




Evaluating Predictive Business Process Monitoring Approaches on Small Event Logs

Martin Käppel¹(✉), Stefan Jablonski¹, and Stefan Schönig²

¹ Institute for Computer Science, University of Bayreuth, Bayreuth, Germany

{martin.kaepfel, stefan.jablonski}@uni-bayreuth.de

² University of Regensburg, Regensburg, Germany

stefan.schoenig@ur.de

Abstract. Predictive business process monitoring is concerned with the prediction how a running process instance will unfold up to its completion at runtime. Most of the proposed approaches rely on a wide number of machine learning techniques. In the last years numerous studies revealed that these methods can be successfully applied for different prediction targets. However, these techniques require a qualitatively and quantitatively sufficient dataset. Unfortunately, there are many situations in business process management where only a quantitatively insufficient dataset is available. The problem of insufficient data in the context of BPM is still neglected. Hence, none of the comparative studies investigates the performance of predictive business process monitoring techniques in environments with small datasets. In this paper an evaluation framework for comparing existing approaches with regard to their suitability for small datasets is developed and exemplarily applied to state-of-the-art approaches in next activity prediction.

Keywords: Process mining · Predictive business process monitoring · Small sample learning · Process prediction

1 Introduction

Predictive business process monitoring aims at predicting how a running process instance will unfold up to its completion at runtime based on its current state of execution. This can help to identify problems before the process instance runs in and enables to take adequate preventive measures to avoid them. One can distinguish several prediction targets, e.g., performance predictions such as the remaining execution time [15], business rule violations [11, 12], predictions regarding the outcome of a process instance [3, 21], and predictions of the next event [6] including further information as when it / they will occur and which resource(s) is/are responsible for it [2, 17]. The majority of the proposed approaches rely on a wide number of different machine learning (ML) techniques to perform these predictions.

© Springer Nature Switzerland AG 2021

A. C. R. Paiva et al. (Eds.): QUATIC 2021, CCIS 1439, pp. 167–182, 2021.

https://doi.org/10.1007/978-3-030-85347-1_13

In the last years numerous comparative studies, reviews, and benchmarks of predictive business monitoring approaches have been published [4, 9, 14, 21, 23]. These studies reveal that ML techniques can be successfully applied for all the mentioned prediction tasks. However, all ML techniques are faced with the fundamental requirement of a qualitatively and quantitatively sufficient dataset. In business process management (BPM) we can have an insufficient dataset since a process (*i*) is executed very seldom or has not been executed often yet, (*ii*) its instances are long running, (*iii*) legal regulations like the General Data Protection Regulation lead to significantly less data, or (*iv*) there are fundamental changes in the intended process execution so that some historic data are not usable anymore. Especially small and medium sized companies frequently cannot fulfill this fundamental requirements of ML since not enough data is recorded. On the other hand, small data can also be a desired objective because of limited computational power or real-time feedback. The latter is, for example, characteristic for Stream Process Mining or Concept Drift.

The problem of insufficient data in the context of BPM is still neglected [8]. Hence, none of the comparative studies or benchmarks investigates the performance of predictive business process monitoring techniques on small data, i.e., small event logs. Hence, the contribution of the paper is two-fold: (*i*) we introduce an evaluation framework for comparing existing approaches w.r.t. their suitability for small event logs, and (*ii*) analyse the suitability of existing state-of-the-art approaches in predictive business process monitoring on small event logs. This analysis is also a step towards answering the question of whether there is a lower bound for a minimum of required data for predictive business process monitoring and, if so, in which range this lower bound is located. Our results show that in many cases the algorithms allow a significant reduction of training data and, hence, training times and computational effort can be significantly reduced.

The remainder of the paper is structured as follows: In Sect. 2 we recall basic terminology and give a short introduction to the area of Small Sample Learning. Section 3 highlights the difference between this comparative study and other surveys. In Sect. 4 we describe our evaluation framework and how it can be tailored to the different areas of BPM. In Sect. 5 we use this framework for comparing selected state-of-the-art approaches for predicting the next activity w.r.t. their suitability for small event logs. Finally, Sect. 6 outlines future work.

2 Background

2.1 Process Mining

The input of process mining techniques is a (*process*) *event log*, i.e., a set of traces of a business process (model). A *trace* (also called *case*) is a temporally ordered sequence of events that are related to the same process instance. An *event* is related to an activity (i.e., a step in a business process) and is characterized by various *event attributes* with at least a case id (\mathcal{C}), the name of the corresponding activity (\mathcal{A}), and the timestamp of occurrence (\mathcal{T}). Optionally, an event contains

Table 1. Sample process event log

| Case ID | Event ID | Activity | Timestamp | Resource | Amount | Key |
|---------|----------|----------|---------------------|----------|--------|-------|
| Case1 | e_{11} | A | 2020-10-09T14:50:17 | MF | | SD-1 |
| Case1 | e_{12} | T | 2020-10-09T14:51:01 | SL | 100 | HG-4 |
| Case1 | e_{13} | W | 2020-11-09T12:54:39 | KH | | HZ-2 |
| Case2 | e_{21} | A | 2019-04-03T08:55:38 | MF | | SD-2 |
| Case2 | e_{22} | T | 2019-04-03T08:55:53 | SL | 340 | HK-7 |
| Case2 | e_{23} | C | 2019-05-19T09:00:28 | KH | | SGH-3 |
| Case3 | e_{31} | A | 2019-11-06T10:47:35 | MK | | SD-3 |
| Case3 | e_{32} | T | 2019-11-06T10:48:53 | PE | 235 | UG-2 |
| Case3 | e_{33} | C | 2019-11-25T08:18:07 | SJ | | KL-6 |
| Case4 | e_{41} | A | 2019-04-05T08:59:38 | MF | | SD-5 |
| Case4 | e_{42} | T | 2019-04-05T09:55:52 | SL | 140 | HK-2 |

further event attributes, such as the resources or systems involved in executing the activity (\mathcal{L}) or further data payload (\mathcal{D}_i). Often additional event attributes (e.g., the role of a process participant) are derived by process mining approaches and added to the events.

