Hierarchical Structuring of Bilaterally Expanding Subtrace Patterns for Efficient Tree-based Activity Suffix Prediction

Simon Rauch
rauch@dbs.ifi.lmu.de
Ludwig-Maximilians-Universität München
Christian M. M. Frey
University of Technology Nuremberg
Andrea Maldonado
University of Technology Nuremberg
Daniel Schuster
Ludwig-Maximilians-Universität München
Gabriel Tavares
Ludwig-Maximilians-Universität München
Thomas Seidl
Ludwig-Maximilians-Universität München

Research Article

Keywords: Sequential Pattern Mining, Tree Mining, Predictive Process Monitoring

Posted Date: December 10th, 2025

DOI: https://doi.org/10.21203/rs.3.rs-8244758/v1

Additional Declarations: No competing interests reported.

Hierarchical Structuring of Bilaterally Expanding Subtrace Patterns for Efficient Tree-based Activity Suffix Prediction

Simon Rauch[1,2,5*], Christian M.M. Frey[3], Andrea Maldonado[4,5], Daniel Schuster[1,5], Gabriel Tavares[1,5], Thomas Seidl[1,5]

[1*]Data Base Systems and Data Mining AI Lab, Ludwig-Maximilians University Munich, Oettingenstr. 67, Munich, 80538, Bavaria, Germany.
[2]Division Supply Chain Services, Fraunhofer IIS, Nordostpark 84, Nuremberg, 90411, Bavaria, Germany.
[3]Department Computer Science & Artificial Intelligence - Machine Learning Lab, University of Technology, Dr.-Luise-Herzberg-Straße 4, Nuremberg, 90461, Bavaria, Germany.
[4]School of Engineering and Design, Technical University of Munich, Lise-Meitner-Str. 9, Ottobrunn, 85521, Bavaria, Germany.
[5]Munich Center for Machine Learning, Ludwig-Maximilians University Munich, Oettingenstr. 67, Munich, 80538, Bavaria, Germany.

*Corresponding author(s). E-mail(s): rauch@dbs.ifi.lmu.de;
Contributing authors: christian.frey@utn.de; andrea.maldonado@tum.de; schuster@dbs.ifi.lmu.de; tavares@dbs.ifi.lmu.de; seidl@dbs.ifi.lmu.de;

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 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 effiecient 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

1 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 into the main drivers of process execution as well as key performance indicators (KPIs) like the scrap rate or throughput times. Moreover, 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 a as early as possible prediction of the future process and not only for the next few activities but 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 the 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' [1] 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 ([2]).

2 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 similarities and differences to a landmark paper for temporal predictions by van der Aalst et al. [2].

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. [3], Camargo et al. [4], Evermann et al. [5], and Hinkka et al. [6] 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. [7] 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. [4] and Evermann et al. [5] for the case of predicting the complete remaining activity sequence. Moreover, the approaches of Tax et al. [3] and Hinkka et al. [6] perform very well for the case of Next Activity Prediction.

Tackling the popularity of LSTM networks in PPM, Wuyts et al. [8] 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 a 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. [7] 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 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 aproaches 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. [9] 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. [10] 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 share 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.

Regarding the mining of hierarchical subtraces, there are already contributions by Diamantini et al. [11] and Tax et al. [12], 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 [2] 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 [3, 6], which integrate time prediction. Although there are similarities with respect to possible insights 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.

3 Preliminaries

For preliminary concepts, we repeat the introduced key concepts from the original contribution [1]. We start with basic notations for business processes [13].

Definition 1 (Event/Trace/Event Log) An *event* is defined as a tuple $\epsilon \coloneqq (c, a, t)$ 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 $\sigma \coloneqq \langle \epsilon_1, \ldots, \epsilon_{|\sigma|} \rangle$ that belong to the same process instance, i.e., case id. An *Event Log* L is a multiset of traces $\{\sigma_1, \ldots, \sigma_{|L|}\}$ that are recorded during the execution of a given business process. The activity sequence S of trace σ is defined as $S(\sigma) \coloneqq \langle a_1, \ldots, a_{|\sigma|} \rangle$. 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 [1] and follow up with a definition of event prefixes and suffixes of process events, similar to [7]:

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 $\langle a_{k-l}^{(i)}, \ldots, a_{k-1}^{(i)} \rangle$ 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 $\langle a_{k+1}^{(i)}, \ldots, a_{k+l}^{(i)} \rangle$ 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 [1], 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. [2]. 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.

4 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 [1]. A repository containing a reference to an implementation of the method as well as all datasets used and results gathered is available online[1].

We provide the needed theory of our approach in the following sections and reuse definitions that were developed for the corresponding conference paper [1]. 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 4.1 and 4.2. Subsequently, the mechanism for the prediction of future activities and complete remaining traces

[1] https://doi.org/10.5281/zenodo.15547095– remark to the editors: we provide supplementary experimental results focussed on the extended parts of this paper via the submission portal.

based on the tree generated structure is outlined in Section 4.3. The definitions and algorithmic procedures listed in the following sections are mostly kept from our original contribution [1]. We add Def. 8 and Equations 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.

