Pfeiffer, Peter; Abb, Luka; Fettke, Peter; Rehse, Jana-Rebecca

Learning from the Data to Predict the Process

Business & Information Systems Engineering

Bus Inf Syst Eng 67(3):357–383 (2025)
https://doi.org/10.1007/s12599-025-00936-4

RESEARCH PAPER

Learning from the Data to Predict the Process

Generalization Capabilities of Next Activity Prediction Algorithms

Peter Pfeiffer · Luka Abb · Peter Fettke · Jana-Rebecca Rehse

Received: 31 July 2024 / Accepted: 27 January 2025 / Published online: 22 March 2025

Abstract Predictive process monitoring (PPM) aims to forecast how a running process instance will unfold in the future, e.g., which activity will be executed next. For this purpose, PPM techniques rely on machine learning models trained on historical event log data. Such models are assumed to learn an implicit representation of the process that accurately reflects the behavior contained in the data, so that they can be used to make correct predictions for new traces with unseen behavior. This capability, called generalization, is fundamental to any machine learning application. However, researchers currently have a limited understanding of what generalization means in a PPM context and how it relates to the characteristics of event logs. In the paper, the authors discuss the generalization capabilities of PPM approaches, focusing on next activity prediction. They develop a framework for generalization in PPM, derived from the understanding of the term in general machine learning. The framework is applied to next activity prediction by developing concrete prediction scenarios, creating corresponding event logs, and using these logs to empirically evaluate the generalization capabilities of state-of-theart models. The evaluation shows that next activity prediction models generalize well in almost all scenarios.

Accepted after one revision by the editors of the Special Issue.

P. Pfeiffer · P. Fettke
German Research Center for Artificial Intelligence (DFKI), Campus D3 2, 66123 Saarbrücken, Germany

P. Pfeiffer (✉) · P. Fettke
Saarland University, Campus D3 2, 66123 Saarbrücken, Germany
e-mail: peter.pfeiffer@dfki.de

L. Abb · J.-R. Rehse
University of Mannheim, L 15, 68161 Mannheim, Germany

Keywords Process prediction · Predictive process monitoring · Next activity prediction · Generalization · Validity issues

1 Introduction and Motivation

Predictive process monitoring (PPM) is a branch of process mining that aims to forecast how a running process instance will unfold in the future (Di Francescomarino and Ghidini 2022). This may concern what the outcome of the process instance will be, how long it will take to complete, or which activities will be executed next (Rehse et al. 2018). The first approaches to PPM relied on explicit models of process behavior, such as transition systems (van der Aalst et al. 2011) or probabilistic automata (Breuker et al. 2016). However, the vast majority of recent research treats PPM as a self-supervised machine learning problem (Neu et al. 2021; Rama-Maneiro et al. 2021; Pasquadibisceglie et al. 2022; Pfeiffer et al. 2021) and aims to solve it with deep neural networks (Evermann et al. 2017).

In contrast to many other process mining techniques, PPM is forward-facing: It aims to identify process execution problems like delays or compliance violations *before* they occur (Di Francescomarino and Ghidini 2022). Based on these predictions, a process manager can preemptively take actions to avoid and mitigate the problems, enabling a real-time management of the business process (Evermann et al. 2017). However, this capability hinges on the quality of the predictions that the machine learning model makes. If they are wrong, the manager might take unnecessary or even harmful actions, which can impede a smooth execution of the business process instead of supporting it.

To make correct predictions about ongoing process executions, the machine learning model creates an implicit representation of the process, which is learned from the data seen during training (Evermann et al. 2017). This representation is meant to accurately cover the underlying process (Pfeiffer 2022). In particular, it is expected to precisely fit the data it was trained on while still being flexible enough to make correct predictions for traces not part of its training data. This capability of machine learning models to handle unseen samples is known as *generalization*. Considered a fundamental goal in machine learning literature (Goodfellow et al. 2016; Bishop 2006), generalization is motivated by the observation that training conditions often differ from the real-world application conditions in which a model needs to perform effectively.

In PPM, several factors can cause discrepancies between training and application settings. For example, because event logs are typically assumed to be incomplete (Buijs et al. 2014), the training data will not capture all possible control flow variants. It is therefore crucial for prediction models to generalize from the limited exampled in the training data to the unseen variants in the application. Furthermore, processes are executed in dynamic environments, so that both the process itself and environmental factors (e.g., the involved resources) can change over time. The models need to cope with these changes to perform well in an application setting. These factors underscore the importance of generalization: To make correct predictions in an application setting, the machine learning model needs to learn a good representation of the *process* instead of merely a good representation of the training data. To achieve this, researchers need a conceptual understanding of what generalization means in a PPM context and how it relates to the characteristics of the event logs used for this task.

Despite its importance, however, the concept of generalization has received minimal attention in existing PPM literature. In this paper, we aim to address this by discussing the generalization capabilities of PPM approaches, with a specific focus on next activity prediction. This prediction task, which is the focus of most state-of-the-art PPM approaches (Neu et al. 2021), intends to determine the most likely next activity in an ongoing process execution. Next activity prediction is typically seen as a classification task (Rama-Maneiro et al. 2021): Based on historical execution data of the business process in question, a machine learning model is trained to predict the most likely next activity for a given trace prefix. Therefore, the model is shown incomplete traces from historical data so that it learns to predict the correct label among a set of known activity classes (Abb et al. 2024).

In the following, we first show that our limited understanding of generalization in PPM has lead to validity issues in previous experiments on next activity prediction. To address these issues, we then derive a conceptual frame for generalization in PPM, which is based on the understanding of the term in broader machine learning research as well as the specific nature of event data. Applying this frame to the concrete task of next activity prediction, we develop concrete prediction scenarios in which models would need to generalize. We create event logs for each scenario and empirically evaluate state-of-the-art prediction models to assess their generalization capabilities. Finally, we validate our findings on more complex as well as real-world event logs and discuss the transferability to other PPM tasks.

This article is an extended and revised version of our original conference publication (Abb et al. 2024). We have significantly extended the alignment between generalization in machine learning with generalization in process prediction by conceptually discussing the aim and challenges of generalization in next activity prediction. The generalization scenarios have been revised and expanded with scenarios that combine multiple single-source scenarios. Further, experiments using simulated event logs based on the generalization scenarios as well as on real-world event logs have been performed to evaluate the ability of state-of-the-art next activity prediction models to generalize on unseen process behavior. Lastly, related work and the validity issues have been revised and extended.

The remainder of the paper is structured as follows: In Sect. 2, we discuss relevant preliminaries, background, and related work. In Sect. 3, we explain the generalization-related validity issues that exist in many experiments on next activity prediction. Next, we discuss the concept of generalization, first in the general machine learning context (Sect. 4) and then specifically for PPM (Sect. 5). Focusing on next activity prediction, we then derive concrete generalization scenarios in event logs in Sect. 6 and perform experiments to assess the generalization capabilities of existing next activity prediction algorithms in Sect. 7. We discuss the results in Sect. 8, before concluding the paper in Sect. 9.

2 Foundations

In this section, we introduce the foundations for our work. We describe the preliminaries in Sect. 2.1, discuss the scientific background in Sect. 2.2, and review related work in Sect. 2.3

2.1 Preliminaries

2.1.1 Log Data

The input to PPM is *event log data*, gathered from the execution of business processes in information systems. An event log L is a collection of cases, each representing a complete process execution. A case is represented by a *trace* c, i.e., a sequence of events $\langle e_1, \ldots, e_n \rangle$ of length n. An event e is a tuple of attribute values and has two mandatory attributes: the *activity* and the *case ID*. In addition, events can have additional attributes, such as a timestamp or an executing resource, which provide further details about them. We refer to these attributes as *context attributes*. We write $\langle A, B, \ldots, H \rangle$ to indicate the trace with events which possess the *activity* attribute values A, B to H. In PPM, we are interested in predicting the future behavior of running cases, which are represented by trace prefixes. A *trace prefix* of a trace c of length p is defined as a subsequence $\langle e_1, \ldots, e_p \rangle$, with $1 \leq p < n$.

2.1.2 Next Activity Prediction

The goal of next activity prediction is to predict which activity will be performed next in a running case. Formally, this problem is framed as a multi-class classification task, where each class represents one activity. For each trace c in a given event log, pairs (x, y) of features x and labels y are created. For a running case, x represents the prefix of c with length p containing the features that are given to the model. The activity at position $p + 1$ of c, i.e., the next activity, which should be predicted by the model, is represented by the label y. These pairs (x, y) are provided to a machine learning model so that it learns a probability distribution $q(y)$ over next activities.

To train and evaluate the prediction model, the event log is split into two parts, the training split L_{train} and the test split L_{test}. The model is trained on the prefix-label pairs from the training split L_{train} and evaluated on those from the test split L_{test}. For evaluation purposes, for each prefix x, the associated prediction $\hat{y}$, i.e., the activity with the highest probability in $q(y)$, is compared with the ground truth label y, which then informs the computation of aggregated performance measures such as accuracy and F1 score.

2.2 Background

In this section, we provide the necessary background for generalization in PPM. We first distinguish generalization in a machine learning context from generalization in process discovery. We then dive deeper into the topic of PPM and how generalization is addressed there.

2.2.1 Generalization in Process Discovery

A fundamental task in process mining is process discovery, which generates a process model from event log data (van der Aalst 2022). These process models are often formal conceptual models, such as Petri nets. Discovered models are typically evaluated along four quality dimensions: Fitness, precision, simplicity, and generalization (Buijs et al. 2014). The idea behind this generalization is similar to generalization in machine learning. It is defined as "the likelihood that the process model is able to describe yet unseen behavior of the observed system" (Buijs et al. 2014). Similar to machine learning, a process model generated with process discovery techniques should not overfit the event log, but generalize from the recorded data (van der Aalst 2016). However, unlike in machine learning, there is no loss function involved in discovering a (symbolic) conceptual model. The data is also not split into a train and test set (Tax et al. 2018), which is why generalization is measured differently. Instead of test set performance, the generalization of process models is evaluated through properties of the discovered process model such as the visiting frequency of model parts (Buijs et al. 2014) or anti-alignments (van Dongen et al. 2016). When measured in the machine learning way, discovered process models have been found to generalize less well than their predictive counterparts based on machine-learning (Tax et al. 2018).

2.2.2 Generalization in PPM

Since generalization is a fundamental part of machine learning, it is discussed in many textbooks. We refer to Goodfellow et al. (2016); Bishop (2006); Bishop and Bishop (2024) and Murphy (2022) for a general overview. More detailed discussions and specific definitions for generalization, paying attention to challenges and requirements of different machine learning tasks, are given in various task-specific papers, such as Zhou et al. (2023); Lu et al. (2024) or Jiralerspong et al. (2024). In the following, we focus on generalization in the domain of PPM.

Generalization in early PPM approaches In contrast to traditional process discovery, PPM does not rely on symbolic models. Rather, a machine learning model is trained to predict characteristics of process instances, thereby learning an implicit process representation (Evermann et al. 2017; Pfeiffer 2022). Ruta and Majeed (2011) were among the first to present a generic framework for PPM that includes the questions of "what will happen next" as

well as "how and when" a process instance will end. Since then, researchers have defined a set of PPM tasks (Di Francescomarino and Ghidini 2022), which mainly differ in terms of their prediction target: Some approaches predict the remaining time until the completion of the process instance (van der Aalst et al. 2011). Others predict the outcome of a process instance (Kratsch et al. 2021) or whether the process instance will deviate from the prescribed behavior (Grohs et al. 2025). The majority of work focuses on the prediction of the next activity (Neu et al. 2021), which is also the focus of this paper.

Shortly after the initial work of Ruta and Majeed (2011), Le et al. (2012) presented an approach specifically focused on next activity prediction. They extended the approach presented in (Ruta and Majeed 2011), building upon Markov models, but combining them with alignment techniques to handle unseen prefixes that have no occurrence in the transition matrix of the Markov model. This avoids predicting the default value for unseen prefixes, which is usually less accurate. Interestingly, the authors of this early PPM approach already acknowledged the need for generalization capabilities, as in real-life applications "the number of unique workflows (process prototypes) can be enormous, their occurrences can be limited, and a real process may deviate from the designed process when executed in real environment and under realistic constraints" (Le et al. 2012). They concluded that an efficient prediction would need to be able to cope with "the diverse characteristics of the data" (Le et al. 2012). Note that this reflects the motivation of generalization in machine learning (see Sect. 4).

Generalization in deep learning PPM approaches Inspired by the success of deep learning models for language modeling, Evermann et al. (2017) used a Long Short-Term Memory (LSTM) model (Hochreiter and Schmidhuber 1997) for next activity predictions. By design, neural networks always produce a prediction regardless of whether a prefix has been seen before, avoiding the problem of symbolic techniques, which predict a default value for unseen samples. Hence, they generalize better to unseen data and are more accurate. The LSTM model by Evermann et al. (2017) included regularization in the form of dropout. Further, it was validated using a 10-fold cross-validation procedure to measure whether the model generalizes to the test set and to prevent overfitting. The results showed that their LSTM model performed better or comparable to the state-of-the-art results of probabilistic automatons for the same task (Breuker et al. 2016). Following work improved the performance of neural-network-based prediction models, e.g., through the use of additional context attributes and specialized network architectures (Tax et al. 2017; Pfeiffer et al. 2021; Pasquadibisceglie et al. 2024) or training procedures (Taymouri et al. 2020). An overview of the field is given by Rama-Maneiro et al. (2021) and Neu et al. (2021).

