LoGo: Combining Local and Global Techniques for Predictive Business Process Monitoring

Kristof Böhmer and Stefanie Rinderle-Ma

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

Abstract. Predicting process behavior in terms of the next activity to be executed and/or its timestamp can be crucial, e.g., to avoid impeding compliance violations or performance problems. Basically, two prediction techniques are conceivable, i.e., global and local techniques. Global techniques consider all process behavior at once, but might suffer from noise. Local techniques consider a certain subset of the behavior, but might loose the “big picture”. A combination of both techniques is promising to balance out each others drawbacks, but exists so far only in an implicit and unsystematic way. We propose LoGo as a systematic combined approach based on a novel global technique and an extended local one. LoGo is evaluated based on real life execution logs from multiple domains, outperforming nine comparison approaches. Overall, LoGo results in explainable prediction models and high prediction quality.

Keywords: Predictive Process Monitoring, Sequential Rule Mining, Local Prediction, Global Prediction, Explainable Prediction Models

1 Introduction

Predictive process monitoring provides insights into the future of ongoing process executions, i.e., process instances that are currently executed. For this historic process executions are analyzed to, for example, predict the next execution event’s activity and timestamp. With this, resource utilization can be optimized or impending Service-Level Agreement *violations* can be *averted*, cf. [23, 22].

Related work typically applies a fuzzy and implicit mixture of *global* and *local* prediction techniques. *Global* techniques take all historic process behaviour into account at once [23]. On the one side, they exploit significant global behaviour, but on the other side require to handle possibly large noisy data collections. In comparison, *local* prediction techniques focus on a certain subset of the historic process behaviour, for example, those traces that are most similar to a ongoing trace which is predicted upon [5]. Local techniques, on the one side, enable data and noise reduction in early stages of the prediction by taking only a subset of the data into account. But on the other side hinder the identification of significant global behaviour reflecting the “big picture” of the process execution behavior.

Hence, it seems promising to combine local and global techniques as they compensate each others drawbacks. However, the currently applied unsystematic or implicit combination of local and global techniques forces the applied machine learning techniques (e.g., recurrent neural networks, cf. [19]) into covering both (local *and* global) in a *single* prediction model at once, and determining if local *or* global prediction and data should be preferred for each prediction task. Given, such learning techniques are not aware of any logical separation between local and global prediction, they struggle to "learn" and represent this separation during their prediction model training phase; with varying success, cf. [2, 19, 23].

Overall, this might result in complex prediction models, high model training times, reduced flexibility, and diminished prediction results, cf. [19, 5, 23]. Accordingly, we propose an explicit combination of specialized local and global prediction techniques instead of an unsystematic or implicit one. Here, both techniques are shallow ones [2] as they are designed for either local or global prediction and yield explainable prediction models. In detail, we propose a novel global prediction technique which could be applied in a stand-alone fashion. To demonstrate and exploit the outlined advantages of combining global and local techniques, the global technique is complemented with a local one that is extended based on previous work in [5].

Specifically, the combined approach works as follows: first, the global technique is applied to exploit global significant process execution behaviour if possible. If no significant behaviour can be exploited the local technique is applied as a fallback, cf. [5], as it is less affected by noise.

Formally: Let p be an execution trace of process P for which the next activity execution should be predicted. Further let L hold all historic execution traces t of P. The key idea is to mine Sequential Prediction Rules (SPR) from L to identify significant global behaviour in the historic data (global prediction). Behaviour is assumed as significant if the related SPRs have a high confidence. Rules have been used in existing data analysis formalisms and mining approaches [26]. This work extends them into SPRs in order to become capable of predicting future behaviour, i.e., upcoming activities and their execution timestamps. Such SPRs form the prediction model M, enabling to predict the next activity in a majority of prediction scenarios, i.e. in about 60% of all predictions as shown in Sect. 4.