4.1 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 Def. 5. After gathering all possible extended patterns, the patterns are organized in the tree structure we define in Def. 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 $P := \langle a_i^{(S)}, \ldots, a_{c(P)}^{(S)}, \ldots, a_j^{(S)} \rangle \subseteq S$ of a sequence $S := \langle a_1^{(S)}, \ldots, a_{|S|}^{(S)} \rangle$, 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 $\tilde{P} := \langle a_{i-l^p}^{(S)}, \ldots, a_i^{(S)}, \ldots, a_{c(P)}^{(S)}, \ldots, a_j^{(S)}, \ldots, a_{j+l^s}^{(S)} \rangle$. 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 $\omega : \mathbb{R} \to E$ denoting a weighting function for edges and $\nu : \mathcal{P} \to V$ 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., $\nu(\emptyset) =: root(BEST)$.
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

$\{P^{S_1}, \ldots, P^{S_\mathcal{L}}\}$ 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 $a^{(P_{pa})}_{c(P_{pa})} = a^{(P_{ch})}_{c(P_{ch})}$.

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 are containing 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

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

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

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

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

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

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 Equation 2 accounting for the length differences always equals to 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 logarithmic local distances of the nodes along the path of traversal between the tree's root node and the pattern in question (see Figure 2).

4.2 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 $\mathcal{S}$ 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

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

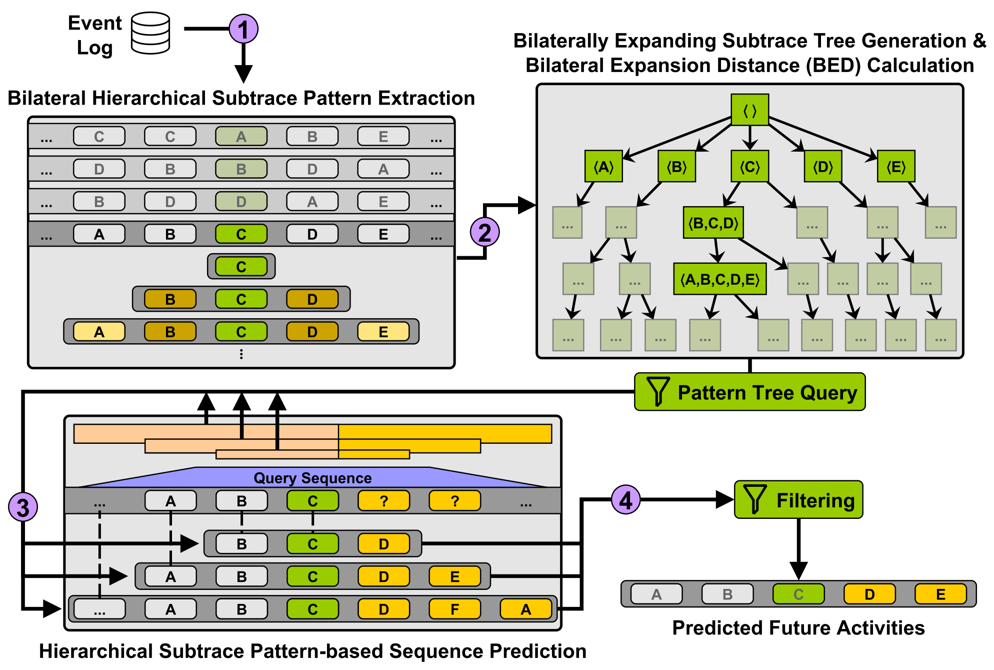

Event Log
1
Bilateral Hierarchical Subtrace Pattern Extraction
Bilaterally Expanding Subtrace Tree Generation & Bilateral Expansion Distance (BED) Calculation
2
⟨ ⟩
⟨A⟩
⟨B⟩
⟨C⟩
⟨D⟩
⟨E⟩
⟨B,C,D⟩
⟨A,B,C,D,E⟩
Pattern Tree Query
Query Sequence
3
4
Filtering
Predicted Future Activities
Hierarchical Subtrace Pattern-based Sequence Prediction

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).

where D_{max} is the maximum tree depth. Having all patterns of the desired sizes extracted from $\mathcal{S}$, 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 the relevance and mere occurrence of subtrace patterns 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.

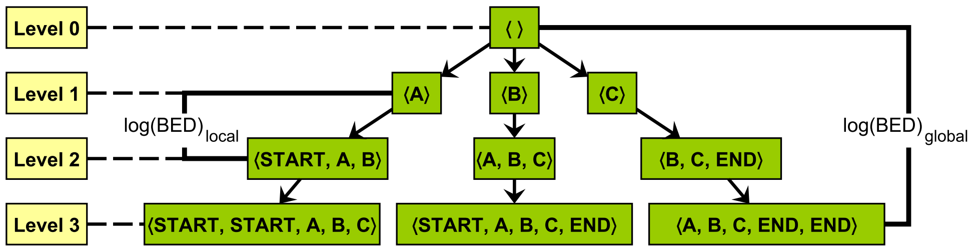


Fig. 2: Hierarchical bilaterally expanding subtrace tree for a single trace event log $L = \{\sigma_{ABC}\}$ with $S(\sigma_{ABC}) = \langle A, B, C \rangle$. Also schematically depicted are the local edge weights in terms of the logarithmic BED and the global logarithmic BED for subtrace pattern $\langle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

4.3 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 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 sequence[2]. The separate branches of our tree come in handy here as we only need to

[2] We show this also schematically in the lower left panel of our framework visualization in Fig. 1

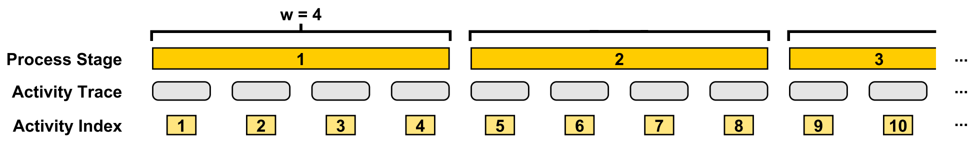


Fig. 3: Process stage slicing of activity traces for process stage width of $w = 4$.