In order to improve generalization in PPM, researchers have employed different techniques that have shown to improve generalization in other domains, e.g., multi-task learning, adversarial training, or transfer learning. As events in traces include additional attributes besides the activity, many approaches perform multi-task learning by predicting multiple attribute values of the next event at once (Evermann et al. 2017; Tax et al. 2017; Pfeiffer et al. 2021; Nolle et al. 2018). Since such models also use other attributes in the input, they are tasked with a much larger variability during training, which should improve their generalization capabilities. Prediction models have also been trained with augmented data or adversarial samples, relying on model-agnostic trace augmentations (Käppel et al. 2023), trying Generative Adversarial Networks (GANs) (Taymouri et al. 2020) or training with adversarial samples (Pasquadibisceglie et al. 2024; Stevens et al. 2023). Others try to improve generalization by adapting the loss function to balance the performance of the predictive model across multiple environments instead of performing best in only one (Venkateswaran et al. 2021). Transfer Learning from one event log to another has shown to be beneficial when training next activity prediction models (Jiralerspong et al. 2024). Further, knowledge gained from the event log of one organization can be transferred to the same process in another organization (Liessmann et al. 2024), indicating that such models may be able to generalize beyond the distribution in one event log. Besides that, others have developed train-test-split strategies that ensure no data leakage between the train and test set caused by traces with temporal overlap (Weytjens and Jochen De Weerdt 2021) or event log sampling approaches that preserve the performance of predictive approaches but allow for more efficient training on a subset of traces (Fani Sani et al. 2023).

2.3 Related Work

In addition to the concrete measures for a higher generalization, some existing works already provide definitions and exemplary assessments of generalization in PPM, particularly in next-activity prediction. This work is reviewed in the following.

Tax et al. (2018, 2020) first concretely addressed the question of generalization in PPM. They compared process discovery approaches to sequence modeling techniques in terms of how well they generalize for next activity prediction. Sequence modeling techniques generate a probability distribution of all activities, which discovery approaches cannot do. To avoid that non-fitting traces

result in zero probability for the next activity, they developed a procedure to generate a probability distribution over next activities by aligning the prefix to the discovered model and sampling from the reachable transitions. With this, they were able to compare discovery methods to other sequence modeling techniques using a loss function. They found that the machine learning approaches are much more accurate than the discovery approaches, at the cost of not being communicable and traceable by humans. As neural approaches reach a lower test error than their discovery counterparts, the authors conclude that they also generalize better.

In comparing the generalization of discovery techniques to neural approaches, Tax et al. (2020) did not consider unseen prefixes specifically. Using standard splitting strategies results in a majority of prefixes in the test set being duplicates of the training set (see Sect. 3). Further, a prefix that cannot be replayed by a discovered model does not necessarily need to be a prefix that has not been seen. If a discovered model does not reach perfect fitness, it is not able to describe all the behavior it has seen in the event log. Thus, it can also happen that seen samples cannot be handled by a discovered process model, a fact that influences the interpretation of the performance in handling unseen behavior. In the following, we discuss the few papers that have investigated generalization with respect to unseen prefixes.

Gerlach et al. (2022) split the log by variants into train and test sets so that not all variants are part of the training set and others are only present in the test set. They trained this model to predict the activity and other context attributes in the next event. After training they used this model to build a likelihood graph to capture and visualize its behavior, by iteratively predicting next steps. They found that their approach can generate unseen variants, manifested in the likelihood graph, which are also valid with respect to the underlying process. They conclude that the next activity prediction model can generalize beyond the event log it was trained on.

Peeperkorn et al. (2022) followed a similar method. They aimed to research whether models trained to predict the next activity in a trace learn the structure of a process model, even if they have not seen all variants during training. For this, they trained the prediction models in the leave-one/some-variants-out fashion and let them generate traces again. In their experimental setting, they used process models of varying size and complexity, obtained a set of traces through simulation, and split the traces such that some process variant(s) were only present in the test split. To measure generalization, they expected that the trained prediction models could generalize the missing variants(s) from the event log, avoid creating variants that are not part of the original log, but generate all variants present in the training log. They found that regularization is required to ensure that the model does not overfit the training log, i.e., that it does not only generate variants it has seen before but also new ones, even for relatively simple models.

In a follow-up work, Peeperkorn et al. (2024) proposed validation set sampling approaches to enhance the generalization capabilities of prediction models in the same setting as their previous work. They found that the sampling techniques can have a strong positive impact on the generalization capabilities but that process structures such as concurrency and long-term dependencies pose challenges for predictive approaches as they do not generate all expected variants. As for other approaches, the completeness of the log with respect to the behavior of the model is a limitation.

While Gerlach et al. (2022) concluded that their model does generalize, Peeperkorn et al. (2022, 2024) conclude that process prediction models do not generalize as they do not generate all (unseen) variants of the original process model. We suspect these differences to come from different settings and techniques being used as well as different expectations being placed on the abilities of the models. For instance, Gerlach et al. (2022) use context attributes as input for their predictive model while Peeperkorn et al. (2022, 2024) use the activity only. Furthermore, the procedure to generate traces from a model that predicts only one next step at a time, (i.e., how to sample from the predicted probability distribution) and the metrics to evaluate the generalization capabilities differ. We will discuss the differences in their definition of generalization to our definition of generalization in Sect. 5.

3 Validity Issues in Next Activity Prediction

In this section, we examine two phenomena that pose threats to the validity of next activity prediction experiments, particularly for their assumed generalization capabilities. The first phenomenon, described in Sect. 3.1, is *example leakage*, i.e., the presence of the same prefixes in both the train and the test split. The second phenomenon, described in Sect. 3.2, is the *accuracy limit*, i.e., the observation that the maximally reachable accuracy of a next activity prediction can be less than 100%. Along a typical evaluation setup for next activity prediction, we propose two new metrics to quantify these phenomena. By relating the performance of state-of-the-art next activity prediction models to these metrics, we can identify potential threats to validity. In Sect. 3.3, we discuss our findings with regard to their impact on the generalization capabilities of next activity prediction models.

In the following, we present empirical evidence generated in a setting that is representative of the typical evaluation setup used in the field. We employ five commonly used event logs (Helpdesk,[1] BPIC12[2] (complete log), BPIC13 Incidents,[3] BPIC17,[4] MobIS[5]) and two common splitting strategies:

1. *5 random splits*: We split the traces in the event log randomly such that 80% of them are part of the training set and 20% are part of the test set
2. *Temporal split*: The split is time-based so that the 20% of traces with the most recent start timestamps end up in the test set.

For all traces in each of the 6 splits, we generate all prefix-label pairs (x, y) with prefix lengths $p \in [1, n-1]$, which constitute our training and test samples. We do not apply log preprocessing or make any other changes to the data. The code and data needed to reproduce the validity issue experiments are available online.[6]

3.1 Example Leakage

In machine learning, leakage refers to exposing a model to information during training that it should not legitimately have access to (Kaufman et al. 2012). This can lead to an unrealistic assessment of the model's performance with respect to the trained prediction task. One particular type of leakage is *example leakage*, which occurs when the same example (more specifically, the same feature vector) is present in both the training and the test set. In this case, the prediction is trivial, as the model is not required to learn a general relationship between features and labels for the prediction to be correct. Due to the repetitive nature of the process executions, which naturally leads to a high portion of duplicate traces, example leakage can be a considerable problem when making predictions on event logs. By building prefixes of traces, the portion of duplicates can increase even further.

In next activity prediction, we denote example leakage as the portion of prefixes in the test split where the test prefix is a duplicate of a prefix in the training split. To quantify this example leakage, we first need to establish when two prefixes are duplicates. When considering the context attributes of events, such as the timestamp, almost every prefix will be unique. However, many prediction approaches do not use the timestamp itself, encode it, e.g., as the weekday or the duration between events. Further, many approaches are restricted to certain types of attributes (Pfeiffer et al. 2021). As the control flow is the most decisive feature for predicting the next activity (the target label is largely subject to the sequence of previous events), and many prediction models use the control flow only, we (for now) determine equality of prefixes based on control flow: Two prefixes are considered duplicates if they exhibit the same sequence of activities. In the following, we use this equality criterion to approximate example leakage.

We can now quantify example leakage on the common event logs and splitting strategies. Let X_{train} be the multiset of prefixes with control flow only of the training set L_{train} and X_{test} be the respective multiset of control flow only test prefixes. The portion of leaked examples is calculated as:

$$\text{Example Leakage} = \frac{|X_{train} \cap X_{test}|}{|X_{test}|}$$

where $X_{train} \cap X_{test}$ contains those prefixes that are found in both the training and the test split.

Figure 1 shows the amount of example leakage in the event logs commonly used for the evaluation of next activity prediction. We observe that, across all datasets and splits, example leakage is close to or above 80%, with almost 100% in the Helpdesk and MobIS event logs. This means that most of the predictions made on the test split should be trivial ones, such that from this evaluation setting, we cannot draw valid conclusions about how well a prediction model would perform on unseen data.

Observation 1

The portion of duplicates between training and test splits is dangerously high.

3.2 Baseline and Accuracy Limit

Another issue relates to the way of reporting and comparing predictive performance in next activity prediction. Commonly, the top-1 accuracy, or simply accuracy, is used for this purpose, evident through its widespread use in papers on next activity prediction (e.g., Le et al. 2012; Ruta and Majeed 2011; Breuker et al. 2016; Evermann et al. 2017; Neu et al. 2021; Pfeiffer et al. 2021; Pasquadibisceglie et al. 2022; Weinzierl et al. 2024). The accuracy is computed as the portion of samples (x, y) that were correctly classified, i.e., the number of samples in which the model assigns the highest probability to the class with the label equal to the ground truth label $\hat{y} = y$. A predictor is considered accurate if it achieves high accuracy values when predicting the next activity for the given prefixes. Without further knowledge of the data, we would also expect that a predictor that has accurately learned the target label distribution reaches high accuracy values. If a

[1] 10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb.
[2] 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.
[3] 10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee.
[4] 10.4121/uuid:5f3067df-f10b-45da-b98b-86ae4c7a310b.
[5] Baier et al. (2020).
[6] https://github.com/ppfeiff/BISEGenPPM.

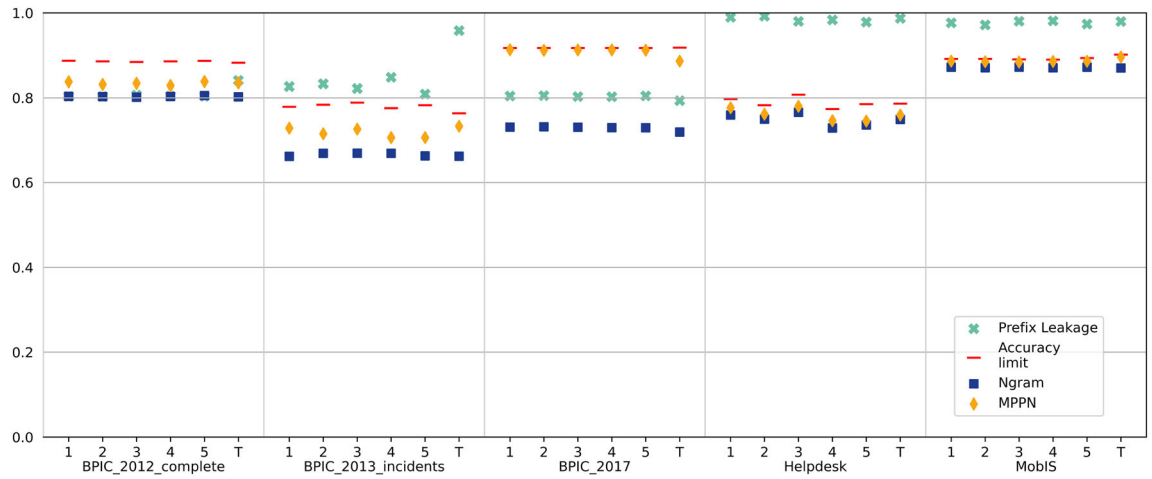

Fig. 1 Example leakage, accuracy limit, and the accuracy of the naive baseline and the MPPN on the test split for the 5 random splits (1-5) and the temporal split (T) on the real-life logs

predictor instead achieves a mediocre accuracy value, e.g., below 80%, we would conclude that it did not perform well and there is room for improvement.

However, certain characteristics of the data and the task, e.g., incomplete information in the input features or the fact that the system is intrinsically stochastic, can make it impossible to achieve a high accuracy, even with an optimal prediction model (Goodfellow et al. 2016, p.442). Goodfellow et al. (2016) refer to this as Bayes error, defined as the minimum error rate one can hope to achieve in a perfect setting by making predictions according to the true probability distribution. For prefixes of running process instances, previous work has found that a large share of samples has multiple valid continuation options and often no attribute that determines which activity follows (Pfeiffer et al. 2023). Therefore, next activity prediction is likely also subject to the Bayes error.

We illustrate this phenomenon with an experiment, which shows that the maximally reachable accuracy in next activity prediction is often much lower than 100%, even if using an oracle that has access to the target probability distribution. Usually, if the target distribution in the test split is known to a machine learning model, it should be capable of reaching almost perfect accuracy by always predicting the class with the highest probability according to the test split. We calculate the *accuracy limit* for next activity prediction as the maximum accuracy one can hope to achieve when employing such an oracle model. This theoretical value is compared to the actual accuracy that a naive, frequency-based baseline and a context-aware neural network reach, when being trained on the training split and without access to the test split.

Again, we consider the control flow only, meaning that the *accuracy limit* is an approximation to its true value based on X_{train} and X_{test}. It is realized through a perfect prediction model with access to the test split X_{test}. For each prefix x of the samples (x, y) in the test split, we predict the activity label $\hat{y}$ that we most commonly observe for this prefix in the test split X_{test}. If this label equals the ground truth label y found in the sample, we say this is a correct prediction. Averaged over all samples in X_{test}, we report the model's accuracy as the *accuracy limit*. Clearly, such a prediction model is considered illegitimate in machine learning research (as it can optimize on the target distribution) and is used exclusively for demonstrating the issues with accuracy in next activity prediction.

For the naive baseline, we train a tri-gram prediction model on the prefixes in the train split and report its accuracy on the test split. Again, we use the control flow only, i.e., the tri-gram model estimates the probability for the next activity using the previous two activities. We use Kneser-Ney smoothing (Kneser and Ney 1995) for unseen prefixes, i.e., the probability for unseen prefixes is approximated using a bi-gram model that considers the last activity. We also report the accuracy of a state-of-the-art prediction model namely the Multi-Perspective Process Network (MPPN) (Pfeiffer et al. 2021). This neural-network-based model is more flexible than the baseline model as it can process any combination of event attributes in the event log and prefixes of length up to 64 events. It has shown to perform very well on different PPM tasks (Pfeiffer et al. 2021; Grohs et al. 2025) and thus represents potential performance gains that can be reached by complex prediction models that are context-aware compared to the naive baseline. We train it on the training split and report the accuracy on the test split.