The proposed global prediction technique is not able to yield any predictions based on M if none of the SPRs contained in M match to p's behaviour. In other words, if no significant execution behaviour was identified in L by the global prediction technique which is relevant for p's current execution state. In this case, the proposed combined approach LoGo employs a local prediction technique, i.e., extended based on [5], as a fallback. The work in [5] relies on the similarity between p and $t \in L$ to identify the most relevant behaviour (traces, resp.) to form a probability based prediction model. Accordingly, even if the similarity between p and $t \in L$ is low there will always be some traces which are the "most" similar ones. These most similar traces serve as a basis to predict future behaviour even if a global, e.g., SPR based, technique is unable to do so.

This paper is organized as follows: Prerequisites and the proposed approach are introduced in Section 2. Details on prediction model generation and its application are given in Section 2 and 3. Section 4 covers the evaluation (real life data, multiple comparison approaches) while Section 5 discusses related work. Finally, conclusions, discussions, and future work are outlined in Section 6.

2 Prerequisites and General Approach

The presented approach enables to predict process instance execution behaviour (e.g., in the form of an process execution event) to be observed next, based on a given ongoing process instance trace p. However, this section, for the sake of simplicity, focuses on the prediction of next event activities. Details on temporal behaviour prediction (execution event timestamps, resp.) are given in Sect. 3.

Overall, the prediction is based on a prediction model M which is generated from a bag of historic process execution traces L, i.e., an execution log. The latter is beneficial as L can be *a*) automatically generated by process engines; *b*) represents real behaviour (including noise and ad-hoc changes); and *c*) it is independent of outdated documentation, cf. [18, 5]. Formally, log L is defined as:

Definition 1 (Execution Log). *Let L be a finite bag of* ***`execution traces`*** *$t \in L$. Let further $t := \langle e_1, \cdots, e_n \rangle$ be an ordered list of execution* ***`events`*** *$e_i := (ea, et)$. Hereby, e_i represents the execution of* ***`activity`*** *$e_i.ea$ which was started at* ***`timestamp`*** *$e_i.et \in \mathbb{R}_{>0}$. The* ***`order`*** *of $e_i \in t$ is determined by $e_i.et$.*

Definition 1 enables the identification of global activity orders and related temporal information. Hence, it it is sufficient for next event prediction. Further Def. 1 is generic. Thus it enables the representation of standardized log data formats, such as, the eXtensible Event Stream[1] or custom organization dependent formats (cf. Sect. 4). In the running example provided in Tab. 4, the event e_2 for trace t_2 (i.e., $t_2.e_2$) represents the execution of activity `E` at timestamp `54`.

Table 1. Running example log L as excerpt of the Helpdesk-Log used in Sect. 4

		Event $e_i := (ea, et)$ where ea=activity, et=timestamp					
Process P	Trace t	e_1	e_2	e_3	e_4	e_5	e_6
P_1	t_1	(A,23) →	(E,32) →	(E,37) →	(F,40) →	(C,47) →	(W,53)
P_1	t_2	(A,49) →	(E,54) →	(F,61) →	(F,68) →	(C,69) →	(R,78)
P_1	t_3	(A,40) →	(F,45) →	(E,49) →	(F,51) →	(C,57) →	(W,63)
P_1	t_4	(E,17) →	(F,21) →	(D,22) →	(F,25) →	(C,30) →	(R,38)

Further, similar to [5], the following *auxiliary* functions are applied:

- $\{S\}^0$ returns the element in the set/bag S, given that S is a singleton.
- $C := A \oplus b$ appends b to a copy C of the collection given by A.
- $\langle \cdot \rangle^l$ retains the last element of a list.
- $\langle \cdot \rangle_i$ retains the list item with index $i \in \mathbb{N}_{>0}$.

