaining traces compared with the ground truth. Experiments showed that the strict pattern selection method starting with a filtering for the minimal local BED performs extraordinarily well on all datasets analyzed as we can record high performance in the prediction of remaining traces.

Our approach has three main parameters—the maximum depth of the tree generated, i.e., the maximum size of the mined subtrace patterns, the respective pattern selection method in the prediction loop, and the total number of distinct process stages we introduce in the training procedure, i.e., how many distinct tree models are trained for the respective event log. This can range from one single tree generated for the whole event log to n trees, where n is the maximum trace length that is observed in the event log.

In the following section, we provide a thorough evaluation of our approach regarding various configurations of those parameters. We provide a comparative benchmark analysis where we compare our results to existing techniques and also touch upon the runtime of our approach in general and for specific parameter combinations.

Evaluation

Conducting a thorough evaluation of our approach, we carry out various experiments with real-world benchmark event logs that show different characteristics regarding the overall event count, log variability in terms of the amount of distinct trace variants and the overall number of distinct activities as well as the length of the different traces recorded in the event log. We keep a steady evaluation routine across all datasets and perform an extensive set of experiments with different parameter combinations. As our predictive task, we perform Next Activity Prediction and Remaining Trace Prediction and evaluate the different experimental runs in terms of their predictive performance.

Evaluation setup

Datasets

For our evaluation, we use the following datasets: Helpdesk (Polato 2017), BPI2012 (van Dongen 2012) with multiple variations thereof that are commonly analyzed in existing contributions on activity prediction tasks (BPI2012 C - COMPLETE events, BPI2012 W - only workflow events, BPI2012 WC - only COMPLETE workflow events), BPI2013 (Closed Problems (Steeman 2013a), Incidents (Steeman 2013b)), Env Permit (Buijs

2022), Sepsis (Mannhardt 2016), Nasa (Leemans 2017), BPI2017 (van Dongen 2017), BPI2019 (van Dongen 2019). See Table 1 for an overview of the analyzed dataset and their core characteristics that are relevant for a sound assessment of predictive performance under different conditions.

Model setting

We run multiple experiments for the main parameters of our model – maximum traversal depth, i.e., maximum pattern size, the number of process stages defined by the process stage width calculated from a percentage of the overall maximum trace length of an event log, and also perform experiments with different selection methods. For a comparative benchmark analysis, we test all combinations of the parameters pattern size—ranging from [3, 5, 7, 11, 17, 21]—and the percentages defining the process stage width—ranging from [0.0, 0.025, 0.1, 0.2, 0.5, 1.0]. We split each dataset into 80% of training data and a hold-out sample of test instances consisting of the remaining 20%. Subsequently, we train our model within a 5-fold cross validation setting for each parameter combination where we split the folds by case id. This leads to each of the folds consisting of 20% of available cases of the training dataset. Effectively, we perform an 80%–20% train-test-split within the initially split training dataset where we use each of the folds once as a set of test instances and train our models on the remaining 4 folds. For each fold, we split the contained activity traces σ_i into full prefixes $\rho_j^{(i)}$ with $j = 0, \dots, |\sigma_i|$ to perform our experiments where j is the length of the resulting prefix starting from an empty prefix. For each dataset, we pick the best parameter combination in terms of the average performance achieved across all 5 folds. Eventually, we refit the model with the selected best parameter combination on the initially split 80% of training data and evaluate the final performance on the 20% hold-out sample. For both tasks—Next Activity Prediction and Remaining Trace Prediction—we report the performance achieved on the hold-out sample in Table 2.

Comparative benchmark analysis

To guarantee a sound assessment of the performances achieved for each prediction tasks, we compare our results with previously presented approaches from deep learning. We rely on the work of Rama-Maneiro et al. (2022) for the achievable performance of multiple approaches (Tax et al. 2017; Evermann et al. 2017; Hinkka et al.

Table 1 Datasets used for performance evaluation

Dataset	events	cases	activities	trace length (mean, median, variants max)			
Helpdesk	21348	4580	14	4.66	4.0	15	226
BPI'12 Full*	262200	13087	36	20.04	11.0	175	4366
BPI'12 C	164506	13087	23	12.57	7.0	96	4336
BPI'12 W*	170107	9658	19	17.61	14.0	156	2921
BPI'12 WC	72413	9658	6	7.50	6.0	74	2263
BPI'13 Closed Problems	6660	1487	7	4.48	3.0	35	327
BPI'13 Incidents	65533	7554	13	8.68	6.0	123	2278
Env Permit	8577	1434	27	5.98	6.0	25	116
Sepsis	15214	1049	16	14.48	13.0	185	845
Nasa	73638	2566	47	28.70	28.0	50	2513
BPI'17 Full*	1202267	31509	66	38.16	35.0	180	16692
BPI'19	1595923	251734	42	6.34	5.0	990	11973

Table 2 Performance comparison for Next Activity Prediction (NAP-Accuracy) and Remaining Trace Prediction (RTP - Damerau-Levenshtein Similarity). The best, second best and third best approaches are shaded in green, blue and orange, respectively. Metrics for (Tax et al. 2017; Evermann et al. 2017; Hinkka et al. 2019; Pasquadibisceglie et al. 2019; Theis and Darabi 2019) were directly taken from (Rama-Maneiro et al. 2022) and metrics for (Rama-Maneiro et al. 2024) were taken from the original publication. The remaining approaches were evaluated within a comparable experimental setup

	Help-desk	BPI'12				BPI'13		Env	Sepsis*	Nasa*
		Full	W	C	WC	Closed Problems	Incidents	Permit		
Next Activity Prediction										
Hinkka et al. (2019)	0.8308	0.8665	0.8478	0.8064	0.7054	0.6347	0.7469	0.8443	0.6350	0.8842
Theis and Darabi (2019)	0.7969	0.8096	0.8686	0.7575	0.8384	0.5948	0.5941	0.8512	0.5814	0.8896
Pasquadi-bisceglie et al. (2019)	0.8393	0.8325	0.8119	0.7460	0.6834	0.4745	0.4603	0.8669	0.5615	0.8827
Tax et al. (2017)	0.8419	0.8546	0.8535	0.8038	0.6979	0.6401	0.7009	0.8571	0.6422	0.8944
Camargo et al. (2019)	0.7657	0.7527	0.2549	0.6775	0.4649	0.4607	0.6349	0.8332	0.5375	–
Wuyts et al. (2024)	0.8443	0.8231	0.8173	0.7897	0.7066	0.6358	0.6481	0.8706	0.5526	0.6792
Polato et al. (2018)	0.6862	0.7489	0.7143	0.6654	0.4692	0.5706	0.5571	0.8462	0.4308	0.7828
Ceci et al. (2014)	0.4058	0.0940	0.1583	0.1702	0.3720	0.5624	0.4517	0.2921	0.2642	0.1774
van der Spoel et al. (2013)	0.8578	0.6557	0.4834	0.4198	0.1910	0.6805	0.5392	0.7967	0.3176	0.3823
Ours	0.8525	0.8428	0.8294	0.7670	0.6627	0.7000	0.7039	0.8735	0.5357	0.8882
Remaining Trace Prediction										
Evermann et al. (2017)	0.8354	0.1986	0.2800	0.2693	0.3372	0.6416	4730	0.5713	0.2693	0.1218
Tax et al. (2017)	0.8695	0.1409	0.0975	0.1717	0.0789	0.5824	0.3336	0.8163	0.1158	0.2320
Rama-Maneiro et al. (2024)	0.8989	0.4061	0.4000	0.4711	0.3829	0.6933	0.6230	0.8369	0.4285	0.4981
Camargo et al. (2019)	0.8784	0.2913	0.1974	0.3480	0.2873	0.6180	0.5105	0.8321	0.3402	–
Wuyts et al. (2024)	0.9191	0.2872	0.1142	0.3043	0.1170	0.7368	0.5112	0.8709	0.1286	0.4402
Polato et al. (2018)	0.8496	0.4132	0.3723	0.4636	0.3619	0.6782	0.5505	0.8252	0.3857	0.3862
Ceci et al. (2014)	0.2346	0.0241	0.0620	0.0649	0.0672	0.2173	0.1350	0.3346	0.0561	0.0280
van der Spoel et al. (2013)	0.9113	0.3192	0.2566	0.3316	0.3282	0.7206	0.5716	0.7463	0.3150	0.2533
Ours	0.9047	0.4215	0.3870	0.4831	0.3637	0.7495	0.5752	0.8385	0.3892	0.5065

2019; Pasquadibisceglie et al. 2019; Theis and Darabi 2019) since they thoroughly evaluated these deep learning approaches also for the tasks of Next Activity Prediction and Remaining Trace Prediction while maintaining a rigorous and unified benchmarking setup³. The achieved accuracies of predicted next activities and similarities of predicted

³The results reported in (Rama-Maneiro et al. 2022) for Theis et al. (Theis and Darabi 2019) are split up between two individual approaches (*w/ attributes* and *w/o attributes*). For each dataset in Table 2 we report results of the better performing approach. We do the same for results we retrieved ourselves for Camargo et al. (2019) and its two separate approaches (*random choice* and *argmax*).

traces we report in Table 2 are identical to the ones reported in (Rama-Maneiro et al. 2022) while always including the respective best performing approach for each dataset. To ensure parity between the results from an external contribution and ours, we recreated their experimental setup (train-test-split ratio, 5-fold cross validation) for the evaluation of our own approach as well as for the approach of Camargo et al. (2019), an additional and more recent approach by Wuyts et al. (2024) using a Transformer network for suffix prediction as well as three additional baselines. These include an approach based on transition systems by Polato et al. (2018), an approach based on path-finding algorithms by van der Spoel et al. (2013) and the approach by Ceci et al. (2014) based on sequential pattern mining in combination with nested machine learning models. Eventually, we also include a recent deep reinforcement learning approach for activity suffix prediction by Rama-Maneiro et al. (2024) and their reported performances into our comparison. They maintain a steady evaluation routine between their contributions, keeping the resulting performance values comparable to the ones reported in their initial benchmark study and also to our approach using the same evaluation setup.