Let us consider the sample event log shown in Table 1 that provides the following event attributes: a *case identifier*, the name of the executed *activity*, the *timestamp* of execution, the involved *resource*, and two further information (*amount*, *key*) in form of data payload.

Definition 1. Let \mathcal{E} be the universe of events, \mathcal{P} the set of event attributes, and ε the empty element. For each event attribute $p \in \mathcal{P}$, we define a function $\pi_p : \mathcal{E} \rightarrow \text{dom}(\mathcal{P}) \cup \{\varepsilon\}$ that assigns a value of the domain of p to an event. However, ε can only be assigned to optional event attributes.

For example, for event e_{13} , holds $\pi_{\mathcal{A}}(e_{13}) = \text{"W"}$, $\pi_{\mathcal{C}}(e_{13}) = \text{"Case1"}$, $\pi_{\mathcal{L}}(e_{13}) = \text{"KH"}$, $\pi_{\mathcal{T}}(e_{13}) = \text{"2020-11-09T12:54:39"}$, $\pi_{\mathcal{D}_{\text{Amount}}}(e_{13}) = \varepsilon$, and $\pi_{\mathcal{D}_{\text{Key}}}(e_{13}) = \text{"HZ-2"}$.

Definition 2. Let S be the universe of traces. A **trace** $\sigma \in S$ is a finite non-empty sequence of events $\sigma = \langle e_1, \dots, e_n \rangle$ such that for $1 \leq i < j \leq n : e_i, e_j \in \mathcal{E} \wedge \pi_{\mathcal{C}}(e_i) = \pi_{\mathcal{C}}(e_j) \wedge \pi_{\mathcal{T}}(e_i) \leq \pi_{\mathcal{T}}(e_j)$, where $|\sigma| = n$ denotes the length of σ and $\sigma(i)$ refers to the i -th element in σ .

This definition states that each event is unique, time within a trace is increasing, and all events with the same case identifier refer to the same process instance.

If a process instance has finished, i.e., no additional events related to this instance are executed in the future, the trace is completed.

Definition 3. A trace $\sigma \in S$ is called **completed** if there is no $e' \in \mathcal{E}$ such that $\pi_{\mathcal{C}}(e') = \pi_{\mathcal{C}}(e)$ with $e' \notin \sigma$ and $e \in \sigma$. An **event log** L is a set $L = \{\sigma_1, \dots, \sigma_l\}$ of completed traces.

As an example, the event log shown in Table 1 consists of four traces, related to the process instances Case1, Case2, Case3, and Case4. We consider traces to be equivalent with respect to one or more process perspectives (e.g., control flow) by introducing the concept of *trace variants*, which defines an equivalence relation on an event log:

Definition 4. Let L be an event log, $\sigma_1, \sigma_2 \in L$ traces, and $\mathfrak{P} \subseteq \mathcal{P}$ a set of event attributes. We write $\sigma_1 \sim_{\mathfrak{P}} \sigma_2$, if σ_1 and σ_2 are equivalent with regard to \mathfrak{P} , i.e., for all $p \in \mathfrak{P}$ there is $\pi_p(\sigma_1(i)) = \pi_p(\sigma_2(i))$ for all $1 \leq i \leq \max\{|\sigma_1|, |\sigma_2|\}$ with $\pi_p(\sigma(i)) = \varepsilon$ if $i > |\sigma|$. The set $[\sigma]_{\sim_{\mathfrak{P}}} := \{\sigma' \in L \mid \sigma' \sim_{\mathfrak{P}} \sigma\}$ is called a **trace variant**.

It is obvious that $\sim_{\mathfrak{P}}$ is an equivalence relation. Applying this relation to an event log provides in dependency of \mathfrak{P} a more abstract or a less fine-grained view on the event log, since we can disregard one or more process perspectives (to be exact event attributes). For example applying the relation with $\mathfrak{P} = \{\mathcal{A}\}$ on the event log in Table 1 results in three trace variants: the trace with id Case1 represents the trace variant $\langle A, T, W \rangle$; a second trace variant $\langle A, T, C \rangle$ is represented by the two remaining traces, since they are identical with regard to the controlflow; a third trace variant $\langle A, T \rangle$ is represented by trace with id Case4. If we additionally consider the organizational perspective, i.e., $\mathfrak{P} = \{\mathcal{A}, \mathcal{L}\}$ four trace variants evolve, since the traces with case id Case2, Case3, and Case4 differ with respect to the involved resources: $\langle (A, MF), (T, SL), (W, KH) \rangle$, $\langle (A, MF), (T, SL), (C, KH) \rangle$, $\langle (A, MK), (T, PE), (C, SJ) \rangle$, and $\langle (A, MF), (T, SL) \rangle$. On the other hand, if we only consider the organizational perspective, the traces with case id Case1 and Case2 represent the same trace variant $\langle (MF, SL, KH) \rangle$. Usually the more event attributes are considered, the more trace variants occur since it is highly probable that they differ in one of the perspectives. We can consider the frequency distribution of the trace variants within the event log.