When comparing the two models to the accuracy limit, we would expect that the *accuracy limit* is at least as high as the share of leaked examples. Further, we expect the prediction models to reach accuracies at least as high as the

prefix leakage, with the MPPN performing better than the naive baseline, given that a lot of research has focused on improving the performance of PPM models over the last years. However, as the results in Fig. 1 show, those expectations are not met. In contrast, the results reveal some unexpected insights: First, and most importantly, the *accuracy limit*, which in theory should be close to 100%, is much lower in practice. Further, the naive baseline reaches almost the same accuracy as the state-of-the-art MPPN model on many logs with all models reaching lower accuracies than the example leakage except for BPIC 17. Even though the example leakage for the Helpdesk log is almost 100%, the maximum achievable accuracy is below 80%. In general, the *accuracy limit* seems not to be related to the example leakage as the values are very different and sometimes even lower than the share of leaked examples, questioning the meaningfulness of this metric for next activity prediction.

Observation 2 The maximum accuracy one can hope to achieve in next activity prediction is much lower than assumed and seems not to be related to the portion of leaked examples.

3.3 Implications

The results show that the widely used evaluation procedure for next activity prediction models is problematic due to the high amount of example leakage in combination with a lower than expected accuracy limit and the questionable meaningfulness of accuracy. Both observations are known evaluation failures in machine learning research (Liao et al. 2021). A high amount of example leakage, which Liao et al. (2021) call "contaminated data", leads to an overestimation of the model's performance and hence poses a threat to its reliability. In contrast, using metrics like accuracy, which do not meaningfully report the performance of a model on a learning task, may underestimate the model's actual performance (Liao et al. 2021). The observation that there is an accuracy limit has so far not been considered in next activity prediction evaluation but influences the interpretation of the models' actual performance.

Given that this evaluation setting has been widely employed in existing publications on next activity prediction, our findings question the reliability of the advancements made in the field. As a research community, we now have a large number of proposed next activity prediction techniques that employ several different neural network architectures, inductive biases, and strategies to incorporate different types of features. However, these techniques have only ever been evaluated in a problematic setting.

In conclusion, we do not know how such models perform on unseen data since most samples used to evaluate their performance are duplicated from the training split and the metric to quantify the performance does not report it meaningfully. Therefore, we cannot say to what extent these approaches would be able to generalize well enough to make good predictions on unseen data – and consequently, if they would be able to provide value in a real-world application.

4 Generalization in Machine Learning

In this section, we reflect on the role of generalization in general machine learning: its aim and challenges and its different types. Based on this, we develop a framework for generalization in PPM in Sect. 5.

4.1 Aim and Challenges of Generalization

Generalization describes the ability of a machine learning model to perform well on unseen samples (see e.g., Goodfellow et al. 2016; Bishop 2006). A machine learning algorithm is said to perform well if it is able to generalize beyond the data it was trained on (Bishop and Bishop 2024). Achieving generalization is one of the most difficult parts of machine learning, due to the following characteristics of a machine learning task:

MC1: The training data contains only a small fraction of the feature variability found in practical application (Bishop 2006, p.2).

MC2: The empirical distribution in the training data is different from the distribution found in application (Murphy 2022, p.121).

MC3: The machine learning model usually has enough capacity to perfectly fit the training data which would, without any countermeasures, not allow it to have enough capacity left for unseen data (Goodfellow et al. 2016, p.112).

For most tasks, we could build a machine learning model that almost perfectly fits the training data (MC3). However, as this data is incomplete with respect to the variability of the application data (MC1) and the empirical distribution of the data might be different in application (MC2), such a model would very likely perform poorly when tasked with samples that differ from the data it was trained on. Thus, the task and data have to be studied in order to apply techniques that ensure that the model also works for unseen data. One has to "understand what kinds of distribution are relevant to the real world that an AI agent experiences" (Goodfellow et al. 2016, p. 118) and what kinds of algorithms perform well on such distributions.

There are two factors that determine how well a machine-learning model works: (1) its ability to perform well on the training data and (2) its ability to minimize the gap between the training and the test error, commonly referred to as generalization error (Goodfellow et al. 2016). If the model performs badly on the training data, corresponding to (1), it is underfitting. If the model fits the training data but cannot reduce the gap between the training and the test error, corresponding to (2), it is overfitting. When training a machine learning model, there is a trade-off to make between its bias and its variance (Goodfellow et al. 2016, p.129) which is associated with overfitting and underfitting. The bias describes the ability of a model to capture the relevant relations between input features and the output. We want the model to have a low bias to avoid underfitting. At the same time, its variance, i.e., how much the predictions vary when the input changes slightly, should be low. As the model capacity becomes larger, the bias decreases while the variance increases, leading to a higher generalization error. The point at which the generalization error is the lowest is also the point where the best trade-off between bias and variance of a model is made.

Depending on the machine learning task, different requirements and challenges arise for generalization. For object detection, we expect the model to perform well when the appearance of objects changes. Further, we might also want to classify objects in different types of images, e.g., in sketches instead of photos (Zhou et al. 2023), or to classify new objects given only a few examples. In contrast, for time series prediction, we want the model to generalize to future data (Lu et al. 2024), where the change of distributions through time (non-stationarity) presents challenges. In the following, we discuss two points that are relevant for generalization on all types of machine learning tasks.

Train-Test-Splitting Splitting the available data into train and test sets allows researchers to estimate how the model will perform on unseen data. To ensure that the test error is reliable, the samples in the test set should not have been seen during training (MC1 and MC2). In leave-one-out cross-validation or k-fold cross-validation, one "variant" or fold at a time is used for testing and the remaining splits are used for training. Repeating this procedure k times, each time using a different part of the data for testing, a precise picture of the generalization capability can be obtained. The model with the best generalization capabilities is chosen as the best model. When splitting the data, one has to ensure that no information about the test set is leaked into the training set (Kaufman et al. 2012; Liao et al. 2021), e.g., the training data should not contain information about the target labels.

Regularization The ability of a machine learning model to perfectly fit the training data (MC3) can lead to overfitting, which means that the model is unable to make correct predictions for unseen samples. One technique to avoid this overfitting is regularization, i.e., modifications that reduce the generalization but not the training error (Murphy 2022; Goodfellow et al. 2016). For example, we can add artificial noise during training to increase feature variability (targeting MC1), mimic other changes (targeting MC2), or stop the training procedure at an early point (targeting MC3). Other regularization types aim to modify the weights of the machine learning models through the loss function or train on multiple tasks at once (Goodfellow et al. 2016).

4.2 Types of Generalization

Many machine learning algorithms assume that the distribution of samples in the test set is the same as in the training set and samples are independent. Together, these are commonly referred to as the independent and identically distributed assumption (Goodfellow et al. 2016). Generalization under this assumption is also called *In-Domain* or *IID* generalization and can be tackled with regularization techniques. In practice, however, this assumption is often violated. This has motivated the development of deep learning models, which can deal better with high-dimensional inputs and higher variability (Goodfellow et al. 2016) and to generalization techniques that enhance model capabilities in such situations.

In contrast to *IID* generalization, so-called *OOD* generalization aims to make a model perform well for samples of unseen domains (Zhou et al. 2023). Such samples are out-of-distribution, i.e., they differ from the samples used for training, which is why this form is also called *Out-of-Domain* generalization. This form of generalization requires more advanced generalization techniques as *In-Domain* generalization. For instance, we would like an image classifier that was trained to detect objects in photos to also detect the same set of objects in sketches. Different methodologies have been developed to enable domain generalization such as data augmentation, meta-learning, or transfer-learning. For an overview, we refer to Zhou et al. (2023).

5 Generalization in Predictive Process Monitoring

Based on the previous Sect. 4, we dedicate this section to discussing generalization for PPM tasks. Although we focus on next activity prediction, many points also apply to other PPM tasks that use prefixes to represent running process instances as input.

5.1 Aim and Challenges of Generalization in Predictive Process Monitoring

The aim of generalization in PPM is to make the models flexible when confronted with running process instances that have not been seen so far, which are likely to occur in application (Le et al. 2012; Peeperkorn et al. 2022). As a machine learning task, characteristics MC1 - MC3 also apply to PPM:

PC1: The traces trained on contain only a small fraction of the process behavior
PC2: The empirical distribution of process variants trained on can be different from the distribution found in application
PC3: PPM models have enough capacity to perfectly fit the data trained on, which would, without any countermeasures, not allow it to make correct predictions for unseen prefixes

To cope with these characteristics, the model has to generalize to unseen process behavior and accurately predict how process instances will continue. However, existing work, reviewed in Sect. 2.3, defines generalization differently as normally done in machine learning: They consider generalization as the ability to generate unseen, valid process variants from seen prefixes, such that the variants represent the whole behavior of the original process model. A prediction model with these abilities would be very powerful and arguably highly valuable for a range of process mining tasks. However, generating valid and unseen traces is not the same as making valid predictions for unseen process instances. The latter requires to make a valid prediction for an unseen input, whereas the former requires generating something unseen from a seen input, which is much more challenging to achieve. It aims more towards the general question of whether prediction models "learn" and generate process model structure rather than making correct predictions for unseen samples, similar to generative language models generalization (Jiralerspong et al. 2024).

In contrast, the view on generalization in PPM, specifically for next activity prediction, that we introduce follows from the standard notion and the challenges MC1 - MC3 of generalization in machine learning. Specifically, we focus on the capability of PPM models to deal with unseen prefixes, as existing evaluation procedures evaluate mostly on seen prefixes. We argue that dealing with unseen prefixes is an important ability of PPM models and a prerequisite for researching more complex types of generalization. Therefore, we define generalization as follows.

Next activity prediction generalization: Predict the correct continuation options for unseen prefixes

A PPM model generalizes well if it (1) performs well on the training split and (2) minimizes the gap between the training and test error, called generalization error. Both factors follow from the definition of generalization in machine learning as introduced in Sect. 4. For the task of next activity prediction, this translates to model that accurately predicts how processes continue and does this equally well on the training split as well as on the unseen samples in the test split. In this case, we say that the model generalizes. If it does not perform well on the training split or on both splits, it does not generalize.

To make accurate predictions for unseen prefixes, PPM models need to learn process behavior beyond the behavior contained in the event log. It is not sufficient to accurately learn the prefixes and repeat the behavior found in the training data. Rather, generalization for PPM requires extrapolating from seen process behavior. This means that the prediction model must learn generally valid patterns – for example, how activities typically can follow one another – rather than specific rules from the training data. In the following, we discuss these characteristics as well as additional challenges for generalization in PPM, which have to be considered when evaluating generalization. They are summarized in Fig. 2.

5.1.1 *Incompleteness and Variability*

The event log, as a collection of traces, is usually incomplete with regard to the process behavior (Buijs et al. 2014), referring to PC1. The training split L_{train}, as a subset of the event log, hence represents an even smaller fraction of the total process behavior. To still train well on this data, the prediction model has to deduce higher-level process behavior. The incompleteness of the event log data is in general likely lower than the incompleteness of data in other machine learning tasks. For example, it is unlikely that two images used as input for object detection are identical in RGB values. There is a very large variability in images leading to a high incompleteness when training an object detection model on real images. In contrast, there are often multiple process trace prefixes with the same control flow and attributes. Even a new, unseen process variant can contain many prefixes that were seen before or differ in one activity or attribute only, increasing the share of samples that are identical in L_{train} and L_{test}. Therefore, the number of duplicates is naturally very high, which explains the high example leakage discussed in Sect. 3. These characteristics have to be considered for splitting the data in anticipation of testing on unseen samples.

Considering context attributes in addition to the control flow increases the variability found in event log data (Evermann et al. 2017). The chance that two samples are identical is much lower if the prediction model can

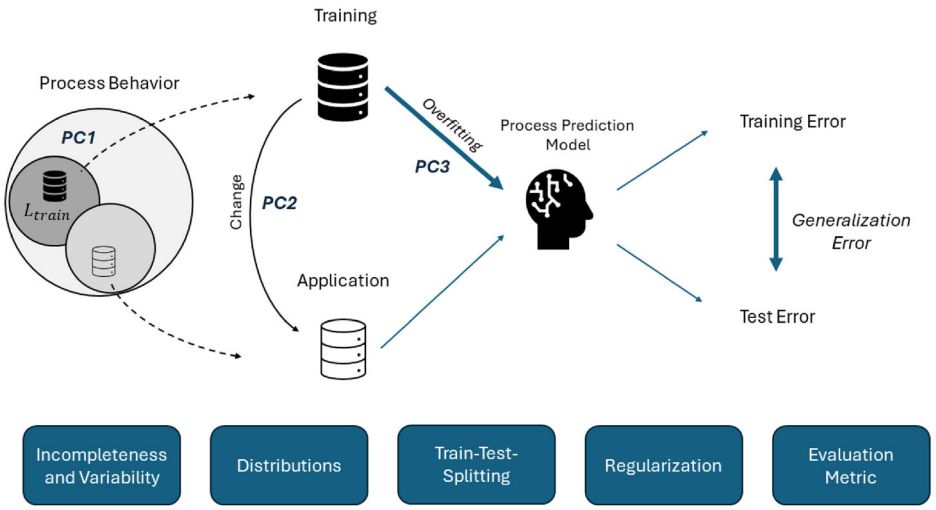

Fig. 2 Overview of generalization in process prediction

consider categorical attributes, e.g., resources. If it also considers numerical attributes, e.g., prices, or temporal ones, e.g., timestamps, each sample is almost certainly unique. Thus, the more context attributes are considered, the higher the variability of the data and its incompleteness. The prediction model must be capable of dealing with this higher variability and potentially incomplete attribute values (combinations) if using context attributes. While context attributes offer additional information for predictions, considering more context attributes significantly increases the number of unseen combinations that can occur in prefixes, making generalization also more challenging.