We performed experiments based on the available implementations and specified parameter setting accompanying their publication. Wuyts et al. (2024) apply a dataset-specific train-test-split and also manipulate the set of process instances instances by filtering longer sequences for their experiments. We executed their approach and computed the results in Table 2 while employing the same 5-fold cross validation procedure as it was used for the remaining deep learning approaches [Tax et al. 2017; Camargo et al. 2019; Evermann et al. 2017; Hinkka et al. 2019 (Pasquadibisceglie et al. 2019; Theis and Darabi 2019), in (Rama-Maneiro et al. 2022), the other baselines (Ceci et al. 2014; van der Spoel et al. 2013; Polato et al. 2018) and our own approach and abstained from performing filtering operations. Overall, we expect to have reached a very high amount of parity in experimental design between the results taken from the deep learning survey by Rama-Maneiro et al. (2022), the deep reinforcement learning approach by Rama-Maneiro et al. (2024), and the remaining approaches included in our comparison which were (re-implemented and) evaluated by ourselves. As the implementation of the Transformer network has no inherent functionality to predict next activities, we extracted the first activity of the resulting predicted suffix for each test case and calculated the performance in an identical manner as for our approach.

Finally, to evaluate our approach in comparison with the remaining contributions, we also use the same and already established evaluation metrics for Next Activity Prediction and Remaining Trace Prediction—accuracy of the forecasted next activity and the normalized Damerau-Levenshtein-Similarity (Boytssov 2011) for the predicted activity suffix in Remaining Trace Prediction.

Looking into our achieved performances for Remaining Trace Prediction, we can observe notably higher performance of BEST in comparison to the existing deep learning techniques for the greater part of the benchmark event logs. However, considering also the deep reinforcement learning approach by Rama-Maneiro et al. (2024), BEST seems to fight a closer battle where it is not consistently the best performer. Nonetheless, we still achieve results that are on par with the remaining contributions in the field on those datasets where we do not achieve the best performance and notice a very high level of consistency of our approach for the task of Remaining Trace Prediction in comparison to most of the deep learning approaches and additional model-based approaches. In

Table 3 Performance comparison on larger datasets BPI'17 Full and BPI'19 for selected models for tasks Next Activity Prediction (NAP - Accuracy) and remaining trace Prediction (RTP - Damerau-Levenshtein Similarity). The best, second best and third best approaches are shaded in green, blue and orange, respectively. Approaches which did not finish within one week (DNF) or ran out of memory (OOM) were marked accordingly

	BPI'17 Full	BPI'19
Next Activity Prediction		
Camargo et al. (2019)	0.6633	0.7053
Wuyts et al. (2024)	0.8568	OOM
Polato et al. (2018)	DNF	DNF
Ceci et al. (2014)	0.0477	OOM
van der Spoel et al. (2013)	0.6557	0.5404
Ours	0.8315	0.7480
Remaining Trace Prediction		
Camargo et al. (2019)	0.4091	0.6829
Wuyts et al. (2024)	0.2198	OOM
Polato et al. (2018)	DNF	DNF
Ceci et al. (2014)	0.0078	OOM
van der Spoel et al. (2013)	0.3192	0.6382
Ours	0.4663	0.7424

fact, only the approaches of Polato et al. (2018) and Rama-Maneiro et al. (2024) seem to be consistent across datasets as the former is able to at least achieve top-3 results for most of the datasets and the latter shows high performance across the board as well.

We attribute the high performance of BEST to the fact that the tree consists only of subtrace patterns that are actively taking place in the recorded event log data. Since these patterns are the direct source of our generated predictions, we can achieve existing patterns with absolute certainty, while the LSTM and Transformer approaches might predict future transitions between activities that their model deems to be the most probable one given the context but might actually be contradicting the underlying process behaviour as they never actually occur inside the data. Choosing only actively occurring patterns seems to be especially beneficial for the case of predicting future activity suffixes when we generate trees with larger pattern sizes.

Regarding Next Activity Prediction, our results suggest that BEST is capable of outperforming existing techniques for some of the analyzed benchmark event logs. However, from the overall perspective, BEST tends to be prone to deficits compared with some of the competing approaches as it cannot outperform them on a consistent basis. For some of the event logs BEST is also not in striking distance, e.g., when comparing to approaches of Hinkka et al. (2019), Tax et al. (2017) and Theis and Darabi (2019) (BPI'12 W, BPI'12 WC, Sepsis). Although BEST is not deemed the best performing approach for all datasets, it takes the third place or higher for the larger part of the analyzed benchmark datasets.

We also provide an assessment of the robustness of the prediction performances for a subset of approaches. The subset consists of representatives for approaches based on (LSTM) neural networks (Camargo et al. 2019), Transformers (Wuyts et al. 2024), transition systems (Polato et al. 2018), pathfinding algorithms (van der Spoel et al. 2013) and sequential pattern mining (Ceci et al. 2014). The choice for Camargo et al. (2019) as the representative for (LSTM) neural network approaches over other listed approaches like Hinkka et al. (2019) and Tax et al. (2017) is based on (Rama-Maneiro et al. 2022) identifying it as the best approach for the task of Remaining Time Prediction that works

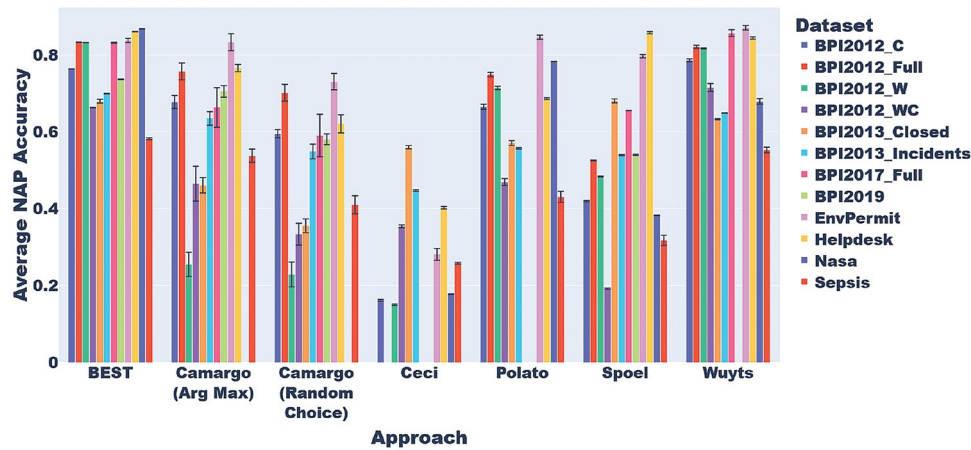


Fig. 4 Forecasting performance including predictive uncertainty of different approaches for the analyzed datasets on the task of Next Activity Prediction. Uncertainty is expressed by means of one standard error across the executed runs for the respective approach and dataset

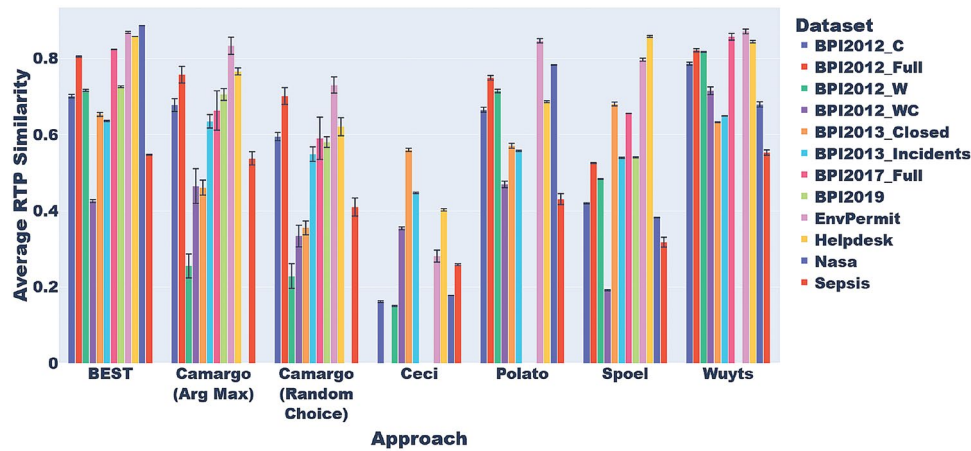


Fig. 5 Forecasting performance including predictive uncertainty of different approaches for the analyzed datasets on the task of Remaining Trace Prediction. Uncertainty is expressed by means of one standard error across the executed runs for the respective approach and dataset

well on a wide range of datasets. Visualized in Fig. 4 and 5, we can observe that BEST achieves a very narrow spread of its achieved performance metrics across the board. For Next Activity Prediction as well as Remaining Trace Prediction, BEST produces forecasts that outperform the remainder of benchmark approaches in a combined assessment of overall predictive performance and robustness. Other approaches are able to achieve a similar level of robustness, e.g. van der Spoel et al. (2013), Ceci et al. (2014) and Polato et al. (2018). However, their overall predictive performance varies across datasets and the models achieve only sub-par results on average when considering the task of Next Activity Prediction. Regarding Remaining Trace Prediction, Polato et al. (2018) are able to also perform well in predictive performance as well as robustness of the generated predictions. The only approach that is able to deliver a similar level of robustness while maintaining high predictive performance in both tasks is the one of Wuyts et al. (2024). As already stated in Table 2, Wuyts et al. (2024) are also able to outperform our approach on some of the datasets in terms of average performance for Next Activity Prediction and Remaining Trace Prediction.