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
```

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 decendants 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 effient as the deeper levels of the tree are only entered in case of a pattern match.

From the set of matching patterns, we apply a pattern selection strategy to arrive a a chosen predicted pattern. In this step, 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:

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

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

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 $S$
while $last(S) \neq END$ do
    $ps \leftarrow psL \leftarrow psR \leftarrow \lfloor |S|/pswidth \rfloor$
    $predP = predictForStage(ps, S, method, params)$
    while $predP = NULL$ do
        $predL \leftarrow predictForStage(psL, S, method, params))$
        $predR \leftarrow predictForStage(psR, S, method, params))$
        if $getLocalBED(predL) < getLocalBED(predR)$ then
            $predP \leftarrow predL$
        else
            $predP \leftarrow predR$
        end if
        $psL \leftarrow max(psL - 1, 0)$
        $psR \leftarrow min(psR + 1, psMax)$
    end while
    $pred \leftarrow predP\left[\left(\left\lceil \frac{|predP|}{2} \right\rceil + 1\right) : |predP|\right]$
    $S = append(S, pred)$
end while
return $S$
```

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

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 $predP$
if $|candidatePatterns| = 0$ then
    return $NULL$
end if
if $method = "BED"$ then
    $predP \leftarrow argmin(getLocalBED(candidatePatterns))$
    if $|predP| > 1$ then
        $predP \leftarrow argmax(getPatternSize(predP))$
        if $|predP| > 1$ then
            $predP \leftarrow argmin(getGlobalBED(predP, \langle\,\rangle)$
        end if
    end if
else
    if $method = "WEIGHTED"$ then
        $predP \leftarrow argmax(getPrW(candidatePatterns, params))$
        if $|predP| > 1$ then
            $predP \leftarrow argmax(getPatternSize(predP))$
        end if
    end if
end if
if $|predP| > 1$ then
    $predP \leftarrow randomChoice(predP)$
end if
return $predP$
```

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 extraordinary 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

Dataset	events	cases	activities	trace length mean	median	max	variants
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

Table 1: Datasets used for performance evaluation

*: In these logs, the activity identifier is generated by a concatenation of the raw activity identifier and the activity lifecycle information (SCHEDULE, START, COMPLETE).

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.

5 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 dinstinct 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 a 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.

5.1 Evaluation Setup

Datasets.