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

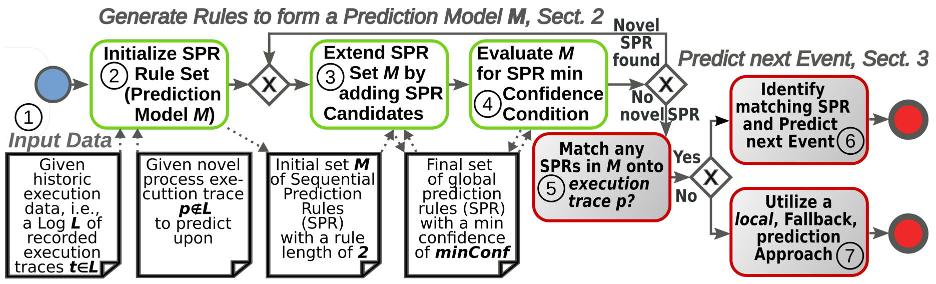
Generate Rules to form a Prediction Model M, Sect. 2
Initialize SPR ② Rule Set (Prediction Model M)
Extend SPR ③ Set M by adding SPR Candidates
Evaluate M for SPR min ④ Confidence Condition
Novel SPR found
No novel SPR
Predict next Event, Sect. 3
Identify matching SPR and Predict next Event ⑥
Utilize a local, Fallback, prediction Approach ⑦
Match any SPRs in M onto ⑤ execution trace p?
Yes
No
①
Input Data
Given historic execution data, i.e., a Log L of recorded execution traces t∈L
Given novel process execution trace p∉L to predict upon
Initial set M of Sequential Prediction Rules (SPR) with a rule length of 2
Final set of global prediction rules (SPR) with a min confidence of minConf

Fig. 1. Proposed SPR based prediction approach – overview

Figure 1 gives an overview on the proposed prediction approach. Its expected *input data* ① comprises historic execution traces $t \in L$ (log, resp.) of a process P to create a prediction model M; and an ongoing execution $p \notin L$ of P for which the next event should be predicted. Hereby, M can be created, even before p's execution starts. Further, the same M can be reused for different traces of P.

Here the *prediction model* M consists of a set of Sequential Prediction Rules (SPR). SPRs expand upon *sequential rules* which have been proposed in sequential rule mining approaches, e.g., [25]. There, a sequential rule s is defined as $s := \langle \mathtt{E}, \mathtt{F}, \mathtt{F} \rangle$ where `E` and `F` refer to process activity executions. The order of the "entries" in s expresses in which order the activities must be observed in a trace t to conclude that s *matches* t. Regarding activity execution orders, existing work names the following patterns that are of interest for this: "`A` *followed by* `B`" means that if `A` is executed eventually `B` must be executed [21]. "`A` *immediately followed by* `B`" requires that as soon `A` has been executed `B` must follow next [21].

This work extends existing sequential rule formalisms, in order to increase their predictive capabilities, into SPRs. In the following a SPR r is defined as

$$r := (pas = \langle pas_1, \cdots, pas_m \rangle, pre, fut) \tag{1}$$

where $pas_1, \ldots, pas_m$, pre, and fut refer to activity executions. The order of these "entries" in r imposes requirements on the expected activity execution order over a corresponding execution trace t. These are, in detail, pas_m *followed by* pre; pre *immediately followed by* fut while for the activities in pas a *followed by* order is expected (similar to sequential rules). Obviously, this constitutes an extension when compared to existing sequential rules which commonly utilize only a single relation pattern. Further, the rule "entries" pas (historic execution behaviour), pre (most recently executed activity), and fut (the activity to be executed next) become assigned a specific semantic meaning. This is another difference to existing sequential rules as they treat each of the "entries" equally.

Overall these extensions give SPRs their prediction capabilities, increases their flexibility and reduce over- *and* underfitting. For example, pas enables to consider global behaviour when creating M or predicting – reducing underfitting. In comparison, overfitting is reduced based on the relaxed *followed by relation* implied on pas (and its "entries") and pre. The latter, combined with the *imme-*

diately followed by relation between *pre* and fut enables to represent in M and subsequently exploit while predicting that, given, *pas* and *pre* was observed for p the to be predicted next (i.e., immediately following) activity is, likely, fut.

Hereby, all SPR entries combined with the expected activity orders provide a foundation to predict p's next activity based on SPRs. For the running example in Table 1, the SPR $r := (\langle \mathtt{E} \rangle, \mathtt{F}, \mathtt{F})$ matches to t_3 while a similar sequential rule $s := (\langle \mathtt{E}, \mathtt{F}, \mathtt{F} \rangle)$ matches to t_2 and t_3; illustrating that the made extensions increase the focus of SPRs over sequential rules. In the following we will also denote SPRs as $\mathtt{pas} \mid \mathtt{pre} \Rightarrow \mathtt{fut}$ in order to be more illustrative – when appropriate.