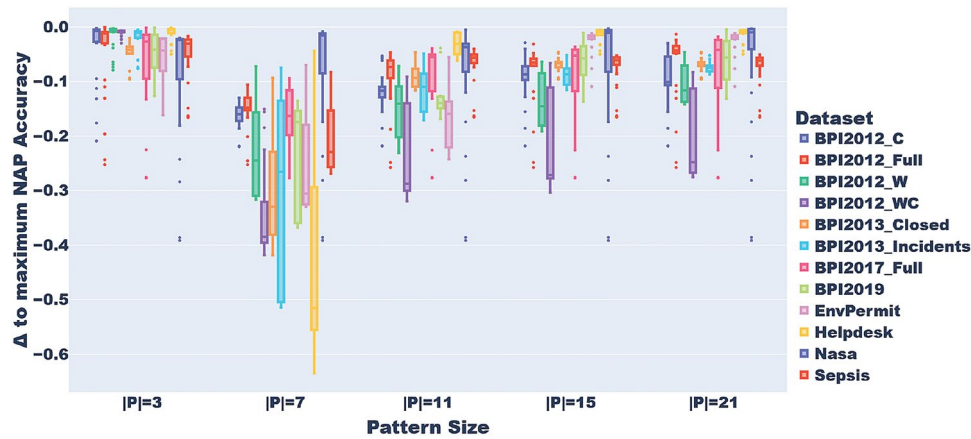


Fig. 6 Distribution of forecasting accuracies in terms of distance to the maximum achieved value across all runs for the specified pattern size used in the prediction

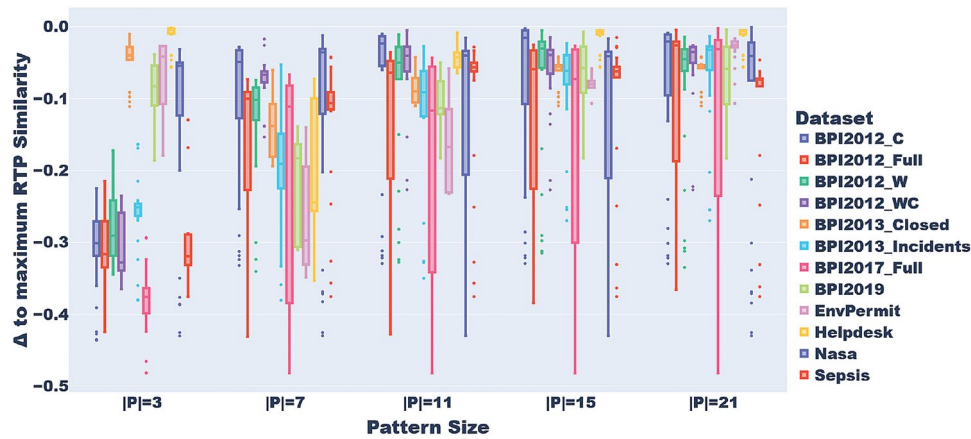


Fig. 7 Distribution of the Normalized Damerau-Levenshtein-Similarity values in terms of distance to the maximum achieved value across all performed experiments for the specified pattern size used in the prediction

Sensitivity analysis

We perform a sensitivity analysis with respect to the main parameters of our approach by looking into how different parameter settings for the maximum pattern size as well as the number of specialized trees generated affect our predictive performances. We show a visual analysis of these effects in Figs. 6 and 7. Increasing the size of the patterns considered in the pattern search, i.e., traversing deeper into the hierarchical pattern tree and scanning for longer matching patterns, leads to an overall increase and a stabilization of the performance recorded for the case of Remaining Time Prediction. Having a deeper tree with longer subtrace patterns in it allows for a more nuanced decision where we can also honor longer distance relationships between activities or subtraces. In contrast, stopping pattern search at shorter pattern lengths translates to only looking into a short activity history and an equally short amount of future activities able to be picked as a forecast. We therefore recommend to increase the tree depth for the task of Remaining Trace Prediction to guarantee higher performances with respect to the achieved pattern similarities. In contrast to this recommendation, we can observe the highest performances of BEST for shorter pattern sizes when considering the task of Next Activity Prediction. Although the accuracies show an increasing trend for an increase of the

pattern length, the maximum accuracy values observed for short patterns of pattern size $|P| = 3$ is almost never surpassed in our experiments. Consequently, we formulate an analogous recommendation for the task of Next Activity Prediction to limit the tree traversal depth to this pattern size.

Although different pattern sizes seem to be favorable for different settings, our approach is still flexible once it is trained. The maximum search depth can simply be specified in the pattern matching and search procedure according to the respective forecasting task without having to retrain the model from scratch.

Shifting to the effects of training multiple specialized tree models for different process stages, we can observe how this parameter affects the accuracies in Fig. 8. Looking at the smaller process stage widths, i.e., the training of *more* specialized models for a given event log, we can recognize a smaller variance in predictive performance and an overall higher performance when considering the task of Remaining Trace Prediction. This effect can, however, not be shown for all analyzed logs (Helpdesk, Env Permit, Nasa). These datasets are excluded from the visualization as they do not exhibit the effect in the same extent as the remaining datasets. Taking the strict slicing mechanism presented in section [BEST: Bilaterally Expanding Subtrace Tree](#) into account, we suspect that the reason for this is that we cannot achieve a perfect mapping from the sliced process stages to actual phases of the process.

Parameter recommendation

Given the results in our sensitivity analysis we can formulate a recommended default parameter setting for our approach. Regarding the maximum pattern size, we advise to tune the parameter to $|P| = 3$ for the task of Next Activity Prediction, as training time is reduced and performance shows superior results in comparison with higher pattern sizes. In the case of Remaining Trace Prediction, we advise a larger pattern size up to the evaluated maximum pattern size of $|P| = 21$ due to our experiments showing the highest and most stable performance for a wide range of datasets. If both downstream tasks are relevant to the user, we recommend training a tree with pattern sizes up to $|P| = 21$ and limiting the pattern matching to patterns of size $|P| = 3$ for Next Activity Prediction and perform unbounded pattern matching up to a pattern size of $|P| = 21$ for Remaining Trace Prediction. Regarding the number of stages, i.e., the amount of

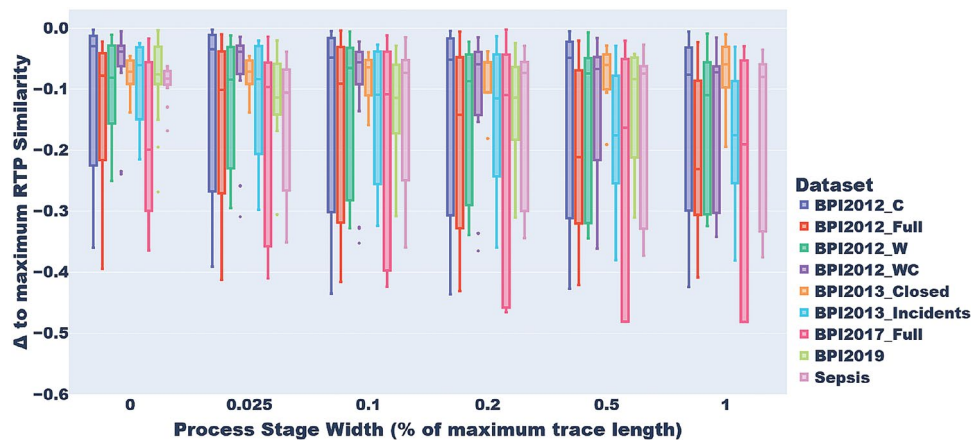


Fig. 8 Distribution of Damerau-Levenshtein-Similarity in terms of distance to the maximum achieved value across all runs for different process stage widths in terms of percentage of the maximum trace length used for mining the patterns. We omitted event logs Helpdesk, EnvPermit, Nasa as the effect is not as pronounced there

hierarchical subtrace trees generated, we advise setting the process stage width in terms of the percentage of the maximum trace length to a smaller value (e.g. < 0.2) based on our experimental results shown in Fig. 8. However, a general one-fits-all setting cannot be recommended, as we experienced contrasting effects between datasets. Nonetheless, the maximum pattern size turns out to be the more critical parameter. Therefore, we assign a secondary importance to setting the *correct* process stage width parameter.

Predictions using the weighted probability metric

In addition to the pattern selection by using the local and global BEDs we analyze the effect of the introduced flexible pattern filtering via the weighted probability metric Pr_W (see Eqs. 3 and 4). Within this analysis, we also highlight the sensitivity of choosing a fitting weight parameter value w with respect to the dataset analyzed and forecasting task performed (NAP or RTP). We provide a visual analysis of our results for different parameter settings, prediction tasks and datasets BPI2012 and its variations in Fig. 9 and a subset of the remaining datasets analyzed in the same manner in Fig. 10.

Regarding the results of the BPI2012 datasets in Fig. 9 we can see the effects of the choice of the weight parameter w in combination with the chosen pattern size for which the tree is searched in the prediction procedure. In the left column, for the results on NAP, we can see a high variation in achieved accuracies if we search the tree for longer patterns. Scanning only for short patterns of size 3, the weight parameter seems to be neglectable as we achieve the maximum accuracy for the respective datasets—as long as we set a non-zero value for w . We can see a similar trend for the datasets in Fig. 10. However, the variance in achieved accuracies seems to be only apparent for smaller pattern sizes between 5 and 11. These results are in line with our previous analyses for the NAP task from Fig. 6 where maximum accuracies were achieved for very small pattern sizes.

Looking into the RTP case, the relationship appears to be somewhat more complex. Datasets BPI2012 W and BPI2012 WC in Fig. 9 as well as dataset EnvPermit and BPI2013 Incidents in Fig. 10 can identify sweet spots for the weight parameter w for given pattern sizes. This highlights the apparent differences in the analyzed real-world datasets in terms of their sequential transitions between activity patterns. At the same time, this makes clear that a more flexible and tunable pattern filtering can help in adapting our model to the specific characteristics of each dataset in addition to the results visualized with respect to the maximum pattern size (see Figs. 6 and 7).