For our evaluation, we use the following datasets: Helpdesk [14], BPI2012 [15] 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 [16], Incidents [17]), Env Permit [18], Sepsis [19], Nasa [20]. 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 $[0.0, 0.025, 0.1, 0.2, 0.5, 1.0]$. For each parameter combination, we train our model in a 5-fold cross validation setting. We split the folds by case id leading to each of the folds consisting of 20% of all available cases. Effectively, we perform a 80%-20% train-test-split 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 $j = 0, \ldots, |\sigma_i|$ to perform our experiments where j is the length of the resulting prefix starting from an empty prefix. For both tasks — Next Activity Prediction and Remaining Trace Prediction — we report the average results over the 5 folds for their individually best parameter combination in Table 2.

5.2 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. [7] for the achievable performance of multiple approaches ([3–6, 21, 22] 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 setup[3]. The achieved accuracies of predicted next activities and similarities of predicted traces we report in Table 2 are identical to the ones reported in [7] 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 an additional and more recent approach by Wuyts et al. [8] using a Transformer network for suffix prediction. We performed experiments based on the available implementations and specified parameter setting accompanying their publication[4]. 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. Eventually, we also include a recent deep reinforcement learning approach for activity suffix prediction by Rama-Maneiro et al. [23] and their reported performances into our comparison. They maintain a steady evaluation routine between their contributions,

[3]The results reported in [7] for Camargo et al. [4] are split up between two individual approaches (*random choice* and *argmax*). For each dataset in Table 2 we report results of the better performing approach. We do the same for Theis et al. [22].

[4]Wuyts et al. [8] 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 with a random split and abstained from performing filtering operations. Overall, we expect to have reached a very high amount of parity between all approaches included in our comparison.

			BPI'12				BPI'13				
		Helpdesk	Full	W	C	WC	Closed Prob-lems	Incidents	Env Permit	Sepsis	Nasa
Camargo [4]	NAP	.8293	.8328	.7640	.7793	🟧 .6895	.5467	.6668	.8578	–	–
Hinkka [6]	NAP	.8308	🟩 .8665	🟧 .8478	🟩 .8064	.7054	.6347	🟩 .7469	.8443	🟦 .6350	.8842
Theis [22]	NAP	.7969	.8096	🟩 .8686	.7575	🟩 .8384	.5948	.5941	.8512	.5814	🟧 .8896
Pasquadibisceglie [21]	NAP	.8393	.8325	.8119	.7460	.6834	.4745	.4603	🟧 .8669	.5615	.8827
Tax [3]	NAP	🟧 .8419	🟦 .8546	🟦 .8535	🟦 .8038	.6979	🟦 .6401	🟧 .7009	.8571	🟩 .6422	🟩 .8944
Wuyts [8]	NAP	🟦 .8443	.8231	.8173	🟧 .7897	🟦 .7066	🟧 .6358	.6481	🟦 .8706	.5526*	.6792*
Our Approach	NAP	🟩 .8609	🟧 .8441	.8329	.7691	.6635	🟩 .6867	🟦 .7039	🟩 .8755	🟧 .5982	🟦 .8915
Camargo [4]	RTP	🟧 .9110	🟧 .3891	🟧 .3110	🟧 .4495	.3191	.6641	🟧 .5294	🟦 .8440	–	–