Definition 5. Let L be an event log, Ω the set of trace variants with regard to $\sim_{\mathfrak{P}}$ on L , and $X : \Omega \rightarrow \mathbb{R}$ a discrete random variable that represents the trace variants. Then the probability for the occurrence of a trace variant is given by:

$$P(X = [\sigma]_{\sim_{\mathfrak{P}}}) = \frac{|[\sigma]_{\sim_{\mathfrak{P}}}|}{|L|}.$$

We call the frequency distribution of X the **distribution of the trace variants**.

For the example from Table 1, we obtain the following frequency distribution: If only the property \mathcal{A} is considered, a frequency distribution of 0.33 to 0.67 follows. If additionally property \mathcal{L} is regarded, three trace variants with equal probability of 0.33 result.

2.2 Small Sample Learning

In recent years a new and promising area in artificial intelligence research called *Small Sample Learning* (SSL) has emerged [18]. SSL deals with ML on quantitatively inadequate datasets. This also encompasses partial insufficient datasets like

imbalanced datasets, where for some classes are significantly more examples than for other classes. Although, SSL has its origins in the field of computer vision, meanwhile many SSL techniques are applied in various application areas. The current SSL research is divided into two main branches: *concept learning* and *experience learning* [18]. Experience learning attempts to solve a SSL problem by applying conventional ML techniques, by either transforming the problem into a classical ML problem through increasing the amount of data or reducing the preliminaries of the required ML algorithms. In contrast, concept learning aims to detect new concepts from only a small number of examples. Within the two branches numerous methods can be identified. Although there are numerous situations with qualitatively insufficient data in the context of BPM, the application of SSL methods is still neglected in this area. This might be caused by the fact that most common SSL techniques are strongly tailored to computer vision or NLP problems and must be adapted to BPM first in order to become applicable [8].

3 Related Work

This work relates to the stream of research in predictive business process monitoring and touches the area of SSL. Since, SSL methods are barely used in BPM so far, related work mainly focuses on comparative studies of existing business process monitoring approaches and the approaches themselves. The problem of quantitatively insufficient data in BPM was systematically addressed the first time in [8], where the authors propose the idea of leveraging SSL methods for this issue. They describe their concept by the example of predictive business process monitoring, suggests SSL methods that seems promising for BPM, and describe an idea of how the effectiveness of such methods can be proven. A survey of existing SSL methods outside of BPM is presented in [18].

The problem of insufficient training data in context of predictive business process monitoring in case of next event prediction was addressed in [20] where the authors proposed the use of Generative Adversarial Networks (GANs) for solving this issue. This network architecture outperforms other existing deep learning approaches w.r.t. accuracy and earliness of prediction. However, the authors use conventional, i.e., not small, event logs for training and evaluation. Hence, it is unclear whether it only improves results on conventional event logs or if it works for small event logs, too.

The necessity of evaluation frameworks for comparing the performance of different algorithms in BPM, respectively process mining, is not new, since the disparity of event logs, experimental setups, and different assumptions makes it often difficult to make fair comparisons [12, 14]. In [16] the authors motivate the need of an evaluation framework for process mining approaches and propose a framework for comparing model discovery algorithms. In the subfield of predictive business process monitoring there is a plenty set of different comparative studies depending on the different prediction tasks [4, 9, 14, 21, 23]. However, all of them depend on large event logs and do not consider environments with a



Fig. 1. Conception of the evaluation framework

small amount of data. Outcome-oriented techniques are reviewed and compared in [21], in [9] with special focus on deep learning techniques. In [23] the authors give a survey and benchmark of remaining time predictions methods. In [14] the authors focus on deep learning techniques for next activity prediction, activity suffix prediction, next timestamp prediction, and remaining time prediction by evaluating approaches with publicly available source code on 12 real-life event logs from various domains.

The above discussed comparative studies observed that deep learning approaches for next activity prediction outperform classical predictions techniques, which use an explicit model representation such as Hidden Markov Models [10], probabilistic finite automata or state-transition [1, 22]. These deep learning approaches are based on different types of neural networks. Most of them use Long-Short-Term-Memory (LSTM) Neural Networks [2, 6, 17, 19], Gated Recurrent Units (GRUs) as a variant of LSTMs [7], or Convolutional Neural Networks (CNN) [5, 13]. The approaches use different encoding techniques for sequences and events and consider different input data for making predictions. An overview about existing deep learning approaches including a detailed description of the underlying architectures is given in [14].

4 Evaluation Framework

In this section we describe the structure of our evaluation framework. We identify different challenges that must be considered by the evaluation framework.