Parameter recommendation

Considering the results on the flexible pattern filtering via the weighted probability metric Pr_W , we can confirm the general setting of the maximum pattern size for the tasks of Next Activity Prediction ($|P| = 3$) and Remaining Trace Prediction ($|P| = 21$) for this selection strategy as well. Regarding a general decision as to which selection method to apply for a specific dataset, our recommendation is to perform pattern selection via the local and global BEDs, as the performance is more consistent and less sensitive to finding the correct weighting parameter w . However, if the computational resources allow it, hyperparameter optimization regarding the weight parameter might pose benefits for individual datasets in terms of predictive performance in comparison to the pattern selection with local and global BEDs.

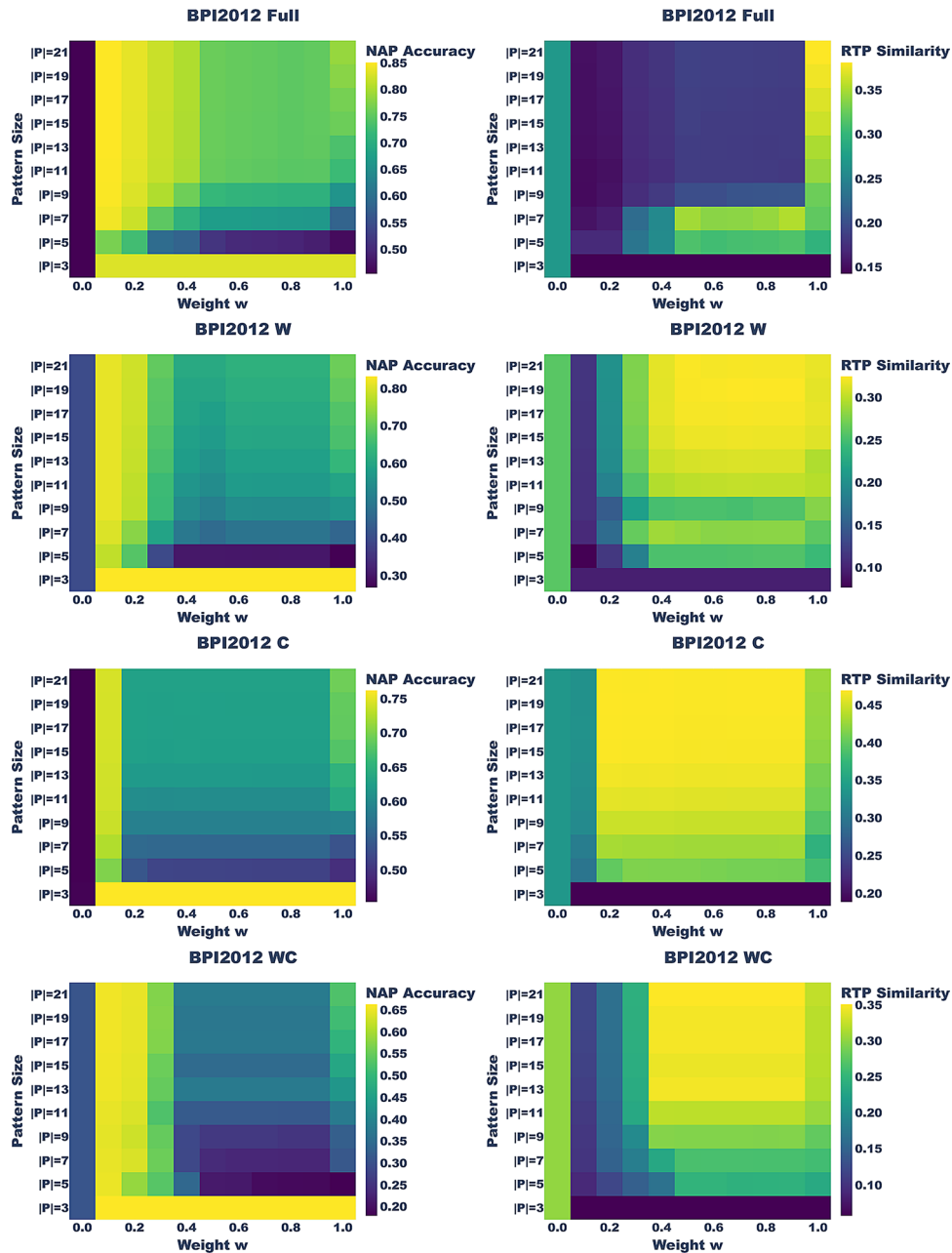


Fig. 9 Visual sensitivity analysis of the weight parameter w for trees generated with different pattern sizes for the dataset BPI2012 along with its variations BPI2012 W, BPI2012 C and BPI2012 WC. The left column of plots shows results for task NAP, the right column of plots shows results for task RTP

Applying pruning

To keep the complexity of our model small enough and to not generate huge tree structures of all possible (but potentially non-informative) subtrace patterns, we can apply pruning strategies. Multiple strategies to prune the tree are possible and can be applied either during tree generation (pre-pruning) or after the tree has been generated completely (post-pruning). We opt for the description and a showcase of the effects of a pre-pruning that is easily applied during the training of the model, specifically setting a global probability cutoff. The probability cutoff decides if the hierarchical tree generation algorithm should go forward with potential extensions of a given subtrace pattern

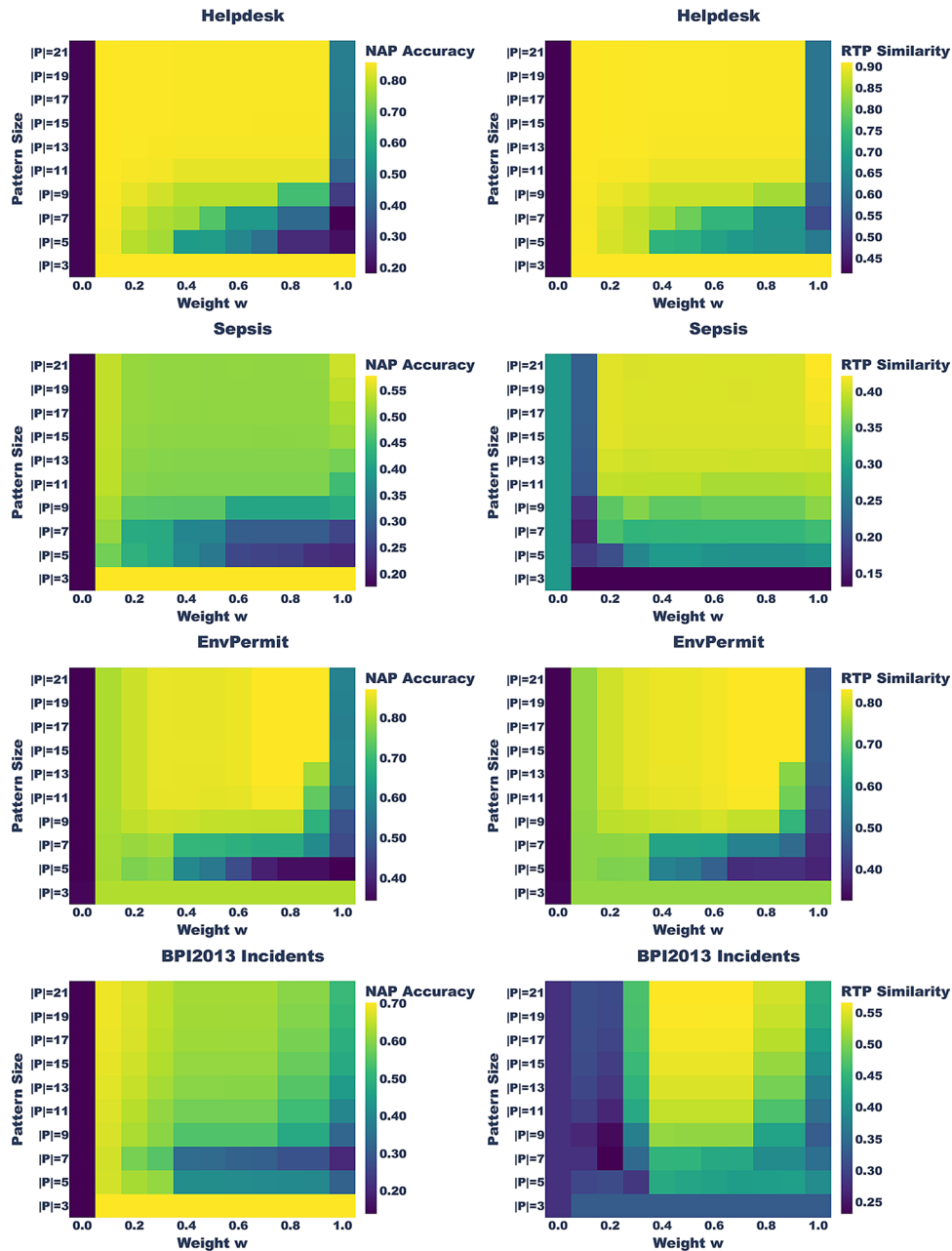


Fig. 10 Visual sensitivity analysis of the weight parameter w for trees generated with different pattern sizes for the datasets Helpdesk, Sepsis, EnvPermit and BPI2013 Incidents. The left column of plots shows results for task NAP, the right column of plots shows results for task RTP

or if the branch of the tree is stopped at the respective node. The decision is based on the global pattern occurrence probability $Pr(P_i)$ of a subtrace pattern P_i and enforces a stoppage of the pattern search and matching when this probability gets lower than the cutoff. A visual representation of this cutoff mechanism is depicted in Fig. 11. In this slightly different visualization of the tree structure, the height of the nodes represents their global occurrence probability and the depth of a node and respectively the size of the contained pattern can be determined by the number of edges between the respective node and the root node. We also want to highlight the case for which patterns of a smaller size can be pruned by the global occurrence probability cutoff it occurs less