5.1.2 Distributions

The distribution of control flow variants in the event log is typically skewed towards a Pareto distribution: Around 80% of traces follow the same few process variants, whereas the other 20% follow a large number of different variants (van der Aalst 2020). The Pareto distribution is also found in the prefixes (with significantly more different variants), but usually not in the target labels, e.g., the next activities in next activity prediction. We can assume this distribution to also be present in application. However, the most frequent variants are not necessarily the same ones as in the training data since processes are typically not assumed to be stationary (Back et al. 2019). The distribution of variants can change and new process instances with new process variants, containing new behavior, are likely to emerge over time (Le et al. 2012; Peeperkorn et al. 2024). This changes the distribution of prefixes (PC2) in application. Tailoring the prediction model for certain variants is thus problematic as we cannot foresee, in general, which variants will be most frequent ones in the later application.

5.1.3 Train-Test-Splitting

To measure the generalization error, we need to split the data in such a way that the test split L_{test} reflects the conditions to be expected in the application. This includes that the training samples do not leak information about the test conditions. The characteristics discussed in the previous paragraphs make it challenging to split the event log realistically. As shown in Sect. 3, splitting the data randomly or by time causes a high example leakage, such that the test error might not reflect the error to be expected in application. Previous work has suggested splitting the traces by variants such that certain variants will only be present in the test split (Peeperkorn et al. 2024). Although this ensures that at least a share of samples in the test split will be unseen, it does not prevent the occurrence of duplicates for next activity prediction, as two different traces can share many identical prefixes.

Following characteristic PC2, we can be certain that new process variants will occur. However, we do not know which variants will occur and with what frequency. Thus, we argue that splitting by time is the most realistic setting as it naturally maintains such distribution changes in time. For this, all traces before a certain point in time t are part of the training split L_{train} and all other traces are put in the test split L_{test}. Afterwards, to keep the example leakage at a reasonably low level, duplicates between the training and test splits, which are created when building prefixes (or windows) from the samples, have to be removed. By doing so, the test split L_{test} will consist of samples where the prefix x has not been seen and qualifies for measuring a meaningful generalization error.

5.1.4 Regularization

When training a PPM model, regularization is required to prevent the model from overfitting the training split L_{train} PC3. The capacity of a neural-network-based prediction model can quickly surpass the variability found in the training data and the number of distinct samples available for training (Goodfellow et al. 2016). If training a prediction on the control flow only, the low variability in the event log can be challenging for the generalization of the model as its capacity is higher than the complexity in the data. Previous work has already pointed out that strong regularization for next activity prediction is required as the models tend to overfit otherwise (Peeperkorn et al. 2024). When designing a PPM model, one should therefore balance the capacity of the model with the number of distinct training examples and the complexity of the process, e.g., in terms of variants. Increasing the variability by using additional attributes can also help to better align the capacity of the model with the variability of the data. Additionally, regularization techniques such as dropout, L1 and L2 regularization, and early stopping are easy to implement and can prevent overfitting effectively.

5.1.5 Evaluation Metrics

Meaningful metrics are important for obtaining validity in the conducted evaluations (Liao et al. 2021). As seen through the analysis in Sect. 3, there are several issues with the use of the accuracy and its interpretation in the next activity prediction task, which affects the model's performance perception. When using a next activity prediction model in application, we are interested in the share of samples where the prediction of the model matches the ground truth. While accuracy reports this share, it focuses exclusively on the prediction with the highest probability and assesses it in an all-or-nothing manner. Thereby, it does not fit the variability of how processes can continue. For a reliable generalization assessment, we need a metric that accurately quantifies how close the predictions of the PPM model on unseen prefixes are to the behavior of the process in the data, valuing that there is not always a single valid next activity. Instead of relying on accuracy, we suggest a probabilistic interpretation that can deal with the variability and considers all continuation options the model has learned $q(y)$ rather than the single most likely prediction.

We motivate the probabilistic assessment on a simple example. Consider a situation where, e.g., due to parallel execution (as shown in Fig. 4), a prefix x has two valid next activities D and E. As processes are not stationary, there may also be a change in the distribution of the very next activities D and E between training and application as exemplified in Table 1. Note that still both activities are executed and only the distribution in their order has changed. We can assume that a machine-learning-based prediction model learns that both activities can happen, which it expresses by giving both activities a high probability, e.g., $q(D) = 47\%$ and $q(E) = 51\%$, based on the frequency found in the training data. If evaluating with accuracy, we only use the most probable prediction of the model which always is E. To compute the accuracy, it is compared to the ground truth. If the samples (x, y) in the test split would follow the same probability as the samples in the training split, only around 52% of all predictions will be evaluated correctly. The remaining samples, containing D as ground truth will be evaluated as wrong predictions. From the accuracy value, one would conclude that the model performs badly. This ignores that the probability distribution learned by the model accurately reflects how the process behaves. Further, a higher accuracy is impossible to achieve.

Table 1 A example with parallel activities D and E to clarify the conceptual issues with accuracy

	Train Setting		Application	
Target activity y	D	E	D	E
Number of examples x	450	500	450	250
Learned prob. distr. $q(y)$	0.47	0.51	0.47	0.51
Cross-entropy (CE)	0.309		0.315	
Accuracy	52.63%		35.71%	

In case the distribution in the application setting changes such that E is executed half as often as during training, the model would reach a significantly lower accuracy with a drop from around 52% to 35%, although its learned probability distribution has not changed. In both scenarios, the performance of the model would be rated much worse than it actually is. When assessing the generalization capabilities of the model, we would conclude that it cannot generalize. However, the model makes accurate predictions in both scenarios which can be shown if measuring in a probabilistic way on $q(y)$.

For this, we opted for using cross-entropy as the error measure, as commonly done in other classification and next element prediction problems (Jurafsky and Martin 2025). In particular, this idea is based on cross-entropy estimation for the next word prediction task in language modeling (Jurafsky and Martin 2025, p. 210), which has motivated deep-learning-based next activity prediction. The cross-entropy formulation for a single sample in context of next word prediction is

$$CE_{basic} = -\sum_{w \in V} p(w) * log(q(w)), \quad (1)$$

where $w \in V$ are the possible classes, $p(w)$ is the observed ground truth probability distribution, and $q(w)$ is the predicted probability distribution returned by the model. Note that, for next activity prediction, the real probability distribution of next activities $p(y)$ in the process that generated the event log is unknown to us. We can therefore, as in next word prediction, only estimate it based on $q(y)$.

Since next activity prediction is a multi-class and single-label classification task and $p(y)$ thus always a one-hot vector, this calculation for a predicted sample (x, y) can be simplified to taking the logarithm of the probability that the model predicts for the true class (Jurafsky and Martin 2025, p.210). For an event log split L_{split} (train or test split) with N samples, where i indexes each sample, the cross-entropy of a next activity prediction model that estimates the probability $q(y_i)$ for the true next activity y_i given the prefix x is therefore calculated as

$$CE(L_{split}) = -\sum_{i=1}^{N} log(q(y_i)). \quad (2)$$

With cross-entropy, the prediction error is thus calculated as the logarithm of the probability that the model assigns to the ground truth activity, whereas with accuracy, we would only determine whether this probability is higher than those assigned to the other activities. When aggregated over all predictions in the event log split, cross-entropy gives a more realistic interpretation of how well the model has learned to predict process behavior, because it more accurately reflects whether the predicted probability of a next activity occurrence is equal to its true probability of occurring. This measure is also in line with the general idea of machine learning to replicate the distribution by giving probabilistic rather than "entirely certain" rules (Goodfellow et al. 2016).

For our above example, the change in cross-entropy between train and application is only minimal and much smaller than the change in accuracy, as shown in Table 1. This is because cross-entropy acknowledges that the model gives a certain probability for D for the samples containing D as it does for samples with E. Note that due to the variability in processes, the cross-entropy might never become 0: Even if having learned the probabilities accurately, it is bound by the variability of the process which is the minimal error that cannot be avoided (Goodfellow et al. 2016). For instance, the cross-entropy for the example is around 0.3 even if the prediction model has learned the probabilities accurately. This is justified as the model gives activity D a reasonably high probability in the samples containing E which accuracy does not value. For next activity prediction, the interpretation of the model using cross-entropy gives an additional perspective on its performance which is more accurate for assessing its generalization performance. Nevertheless, the usage of cross-entropy might not be universally beneficial for all PPM tasks and should be considered per task.

5.2 Types of Generalization

For process prediction, we can also differentiate between *In-Domain* and *Out-of-Domain* generalization. The *In-Domain* setting occurs in situations where the distribution of samples in the test split is identical to the training split. Considering only the control flow, an unseen prefix that is *In-Domain* can be caused by loops or parallel activities. Such situations do not cause significant changes to the distributions and data. If considering context attributes, we can differentiate between an unseen combination of seen attribute values or completely new attribute values. An unseen combination of context attributes, e.g., a new combination of resource and cost, would still be *In-domain*. Further, we argue that even unseen attribute values are not necessarily Out-of-Domain, as long as there is no significant change to the process.

If changes are more significant, they require generalization to out-of-distribution samples. As process data is known to be non-stationary (Back et al. 2019), i.e., probabilities change over time, prediction models should be capable of generalizing in such settings. There can also be other situations that cause more significant changes to the process and the distribution of the data, e.g., if an unseen variant becomes the most frequent one. Other examples could be a new decision point, the introduction of new activities or subprocesses, or an event log from the same process in a different company. Other types of generalization include generating unseen and valid process variants from seen prefixes (Peeperkorn et al. 2022, 2024).

6 Generalization Scenario Examples

In this section, we present nine example scenarios that exemplify the aim of generalization for the task of next activity prediction. The scenarios require different abilities from PPM models in order to achieve generalization and are inspired by typical situations faced when dealing with process data in event logs. Each scenario presents a situation where the prediction model is faced with an unseen sample (x, y), i.e., where the prefix x has not been seen before. For each scenario, we discuss what we expect a generalizing prediction model to predict, i.e., what a valid prediction $\hat{y}$ would be. In the following section, we will simulate event logs that implement the scenarios and

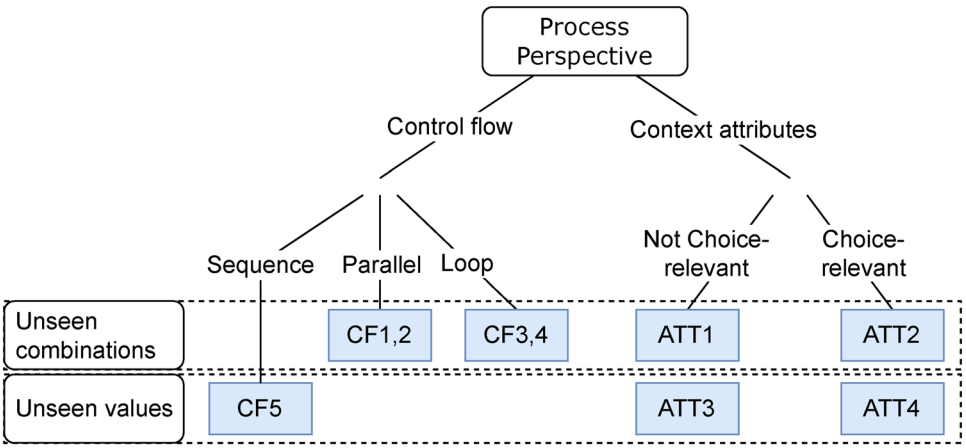

Fig. 3 Structure of the generalization scenarios

evaluate how well existing next activity prediction models generalize.

An overview of the scenarios is shown in Fig. 3. We structure the scenarios based on the process perspective that is involved (only control flow or including context attributes) and whether the prefix contains unseen combinations of previously seen values or whether it contains values never seen before. A prefix can be unseen if certain values appear in an unseen combination. For instance, if a certain person performs an activity for the first time (in the log) while both the activity and person have been seen in the samples before. A prefix can also be unseen if a value appears that it has never seen before, e.g., a completely new activity or completely new resources. All scenarios relate to the characteristics PC1 and PC2. Five scenarios (CF1-5) cover generalization to unseen control flow variants in the prefix, further divided into the possible process execution patterns sequence, concurrency, and loop. The other four refer to unseen (combinations of) context attributes that can either determine the continuation of the process (e.g., as the condition of a choice; ATT2 and ATT4) or be "noise", i.e., not relevant for the prediction of the next activity (ATT1 and ATT3). In the following, we introduce the scenarios conceptually and illustrate them using minimal example event logs.

Table 2 CF1 - prediction after parallel activities

Event Log $L1$
⟨A, B, C1, C2, C3, D, E⟩
⟨A, B, C2, C1, C3, D, E⟩
⟨A, B, C2, C3, C1, D, E⟩
⟨A, B, C3, C1, C2, D, E⟩
⟨A, B, C3, C2, C1, D, E⟩

Table 3 CF2 - prediction within parallel activities

Event Log $L2$
$\langle A, B, C, D, E, F, \underline{G, H}\rangle$
$\langle A, B, C, F, D, \underline{G, E}, H\rangle$
$\langle A, B, C, D, F, E, \underline{G, H}\rangle$
$\langle A, B, F, C, D, \underline{G, H}, E\rangle$

Table 4 CF3 and CF4 - prediction after and within loops

Event Log $L3$
⟨A, B, C, D⟩
⟨A, B, B, C, D⟩

6.1 Scenarios with Unseen Control Flow

The event logs $L1$, $L2$ and $L3$ cover scenarios that are created by parallel activities or loops. Log $L1$ in Table 2 shows an example of three activities $C1$, $C2$ and $C3$ that can occur in any order. $L2$ in Table 3 shows a similar, yet more complex scenario with C, D, E, F, G, H in any order. This is typically caused by parallel activities and is a common phenomenon in real-world event logs. Another common scenario is the appearance of activities that can be executed multiple times after another as shown in $L3$. For event logs with such patterns, four interesting scenarios (CF1-4) can occur:

- **CF1**: $L1$ and prefix ⟨A, B, C1, C3, C2, D⟩. Expected prediction: E. Although the model has not seen this prefix due to a new order of $C1$, $C2$ and $C3$, it should have learned that the case always continues with E after D, regardless of the order of the previous activities. The same holds for predicting D after having seen $C1$, $C2$ and $C3$.