4.1 The Issue of Small Event Logs

The evaluation framework (cf. Fig. 1) gets two inputs: the approaches to be compared and small event logs. However, providing small event logs is a crucial challenge. This is due to the missing definition of “small event log”. This question is strongly related to the question what is “big data”. Also, this is still an open question in research. Due to inconsistent and sometimes contradicting definitions, that often include time dependency (i.e., define as big data what is today the largest available amount of data), domain dependency or circular reasoning (e.g., big data is the opposite to small data) this question cannot be conclusively clarified. We bypass this technical and conceptual problem by reducing event logs with various reduction factors and thereby generate small event logs of different sizes. The use of different reduction factors enables us to fully cover the broad range of “smallness”, which ranges from zero or a single case up to, for example, several 1000 cases. Hence, we are independent of concrete

definitions. In a strict sense, the generated small event logs are rather “relatively small” event logs, than “small” event logs. The exact procedure of generating small event logs is described in detail in Sect. 4.3.

In addition to bypassing this definition problem, generating small event logs by reducing conventional event logs has another advantage: Since we can fall back to the non-reduced event log (so called *reference log*), it is possible to compare results of an analysis achieved on a small log with the results obtained on the reference log¹. This comparison allows us to make quantitative statements about the impact of reducing an event log. Such a comparison is necessary to determine whether any potential loss of quality that goes in hand with the data reduction, can be better compensated by one method or by another. Hence, we measure how an approach performs depending on the reduction factor. It is likely that the achieved results also depend on the domain and structure of the considered event log (e.g., number of event per case).

4.2 Preserving Comparability

When we talk about comparability in our approach, we mean to compare the prediction quality of predictive business process monitoring approaches measured on the reference log and the generated small event logs. For preserving comparability, it is essential to evaluate all trained models with the same test data. Hence, we first divide the event log into training and test data and afterwards we reduce only the training data and not the whole reference log. The selection of the test data is discussed in Sect. 4.4. However, excluding the test data from reduction has a far-reaching consequence: Usually ML techniques, which require a split into training and test data, use training-test ratios like 80:20 or 70:30. Since we keep the test data and only reduce the training data the ratio between train and test data shifts more and more towards test data with increasing reduction factor. Hence, the less training data are used, the relatively more tests the model must pass afterwards. This seems to be unusual, however it ensures comparability and guarantees that the quality of the trained model and, as a result, the performance of the method used for training, are not overestimated but rather underestimated. In consequence, also the frequently used cross validation that enables the use of all available data for training as well as for testing are not applicable anymore, since through the reduction step reference log and small event logs differ. It should be noted that dispense on comparability would avoid this shift problem but would be accompanied with interpretation problems due to meaningless splits in training and test data. Suppose that a reduced event log would only contain 10 traces left and we would split this event log into training and test data using a ratio of 80:20. Then the metrics used for evaluation would hardly be meaningful. For example the accuracy metric could only attain three different values: 0%, 50%, or 100%.

¹ At this point we implicitly assume that the event logs currently used in research can be considered as quantitatively sufficient.

It is obvious that comparability and keeping the ratio between training and test data are diametrically opposed to each other. Since, the comparability between the reduced event logs is essential for our aims and cannot be neglected, we accept the shift of the training-test ratio in our evaluation framework.

4.3 Reducing Event Logs

The amount of training data is reduced by removing as many process instances from the training data such that a given reduction factor is reached. We select process instances for removal either randomly or along the time dimension. Removing instances randomly means selecting process instances randomly. When reducing along the time dimension, we order the process instances ascending by its first event timestamp and then remove the first process instances according to the reduction factor. Note that the way how the process instances to be removed are selected leads to different interpretations: In case of randomly removing process instances, we simulate that a process is executed very seldom or due to legal regulations there are only a few records available for analysis. Reason for this interpretation is that the underlying time window of the event log (spanned by the earliest and latest timestamp of an event) stays nearly unchanged. However, in case of removing process instances along the time dimension, the time window is shortened. Hence, this reduction method reflects a scenario where a process has not been executed often yet or due to fundamental changes in the intended process execution some historic data (up to a specific time) are not usable anymore. Hence, these two reduction methods are sufficient to simulate all in the introduction mentioned reasons for quantitatively insufficient event logs. Nevertheless, we implement some alternative selection methods, like removing the most recent data or removing only specific trace variants, as defined in Definition 4.

However, the reduction of the training data bears two further issues: *(i)* the possible loss of activities and resources, and *(ii)* statistical bias.

Loss of Activities and Resources. Since we generate small event logs out of large event logs, there is a risk that activities or resources get completely lost or are finally only represented in the test data. In case of getting completely lost, the trained model would not be able to handle these activities or resources, if they occur later in productive use, since they are not encoded and therefore are unknown to the model. Therefore, we extract and buffer all occurring activities and resources from the reference log before splitting into training and test data and before generating the small event logs. Hence, these activities and resources can be considered even if no training sample reflects them. However, this also implies that process instances in the test data that contain activities or resources that are not represented in the training data cannot be predicted well.

Statistical Bias. The reduction of the training data may be accompanied by statistical bias in the probability distribution of the trace variants (cf. Definition 5). Since most of the ML techniques are statistical methods, it affects the model quality and must therefore be considered adequately. Suppose that a trace

variant is represented by exactly one representative in a training dataset, which consists of 100 traces. Then this trace variant has an empirical probability of 1%. If we reduce the training data by 50% and the representative of the considered trace variant is not removed, then the empirical probability of this trace variant increases to 2%. At the same time other trace variants are either completely eliminated or their empirical probability decreases. In case of an even stronger reduction and under the assumption that the considered trace variant is not removed, the increase of empirical probability could be much stronger. Hence, the probability distributions of the trace variants in the considered event logs can differ significantly or reflects no longer their occurrence frequency in reality.