How to mine SPRs as defined in Eq. 1 from an execution log L? As depicted in Fig. 1, the mining approach comprises three interconnected steps. First, the prediction model M is *initialized* ② with the set of all *minimal* SPRs over a log L. Here, a SPR $r := (pas, pre, fut)$ is called minimal if $pas = \emptyset$. Formally, the initial prediction model M_{init} is defined and generated as follows:

$$M_{init} := \{(\emptyset, e.ea, e'.ea) | e, e' \in t, t \in L\} \tag{2}$$

Assuming that L solely holds t_1, see Tab. 1, then $M_{init} := \{(\langle\rangle, \mathtt{A}, \mathtt{A}), (\langle\rangle, \mathtt{A}, \mathtt{E}), (\langle\rangle, \mathtt{A}, \mathtt{F}), (\langle\rangle, \mathtt{A}, \mathtt{C}), (\langle\rangle, \mathtt{A}, \mathtt{W}), (\langle\rangle, \mathtt{E}, \mathtt{A}), \cdots\}$, i.e., all possible pre/fut combinations (similar to a Cartesian product). For the sake of brevity, M_{init} is given in parts.

Subsequently, iterative SPR *extension* ③ and *evaluation* ④ steps are performed (cf. Fig. 1). *Extension*: For each SPR $r := (pas, pre, fut) \in M_{init}$ all possible extensions of *r.pas* are explored (i.e., by adding one activity in L's activities to it) as potential SPRs for an extended prediction model M_{ext}. For this, the first iteration exploits M_{init} while later utilize the most recent M_{ext}. Formally: $\forall r \in M_{init}, a \in A := \{e.ea | e \in t, t \in L\}$ an extended SPR set is built:

$$M_{ext} := \{(r.pas \oplus a, r.pre, r.fut) | a \in A\} \tag{3}$$

$M_{ext} := \{(r.pas \oplus a, r.pre, r.fut) | a \in A\}$.

Evaluation: All rules in M_{ext} are then evaluated based on their *confidence* (cf. Def. 2) [26]. The confidence reflects the "likelihood" that a given combination of activity observations, as defined in *r.pas*, results in the future behaviour represented in r to be observed. In other words, the most recently executed activity *r.pre* will be directly succeeded by $r.fut$. Here we exploit the common confidence metric [26] to measure global behaviour (and thus SPR) significance.

Definition 2 (SPR Confidence). *Let SPR $r := (pas, pre, fut)$ be a sequential rule and L be a log. The confidence $conf(L, r) \in [0, 1]$ of r is defined as:*

$$conf(L, r) := \frac{count(pas \mid pre \Rightarrow fut, L)}{count(pas \mid pre \Rightarrow \cdot, L)}$$

where count determines the number of traces in L which r matches to. The numerator considers all "entries" of r (`full match`) while the denominator ignores $r.fut$ (`partial match`), cf. Sect. 3.1. This follows commonly applied association rule confidence calculation techniques, as described in [25].

An example for the confidence calculation is given in Section 3.1. We found alternatives to the confidence metric such as lift or support [26] to have no or even a negative impact as the use of even low support values (e.g., a minimal support of 0.001) frequently results in generating underfitted prediction models.

The evaluation of each rule is performed based on a user configurable minimal confidence threshold $minc \in [0, 1]$. Accordingly, only SPRs with a confidence above $minc$, i.e., $conf(L, r) > minc$, will be retained from M_{ext} to form this iterations version of M. The assumption behind that is: the higher the confidence, the higher the significance of a given rule and, in return, the global execution behaviour it represents. Finally, the resulting prediction model M turns out as:

$$M := \{r \in \bigcup_{M_{ext}} \mid conf(r) > minc\} \tag{4}$$

Rule extension ③ and evaluation ④ cycles (cf., Fig. 1) are repeated until no additional novel SPRs can be identified. Altogether, the repeated extension and evaluation of rules in M aims at identifying and extracting global significant execution behaviour. For this work this means that a specific unique combination of activity observations (see pas) results in a high likelihood that the next occurrence of pre will directly be followed by the activity fut, cf. Sect. 3.