- **CF2**: *L2* and prefix: ⟨A, B, C, D, F, G⟩. In this scenario, we are making a prediction during the execution of concurrent activities, where more than a single activity is possible. As seen in *L2*, both *E* and *H* have been observed immediately after *G* and are thus valid continuations and valid predictions.
- **CF3**: *L3* and prefix: ⟨A, B, B, B, C⟩. Expected prediction: *D*. In this scenario, the model should have learned that the case always continues with *D* after *C*, no matter how often *B* has happened.
- **CF4**: *L3* and prefix: ⟨A, B, B, B⟩. In this scenario, we are essentially predicting whether the loop continues (prediction *B*) or stops (prediction *C*). Similar to scenario CF2, both would be valid continuations.

The training data can be incomplete, e.g., with respect to the activities that can occur in the process. For instance, if obtaining an event log from the same process but a different system that records one additional activity. Another reason could be that a process change may lead to a new activity being introduced between training and prediction time, which would then be unknown to the prediction model:

- **CF5**: *L3* and prefix ⟨A, F, C⟩. As *F* is an activity the prediction model has never seen before, there is no evidence from the event log how to continue. One option would be to predict a label from the event log that could potentially follow, e.g., *D*, as this has occurred after *C* in all traces in the training data. Otherwise, the model could also indicate that it does not know, e.g., by making a special prediction *UNKNOWN*.

6.2 Scenarios with Unseen Context Attributes

In some situations, the context attributes like involved resources, timestamp or cost carry important information to determine the continuation of the process instance (Brunk et al. 2020). Considering this contextual information is therefore an important capability when dealing with event logs as the next element to predict is often not determined by the previous activities only. In scenarios ATT1 and ATT2, we expect the prediction model to generalize in the presence of context attributes. Note that in these scenarios, the models have seen the context attribute values before, i.e., they are not completely new. It is only the combination of activity and a context attribute that has not been observed in the training data. The first example, *L4* in Table 5, shows a situation in which different resources are involved in the activities, but do not influence the way the process continues. In contrast, Log *L5* in Table 6 shows an example in which the next activity to execute depends on the cost value observed in a previous one. If *cost* is lower than *500€* then *C* follows. If it's higher than *500€* then *D* follows. In this scenario, the model has to learn to make predictions based on the *cost* attribute.

Table 5 **ATT1**-different resources *R* performing *B*

Event Log *L4*
$\langle(\mathbf{A}, R1), (\mathbf{B}, R100), (\mathbf{C}, R2)\rangle$
$\langle(\mathbf{A}, R1), (\mathbf{B}, R101), (\mathbf{C}, R2)\rangle$
$\langle(\mathbf{A}, R1), (\mathbf{B}, R101), (\mathbf{C}, R2)\rangle$

Table 6 **ATT2**-decision depending on cost after *B*

Event Log *L5*
<(**A**, 2€), (**B**, 2€), (**C**, 2€)>
<(**A**, 499€), (**B**, 499€), (**C**, 499€)>
<(**A**, 501€), (**B**, 501€), (**C**, 501€)>

- **ATT1**: *L4* and prefix $\langle(\mathbf{A}, R1), (\mathbf{B}, R1)\rangle$. Expected prediction: *C*. In *L4*, different resources are involved in activity *B*. However, *C* follows *B* every time. Thus, the prediction model should know that regardless of the resource *R* in activity *B*, *C* always follows.
- **ATT2**: *L5* and prefix <(A, 2€), (B, 499€)>. Expected prediction: *C*. While this exact combination of activities and cost values has not been observed before, the model should have learned that with a cost of 499€ in the second event, *C* follows.

Similar to the unseen activities discussed in the previous section, prefixes may also contain unseen context attribute values, such as a new resource due to a new employee becoming involved in the process after the model has been trained. To demonstrate these scenarios, we also employ logs *L4* and *L5* but discuss other prefixes.

- **ATT3**: *L4* and prefix ⟨(A, R1), (B, R37)⟩. In this scenario, resource *R37* in the second event has never been seen before. Similar to the situation with an unseen activity in CF5, the model could predict any label on a positional basis, e.g., *C* or indicate that it does not know.
- **ATT4**: *L5* and prefix <(A, 200€), (B, 200€)>. Though 200€ is an unseen value for the quantitative attribute cost, it is between the seen values 2€ and 499€. Thus, we argue that an optimal prediction model should have learned to predict *C*.

7 Experiments

Based on the generalization scenarios introduced in the previous section, we will now evaluate if, and to what extent, existing next activity prediction models can generalize, according to the definition given in Sect. 5. We use an evaluation setup following the requirements and observations made in previous sections. Specifically, we simulate event logs that are based on the scenarios presented in Sect. 6. The event logs are split, per scenario, such that all samples in the test split are unseen, which allows us to compute a meaningful generalization error. In addition to the single scenarios, we experiment with combinations of these scenarios to increase the complexity and difficulty for generalization. Further, we also evaluate the PPM models on real-life event logs, containing real scenarios with presumably different characteristics and different variability, to validate the findings made in the simulated settings. Based on the results, we can assess whether and to what extent existing methods can generalize in which scenarios.

We present the process models, event logs and scenario splits in Sect. 7.1, the prediction models and training setup in Sect. 7.2, and discuss how to measure the performance in Sect. 7.2.5. The experiments on the single scenarios are discussed in Sect. 7.3, on the combinations in Sect. 7.4, and on the real-life event logs in Sect. 7.5.

7.1 Process Models and Event Log Data for the Simulated Single Scenarios

We create a set of 5 process models and accompanying event logs that implement the generalization scenarios *CF1-5* and *ATT1-4* introduced in the previous section, allowing to test the generalization in a controlled environment. Note that not all process models have the same activities or attributes as the sample logs presented before. Nevertheless, they are used to generate event logs for these scenarios, as they implement the scenarios on a conceptual level.

In addition to the separation between control flow and context attributes, as shown in Fig. 3, we evaluate all scenarios with two complexity levels: simple and advanced. This is intended to make the evaluation more representative and account for the different levels of complexity that can occur in real-world processes. The 5 process models (Figs. 4, 5 and 6) are used to generate event logs for the 9 scenarios and both complexity levels. In total, we evaluate on 18 event logs. The event logs are split manually by a ratio of 80/20 such that there are no duplicates between the prefixes in the training and the test split. Further, the manual split ensures that we evaluate in each log one specific scenario. Table 7 shows an overview of the scenarios, process models, and prediction targets. Each of the 18 settings that we evaluate is characterized by a scenario that it replicates, the process model used to generate

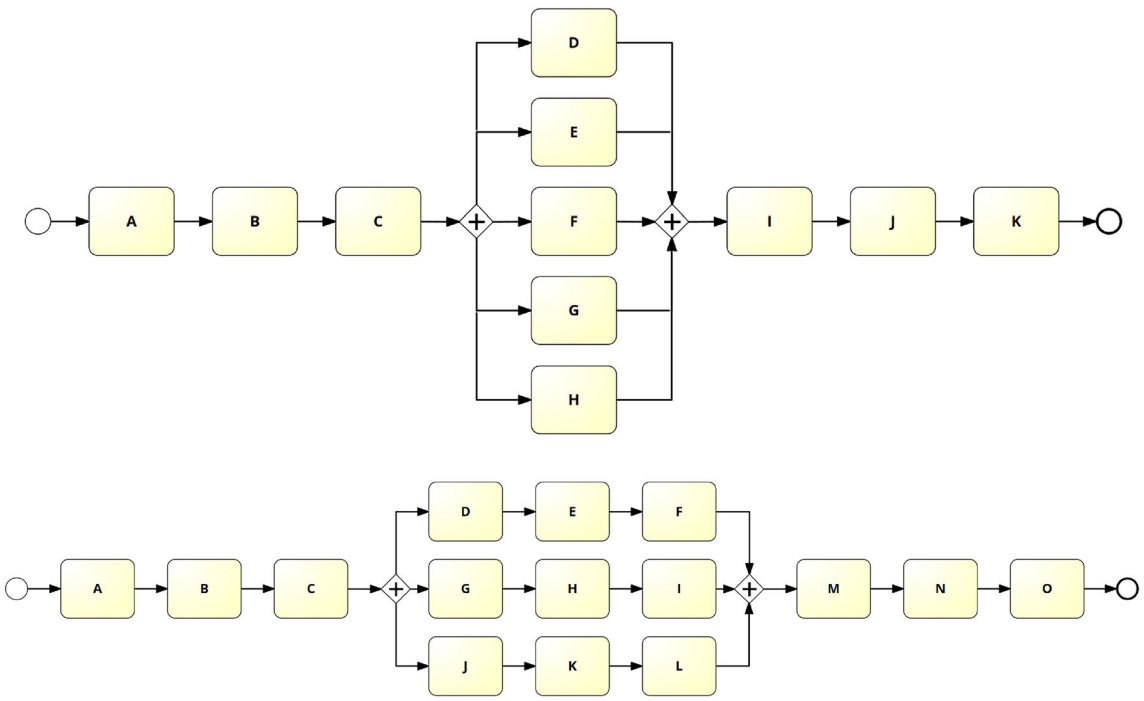

Fig. 4 BPMN models 1 (top) with concurrency simple and model 2 (bottom) for concurrency advanced version for scenarios CF1 and CF2

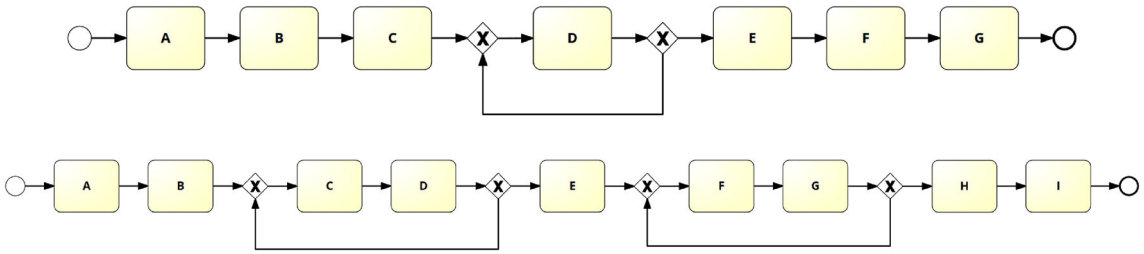

Fig. 5 BPMN models 3 (top) with loops simple and model 4 (bottom) for loops advanced for scenarios CF3 and CF4

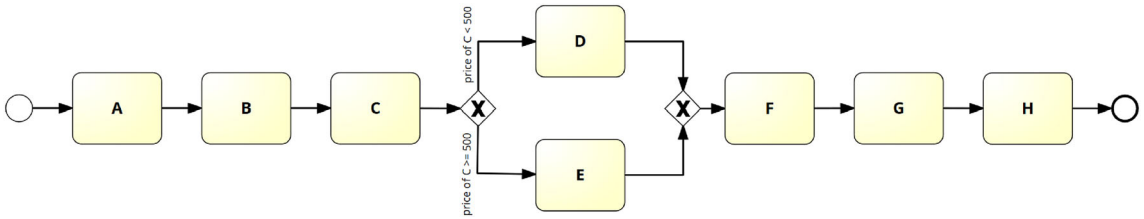

Fig. 6 BPMN model 5 with attribute-dependent exclusive choice for scenarios ATT1-4 and CF5

Table 7 Overview of generalization scenarios and the respective prediction targets

		Simple		Advanced	
Scenario	Description	Targets	Model	Targets	Model
CF1	Unseen concurrent permutation, prediction after concurrent block	*I, J, K*	1	*M, N, O*	2
CF2	Unseen concurrent permutation, prediction within concurrent block	*D, E, F, G, H*	1	*D, E, F, G, H, I, J, K, L*	2
CF3	Unseen loop count, prediction after loop	*F, G*	3	*I*	4
CF4	Unseen loop count, prediction of loop	*D, E*	3	*C, D, E, F, G, H*	4
CF5	Unseen activity	*D, E, F, G, H*	5	*D, E, F, G, H*	5
ATT1	Unseen combination of activity and resource	*D, E, F, G, H*	5	*D, E, F, G, H*	5
ATT2	Unseen combination of activity and price	*D, E*	5	*D, E*	5
ATT3	Unseen resource	*D, E, F, G, H*	5	*D, E, F, G, H*	5
ATT4	Unseen price	*D, E*	5	*D, E*	5

the event log, the prediction targets that we evaluate generalization on, and a complexity level.

7.1.1 Process Models

Figures 4, 5 and 6 show the BPMN models used to generate the event logs:

- *BPMN models 1 and 2* (Fig. 4) feature parallel activities for scenarios *CF1* and *CF2*. BPMN model 1 is simpler with 5 single parallel activities (leading to a total of 120 possible activity permutations), whereas the advanced one features 9 activities with a partial order among 3 sets of 3 activities (leading to a total of 1680 possible permutations). Model 1 is used for the simple setting and model 2 for the advanced setting.
- *BPMN models 3 and 4* (Fig. 5) feature loops for scenarios *CF3* and *CF4*. In the simple model 3, a single activity can repeat up to 25 times (which leads to 25 possible variants), whereas in the advanced model, two sequences of two activities can independently repeat up to 10 times (leading to 100 possible variants). These models are used for the simple and advanced setting respectively.
- *BPMN model 5* (Fig. 6) features a context attribute-dependent choice and is used for scenarios *ATT1-4* and *CF5*. Depending on the value of attribute *price* in activity *C*, either *D* or *E* is executed next. Because scenarios *ATT1-4* are related to the context perspective, the control flow of the process remains constant for these. For *ATT1* and *ATT2*, we instead increase complexity by expanding the domain of the relevant attributes (attribute resource for *ATT1* or attribute price

for *ATT2*) from 100 possible values in the simple process to 1000 in the advanced one. For *ATT3* and *ATT4*, we assign unseen attribute values to one event (activity *C*) in the simple process and to two events (activities *B, C*) in the advanced one. We also use this process model to generate training data for scenario CF5. For the test data, we then replace one activity (*C*) in the simple setting and two activities (*B, C*) in the advanced setting with new, unseen ones (*X* and *Y*).