Evermann [5]	RTP	.8354	.1986	.2800	.2693	🟧 .3372	.6416	4730	.5713	🟧 .2693	.1218
Tax [3]	RTP	.8695	.1409	.0975	.1717	.0789	.5824	.3336	.8163	.1158	.2320
Wuyts [8]	RTP	🟩 .9191	.2872	.1142	.3043	.1170	🟩 .7368	.5112	🟩 .8709	.1286*	🟧 .4402*
Rama-Maneiro [23]	RTP	.8989	🟦 .4061	🟩 .4000	🟦 .4711	🟩 .3829	🟧 .6933	🟩 .6230	.8369	🟩 .4285	🟩 .4981
Our Approach	RTP	🟦 .9116	🟩 .4449	🟦 .3794	🟩 .4846	🟦 .3628	🟦 .7352	🟦 .5671	🟧 .8388	🟦 .4103	🟦 .4887

*: Results were achieved with the non-data-aware (NDA) architecture of Wuyts et al. [8].

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.

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. Finally, to evaluate our approach in comparison with the remaining contributions, we also use the same and already established evluation metrics for Next Activity Prediction and Remaining Trace Prediction — accuracy of the forecasted next activity and the normalized Damerau-Levenshtein-Similarity [24] 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 learning approach by Rama-Maneiro et al. [23], 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. In fact, only the approaches of Camargo et al. [4] and Rama-Maneiro et al. [23] 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.

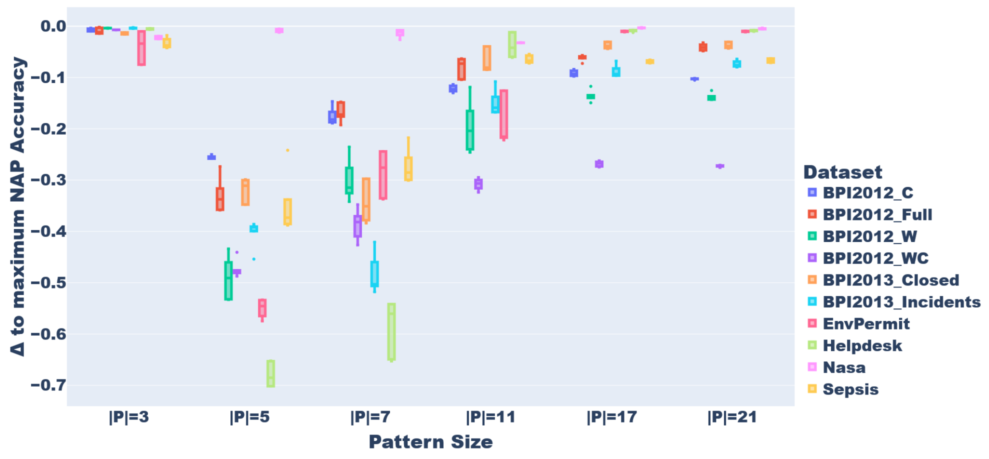


Fig. 4: Distribution of forecasting accuracies in terms of distance to the maximum achieved value across all runs for the specified pattern size used in the prediction.

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 shows 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. [6] and Theis et al. [22] (BPI'12 W, 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.

5.3 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 Figures 4 and 5. 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

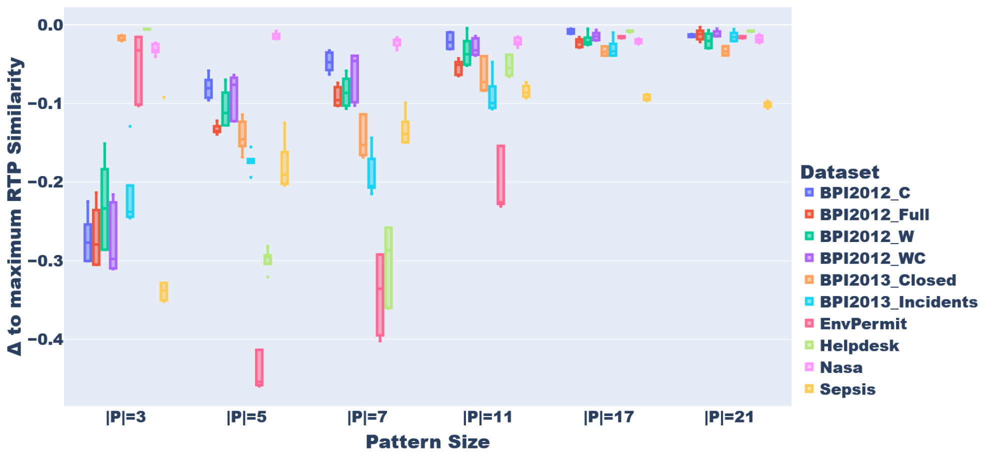


Fig. 5: Distribution of the Normalized Damerau-Levenshtein-Similarity values in terms of distance to the maximum achieved value across all performed experiments for the specified pattern size used in the prediction.

trend for an increase of the pattern length, the maximum accuracy values observed for short patterns of pattern size $|P| = 3$ is 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 show 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 Figure 6. 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 4 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.

5.4 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 chosing a fitting weight parameter value w with respect to the dataset analyzed and

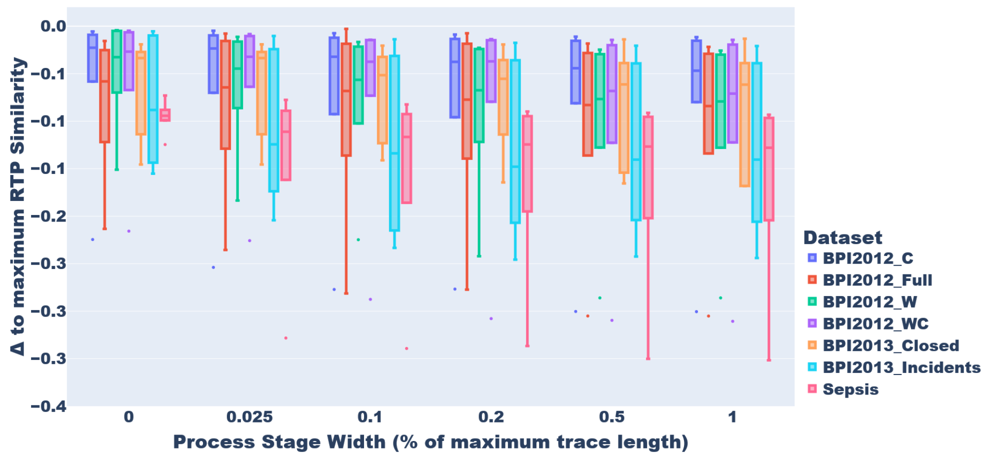


Fig. 6: Distribution of Damerau-Levenshtein-Similarity in terms of distance to the maximum achieved value across all runs for different process stage widths in terms of percentage of the maximum trace length used for mining the patterns. We omitted event logs Helpdesk, EnvPermit, Nasa as the effect is not as pronounced there.