However, the problem becomes less relevant the more process perspectives are considered, since process perspectives foster the singularity of traces and the number of trace variants represented in the event log tends to the number of traces in the event log. In the case that each trace variant occurs only once in the event log, removing a trace directly leads to a loss of a trace variant. This observation prohibits to reduce the event log along the distribution of its trace variants. However, considering this issue only from the perspective of trace variants is not sufficient. Because from the ML perspective there may be a significantly lower statistical bias since this perspective also takes the similarity between the trace variants into account (for example two trace variants only differ in a single event). Hence, often removing a trace variant is sufficiently compensated by another very similar trace variant that is still included in the event log. However, it is difficult to determine this compensatory effect, because it strongly depends on the considered ML technique and therefore cannot be adequately considered in the evaluation framework.

The effects discussed above are affected by the chosen reduction method. In case of a randomly reduction, the distribution of trace variants can be extremely distorted. The reduction along the time dimension alleviates this issue, since the distribution of the trace variants in a sufficiently large time window should be more similar to the distribution of the entire event log than the distribution in a randomly selected subset of the event log. This assumption also holds for the compensatory similarity effect. Hence, it is expected that via reduction along the time dimension the statistical bias can be reduced. However, it is clear, that it also depends on the particular event logs and in case of strong reduction the statistical bias cannot be longer compensated.

4.4 Splitting into Training and Test Data

Still, the question remains how the event log should be split into training and test data. We use the same procedure for selecting test data as for reducing the training data, i.e., the test data is either selected randomly or along the time dimension. In the latter case, the newest process instances are used for testing and the oldest for training. The chosen split procedure may affect the achieved results, since training and test data may overlap in time by splitting the event log randomly. This could be problematic if the underlying process evolves

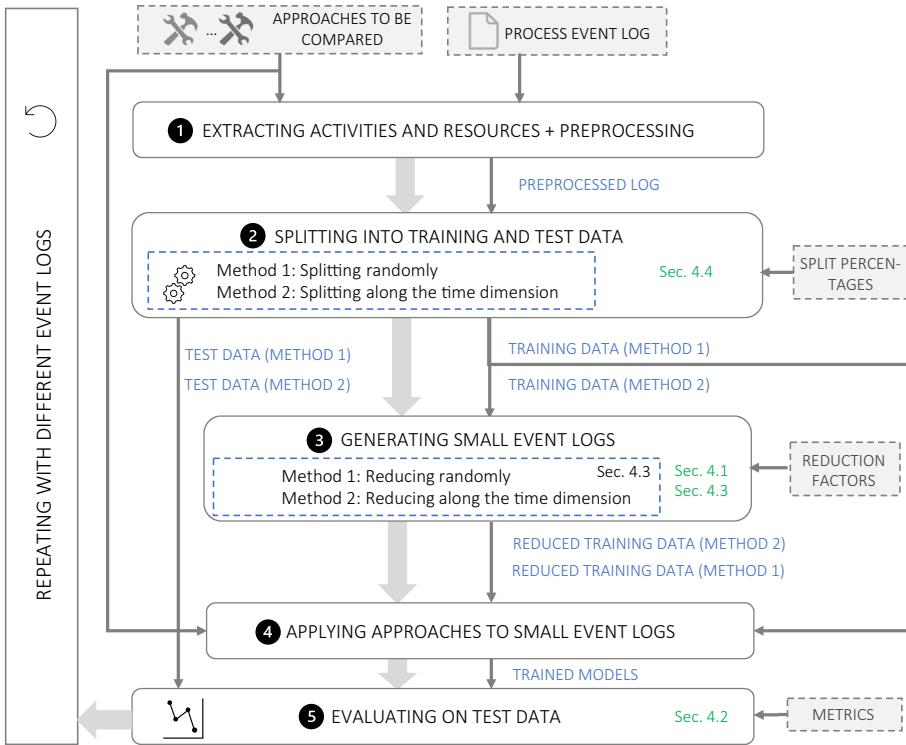


Fig. 2. Architecture of the evaluation framework

during time and, hence, different process variants are mixed up in the event log. However, splitting along the time dimension does not prevent this issue, since the splitting can lead to a separation of a specific process variant (i.e., the test data represents a process variant that does not occur in the training data). Nevertheless, splitting along the time dimension appears closer to reality, since knowledge of the past is used to predict the future. Furthermore, the splitting along the time dimension also provides a better reproducibility.