7.1.2 Scenario-Specific Prediction Targets

To measure the generalization properties that we are interested in as specifically as possible, we only evaluate on a subset of prefix-label pairs in each event log. If we want to assess the ability of a prediction model to generalize to unseen activity orderings in the process with parallel activities (Fig. 4), for example, it would not be meaningful to measure predictive performance on the prefix-label pair [*A, B*] - *C*, because it occurs before the concurrent activities. Instead, the meaningful activities to predict for scenario CF1 simple (prediction after concurrent block) would be *I, J* and *K*, and for scenario CF2 simple (prediction in concurrent block) they would be *D, E, F, G* and *H*. We call the labels of interest prediction targets. A summary of the 9 prediction scenarios and the prediction targets for the simple and advanced versions of each is shown in Table 7.

7.1.3 Event Log Generation and Splits

For each event log, we generate 10,000 cases following the respective process model. Each event has the same five attributes: *case_id, activity, timestamp, resource, price*. The start timestamp of each case is randomly sampled from an interval from January 2015 to December 2019. The timestamps of following events are then iteratively sampled from 48-hour windows beginning with the timestamp of the previous event. The resource and price per event are sampled independently from a set of possible values (100 for the simple settings, 1000 for the advanced settings),

We manually split each event log into 80% training and 20% test data so that there are no prefix samples leaked from the training split to the test split, with regard to the generalization characteristic that is evaluated in that log. For instance, in scenario CF1, we split in such a way that the permutations of the parallel activity execution in the test split have not been seen in the train split. Each sample (x, y) in the test split has an ordering of activities that is not present in the training split. For scenario ATT1, traces and prefixes in both splits may have the same control flow, but the split instead ensures that all combinations of activity *C* and resource in the test split are not included in the train split. As a consequence of this approach, we must split the event logs for scenarios CF2 (prediction in parallel block) and CF4 (prediction in loop) by prefixes instead of traces. Otherwise, it would be impossible to avoid leakage among prefix-label pairs in these scenarios. Consider, for example, the variant [*A, B, C, D, D, D, E, F, G*] generated from the simple loop process model (Fig. 5). Even if this full execution variant only exists in the test split, the prefix [*A, B, C, D*] that it entails would also be part of any other variant that this model may generate, and thus always be found in both training and test split if we were to split on trace level. By splitting on prefixes we can ensure that there are no duplicates.

The result is two splits – training and test – where the samples (x, y) in the test split have not been seen during training and make up 20% of all variants. For instance, for *CF1* we test on 72 unique prefix variants in the simple setting and 1008 unique prefix variants in the advanced setting. This test setting is more extreme than found in reality, but it ensures an accurate assessment of generalization capabilities because the model is trained on a fraction of the variability of the data (see PC1). Further, as no test prefix has been seen during training, their distribution is different from the training data (see PC2).

7.2 PPM Models and Training Setup

For the experiments, we aim at having a representative set of prediction models that reflect the current landscape of PPM approaches, which is mainly driven by deep neural networks (Weinzierl et al. 2024). Aligned with the two types of generalization scenarios (control flow and context attributes), we use two next activity prediction models: one that uses the sequence of activities only and another one that uses additional attributes. For the control-flow-only model, we use an LSTM model. LSTM models, developed for sequential data, have shown to be a good fit for PPM tasks where prefixes and traces are considered as sequences of events (Weinzierl et al. 2024; Neu et al. 2021).

The context-aware model has to deal with all attribute types that are included in the scenarios *ATT1-4*, i.e., *resource* and *price*. Given that *resource* is usually encoded as a categorical attribute while *price* is a numerical one, the model must be flexible in dealing with categorical and numerical attributes. The number of next step prediction models that can deal with categorical and numerical attributes at once is very limited (we refer the interested reader to Rama-Maneiro et al. (2021, Tab. 3) and Pfeiffer et al. (2021, Tab. 4)). To ensure consistency in the results, we choose one context-aware model, the MPPN, as it can process both types of attributes. The MPPN is very flexible in the number and types of context attributes that can be

considered and has shown to perform very well on a range of PPM tasks that require context information, including next step and outcome prediction (Pfeiffer et al. 2021), task abstraction (Rebmann et al. 2023) or deviation prediction (Grohs et al. 2025). In the following, we will elaborate on the prediction models, training strategy, and hyperparameter settings.

7.2.1 Prediction Model Architecture

LSTM model We used a simple LSTM model that is similar to the model proposed by Evermann et al. (2017), featuring an embedding layer for the activities, a single LSTM layer (16 neurons), followed by one fully connected layer for classification.

MPPN model The MPPN is a process representation learning model designed to solve a variety of PPM tasks while being flexible with respect to the attributes in events to use for prediction. It consists of three parts: A single CNN model that extracts features for each perspective, i.e., the sequence of attribute values; a fully connected part that pools and combines the features extracted per perspective; and a configurable number of prediction heads which can be used to solve a variable number classification or regression tasks. More details about its architecture can be found in Pfeiffer et al. (2021). In all experiments with synthetic data, the MPPN uses the attributes *activity*, *resource*, *timestamp*, and *price* in the input, even if the attributes are not relevant for the next activity nor being predicted. The CNN part got pre-trained on variant classification as described in Pfeiffer et al. (2021), while the fully connected part for pooling consists of a single layer with 16 neurons followed by a single classification head for next activity prediction.

7.2.2 Event Log Preprocessing

For all 18 event logs and both splits, we created prefixes of length 64. Shorter prefixes were padded with a distinct token.

7.2.3 Hyperparameter Setting and Tuning

Since the synthetic scenarios are rather simple (in order to be comprehensively evaluable) and much simpler than real-life event logs, the prediction models are at risk of overfitting, as their capacity easily surpasses the variability in the event log (see PC3). For instance, *CF1* simple features 120 permutations of parallel activities, resulting in 288 unique samples (x, y) of prefixes and targets in the training log. The number of parameters of the LSTM model is at least an order of magnitude larger. Thus, we performed a hyperparameter search for the number of neurons (4, 8, 16, 32, 64) per layer (embedding layer and LSTM layer for the LSTM model; fully connected layer in the MPPN model) and found that increasing the number of neurons beyond 16 did not lead to better performance in the single scenario experiments. For the experiments with combinations of scenarios and on real-life event logs, the number of neurons per layer had to be increased as detailed in the respective paragraph. In result, both the LSTM and MPPN feature around 5000 - 10,000 trainable parameters in the single scenario experiments, depending on the number of activities in the event log.

7.2.4 Training Procedure

We trained all models on all event logs on the respective training split using AdamW as optimizer on the cross-entropy and a learning rate of 1e-4. The training log was split into a training and evaluation part. Note that the training split contains all prefixes, i.e., the model is trained also on prefixes with other target values indicated in Table 7. For instance, while we evaluate in *CF2* only predictions within concurrency (activities *D*, *E*, *F*, *G*, *H*) and report the performance on samples with those target labels only, we still train the prediction model on samples with target labels *B*, *C* and *I*, *J*, *K* to ensure that they learn the whole process and not only a part of it.

Regularization In addition to using small model sizes, we used regularization techniques to prevent overfitting, addressing PC3. First, we used early stopping to stop training when the evaluation loss did not reduce any further (which has shown to be the most effective countermeasure against overfitting in previous work (Peeperkorn et al. 2024)). Further, we used dropout with 10% in both models.

7.2.5 Generalization Performance Evaluation

We evaluate the generalization performance of the prediction models as motivated in Sect. 5. We primarily asses the prediction models with the cross-entropy estimation, as defined in Eq. 2. In addition, we also report the accuracy, since it has been frequently used in next activity prediction. To evaluate whether the models generalize in predicting next activities, we use the following criteria, which follow from the conceptualization in Sect. 5.

1. The prediction model reaches a low training error
2. The prediction model reaches a low generalization error

To assess criterion (1), we inspect the absolute values of cross-entropy and accuracy. To assess criterion (2), we measure the generalization error as the difference between

the error on the training and test split. If both are fulfilled, i.e. the model performs very well and almost identically on the test split as on the training split, it generalizes well. If the differences between training and test error are more significant or the performance on the training or test split bad, the model generalize less well. We argue that generalization is a capability that should be quantified. We cannot say that a model does generalize if and only if the generalization error is smaller than a certain value. Rather, the smaller the error and the smaller the generalization error, the better the model generalizes. The higher the error or delta, the less the model can generalize. Thus, we refrain from using a fixed threshold to quantify generalization.

Remember that the test splits contain only unseen prefixes with the target next activities specified in Table 7. In contrast, the training and evaluation splits contain prefixes with all activities as target values. If reporting the performance on the whole training set, the numbers would not be comparable to the numbers obtained on the test set. Therefore, we report the cross-entropy and accuracy of the models reached on the training split as training performance, limited to prefixes with the same set of activities as used in the test split. This ensures that the training and test performance can be compared adequately. All PPM models are trained and tested 5 times in all scenarios and the results are averaged.

7.3 Single Scenarios

Table 8 shows the cross-entropy estimation and accuracy that the models achieve on the train and test split in each of the 18 evaluation event logs. In most scenarios there is at least one model that generalizes well if considering the specifics of the scenario. Scenario CF5 is the only scenario where the performance of all models decreases considerably, suggesting insufficient generalization. We will discuss the results in the following in detail.

7.3.1 Results per Scenario

In *CF1* and *CF3* the prediction models generalize very well with perfect accuracy and cross-entropy - both in simple as in the advanced setting. This shows that they have learned that always the same activity follows behind the block of concurrent or looping activities, no matter in which order the activities occur. In scenario *CF2*, the LSTM performs as good on the test split as it performs on the training split while the MPPN performs a little worse on the test split. The high cross-entropy and low accuracy suggest that the models do not perform well and, thus not fulfill criteria (1). As this scenario involves predictions within concurrency only, it is strongly affected by having multiple valid continuation options that induce a minimal error that cannot be avoided. In these conditions, the performance has to be considered good as we will show and discuss in depth

Table 8 Cross-entropy (CE) according to Eq. 2 and accuracy (ACC) for the samples with the respective target labels given in Table 7 on the training and test split

		LSTM				MPPN			
		Train		Test		Train		Test	
		CE	ACC	CE	ACC	CE	ACC	CE	ACC
Simple	CF1	0.001	100.00%	0.002	100.00%	0.0	100.00%	0.0	100.00%
	CF2	0.841	52.28%	0.844	52.60%	0.800	56.11%	1.037	49.51%
	CF3	0.000	100.00%	0.000	100.00%	0.0	100.00%	0.0	100.00%
	CF4	0.256	92.19%	0.236	93.47%	0.253	92.32%	1.253	82.85%
	CF5	0.177	88.63%	0.266	85.74%	0.009	99.72%	0.926	84.16%
	ATT1	0.176	88.62%	0.177	88.05%	0.040	97.51%	0.041	97.41%
	ATT2	0.702	50.41%	0.700	61.26%	0.026	99.15%	0.090	96.16%
	ATT3	0.177	88.70%	0.177	88.67%	0.007	99.78%	0.009	99.71%
	ATT4	0.700	54.00%	0.698	55.26%	0.087	96.91%	0.326	90.91%
Advanced	CF1	0.001	100.0%	0.001	100.0%	0.0	100.0%	0.0	100.0%
	CF2	0.829	55.76%	0.895	53.22%	0.798	57.52%	1.491	48.06%
	CF3	0.000	100.0%	0.000	100.0%	0.0	100.0%	0.001	99.43%
	CF4	0.205	92.04%	0.205	92.45%	0.200	92.26%	0.231	91.20%
	CF5	0.177	88.53%	0.821	75.00%	0.007	99.80%	2.720	56.12%
	ATT1	0.176	88.72%	0.176	88.66%	0.014	99.56%	0.014	99.54%
	ATT2	0.701	51.03%	0.702	50.75%	0.104	96.18%	0.145	94.15%
	ATT3	0.177	88.31%	0.177	88.60%	0.005	99.83%	0.006	99.78%
	ATT4	0.699	54.24%	0.697	55.18%	0.055	98.05%	0.321	91.92%

Table 9 Probability distribution $q(y)$ learned by the LSTM for activities with an unseen prefix on *CF2* simple. Probabilities in bold are those assigned to the ground truth activities y

Unseen prefix	A	B	C	D	E	F	G	H	I	J
$\langle A, B, C, E \rangle$	0.00	0.00	0.00	**0.22**	0.01	**0.30**	**0.23**	**0.24**	0.00	0.00
$\langle A, B, C, F, D \rangle$	0.00	0.00	0.00	0.01	**0.28**	0.00	**0.36**	**0.35**	0.00	0.00
$\langle A, B, C, D, E, G \rangle$	0.00	0.00	0.00	0.01	0.00	**0.49**	0.01	**0.48**	0.00	0.00
$\langle A, B, C, D, E, F, G \rangle$	0.00	0.00	0.00	0.01	0.01	0.01	0.01	**0.97**	0.00	0.00

along Table 9. Similar observations can be made for *CF4*. In this scenario, the prediction model has to choose between two activities (*D* or *E*) while in *CF2* there are up to five alternative activities (*D*, *E*, *F*, *G*, *H*) per sample. We argue that this explains why the accuracy in *CF4* is much higher than in *CF2* and cross-entropy lower.

Scenario *CF5* is the scenario with the strongest increase in generalization error (both in cross-entropy and accuracy). In the simple version, the LSTM model still performs well with little differences between training and test performance. This means that the model is still able to make correct predictions if one activity is replaced by a unseen one. In the advanced version of *CF5*, the generalization error is much higher. Following our definition of generalization, we have to conclude that both models do not generalize well in this scenario. Nevertheless, the LSTM still reaches an accuracy of 75%, meaning that although the cross-entropy is much higher (caused by higher variance in the model's predictions), the model still gives the correct activity the highest probability in most of the unseen samples.