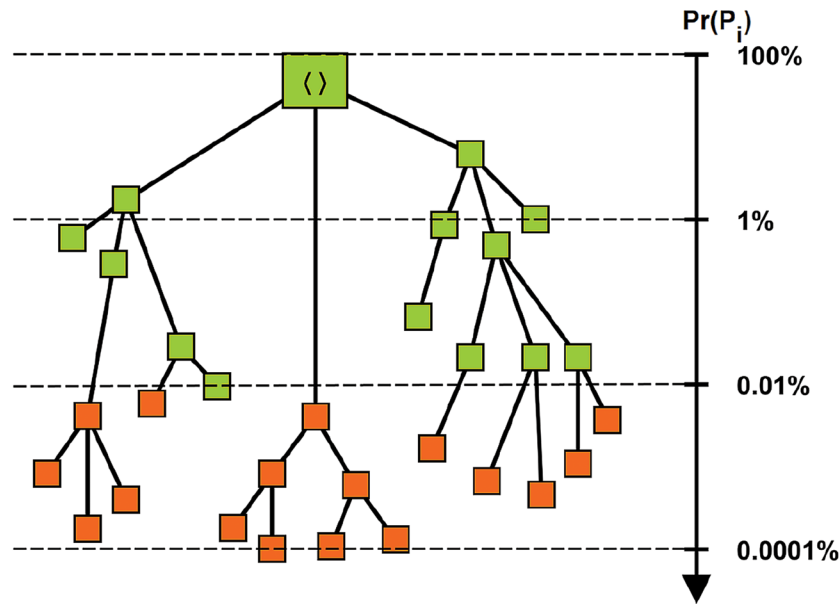


Fig. 11 Schematic visualization of the effect of a global occurrence probability cutoff threshold of 0.01%. Pattern nodes of the tree are visualized (•: pruned, •: unpruned) comparison of non-constrained tree generation and tree generation with applied pre-pruning via global pattern probability cutoff

frequently than an arbitrary larger pattern. This case can be observed in the middle branch of the tree, where a pattern of size 1 is pruned due to a global occurrence probability of less than the cutoff probability but other, larger patterns are still unpruned.

The pruning drastically affects the number of patterns generated and therefore can reduce the complexity of the generated trees heavily. Experiments showed that the reduction in mined and matched subtrace patterns for trees amounts to $\approx 99\%$ when setting a probability cutoff value of 0.001%. The effects of this reduction in model complexity are also directly apparent in terms of training and inference times of our approach. We elaborate on this in section [Runtime analysis](#) while also considering the implications for the resulting predictive performance.

Runtime analysis

In the following, we compare average computation times split up between training time and inference time for the complete suffix prediction for all datasets. In that, we report the times for BEST and a subset of existing deep learning approaches where implementations were available, i.e., Camargo et al. (2019) (LSTM), Wuyts et al. (2024) (Transformer), Polato et al. (2018) (Transition System), van der Spoel et al. (2013) (Pathfinding Algorithm) and Ceci et al. (2014) (Sequential Pattern Mining). We provide the respective parameter setting for each approach in Table 4. For the approach of Polato et al. (2018), we selected the ‘sequence’ abstraction as it was the one achieving the best predictive performance across all datasets and also the minimum inference duration for processing all prefix traces for a given dataset. Regarding the approach of Ceci et al. (2014), we abstained from using the sequence tree types that only mine frequent contiguous and contiguous opening sequences as specified in the original contribution because usage of the original FAST tree generation algorithm performed best in terms of achieved predictive performance. However, the tree generation with the original FAST algorithm comes with a caveat as the amount of tree nodes grows drastically in comparison with the more

Table 4 Parameter settings for all benchmark approaches within the evaluation for training and inference duration

Approach	Training Parameters
Ours	Maximum Pattern Size: 21 Probability Cutoff Value: 0.01% Process Stage Width (percentage of max trace length): [0.00, 0.025, 0.10, 0.20, 0.50, 1.00]
Camargo et al. (2019)	Max Epochs (embedding training): 100 Max Epochs (model training): 200 Early Stopping (embedding training): False Early Stopping (model training): True Patience (model training): 50 Batch Size: 64
Wuyts et al. (2024)	Max Epochs: 200 Early Stopping: True Patience: 24 Batch Size: 128 Batch Size (validation): 2048
Polato et al. (2018)	Abstraction: sequence
van der Spoel et al. (2013)	—
Ceci et al. (2014)	Sequence Tree Algorithm: FAST Minimum Support: [0.25, 0.3, 0.4] ML model: C4.5 Decision Tree

restrictive tree generation algorithms mining for only frequent contiguous and contiguous opening sequences. As the approach requires to learn distinct nested models for each tree node, the model grows increasingly large in terms of memory as well as time needed for training the nested models. Additionally, the approach leverages all available process event attributes to predict future activities. Especially for larger datasets, e.g., BPI'17 and BPI'19, this results in increased training/inference durations and memory overflow, respectively.

For a more detailed assessment, Figs. 12 and 13 provide a visual comparison of training and inference times. Moreover, we provide an analysis on the effect of the pruning by a global probability cutoff in terms of training and inference times as well as the overall model size, while also looking at the effects on pruning on the predictive accuracy of our approach. Figures 14 and 15 depict the interconnection between setting different probability cutoff values for training time and inference time, respectively.

Regarding the training times, we achieve runtimes that are orders of magnitude smaller than those for existing deep learning approaches. The runtime of our approach for the different datasets is more in the vicinity of the approaches of van der Spoel et al. (2013) and Polato et al. (2018). However, the approach of Polato et al. (2018) suffers greatly in terms of inference speed, recording longer inference runtimes than all of the remaining approaches. While BEST achieves drastically lower inference times, it is only outperformed by the lightweight pathfinding algorithm of van der Spoel et al. (2013). In a direct comparison with the deep learning approaches of Camargo et al. (2019) and Wuyts et al. (2024), BEST also records inference times that are around one order of magnitude lower for a wide range of datasets compared to Camargo et al. (2019), showcasing even larger improvements compared to the Transformer model proposed by Wuyts et al. (2024). To further put the results into perspective, we want to note that the deep learning architectures rely on popular libraries like PyTorch and Tensorflow that provide CUDA support and enable heavily parallelized and efficient training of large model architectures on GPUs. In contrast to this, the implementation of our proposed

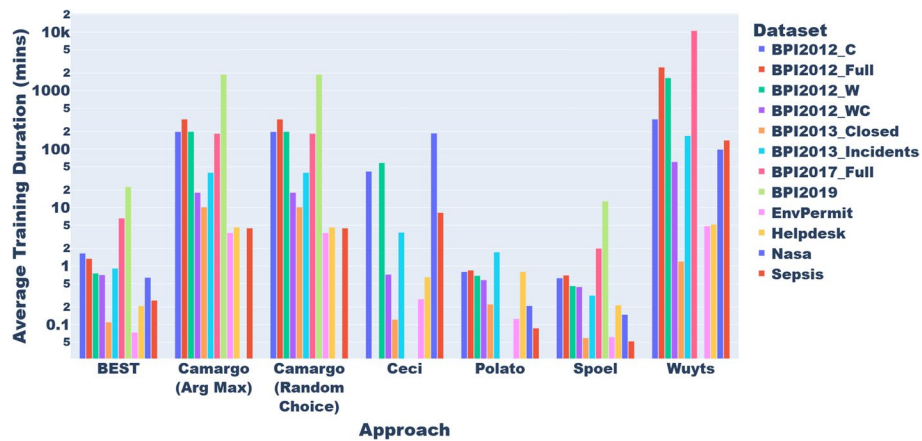


Fig. 12 Training duration in minutes for all datasets for BEST and a subset of competitors from different architectures (LSTM - Camargo et al. (2019), Transformer - Wuyts et al. (2024), Transition Systems - Polato et al. (2018), Pathfinding Algorithms - van der Spoel et al. (2013), Sequential Pattern Mining (Ceci et al. (2014)). Durations are visualized with logarithmic scale. Durations for BEST were achieved with pre-pruning enabled with a probability cutoff value of 0.01% and maximum pattern size of 21

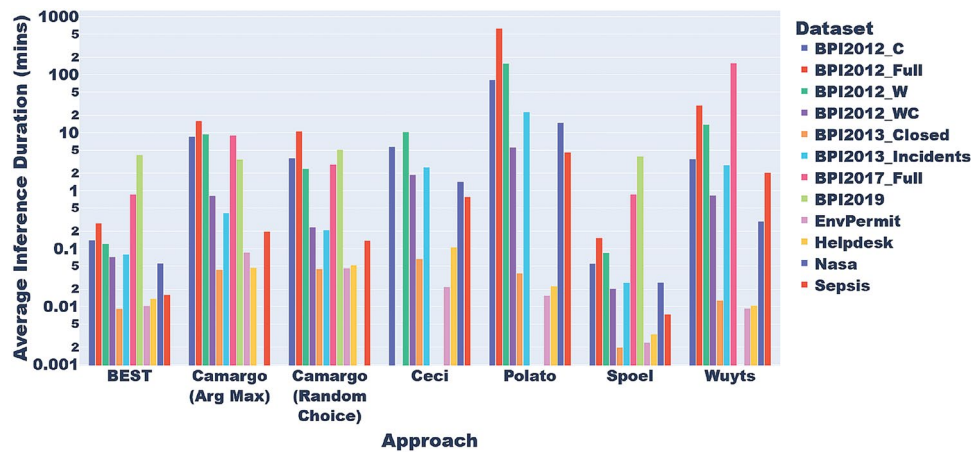


Fig. 13 Inference duration in minutes for all datasets for BEST and a subset of competitors from different architectures (LSTM - Camargo et al. (2019), Transformer - Wuyts et al. (2024), Transition Systems - Polato et al. (2018), Pathfinding Algorithms - van der Spoel et al. (2013), Sequential Pattern Mining (Ceci et al. (2014)). Durations are visualized with logarithmic scale. Durations for BEST were achieved with pre-pruning enabled with a probability cutoff value of 0.01% and maximum pattern size of 21

framework is evaluated on the CPU only. The training and inference times of the evaluated deep learning architectures were performed on an NVIDIA L40S GPU featuring 48 GB of GDDR6 VRAM, experiments of our approach were executed on a virtual machine running with 16 virtual CPUs with base frequency of 2.10 GHz and 128 GB of RAM.