4.5 Architecture of the Evaluation Framework

In summary, the proposed evaluation framework depicted in Fig. 2 comprises five successive steps. In a first step, an event log is preprocessed depending on the selected approaches to be compared and all activities and resources of the event log are extracted and registered. Further preprocessing encompasses, for example, the removal of traces with events that have missing values. Afterwards (cf. Step 2) a preprocessed event log is split into training and test data according to one or more split ratios. The resulting training data is then reduced in the third step according to various reduction factors to generate a set of small event

Table 2. Statistic of the used event logs. Time related measures are shown in days.

| Event Log | BPIC12 | Helpdesk | BPIC13 | BPIC15.2 | BPIC15.5 |
|----------------------|--------|----------|---------|----------|----------|
| Number of cases | 9559 | 4580 | 1487 | 832 | 1156 |
| Number of activities | 36 | 14 | 7 | 410 | 389 |
| Number of roles | 8 | 4 | 4 | 11 | 10 |
| Number of events | 140863 | 21348 | 6660 | 44354 | 59083 |
| Maximal case length | 163 | 15 | 35 | 132 | 154 |
| Minimal case length | 3 | 2 | 1 | 1 | 5 |
| Average case length | 14.74 | 4.66 | 4.49 | 53.31 | 51.11 |
| Maximal duration | 76.90 | 59.99 | 2254.85 | 1325.96 | 1343.96 |

logs. The framework selects the traces to be removed with the same methods as used for selecting test and training data. The approaches to be compared are trained with the training data resulting from the splitting step as well as the reduced training data of the reduction step (cf. Step 4). In order to measure the performance of the considered approaches, the trained models are evaluated on the corresponding test data (cf. Step 5). For measuring the performance one or more suitable metrics are used. The selection of the metrics primarily depends on the types of approaches. After the completion of the evaluation step, all steps are repeated with further event logs to get representative results.

In general, the framework is extensible by adding alternative methods for splitting and reducing the processed data. In Fig. 2, all parts of the framework that can be adapted and configured are marked with dashed lines. Fields with a grey background represent input parameters that have to be set to configure the framework. The remaining dashed fields can be extended to add more functionality to the framework.

We conclude this section by a brief discussion whether the evaluation framework is tailored to specific process mining methods. Since the framework offers flexible preprocessing and the splitting into test and training data can be skipped it is also possible to evaluate unsupervised ML techniques. For ML techniques, which require a split into training, validation, and test data some smaller adaptations would be necessary. Since the evaluation framework does not implement any approach specific particularities, the framework can be considered as approach-agnostic.

5 Evaluation of Existing Approaches on Small Event Logs

5.1 Dataset Description and Experimental Setup

The first three steps of the framework that are responsible for generating small event logs are implemented as a Java application. Step 4 is covered by the modified implementations of the considered approaches. Modification becomes necessary, since approaches must deal with the generated small event logs and the test

data generated in the previous steps. Hence, parts in the implementations of the considered approaches that are responsible for splitting into training and test data or extracting activities and resources must be modified. Also, the approach specific evaluation components must be replaced by the evaluation component of the framework (Step 5) to ensure a consistent evaluation procedure.

We evaluate our framework with a small comparative study of selected state-of-the-art approaches for next activity and role prediction. We select approaches [2] and [13] since they represent the most frequently used deep learning architectures (LSTM and CNN respectively) for next activity prediction, provides publicly available source code, and achieve good results in various comparative studies. We modified the approaches in the above described way and additionally changed the implementations to run with Python 3.7 and to support training on GPU. The experiments are run on a system equipped with a Windows 10 operating system, an Intel Core i9-9900K CPU 3.60GHz, 64GB RAM, and a NVIDIA Quadro RTX 4000 having 6GB of memory.

We perform our experiment using 5 real-life event logs from different domains with diverse characteristics (cf. Table 2) extracted from the *4TU Center for Research Data*². For our experiment, we preprocess the event logs in the same way as it is done in the considered approaches. Due to the missing resource event attribute in some traces we removed 3528 of the 13087 traces in the BPIC12 log.

5.2 Tailoring the Evaluation Framework to Predictive Monitoring

We tailor the evaluation framework for evaluating predictive business process monitoring approaches in the following way. We use a training-test ratio of 70:30. The test data is selected by applying the two methods from Sec. 4.4: (i) splitting randomly, and (ii) splitting along the time dimension. Training data are reduced according to the following reduction factors: 0.2, 0.4, 0.6, 0.8, 0.9, 0.95, and 0.99. Hereby, the traces are either removed randomly or along the time dimension. For evaluation, we use the following common metrics that can be derived from the confusion matrix³: (i) *Recall* (also called sensitivity) defined as $R = TP / (TP + FN)$, (ii) *Precision* $P = TP / (TP + FP)$, (iii) *F-Measure* defined as harmonic mean $F = 2RP / (R + P)$ of precision and recall, and (iv) *Accuracy* defined as $A = (TP + TN) / (TP + FN + FP + TN)$. These metrics coincide with those from other comparative studies.

5.3 Results and Discussion

Due to the large number of trained models and measures and since the achieved results are comparable for all considered approaches we report in this paper only an exemplarily excerpt of the detailed measures (cf. Table 3 and 4) and

² <https://data.4tu.nl>.

³ i.e., it can be calculated from true positives (TP), true negatives (TN), false positives (FP), and true positive (TP).