We assume that the models can make predictions on positional basis, i.e., they know that a certain activity appears in a certain position even if an unseen activity is introduced, but assigning this activity a little lower probability as normally which explains the higher cross-entropy. This means, that the models have learned that after seeing $\langle A, UNK, UNK \rangle$ (UNK represents the encoded unknown activity) in the fourth position either D or E follows. Further, even if seeing unknown activities in the prefix, they have learned that after F always G follows. Given the difficulty of this task, the results are remarkable and suggest that generally valid patterns have been learned to a certain extent.

In the scenarios *ATT1* to *ATT4* that require context attributes, the MPPN performs, as expected, much better than the LSTM, which does not consider these attributes. In the scenarios *ATT1*, *ATT2*, and *ATT3*, the MPPN generalizes well. Scenario *ATT4* (unseen price attribute) is the most challenging scenario among the context-aware ones, and we see strong increase in cross-entropy and a drop in accuracy. Nevertheless, the performance is still reasonably high, which is why we argue that the model still generalizes in this scenario, too. This means that the model can deal with unseen combinations of attribute values and also unseen attribute values and predict the correct activity (D or E) based on the context attributes.

7.3.2 Differences Between Models and Capacities

Comparing the models reveals that the LSTM model performs better on the scenarios with control flow only (*CF1-5*) and the MPPN performs better on the scenarios with context attributes. For *CF1-4*, the LSTM performs almost equally well on the training and test split, but the MPPN cannot reduce the generalization error as far as the LSTM can. In scenarios *CF4*, *ATT1*, *ATT3*, and *ATT4*, the models perform better in the advanced setting than in the simple setting. We attribute this to a better fit of the capacity of the prediction models to the complexity of the prediction task. The prediction models used for all scenarios are very small in terms of models size and the number of neurons. For some scenarios, the capacity of the model might still be too high in the simple setting while the advanced settings, featuring higher variability, fits the capacity better.

7.3.3 Probability Distribution in Predictions

In scenarios like *CF2*, the prediction model reaches a low generalization error, but the overall performance, at first, seems not to be that good. As described above, we attribute this to the high number of parallel activities. We will show that next activity prediction models actually learn that multiple activities can follow, which they express by giving them certain probabilities in $q(y)$, showing that they make accurate predictions in such scenarios. For this, we show the probabilities $q(y)$ that the LSTM model gives to different next activities y, i.e., the result after applying the softmax function on the output of the classification layer (these probabilities sum up to 1).

Table 9 shows the probability distribution over the activities for some prefixes in the test split (note that they are all unseen) of *CF2*. As we can see, the predicted probabilities resemble the process behavior accurately. The

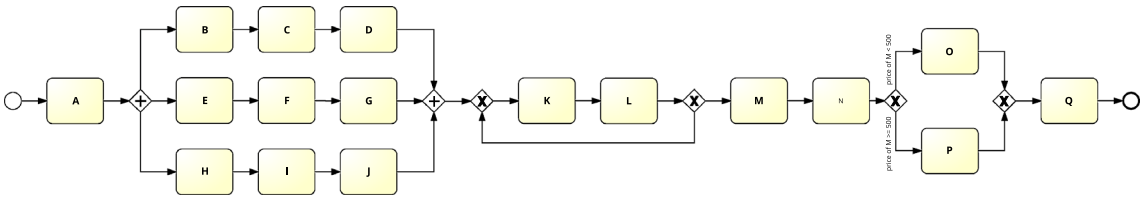

Fig. 7 BPMN model with a combination of concurrency, loop, and choice

LSTM model gives reasonably high probabilities to all next activities that are possible. For instance for the prefix $\langle A, B, C, E \rangle$ it gives activities *D*, *F*, *G*, *H* high probabilities and all other activities very low probabilities. While the probabilities for *D*, *G* and *H* are almost identical, *F* is given a higher probability. This might represent the distribution found during training, i.e., that in more situations during training *F* did follow compared to the other three activities.

7.4 Scenario Combinations

In addition to evaluating prediction models' generalization capabilities on models with isolated characteristics, we also evaluate them in settings where several characteristics occur at the same time, as is typically the case in real-world processes. To this end, we combine parts of the previous process models into a new one that features concurrency, a loop, and an attribute-dependent choice (Fig. 7). We then generate 3 combination scenarios with progressively increasing complexity and evaluate the same two PPM models on them:

- Comb1: This scenario combines CF1 and CF3, featuring an unseen permutation of concurrent activities and a loop.
- Comb2: Extends scenario *Comb1* by additionally replacing 3 activities and resources in the prefix with unseen ones. It is therefore a combination of CF1, CF3, CF5, and ATT3.
- Comb3: Extends scenario *Comb2* by additionally assigning an unseen price value in activity *N*, so that it additionally includes ATT4.

Like in the isolated scenarios, we evaluate performance only for the prediction of generalization-relevant target activities (activities *N, O, P, Q* for *Comb1* and *Comb2*; activities *O, P, Q* for *Comb3*). Due to the higher complexity in the data, we increased the number of neurons in the LSTM and MPPN layer to 32 and used the same training and testing strategy as before, in which the test split contains unseen prefixes only. We averaged the performance values across 5 runs.

Results The results shown in Table 10 are very much in line with the results of the single scenario experiments. The LSTM reaches almost the same cross-entropy and accuracy in the training as in the test split. This indicates that the model performs equally well on the unseen prefixes as on the seen ones. However, as the MPPN shows, higher performance can be reached if using context attributes. Although the MPPN reaches very good performance on the training split, it performs less well on the test split. This indicates that the model can make accurate predictions, but not for all unseen prefixes. Interestingly, it still performs well on *Comb2*, which includes unseen activities as in *CF5* where the model did not perform well. However, on *Comb3*, which additionally includes unseen price values (as *ATT4* where the model performed well), its generalization error increases drastically. This indicates that the model cannot generalize well if these scenarios occur in combination.

7.5 Real-Life Event Logs

The previous experiments have evaluated the generalization capabilities of PPM models on specific scenarios which allowed to assess their performance for individual process characteristics. Although they exhibit a high variability, real-life processes might contain other or additional characteristics in their traces. Therefore, we conduct an additional experiment on real-life data to validate the

Table 10 Cross-entropy (CE) according to Eq. 2 and accuracy (ACC) for the combinations of scenarios on the samples with respective target activity

	LSTM				MPPN			
	Train		Test		Train		Test	
	CE	ACC	CE	ACC	CE	ACC	CE	ACC
Comb 1	0.233	83.82%	0.233	83.85%	0.074	97.07%	0.169	93.05%
Comb 2	0.232	83.34%	0.251	83.60%	0.076	97.00%	0.324	87.63%
Comb 3	0.349	75.29%	0.379	75.45%	0.068	97.63%	2.319	56.96%

Table 11 Variant leakage and prefix leakage if splitting BPIC event logs by time into train and test with a ratio of 50:50, i.e. such that the last half of the traces is in the test split

Log	Traces	Variants	Variant Leakage	Unique Pref	Example Leakage	Unseen Test Pref
BPIC12	13,087	4,366	66.38%	60,866	78.69%	29,861
BPIC17	31,509	15,930	50.89%	279,796	77.54%	50,530

Table 12 Cross-entropy (CE) according to Eq. 2 and accuracy (ACC) for setting *50:50* (no removal of leaked prefixes) and setting *unseen* (all leaked prefixed removed)

	LSTM				MPPN			
	Train		Test		Train		Test	
	CE	ACC	CE	ACC	CE	ACC	CE	ACC
BPIC 12 *50:50*	0.381	85.29%	0.394	84.55%	0.448	83.36%	0.448	83.13%
BPIC 12 *unseen*	0.381	85.29%	0.566	81.49%	0.448	83.36%	0.890	72.22%
BPIC 17 *50:50*	0.376	87.22%	0.349	87.20%	0.246	91.13%	0.241	91.21%
BPIC 17 *unseen*	0.376	87.22%	0.549	81.15%	0.246	91.13%	0.513	82.70%

findings made in the previous experiments on simulated data.

For this purpose, we chose the BPIC17 event log, due to its size and the observation that many new variants appear over time (Peeperkorn et al. 2024, Fig. 1). To force a scenario where the test set contains many unseen samples, we split the event log by time such that the first 50% of traces are used for training and the last 50% for testing. However, as Table 11 shows, there is still a significant leakage of variants and prefixes from the training to the test split if splitting that way. In detail, 50.89% of full-trace variants are leaked from the training to the test set; the prefix leakage (see Sect. 3) is even higher at 77.54%. Therefore, as an additional setting, we removed all prefixes from the test split that are also included in the training split. This results in 29,861 unseen samples (x, y), which were not seen during training and can be used for testing. Evaluating both settings also allows to see how the performance changes when being evaluated on only a portion of unseen samples to unseen samples only. In addition, we evaluate on the BPIC12 (complete) event log, which is an earlier version of the same process as in BPIC17, using the same setting, too.

In contrast to the experiments on simulated data where we evaluated on prefixes with specific target activities only, no such prefix sampling with specific target activities is done. For the real-life event logs used in the experiments, we do not know, e.g., which activities are executed concurrently, in a loop, or in other specific circumstances. Therefore, we report as training performance the cross-entropy and accuracy reached on the hold-out evaluation split in the last epoch of training. As test performance, we report the cross-entropy and accuracy on the full test split (setting "*50:50*") and on the test split with unseen prefixes only (setting "*unseen*"). Note that the training performance therefore is the same in both settings, as we use the same model after training for both test settings.

As the logs are much larger and the processes more complex, we increased the capacity of the models. For the LSTM model, we used 2 LSTM layers with 64 neurons each and for the MPPN model, we used 2 layers in the fully-connected part with 256 neurons each. Smaller models have been found to reach a lower test performance while increasing their size led to overfitting. We trained each model 5 times and report the average of accuracy and cross-entropy across all runs.

Results Similar to the experiments before, both prediction models perform considerably well on both real-life event logs and both settings. The generalization error is almost zero in the *50:50* setting. When testing on unseen prefixes only, the generalization error in cross-entropy and accuracy increases while maintaining a high accuracy. Considering the much higher complexity of real-life data, the performance on the unseen samples in the *unseen* setting validates that PPM models can also generalize on challenging, real-life settings.

7.6 Summary

We conclude that next activity prediction models generalize well in almost all scenarios, from simulated to real event logs. The results show that such models can make correct predictions for unseen prefixes even in challenging settings like high concurrency or completely unknown activities. Further, the learned probability distribution

$q(y)$ reflects how prefixes can continue in the process. While the results are not always optimal and there is room for improvement they definitely show that generalization capabilities have been learned by the prediction models.

However, there are differences between the scenarios. In general, scenarios where all values (regardless if activity or any other attributes) have been seen before, e.g., *CF1* - *CF4* and *ATT1* and *ATT2* work better than scenarios where completely unknown values appear in the prefix, e.g., *CF5*, *ATT3*, and *ATT4*. Unknown resource values (*ATT3*) are handled more sophisticatedly than unknown price values (*ATT4*) by the MPPN, hinting towards a better capability in handling categorical values than numerical ones. However, this could also be specific to the architecture of the prediction model.

The detailed analysis of the learned probability distribution in Table 9 also shows that the models learn all valid continuation options, which cross-entropy does account for. In such situations, the next activity to follow is arbitrary, causing a certain error that cannot be avoided. We assume that the cross-entropy of the LSTM reached in *CF2* is bound by the entropy, i.e., the cross-entropy of the prediction model cannot become smaller than the respective entropy of the process (Pfeiffer and Fettke 2024).

In the more complex scenario combinations and real-life event logs, the models still generalize well when being tasked with unseen prefixes. However, certain combinations of unseen categorical and numerical values in the prefix seem to be challenging for state-of-the-art PPM models. Since the generalization error increased the most in scenarios where context is relevant, PPM models should be enhanced in their ability to handle unseen context-attribute combinations which will further increase their overall performance.

8 Discussion

8.1 Discussion of the Experimental Results

As the experimental results show, next activity prediction models generalize well in almost all scenarios with respect to the definition given in Sect. 5. There are differences per scenario and prediction models used. While the LSTM model, which uses control flow information only, performs better in *CF1-5*, the MPPN shows strong generalization in the scenarios requiring context awareness. In addition, there are differences in generalization depending on whether the prefixes contain entirely unseen attribute values or not. The prediction models generalize less well when there are completely unseen values in the prefix, e.g., unseen activities (*CF5*) or unseen attribute values (*ATT3* and *ATT4*), than in scenarios where the values appear in unseen constellations. Further, certain combinations of unseen context attributes are more challenging than others. However, they generalize surprisingly well to unseen prefixes.

We observe that scenarios with unseen values (*CF5*, *ATT3*, and *ATT4*) are harder to generalize. In scenario *CF5* there are unseen values in the control flow, which is the only dimension available. As the activities are out-of-distribution and not part of the training set, we argue that the models show limited OOD generalization capabilities. The unseen attribute values in *ATT3* and *ATT4* are also out-of-distribution but do not necessarily constitute a significant change from the original data. Further, the control flow remains constant with no new activities being introduced. Thus, they qualify less for OOD generalization.

The MPPN generalizes better than the LSTM on the single scenarios involving context attributes. This could validate our assumption that additional information in form of context attributes supports generalization, which is in line with previous work (Gerlach et al. 2022). In the more complex scenarios, however, the picture becomes more diverse. While the MPPN reaches a better training performance and often overall higher accuracies, its generalization error is often much higher than the generalization error of the LSTM. While the model can learn dependencies between the attributes on the training set, as indicated by a low error, it does not always seem to be able to transfer this ability to the test set. Conversely, models require more generalization capabilities to include additional context attributes.

The results also show that next activity prediction models accurately learn how unseen prefixes continue by assigning appropriate probabilities to next activities, as seen in Table 9. We conclude that prediction models learn how processes are structured, i.e., how activities are structured and which activities follow next in which situations.

Previous work has defined generalization differently from our conceptualization as the ability to generate unseen traces (Peeperkorn et al. 2022). As our work shows, next activity prediction models can deal very well with unseen prefixes. Thus, their limitations in generating unseen traces must be tied to their generative ability rather than their predictive ability. Although they are accurate in predicting next activities (even for unseen prefixes), they might be unable to accurately predict longer future behavior. From this, it follows that they may not be able to generate unseen variants as expected and measured by Peeperkorn et al. (2022). To achieve this ability, a more forward-looking training with a larger prediction horizon might help. It also shows that our definition of generalization, focusing specifically on next activity prediction, cannot be applied universally to all PPM tasks.