Figures 14 and 15 show the stark positive effect of employing a pre-pruning strategy in the training procedure by executing a global pattern probability cutoff across all datasets. The reduction in average training time is more pronounced for the more complex datasets, i.e., BPI'12 and its variations, BPI'13 Incidents and Nasa, as the remaining datasets also experience lower training times without applying any pruning. Recorded times for datasets BPI'17 Full and BPI'19 experience less improvement due to applying pruning. However, we were only able to run those datasets with higher probability cutoff values of $> 0.01\%$. Therefore, a reduction in model complexity by applying pruning is

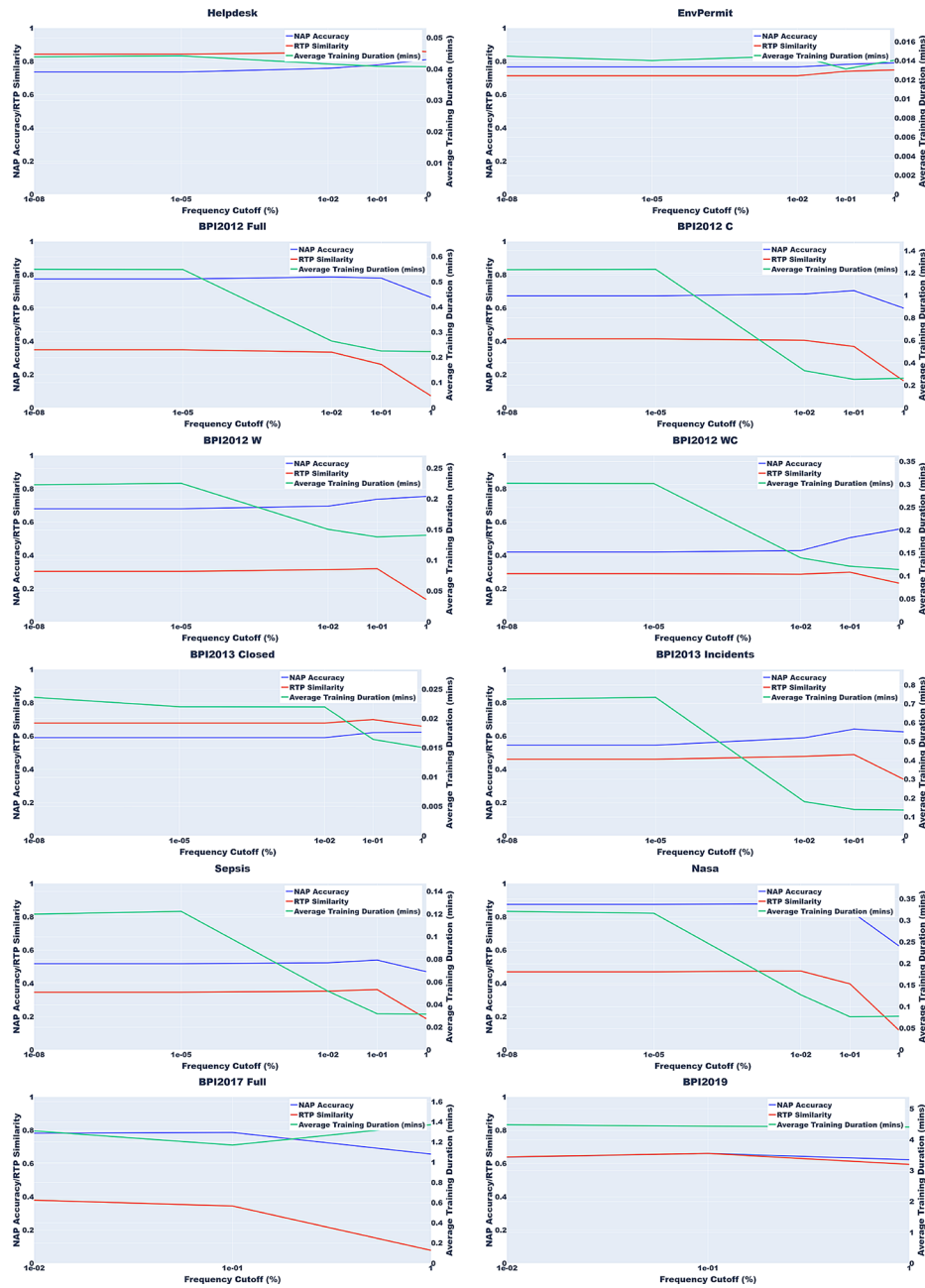


Fig. 14 Connection between different probability cutoff values for training times and the corresponding achieved accuracies and similarities across all datasets processed with BEST. Due to memory constraints, training times for BPI'17 Full and BPI'19 could not be gathered for smaller probability cutoff values than 0.01%

expected. Notably, the performance in terms of the achieved metrics for Next Activity Prediction and Remaining Trace Prediction seem to stay constant for a probability cutoff value lower than 0.01%, highlighting the potential runtime improvement without sacrificing any predictive capabilities. Regarding inference time, we can see a similar pattern, and inference time can be reduced for the larger datasets with many traces to be predicted, i.e., BPI'12, BPI'17 and BPI'19. Looking at the slight trade-off in applying pruning and reducing inference times, we propose to set the probability cutoff value at a value of 0.01% to keep training and inference times minimal while still being able to receive

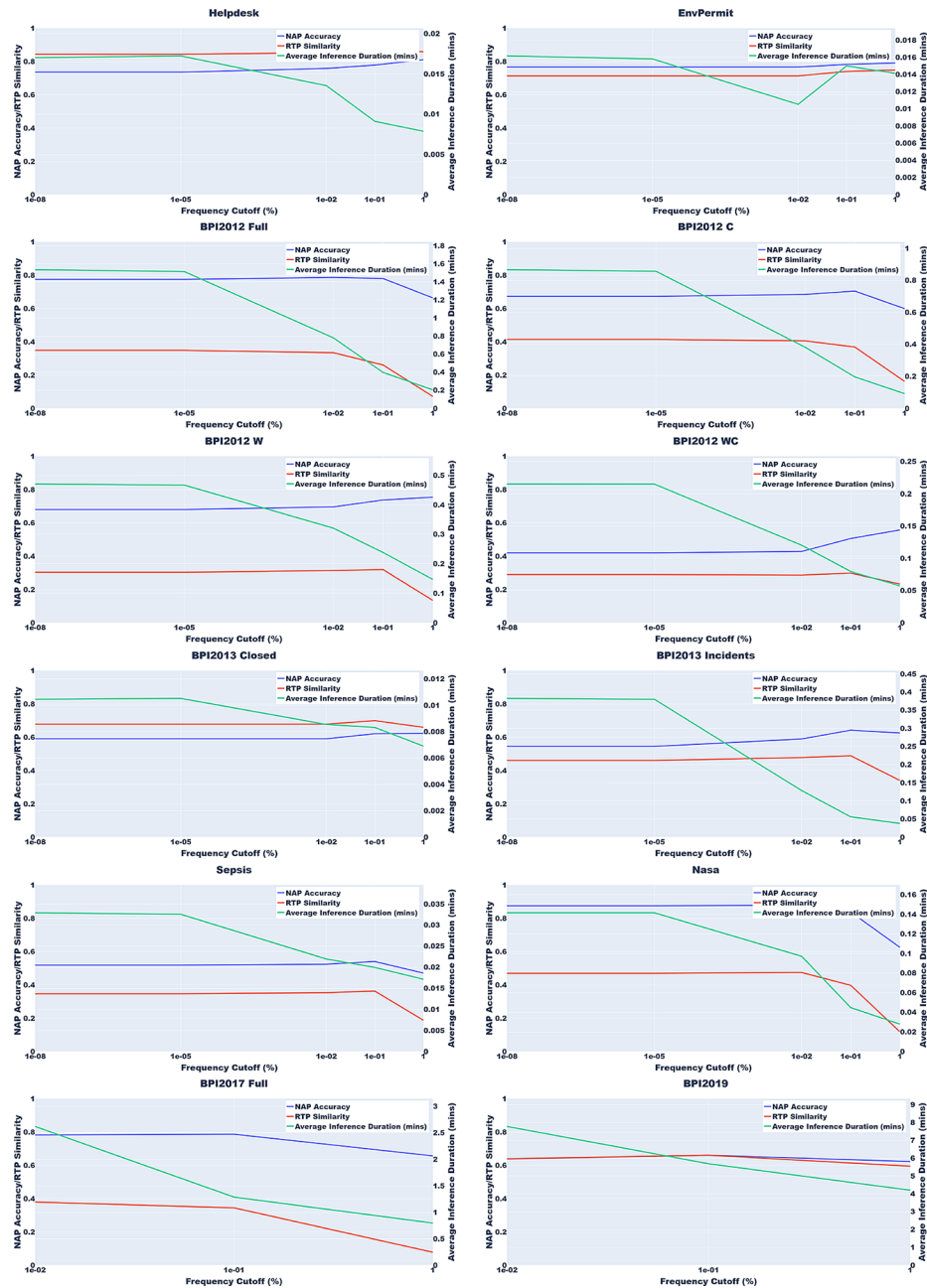


Fig. 15 Connection between different probability cutoff values for inference times and the corresponding achieved accuracies and similarities across all datasets processed with BEST. Due to memory constraints, inference times for BPI'17 Full and BPI'19 could not be gathered for smaller probability cutoff values than 0.01%

high-performing prediction models. In terms of model size, we can also show a decrease across all datasets as depicted in Fig. 16 when introducing the pre-pruning strategy during training. Across all datasets analyzed, we can drastically reduce the model size in terms of patterns mined and arranged into the hierarchical tree structure. This translates into an overall speed-up in the inference phase for Remaining Trace Prediction as the curves for the numbers of prefix traces processed per second convincingly show. Some datasets experience peaks in inference speed for intermediate probability cutoff values. These can be explained by the fact that the probability cutoff value trims longer, more