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 Figure 7 and a subset of the remaining datasets analyzed in the same manner in Figure 8.

Regarding the results of the BPI2012 datasets in Figure 7 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 Figure 8. 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 Figure 4 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 Figure 7 as well as dataset EnvPermit and BPI2013 Incidents in Figure 8 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 Figures 4 and 5).

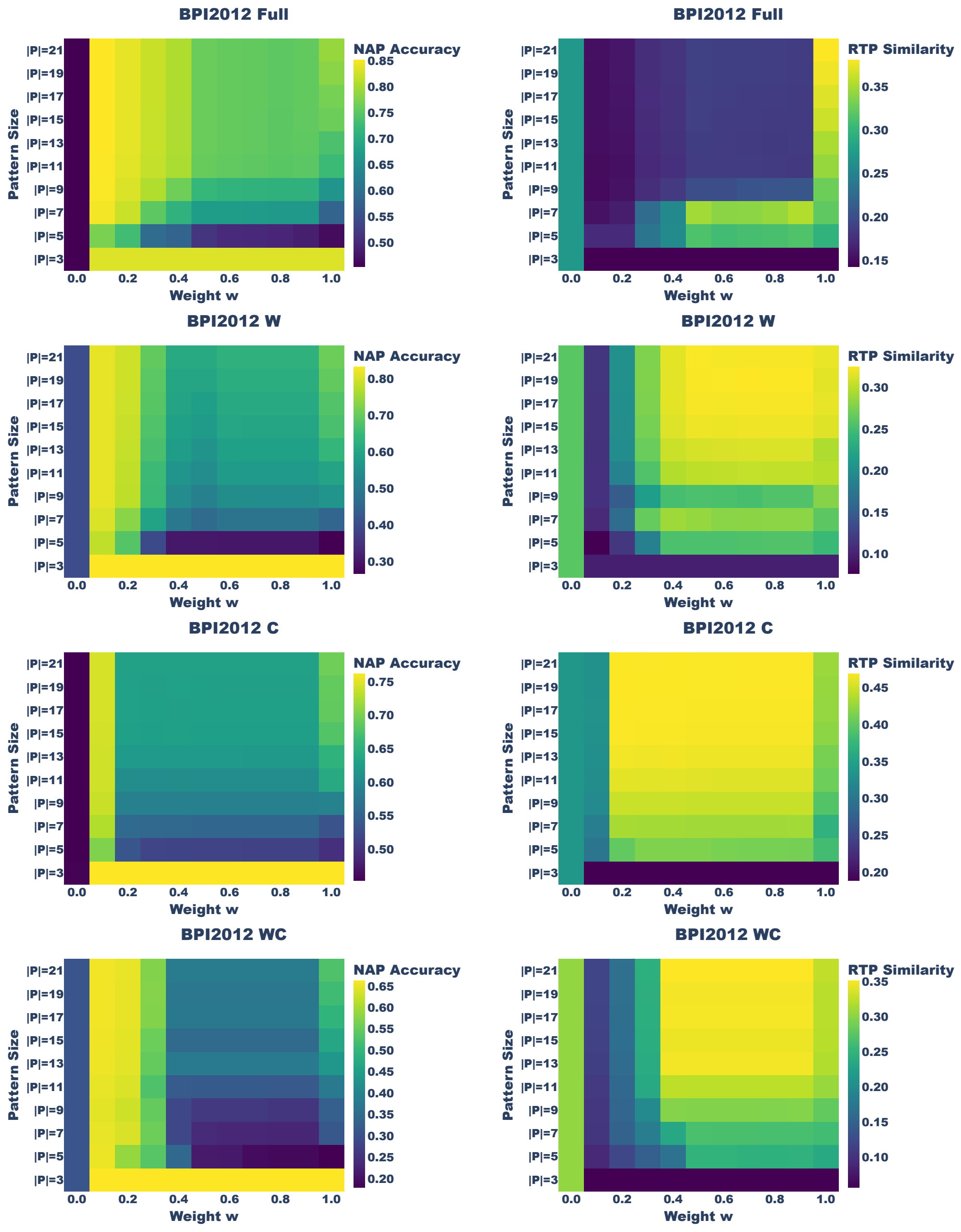


Fig. 7: Visual Sensitivity Analysis of the weight parameter w for trees generated with different pattern sizes for the dataset BPI2012 along with its variations BPI2012 W, BPI2012 C and BPI2012 WC. The left column of plots shows results for task NAP, the right column of plots shows results for task RTP

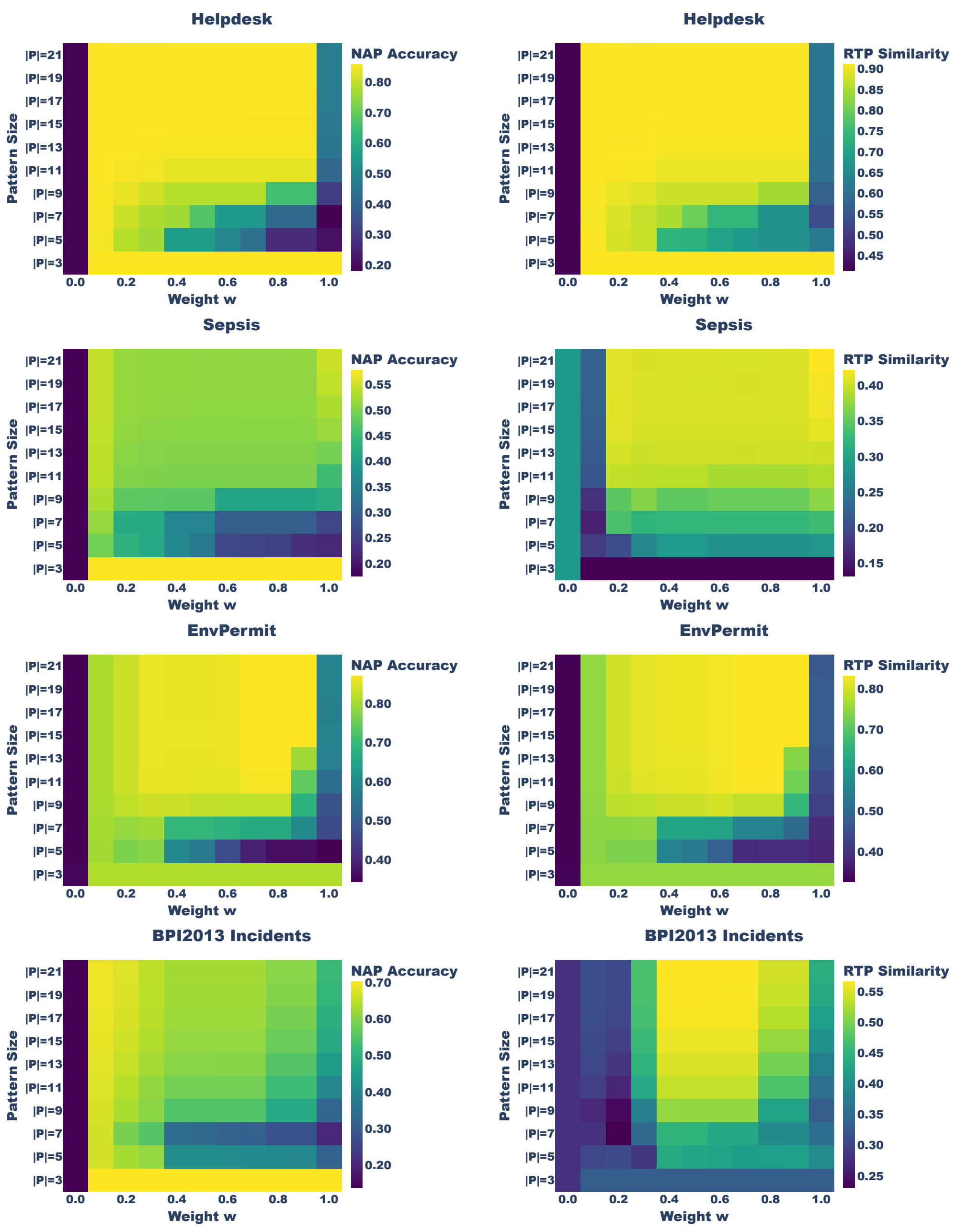


Fig. 8: Visual Sensitivity Analysis of the weight parameter w for trees generated with different pattern sizes for the datasets Helpdesk, Sepsis, EnvPermit and BPI2013 Incidents. The left column of plots shows results for task NAP, the right column of plots shows results for task RTP

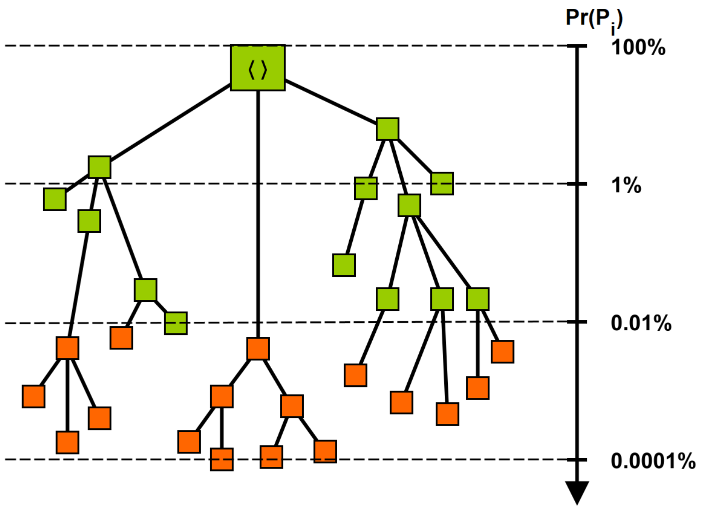
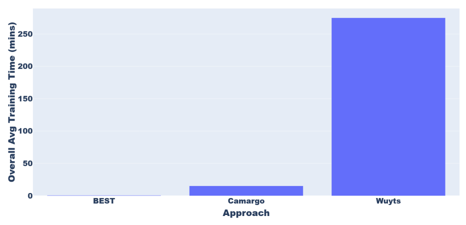


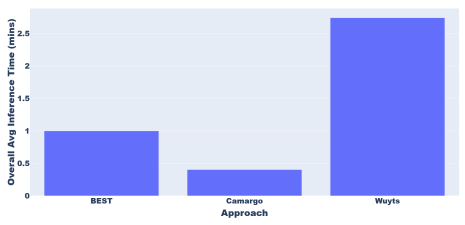
Fig. 9: Schematic visualization of a global occurrence probability cutoff at a cutoff probability of 0.01%. Pattern nodes of the tree are visualized (●pruned, ●: unpruned) comparison of non-constrained tree generation and tree generation with applied pre-pruning via global pattern probability cutoff.

5.5 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 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. 9. 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 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.



(a) Average training times



(b) Average inference times

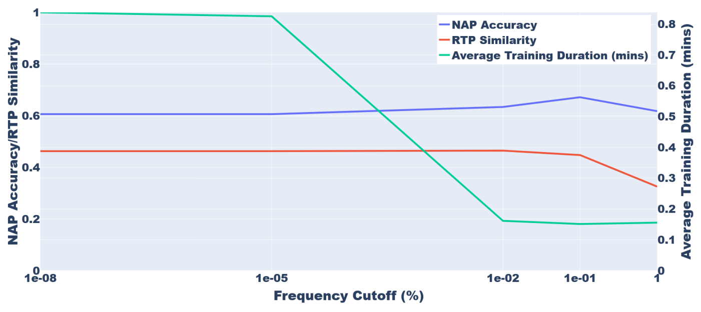
Fig. 10: Training and inference times averaged over all datasets for BEST and a subset of competitors from different architectures (LSTM - Camargo [4], Transformer - Wuyts [8])

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 ≈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 5.6 while also considering the implications for the resulting predictive performance.

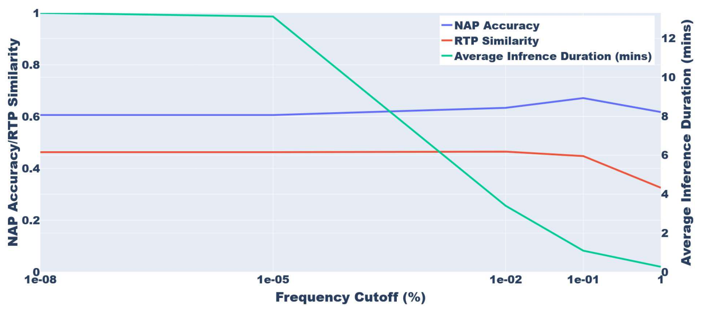
5.6 Runtime Analysis

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. [4] (LSTM) and Wuyts et al. [8] (Transformer). Figure 10 provides a visual comparison of training and inference times, respectively. Moreover, we provide an analysis on the effect of the pruning by a global probability cutoff in terms of training and inference times, while also looking at the effects on predictive accuracy of our approach. Figs. 11a and 11b depict the interconnection between setting different probability cutoff values for training time and inference time, respectively.