8.2 Discussion of Limitations

Several limitations apply to this work. First, our definitions of example leakage and accuracy limit do not consider the context attributes found in event logs. Thus, they are only approximations of the values that we would obtain if considering, e.g., all decisive attributes for the next activities. However, as discussed, not all attributes are decisive for the next activity which is why it is complicated to compute the values on the respective attribute set.

Second, the scenarios used in the experiments are not complete, meaning that there might be other situations that require generalization. For instance, a concept drift that changes the process after a certain date might require more challenging forms of generalization which we have not analyzed yet.

Third, although we tested different hyperparameter settings for each experiment, the generalization performance can differ in other settings. This might result in better performance as reported, potentially resulting in better generalization. For instance, the models might have overfitted in certain scenarios although countermeasures have been implemented. Also, other PPM approaches, which we have not explored so far, can perform very differently.

Fourth, the incompleteness of the training event log could be higher than assumed. We trained the PPM models on 50%-80% of all traces. While we found that even with a split of 50% most prefixes (close to 80%) exhibit the same control flow, this ratio could be much smaller in other real-world applications. Thus, it remains open whether next activity prediction models would also generalize when having seen, e.g., only 30% of the total behavior.

Lastly, the conceptualization of generalization and the experiments have been conducted with a focus on the task of next activity prediction. PPM consists of many more tasks that we have not analyzed yet. As many other tasks also involve an unseen prefix but differ in the attribute to predict, a similar procedure as suggested in this paper might be applicable to those tasks, too. For instance, the observation and characteristic that prefixes often recur in training and application also applies to other PPM tasks. Similarly, the training event log should always be assumed to be incomplete and regularization applied. In contrast, the distribution of target variables can differ in other PPM tasks while cross-entropy cannot and should not universally be applied to all PPM tasks. Therefore, more research is required on generalization in other PPM tasks.

9 Conclusion

In this work, we analyzed generalization in PPM conceptually and empirically. We showed that the current evaluation procedures used for next activity prediction research are flawed and do not allow researchers to draw valid conclusions about their generalization capabilities. This follows from the observation that most samples used for testing are duplicates and that these procedures are not able to reliably communicate the actual performance of the model regarding the prediction of what will happen next. Following these observations, we introduce a novel conceptualization of generalization for the task of next activity prediction, which defines generalization based on the definition of generalization in machine learning research: to accurately predict the process behavior for unseen prefixes. Further, we discuss challenges for generalization in next activity prediction to guide adequate evaluation procedures, such as variability and incompleteness of event logs, adequate train-test-splitting, and the choice of evaluation metric. While this conceptualization has been developed focusing on next activity prediction, certain challenges for generalization also apply to other PPM tasks that work with trace prefixes.

To demonstrate what it means for a next activity prediction algorithm to generalize, we presented various example scenarios. We evaluated two state-of-the-art next activity prediction models with regard to how they performed in each scenario, which allowed for the evaluation of generalization capabilities for each situation. As the results show, existing models generalize well in predicting the next activity in almost all scenarios, considering the control flow only and when using context attributes. Further, the models were also able to generalize very well in more complex scenarios and on real event logs.

While the overall accuracy on, e.g., BPIC 12 and 17 is already high, there is a decline between training and test performance. In turn, overall better performance can be achieved by developing context-aware models that can generalize to unseen context attribute combinations. Confidently handling unseen context attribute combinations seems to be a limiting factor for state-of-the-art next activity prediction models and potentially for PPM models in general. Note that the combination of diverse attributes of different types found in event logs is also a unique challenge for machine learning in general. Further, as processes are known to change over time, generalization in PPM also requires developing models that can adapt to changing process behavior in order to improve their generalization capabilities.

In the experiments, we have tested the PPM models on unseen prefixes only, which we have argued to resemble the to-be-expected conditions in application. However, our analysis also revealed that, although new variants are introduced over time, most prefixes of traces in the real data are highly repetitive and only a small share of samples contain unseen variants, even when 50% of the most recent

traces are used as test split. Thus, our experimental conditions, which allowed us to measure a meaningful generalization error, might be stricter and more difficult than real-life conditions. Nevertheless, we encourage the community to test their models specifically for generalization in a setting with unseen prefixes only. This might reveal larger performance differences between models and provide new insights. Further, aiming for better generalization on the unseen traces should make the models robust for various real-life conditions.

Finally, the ability of the tested models to make correct predictions on unseen prefixes shows that they must have learned a representation of the process from the data that closely resembles process behavior. This representation presumably contains high-level process features that describe the behavior of the process.

In future work, we plan to tackle more sophisticated instances of OOD generalization, i.e., investigate more challenging prediction settings such as concept drifts that introduce more significant changes to the process than those present in the scenarios in this paper. Further, prediction models that can adapt their learned probability distribution after training to account for changes in the data are of particular interest.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Abb L, Pfeiffer P, Fettke P, Rehse JR (2024) A discussion on generalization in next-activity prediction. BPM workshops. Springer, Heidelberg, pp 18–30

Back CO, Debois S, Slaats T (2019) Entropy as a measure of log variability. J Data Semant 8:129–156

Baier S, Dunzer S, Fettke P, Houy C, Matzner M, Pfeiffer P, Rehse JR, Scheid M, Stephan S, Stierle M (2020) The MobIS-challenge 2019. Enterpr Model Inf Syst Archit 15:1–25

Bishop CM (2006) Pattern recognition and machine learning. Springer, Heidelberg

Bishop CM, Bishop H (2024) Deep learning foundations and concepts. Springer, Heidelberg

Breuker D, Matzner M, Delfmann P, Becker J (2016) Comprehensible predictive models for business processes. MIS Q 40:1009–1034

Brunk J, Stottmeister J, Weinzierl S, Matzner M, Becker J (2020) Exploring the effect of context information on deep learning business process predictions. J Decis Syst 29:328–343

Buijs JCAM, van Dongen BF, van der Aalst WMP (2014) Quality dimensions in process discovery: the importance of fitness, precision, generalization and simplicity. Int J Coop Inf Syst 23(1440):001

Di Francescomarino C, Ghidini C (2022) Predictive process monitoring. Process mining handbook. Springer, Heidelberg, pp 320–346

Evermann J, Rehse JR, Fettke P (2017) Predicting process behaviour using deep learning. Decis Support Syst 100:129–140

Fani Sani M, Vazifehdoostirani M, Park G, Pegoraro M, van Zelst SJ, van der Aalst WMP (2023) Performance-preserving event log sampling for predictive monitoring. J Intell Inf Syst 61:53–82

Gerlach Y, Seeliger A, Nolle T, Mühlhäuser M, Franch X, Poels G, Gailly F, Snoeck M (2022) Inferring a multi-perspective likelihood graph from black-box next event predictors. Advanced information systems engineering. Springer, Heidelberg, pp 19–35

Goodfellow I, Bengio Y, Courville A (2016) Deep Learning. MIT Press, Cambridge

Grohs M, Pfeiffer P, Rehse JR (2025) Proactive conformance checking: An approach for predicting deviations in business processes. Inf Syst 127(102):461

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9:1735–1780

Jiralerspong M, Bose J, Gemp I, Qin C, Bachrach Y, Gidel G (2024) Feature likelihood divergence: Evaluating the generalization of generative models using samples. In: NeurIPS, Curran Associates, pp 33,095 – 33,119

Jurafsky D, Martin JH (2025) Speech and language processing: an introduction to natural language processing, computational linguistics, and speech recognition with language models, 3rd draft edn. https://web.stanford.edu/~jurafsky/slp3/, online manuscript released 12 Jan 2025, accessed 13 Feb 2025

Käppel M, Jablonski S, Indulska M, Reinhartz-Berger I, Cetina C, Pastor O (2023) Model-agnostic event log augmentation for predictive process monitoring. Advanced information systems engineering. Springer, Heidelberg, pp 381–397

Kaufman S, Rosset S, Perlich C, Stitelman O (2012) Leakage in data mining: formulation, detection, and avoidance. ACM Transact Knowl Discov Data 6:1–21

Kneser R, Ney H (1995) Improved backing-off for m-gram language modeling. In: International conference on acoustics, speech, and signal processing, IEEE, pp 181–184

Kratsch W, Manderscheid J, Röglinger M, Seyfried J (2021) Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. Bus Inf Syst Eng 63:261–276

Le M, Gabrys B, Nauck D, Bramer M, Petridis M (2012) A hybrid model for business process event prediction. Research and development in intelligent systems. Springer, Heidelberg, pp 179–192

Liao T, Taori R, Raji ID, Schmidt L (2021) Are we learning yet? a meta review of evaluation failures across machine learning. In: NeurIPS datasets and benchmarks, Curran Associates

Liessmann A, Wang W, Weinzierl S, Zilker S, Matzner M (2024) Transfer learning for predictive process monitoring. ECIS

Lu W, Wang J, Sun X, Chen Y, Ji X, Yang Q, Xie X (2024) Diversify: a general framework for time series out-of-

distribution detection and generalization. IEEE Trans Patt Anal Mach Intell 46(6):4534–4550
Murphy KP (2022) Probabilistic machine learning: an introduction. MIT press, Cambridge
Neu DA, Lahann J, Fettke P (2021) A systematic literature review on state-of-the-art deep learning methods for process prediction. Artif Intell Rev 55:801–827
Nolle T, Seeliger A, Mühlhäuser M (2018) Binet: Multivariate business process anomaly detection using deep learning. Business process management. Springer, Heidelberg, pp 271–287
Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2022) A multi-view deep learning approach for predictive business process monitoring. IEEE Transact Serv Comput 15(4):2382–2395
Pasquadibisceglie V, Appice A, Castellano G, Malerba D (2024) JARVIS: Joining adversarial training with vision transformers in next-activity prediction. IEEE Transact Serv Comput 17(4):1593–1606
Peeperkorn J, van den Broucke S, De Weerdt J (2022) Can recurrent neural networks learn process model structure? J Intell Inf Syst 61:27–51
Peeperkorn J, van den Broucke S, De Weerdt J (2024) Validation set sampling strategies for predictive process monitoring. Inf Syst 121(102):330
Pfeiffer P (2022) Business process representation learning. In: BPM doctoral consortium, CEUR, Münster, Germany
Pfeiffer P, Lahann J, Fettke P (2021) Multivariate business process representation learning utilizing gramian angular fields and convolutional neural networks. Business process management. Springer, Heidelberg, pp 327–344
Pfeiffer P, Lahann J, Fettke P (2023) The label ambiguity problem in process prediction. BPM workshops. Springer, Heidelberg, pp 37–44
Pfeiffer P, Fettke P (2024) Trace vs. time: Entropy analysis and event predictability of traceless event sequencing. In: BPM forum, Springer, Heidelberg, pp 72–89
Rama-Maneiro E, Vidal J, Lama M (2021) Deep learning for predictive business process monitoring: review and benchmark. IEEE Transact Serv Comput 16:739–756
Rebmann A, Pfeiffer P, Fettke P, van der Aa H (2023) Multi-perspective identification of event groups for event abstraction. ICPM workshops. Springer, Heidelberg, pp 31–43
Rehse JR, Dadashnia S, Fettke P (2018) Business process management for industry 4.0 - three application cases in the DFKI-Smart-Lego-Factory. IT - Inf Technol 60:133–141
Ruta D, Majeed B (2011) Business process forecasting in telecom industry. In: IEEE GCC conference and exhibition, IEEE, pp 389–392
Stevens A, Peeperkorn J, De Smedt J, De Weerdt J (2023) Manifold learning for adversarial robustness in predictive process monitoring. In: International conference on process mining, IEEE, pp 17–24
Tax N, Verenich I, La Rosa M, Dumas M (2017) Predictive business process monitoring with LSTM neural networks. Advanced information systems engineering. Springer, Heidelberg, pp 477–492
Tax N, van Zelst SJ, Teinemaa I, Gulden J, Reinhartz-Berger I, Schmidt R, Guerreiro S, Guédria W, Bera P (2018) An experimental evaluation of the generalizing capabilities of process discovery techniques and black-box sequence models. Enterprise, business-process and information systems modeling. Springer, Heidelberg, pp 165–180
Tax N, Teinemaa I, van Zelst SJ (2020) An interdisciplinary comparison of sequence modeling methods for next-element prediction. Softw Syst Model 19:1345–1365
Taymouri F, La Rosa M, Erfani S, Dasht Bozorgi Z, Verenich I (2020) Predictive business process monitoring via generative adversarial nets: The case of next event prediction. Business process management. Springer, Heidelberg, pp 237–256
van der Aalst WMP (2016) Process mining: data science in action, 2nd edn. Springer, Heidelberg
van der Aalst WMP (2022) Process mining: a 360 degree overview. Process mining handbook. Springer, Heidelberg, pp 3–34
van der Aalst WMP, Schonenberg MH, Song M (2011) Time prediction based on process mining. Inf Syst 36:450–475
van Dongen BF, Carmona J, Chatain T (2016) A unified approach for measuring precision and generalization based on anti-alignments. Business process management. Springer, Heidelberg, pp 39–56
van der Aalst W (2020) On the pareto principle in process mining, task mining, and robotic process automation. In: International conference on data science, technology and applications, SciTePress, pp 5–12
Venkateswaran P, Muthusamy V, Isahagian V, Venkatasubramanian N (2021) Robust and generalizable predictive models for business processes. Business process management. Springer, Heidelberg, pp 105–122
Weinzierl S, Zilker S, Dunzer S, Matzner M (2024) Machine learning in business process management: a systematic literature review. Exp Syst Appl 253(124):181
Weytjens H, De Weerdt Jochen (2021) Creating unbiased public benchmark datasets with data leakage prevention for predictive process monitoring. BPM workshops. Springer, Heidelberg, pp 18–29
Zhou K, Liu Z, Qiao Y, Xiang T, Loy CC (2023) Domain generalization: a survey. IEEE Transact Pattern Anal Mach Intell 45:4396–4415