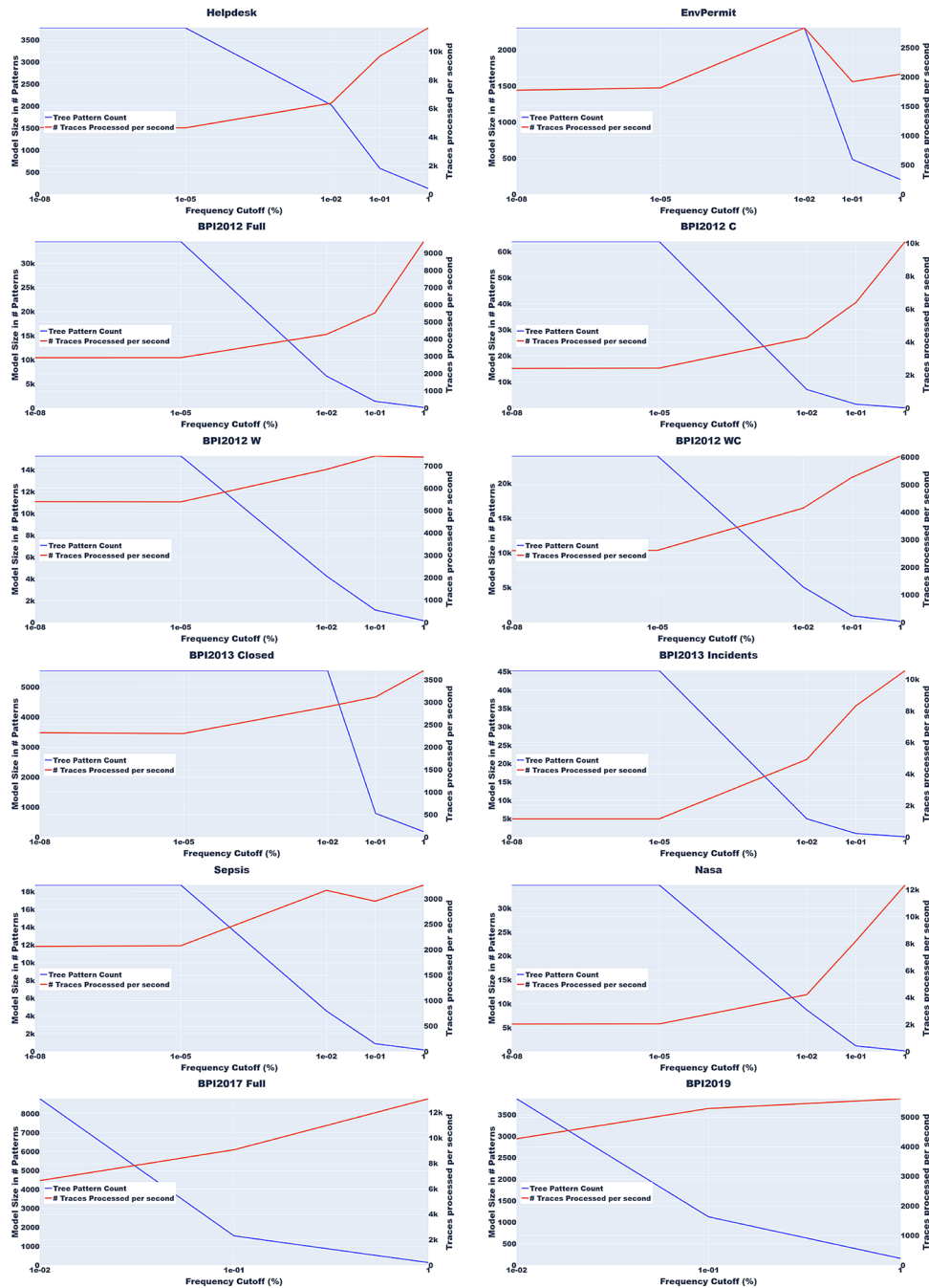


Fig. 16 Connection between different probability cutoff values and model size in terms of patterns contained in the generated trees as well as corresponding inference speed in terms of prefix traces processed per second for the task of Remaining Trace Prediction across all datasets processed with BEST. Due to memory constraints, model sizes for BPI'17 Full and BPI'19 could not be gathered for smaller probability cutoff values than 0.01%

seldomly occurring patterns from the tree which would lead to a termination of the prediction of a future activity traces in fewer iterations. Instead, the tree showcases only smaller patterns, which require more iterations to reach the respective 'END' token. In total, BEST seems to be capable of handling high loads during inference tasks as for all datasets at least 1000 prefix traces can be processed in one second even without leveraging pruning strategies. Using the previously proposed probability cutoff value of lower

than 0.01%, this speed can be increased even higher, reaching up to several thousand complete predictions for a wide range of datasets.

Outlook: integrating and predicting process event attributes

As BEST is mainly designed for forecasting activity suffixes, we focused on building the tree of hierarchical subtrace patterns from control flow information alone until now. However, the tree structure is also capable of integrating attribute information of events as well. In the following, we present in exemplary manner, how BEST could leverage temporal attribute information and consequently be used as a predictive model to forecast not only activity suffixes but temporal event information and overall case durations as well. To achieve this, the basic framework of the approach as well as the tree generation and forecasting logic can remain unchanged. Temporal attribute integration is accomplished by transforming the activity subtraces into subtraces carrying activity and discrete temporal information. In this proof of concept, we include temporal information of the analyzed event log into the patterns, by performing a concatenation of the respective activity with the current case duration at each respective event in a discretized form. We provide a small example for this transformation in Table 5. For the remainder of this outlook, we perform a discretization of the overall durations across all cases into 10 bins of equal width. The addition of the discretized temporal perspective into the subtrace patterns comes with an increase in the cardinality of the item space we build our hierarchical trees with. In theory, this increase could amount to the cartesian product of the activity space times the number of bins created within the discretization of the overall case duration. However, as our experiments showed, the combinations between activities and overall duration classes are quite sparse, as earlier activities in cases tend to show smaller overall durations, whereas later activities are combined with higher overall durations.

Applying the tree generation to the transformed subtrace patterns consisting of activity and duration information enables prediction of suffixes of activity-duration pairs, allowing for both, Remaining Trace Prediction as well as the prediction of the future overall process duration while also using the temporal perspective as input information in both tasks. We carry out experimental evaluations for both tasks while using temporal attributes for tree generation in Table 6.

Firstly, we can recognize that the performance with respect to suffix prediction experiences a decrease over all datasets analyzed when including the temporal information as it was specified previously. We suspect that this is a side effect of the increased cardinality of the created activity-duration pairs in comparison to only using the pure activity information. There appears to be a trade-off between the positive influence of including temporal information and the deteriorating effect of the increased complexity of the resulting trees. Hence, we see a need to further investigate strategies to incorporate attribute information into the tree generation and prediction procedures without

Table 5 Exemplary concatenation of activity identifiers $\{A, B\}$ and different temporal information (current overall case duration) classes into activity-duration pairs

Activity	Overall Duration Class		Activity-Duration Pair
A	3	$\Rightarrow$	(A, 3)
A	4	$\Rightarrow$	(A, 4)
B	7	$\Rightarrow$	(B, 7)

Table 6 Results for Remaining Trace Prediction (RTP - Damerau-Levenshtein Similarity) and Overall Duration Prediction (ODP - Accuracy) on different datasets with 10 overall duration classes

	Helpdesk	BPI'12				BPI'13		Env	Sepsis	Nasa	BPI'17	BPI'19
		Full	W	C	WC	Closed Problems	Inci-dents	Permit			Full	
RTP	0.8888	0.4148	0.3590	0.4521	0.3481	0.6718	0.5594	0.8202	0.3629	0.4708	0.4670	0.7446
ODP	0.3143	0.4876	0.4119	0.4225	0.1933	0.5296	0.8262	0.7883	0.6516	0.9586	0.5496	0.8436

suffering the experienced performance decrease that is apparent within the results of Table 5. Shifting from the demonstrated effects with respect to Remaining Trace Prediction, the added functionality and experimental results of predicting the future overall duration of a case in a discretized manner can be observed. While some of the datasets show a rather low accuracy in predicting the correct of the 10 introduced overall duration categories, other datasets (BPI'13 Closed Problems, BPI'13 Incidents, Env Permit, Nasa, BPI'17 Full) achieve good accuracy in forecasting the remaining duration class of a case. However, as the performance seems to exhibit stark differences for the different datasets, there seems to be a need for including the temporal information in a more efficient way or to also include other attributes that could influence the overall duration of a process into our method.

Limitations and future work

As the gathered results from our comparative benchmark analysis show, BEST suffers from slight drawbacks for the case of predicting next activities. When considering that we achieve the best results for our approach when limiting the patterns to smaller sizes (see Fig. 6), the main reason for this might be that only a smaller amount of prefix context can be taken into account. In turn, allowing for larger patterns shifts the focus from the next activity to activities further down the road. As a result, due to using the conditional probability of an extended pattern as the main driver of our pattern selection procedure, we select patterns that are not the best fit regarding the next activity. To tackle this problem and to also be able to take a longer prefix context into account also for the prediction of next activities, we want to investigate asymmetrical subtrace pattern extension with longer prefixes and 1-suffixes. Additionally, BEST is mainly developed for the task of activity prediction. In future research, we want to build upon the short proof of concept shown in section [Outlook: integrating and predicting process event attributes](#) where we demonstrated how the integration of additional attribute information could be accomplished and also look into other ways of incorporating larger amounts of additional event information into the training and inference phases in a standardized and robust manner. By taking additional event information into account, BEST could be elevated into a generally applicable framework for other common prediction tasks from PPM like attribute or outcome prediction.