Table 3. Next activity prediction results (accuracy measures in %) for approach [2] (architecture “shared categorical”)

| | Reduced along time dimension | | | | | | | | Reduced randomly | | | | | | | |
|-----------|----------------------------------|-------------|-------------|-------------|-------------|-------------|------|------|----------------------------------|------|------|------|------|------|------|------|
| | <i>Applied reduction factors</i> | | | | | | | | <i>Applied reduction factors</i> | | | | | | | |
| Event Log | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 | 0.95 | 0.99 | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 | 0.95 | 0.99 |
| BPIC_15_2 | 0.10 | 0.09 | 0.09 | 0.08 | 0.11 | 0.21 | 0.17 | 0.07 | 0.11 | 0.09 | 0.09 | 0.06 | 0.08 | 0.05 | 0.06 | 0.04 |
| BPIC_15_5 | 0.16 | 0.20 | 0.24 | 0.28 | 0.28 | 0.26 | 0.22 | 0.22 | 0.16 | 0.14 | 0.13 | 0.14 | 0.09 | 0.09 | 0.08 | 0.03 |
| BPIC_13 | 0.54 | 0.64 | 0.58 | 0.61 | 0.60 | 0.57 | 0.51 | 0.39 | 0.56 | 0.52 | 0.53 | 0.51 | 0.53 | 0.49 | 0.50 | 0.44 |
| Helpdesk | 0.74 | 0.74 | 0.79 | 0.79 | 0.78 | 0.79 | 0.78 | 0.70 | 0.74 | 0.74 | 0.73 | 0.74 | 0.73 | 0.73 | 0.72 | 0.72 |
| BPIC_12 | 0.85 | 0.86 | 0.85 | 0.84 | 0.85 | 0.84 | 0.84 | 0.79 | 0.85 | 0.85 | 0.85 | 0.85 | 0.84 | 0.84 | 0.84 | 0.74 |

Table 4. Next role prediction results (accuracy measures in %) for approach [2] (architecture “shared categorical”)

| | Reduced along time dimension | | | | | | | | Reduced randomly | | | | | | | |
|-----------|----------------------------------|-------------|-------------|-------------|-------------|-------------|-------------|------|----------------------------------|-------------|-------------|-------------|-------------|-------------|------|------|
| | <i>Applied reduction factors</i> | | | | | | | | <i>Applied reduction factors</i> | | | | | | | |
| Event Log | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 | 0.95 | 0.99 | 0.0 | 0.2 | 0.4 | 0.6 | 0.8 | 0.9 | 0.95 | 0.99 |
| BPIC_15_2 | 0.85 | 0.87 | 0.86 | 0.86 | 0.86 | 0.82 | 0.84 | 0.55 | 0.87 | 0.78 | 0.75 | 0.85 | 0.85 | 0.82 | 0.68 | 0.55 |
| BPIC_15_5 | 0.90 | 0.91 | 0.91 | 0.92 | 0.93 | 0.88 | 0.91 | 0.89 | 0.90 | 0.91 | 0.90 | 0.91 | 0.87 | 0.87 | 0.87 | 0.52 |
| BPIC_13 | 0.96 | 0.96 | 0.92 | 0.96 | 0.95 | 0.93 | 0.83 | 0.63 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.93 | 0.91 | 0.95 |
| Helpdesk | 0.95 | 0.95 | 0.94 | 0.95 | 0.94 | 0.95 | 0.94 | 0.84 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.94 | 0.93 |
| BPIC_12 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.94 | 0.97 | 0.97 | 0.97 | 0.97 | 0.97 | 0.96 | 0.96 | 0.93 |

limit ourselves to the discussion of the overall results. All further measures are provided in the repository of our implementation⁴.

The most surprising observation is that only a very strong reduction factor of 95% or 99% significantly worsens the performance measured by different metrics. This applies for event logs where good results are achieved as well as for event logs that show poor results. A further surprising fact is that in case of a reduction along the time dimension the best results (highlighted in red) are not achieved by the reference values (reduction factor 0.0), i.e., often a reduced amount of training data achieves better results. However, this observation does not hold for a randomly performed reduction. This reduction method either achieves similar or slightly worse results. Hence, this observation supports the hypothesis of Sect. 4.3 that reduction along the time dimension leads to a more representative dataset. Since, we remove those traces that have the earliest first timestamp this behaviour might also be an indicator that there is some process evolution within the logs. We also observe that the most complex event logs BPIC15.2 and BPIC15.5, which contain a relatively large number of activities for a comparatively small number of traces shows poor results. The reason for this behaviour seems to be the high number of activities per number of traces (cf. Table 2). This interpretation is supported by the fact that the prediction of

⁴ <https://github.com/mkaep/SSL-Evaluation-Framework>.

the next roles performs significantly better. Hence, for learning such complex logs we would need significantly more training data.

In a deeper analysis of the results we analysed whether the prediction quality between the activities differ. This analysis reveals that all trained models (also the reference models) are good in predicting frequent activities but perform poor in predicting rare activities. As a result, frequent trace variants (i.e., standard cases) are predicted very well, while rare trace variants are barely predicted correct, since the prediction model treat them as standard cases. This is somehow natural, since ML methods try to generalize the data in a simple way by neglecting rare activities.

Hence, we can draw the following conclusions: for learning to predict frequent trace variants of less complex logs even a significantly reduced amount of data is sufficient. For learning rare trace variants, however, it is necessary to increase the amount of data, especially by better representing rare trace variants. For complex logs, like the BPIC15 logs, the currently available amount of data is not sufficient, to achieve acceptable results. Hence, our results reinforce the need for SSL methods in the area of predictive business process monitoring.