Regarding the training times, we achieve an average duration (across all datasets) of training our models that is orders of magnitude smaller than existing approaches. 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 framework is evaluated on the CPU only. The training and inference times of the evaluated deep learning architectures were performed on a NVIDIA GeForce RTX 2080 Ti GPU featuring 11 GB of GDDR6 VRAM, experiments of our approach were executed on a virtual machine running with an Intel Xeon Skylake processor featuring 8 CPU cores clocked at 2.7 GHz and 64GB of RAM.



(a) Average training times



(b) Average inference times

Fig. 11: Connection between different probability cutoff values for training and inference times and the corresponding achieved average accuracies and similarities across all datasets averaged over all datasets for BEST

In Fig. 11a shows the stark positive effect of employing a pre-pruning strategy in the training procedure by executing a global pattern probability cutoff. The reduction in average training time amounts to about a decrease of up to 80%. Consequently, we can record a drop off in similarity of predicted future activity traces for a probability cutoff value of 1%. However, the decreased training duration can still be upheld by using a smaller probability cutoff of 0.001% while still being able to have steady performance for the prediction of activity suffixes. Regarding inference time, we can see a similar pattern, and inference time can be reduced by an even higher factor of >97%. 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 high-performing prediction models.

5.7 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. 4), 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 also look into ways of incorporating additional event information into the training and inference phases. 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.

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 [25] or to perform hierarchical modeling of process phases of different granularities.

6 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 hierarchical 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.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

Declarations

Ethical Approval. Not applicable.

Funding. Not applicable.

Availability of data and materials. Implementations of the approach along with datasets used and experiments gathered are available under https://github.com/lmu-dbs/BEST.

References

[1] Rauch, S., Frey, C.M.M., Maldonado, A., Seidl, T.: BEST: Bilaterally Expanding Subtrace Tree for Event Sequence Prediction, pp. 415–432. Springer, ??? (2025)

[2] van der Aalst, W.M.P., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information Systems **36**(2), 450–475 (2011) https://doi.org/10.1016/j.is.2010.09.001 . Special Issue: Semantic Integration of Data, Multimedia, and Services

[3] Tax, N., Verenich, I., Rosa, M.L., Dumas, M.: Predictive Business Process Monitoring with LSTM Neural Networks. In: CAiSE, pp. 477–492. Springer, ??? (2017)