Furthermore, as BEST relies on the historical occurrence probabilities of subtrace patterns, the hierarchical subtrace tree could be prone to concept drifts in the control flow. Currently, the occurrence of such a concept drift would require retraining of the model in an updated training set containing sequences subject to the concept drift. As we showed in our extended runtime analysis, the favorable training runtime of our proposed approach would feature reduced cost for retraining compared to established deep learning techniques. Nevertheless, we want to further investigate opportunities to

mitigate effects of changes in control flow over time, possibly via incremental learning strategies or improved pattern selection strategies that are robust to such disturbances.

Since the training of multiple specialized tree models for different process stages showed interesting but not generally improved results, we want to look into possibilities of refining the slicing method to be able to better grasp the starts and ends of actual phases in the process. Although we might take the current sequence length as an indication of a currently active process phase, there should still be potential in including also additional information from event attributes. Finally, as the process stages showed beneficial effects only for some of the datasets, we want to investigate how to perform a smarter stage slicing across the event sequences. We hope to be able to eliminate the detrimental effects that most likely occurred due to slicing of potentially coherent process phases into distinct stages. Possible avenues for improving the slicing mechanism implemented here might be to look into event abstractions (van Zelst et al. 2021) or to perform hierarchical modeling of process phases of different granularities.

Conclusion

In this work, we described BEST, an efficient tree-based prediction model for the task of Next Activity Prediction and Remaining Trace Prediction in the domain of Predictive Process Monitoring. Our model leverages a hierarchical structuring of bilaterally expanding subtrace patterns and makes use of efficient pattern matching as well as a choice of pattern selection methods to arrive at a suitable activity suffix prediction for ongoing process instances. As we can show with our experiments, BEST is able to achieve superior predictive performance for the task of Remaining Trace Prediction in comparison to established techniques from the domain of deep learning while being orders of magnitude faster to train. BEST is also capable of grasping intra-trace dynamics by training individual and specialized models for different process stages an ongoing case currently resides in. Moreover, we provide a thorough empirical evaluation using also a weighted probability metric in pattern selection, showing the trade-off between favoring local, i.e., conditional, and global pattern occurrence probabilities, the effect of pruning strategies for efficiency. Eventually, our approach shows to be lightweight and efficient, as we demonstrate in a dedicated runtime analysis and comparison with existing techniques for identical tasks.

Supplementary information

The online version contains supplementary material available at <https://doi.org/10.1007/s44311-026-00050-y>.

Supplementary Material

Author contributions

Simon Rauch led the development of the methodological core of the work, introducing the novel BEST framework, designing the experimental setup, implementing the full system, and driving the methodological refinements throughout the original and extended versions. Christian Frey contributed to the theoretical heart of BEST and contributed the formal underpinnings of the framework, providing the formalization of the model's central construct and supporting the experimental design and manuscripts' preparations. Andrea Maldonado shaped the conceptual positioning of the work within the broader literature, delineating the contribution relative to prior research, and additionally supported the experimental design and helped in writing the manuscript. Thomas Seidl is the PhD supervisor of the first author, Simon Rauch, and contributed to the supervision of the project, as well as providing constructive feedback on the overall structure of the paper. Additional co-authors for the extended version: Daniel Schuster contributed to the overall structure of the extended version and also provided support in the experimental design of the novelties being presented in the extended version. Gabriel Marquez Tavares helped in the experimental design of the extended version.

Funding

Open Access funding enabled and organized by Projekt DEAL.

Data availability

The implementation of the approach along with datasets used and experiments gathered are available und <https://github.com/lmu-dbs/BEST>. The used datasets and gathered experiments are additionally added to the supplementary files of this submission.

Declarations

Ethical approval

Not applicable.

Competing interests

The authors declare no competing interests.

Received: 30 November 2025 / Accepted: 1 June 2026

Published online: 19 June 2026

References

- Boytsov L (2011) Indexing methods for approximate dictionary searching: comparative analysis. *ACM J Exp Algorithmics* 16. <https://doi.org/10.1145/1963190.1963191>
- Buijs J (2022) Receipt phase of an environmental permit application process (WABO), CoSeLoG project. Eindhoven University of Technology. <https://doi.org/10.4121/12709127.v2>
- Camargo M, Dumas M, González-Rojas O (2019) Learning accurate LSTM models of business processes. In: LNCS. Springer, Cham, pp 286–302. https://doi.org/10.1007/978-3-030-26619-6_19
- Ceci M, Lanotte PF, Fumarola F, Cavallo DP, Malerba D (2014) Completion time and next activity prediction of processes using sequential pattern mining. In: *Discovery science*. Springer, Cham, pp 49–61. https://doi.org/10.1007/978-3-319-11812-3_5
- Ceravolo P, Comuzzi M, De Weerd J, Di Francescomarino C, Maggi FM (2024) Predictive process monitoring: concepts, challenges, and future research directions. *Process Sci* 1(1). <https://doi.org/10.1007/s44311-024-00002-4>
- Di Francescomarino C, Ghidini C (2022) Predictive process monitoring. Springer, Cham, pp 320–346. https://doi.org/10.1007/978-3-031-08848-3_10
- Diamantini C, Genga L, Potena D (2016) Behavioral process mining for unstructured processes. *J Intell Inf Syst* 47(1):5–32. <https://doi.org/10.1007/s10844-016-0394-7>
- Evermann J, Rehse J-R, Fettke P (2017) Predicting process behaviour using deep learning. *Decis Support Syst* 100:129–140. <https://doi.org/10.1016/j.dss.2017.04.003>
- Hinkka M, Lehto T, Heljanki K, Jung A (2019) Classifying process instances using recurrent neural networks. Springer, Cham, pp 313–324. https://doi.org/10.1007/978-3-030-11641-5_25
- Leemans M (2017) NASA crew exploration vehicle (CEV) software event log. Eindhoven University of Technology. <https://doi.org/10.4121/uuid:60383406-ffcd-441f-aa5e-4ec763426b76>
- Mannhardt F (2016) Sepsis cases - event log. Eindhoven University of Technology. <https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460>
- Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2019) Using convolutional neural networks for predictive process analytics. In: *ICPM*. pp 129–136. <https://doi.org/10.1109/icpm.2019.00028>
- Polato M (2017) Dataset belonging to the help desk log of an Italian company. University of Padova. <https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb>
- Polato M, Sperduti A, Burattin A, Leoni MD (2018) Time and activity sequence prediction of business process instances. *Computing* 100(9):1005–1031. <https://doi.org/10.1007/s00607-018-0593-x>
- Rama-Maneiro E, Vidal J, Lama M (2022) Deep learning for predictive business process monitoring: review and benchmark. *IEEE Trans Serv Comput* 1–1. <https://doi.org/10.1109/TSC.2021.3139807>
- Rama-Maneiro E, Patrizi F, Vidal J, Lama M (2024) Towards learning the optimal sampling strategy for suffix prediction in predictive monitoring. Springer, Switzerland, pp 215–230. https://doi.org/10.1007/978-3-031-61057-8_13
- Rauch S, Frey CMM, Maldonado A, Seidl T (2025) BEST: bilaterally expanding subtrace tree for event sequence prediction. Springer, Cham, pp 415–432. https://doi.org/10.1007/978-3-032-02867-9_25
- Steeman W (2013b) BPI challenge 2013, incidents. Ghent University. <https://doi.org/10.4121/uuid:500573e6-acc6-4b0c-9576-a5468b10cee>
- Steeman W (2013a) BPI challenge 2013, closed problems. Ghent University. <https://doi.org/10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11>
- Tax N, Genga L, Zannone N (2017) On the use of hierarchical subtrace mining for efficient local process model mining. In: *SIMPDA*. pp 8–22
- Tax N, Verenich I, Rosa ML, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. In: *CAiSE*. Springer, Cham, pp 477–492. https://doi.org/10.1007/978-3-319-59536-8_30
- Theis J, Darabi H (2019) Decay replay mining to predict next process events. *IEEE Access* 7:119787–119803. <https://doi.org/10.1109/ACCESS.2019.2937085>
- van der Aalst WMP (2022) Process mining: a 360 degree overview. In: van der Aalst WMP, Carmona J (eds) *Process Mining Handbook*. Springer, Cham, pp 3–34. https://doi.org/10.1007/978-3-031-08848-3_1
- van der Aalst WMP, Schonenberg MH, Song M (2011) Time prediction based on process mining. *Inf Syst* 36(2):450–475. <https://doi.org/10.1016/j.is.2010.09.001>. Special Issue: Semantic Integration of Data, Multimedia, and Services
- van der Spoel S, van Keulen M, Amrit C (2013) Process prediction in noisy data sets: a case study in a Dutch hospital. Springer, Berlin, Heidelberg, pp 60–83. https://doi.org/10.1007/978-3-642-40919-6_4

- van Dongen B (2012) BPI challenge 2012. Eindhoven University of Technology. <https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f>
- van Dongen B (2017) BPI challenge 2017. Eindhoven University of Technology. <https://doi.org/10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b>
- van Dongen B (2019). BPI challenge 2019. 4TU.Centre for research data. <https://doi.org/10.4121/uuid:d06aff4b-79f0-45e6-8ec8-e19730c248f1>
- van Zelst SJ, Mannhardt F, Leoni M, Koschmider A (2021) Event abstraction in process mining: literature review and taxonomy. *Granul Comput* 6(3):719–736. <https://doi.org/10.1007/s41066-020-00226-2>
- Wuyts B, Vanden Broucke S, De Weerd J (2024) SuTraN: an encoder-decoder transformer for full-context-aware suffix prediction of business processes. In: 2024 6th International Conference on Process Mining (ICPM), pp 17–24. <https://doi.org/10.1109/icpm63005.2024.10680671>
- Yap G-E, Li X-L, Yu PS (2012) Effective next-items recommendation via personalized sequential pattern mining. In: DASFAA. Springer, Berlin, Heidelberg, pp 48–64. https://doi.org/10.1007/978-3-642-29035-0_4

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.