6 Future Work

In this paper, we propose a customizable evaluation framework for investigating predictive business process monitoring approaches w.r.t their suitability for small event logs. Our experiments reveal that training times and computational effort can be significantly reduced without any loss of quality with regard to the common metrics. For further improvement, however, it would be necessary to cope with the problem of rare trace variants. In future work the study should be extended to further approaches, event logs, and should investigate how different types of sequence and event encoding affect the performance. Furthermore, other prediction tasks, like the prediction of suffixes or the remaining time should be investigated. It is also necessary to adopt the framework for use in other subfields of process mining, like process model discovery or conformance checking.

References

1. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. *MIS Q.* **40**(4), 1009–1034 (2016)
2. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate LSTM models of business processes. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) *BPM 2019*. LNCS, vol. 11675, pp. 286–302. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26619-6_19
3. Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M.P.: Supporting risk-informed decisions during business process execution. In: Salinesi, C., Norrie, M.C., Pastor, Ó. (eds.) *CAiSE 2013*. LNCS, vol. 7908, pp. 116–132. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38709-8_8

4. Di Francescomarino, C., Ghidini, C., Maggi, F.M., Milani, F.: Predictive process monitoring methods: which one suits me best? In: Weske, M., Montali, M., Weber, I., vom Brocke, J. (eds.) BPM 2018. LNCS, vol. 11080, pp. 462–479. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-98648-7_27
5. Di Mauro, N., Appice, A., Basile, T.M.A.: Activity prediction of business process instances with inception CNN models. In: Alviano, M., Greco, G., Scarcello, F. (eds.) AI*IA 2019. LNCS (LNAI), vol. 11946, pp. 348–361. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-35166-3_25
6. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. *Decis. Supp. Syst.* **100**, 129–140 (2017)
7. Hinkka, M., Lehto, T., Heljanko, K.: Exploiting event log event attributes in RNN based prediction. In: Welzer, T., et al. (eds.) ADBIS 2019. CCIS, vol. 1064, pp. 405–416. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30278-8_40
8. Käppel, M., Schönig, S., Jablonski, S.: Leveraging small sample learning for business process management. *Inf. Softw. Technol.* **132**, 106472 (2020)
9. Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J.: Machine learning in business process monitoring: a comparison of deep learning and classical approaches used for outcome prediction. *BISE* **63**, 261–271 (2020). <https://doi.org/10.1007/s12599-020-00645-0>
10. Lakshmanan, G.T., Shamsi, D., Doganata, Y.N., Unuvar, M., Khalaf, R.: A Markov prediction model for data-driven semi-structured business processes. *Knowl. Inf. Syst.* **42**(1), 97–126 (2013). <https://doi.org/10.1007/s10115-013-0697-8>
11. Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C.: Predictive monitoring of business processes. In: Jarke, M., et al. (eds.) CAiSE 2014. LNCS, vol. 8484, pp. 457–472. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-07881-6_31
12. Metzger, A., et al.: Comparing and combining predictive business process monitoring techniques. *IEEE Trans. Syst. Man Cybern.* **45**(2), 276–290 (2015)
13. Pasquadibisceglie, V., Appice, A., Castellano, G., Malerba, D.: Using convolutional neural networks for predictive process analytics. In: Proceedings of ICPM 2019 (2019)
14. Rama-Maneiro, E., Vidal, J., Lama, M.: Deep learning for predictive business process monitoring: review and benchmark. *ArXiv* [arXiv:2009.13251](https://arxiv.org/abs/2009.13251) (2020)
15. Rogge-Solti, A., Weske, M.: Prediction of business process durations using non-Markovian stochastic petri nets. *Inf. Syst.* **54**, 1–14 (2015)
16. Rozinat, A., de Medeiros, A.K.A., Günther, C.W., Weijters, A.J.M.M., van der Aalst, W.M.P.: The need for a process mining evaluation framework in research and practice. In: ter Hofstede, A., Benatallah, B., Paik, H.-Y. (eds.) BPM 2007. LNCS, vol. 4928, pp. 84–89. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-78238-4_10
17. Schönig, S., Jasinski, R., Ackermann, L., Jablonski, S.: Deep learning process prediction with discrete and continuous data features. In: Proceedings of ENASE 2018 (2018)
18. Shu, J., Xu, Z., Meng, D.: Small sample learning in big data era. *CoRR* [abs/1808.04572](https://arxiv.org/abs/1808.04572) (2018). [arXiv:1808.04572](https://arxiv.org/abs/1808.04572)
19. Tax, N., Verenich, I., La Rosa, M., Dumas, M.: Predictive business process monitoring with LSTM neural networks. In: Dubois, E., Pohl, K. (eds.) CAiSE 2017. LNCS, vol. 10253, pp. 477–492. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59536-8_30

20. Taymouri, F., Rosa, M.L., Erfani, S., Bozorgi, Z.D., Verenich, I.: Predictive business process monitoring via generative adversarial nets: the case of next event prediction. In: Fahland, D., Ghidini, C., Becker, J., Dumas, M. (eds.) BPM 2020. LNCS, vol. 12168, pp. 237–256. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58666-9_14
21. Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M.: Outcome-oriented predictive process monitoring: review and benchmark. TKDD **13**(2), 1–57 (2019)
22. Unuvar, M., Lakshmanan, G.T., Doganata, Y.N.: Leveraging path information to generate predictions for parallel business processes. Knowl. Inf. Syst. **47**(2), 433–461 (2015). <https://doi.org/10.1007/s10115-015-0842-7>
23. Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I.: Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM TIST **10**(4), 1–34 (2019)