[4] Camargo, M., Dumas, M., González-Rojas, O.: Learning Accurate LSTM Models of Business Processes. In: LNCS, pp. 286–302. Springer, ??? (2019)

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

[6] Hinkka, M., Lehto, T., Heljanko, K., Jung, A.: Classifying Process Instances Using Recurrent Neural Networks, pp. 313–324. Springer, ??? (2019)

[7] Rama-Maneiro, E., Vidal, J., Lama, M.: Deep Learning for Predictive Business Process Monitoring: Review and Benchmark. IEEE Transactions on Services Computing, 739–756 (2023)

[8] Wuyts, B., Vanden Broucke, S., De Weerdt, J.: SuTraN: an Encoder-Decoder Transformer for Full-Context-Aware Suffix Prediction of Business Processes. In: ICPM, pp. 17–24. IEEE, ??? (2024)

[9] Yap, G.-E., Li, X.-L., Yu, P.S.: Effective next-items recommendation via personalized sequential pattern mining. In: DASFAA, pp. 48–64 (2012). Springer

[10] Ceci, M., Lanotte, P.F., Fumarola, F., Cavallo, D.P., Malerba, D.: Completion Time and Next Activity Prediction of Processes Using Sequential Pattern Mining. In: Discovery Science, pp. 49–61. Springer, ??? (2014)

[11] Diamantini, C., Genga, L., Potena, D.: Behavioral process mining for unstructured processes. Journal of Intelligent Information Systems **47**(1), 5–32 (2016)

[12] Tax, N., Genga, L., Zannone, N.: On the use of hierarchical subtrace mining for efficient local process model mining. In: SIMPDA, pp. 8–22 (2017)

[13] Aalst, W.M.P.: Process Mining: A 360 Degree Overview, pp. 3–34. Springer, ??? (2022)

[14] Polato, M.: Dataset belonging to the help desk log of an Italian Company. University of Padova (2017)

[15] Dongen, B.: BPI Challenge 2012. Eindhoven University of Technology (2012)

[16] Steeman, W.: BPI Challenge 2013, closed problems. Ghent University (2013)

[17] Steeman, W.: BPI Challenge 2013, incidents. Ghent University (2013)

[18] Buijs, J.: Receipt phase of an environmental permit application process (WABO), CoSeLoG project. Eindhoven University of Technology (2022)

[19] Mannhardt, F.: Sepsis Cases - Event Log. Eindhoven University of Technology (2016)

[20] Leemans, M.: NASA Crew Exploration Vehicle (CEV) Software Event Log. Eindhoven University of Technology (2017)

[21] Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using Convolutional Neural Networks for Predictive Process Analytics. In: ICPM, pp. 129–136. IEEE, ??? (2019)

[22] Theis, J., Darabi, H.: Decay Replay Mining to Predict Next Process Events. IEEE Access **7**, 119787–119803 (2019)

[23] Rama-Maneiro, E., Patrizi, F., Vidal, J., Lama, M.: Towards Learning the Optimal Sampling Strategy for Suffix Prediction in Predictive Monitoring, pp. 215–230. Springer, ??? (2024)

[24] Boytsov, L.: Indexing methods for approximate dictionary searching: Comparative analysis. ACM Journal of Experimental Algorithmics **16** (2011)

[25] Zelst, S.J., Mannhardt, F., Leoni, M., Koschmider, A.: Event abstraction in process mining: literature review and taxonomy. Granular Computing **6**(3), 719–736 (2020) https://doi.org/10.1007/s41066-020-00226-2