Finally, the SPRs in prediction model M are exploited to predict the activity to be executed next for an ongoing process execution trace $p \notin L$. For this, it must first be determined if *appropriate* SPRs are available in M for the prediction task at hand (⑤, Fig. 1). A SPR $r := (pas, pre, fut)$ is appropriate if pas matches p and $r.pre = p^l.ea$, i.e., the present activity $r.pre$ is equal to the most recently executed process activity in p. Out of these rules r, the appropriate SPR with the highest confidence is assumed as the most relevant one such that its $r.fut$ becomes the expected (predicted, resp.) activity to be executed next for p ⑥.

If appropriate SPRs (global behaviour, resp.) cannot be identified we propose to *fall back* to a local prediction technique (⑦, Fig. 1). Such techniques, typically, are not solely relying on matching specific global behaviour and hence can be applied in a more flexible manner. Throughout the evaluation, a local prediction technique [5] was chosen and extended for this. Further details on the outlined prediction algorithms, their application and SPR matching are given in Sect. 3.

3 SPR based Predictive Monitoring

This section discusses details on SPR mining including the exploitation of existing sequential rule mining optimization strategies and the relation between and prediction of control and temporal execution behaviour.

3.1 SPR Mining & Optimizations

Creating the prediction model and mining the related SPRs requires to apply the proposed *extension* and *evaluation cycle*, cf. Fig 1. The latter requires to determine if and how recorded execution traces $t \in L$ match to a given SPR r. We assume that a SPR can be matched to t either *partly* and/or *fully*, specifically:

Full match: Given a trace t, it is *fully* matched by r if all activity executions specified by r (i.e., past $r.pas$, present $r.pre$, and future $r.fut$ behaviour) can be observed in t in the specified order. Hereby, the activities in $r.pas$ must occur in the orders specified by their indexes $r.pas_i$ (i.e., activity $r.pas_2$ must occur *after* activity $r.pas_1$ was observed, i.e., a *followed by relation* is used). To increase the flexibility of this matching technique arbitrary gaps, i.e., $n \in \mathbb{N}_0$ activity executions, in t are permitted between each matching occurrence of $r.pas_i$. The latter results in less strict prediction models which are less prone to overfitting.

For example, an assumed $r.pas := \langle \texttt{A}, \texttt{E} \rangle$ would match to t_1, t_2, *and* t_3 in the running example, cf. Table 1. Hereby, A and E are only direct successors in t_1 and t_2. In comparison an unrelated activity execution (i.e., F) takes place between A and E in t_3. In future work more complex SPR representations will be explored which, inter alia, restrict the maximum gap between matches for $r.pas$.

In addition, each SPR r consists of $r.pre$ and $r.fut$, cf. Def. 1. Hereby, $r.fut$ is required to be a *direct* successor to $r.pre$ in a given trace t to conclude that r matches to t (i.e., an *immediately followed by* relation is used). This is because here we are interested in predicting the direct successive activity execution (i.e., $r.fut$) for a given most recently executed activity (i.e., $r.pre$).

Partial match: In comparison a *partial* match is already given when $r.pas$ and $r.pre$ can be observed in a given trace t (i.e., $r.fut$ is ignored). Similarly to a full match gaps of arbitrary length are permitted between activity observations (*followed by relation*, resp.). Overall, the described full and partial match concept pave the ground for the SPR confidence calculation outlined previously in Def. 2. Hereby, the full match is implemented accordingly for $count(pas \mid pre \Rightarrow fut, L)$ while a partial match is utilized for $count(pas \mid pre \Rightarrow \cdot, L)$ on the traces in L.

For example, given SPRs $r_1 := (\langle \texttt{E} \rangle \mid \texttt{C} \Rightarrow \texttt{R})$ and $r_2 := (\langle \texttt{E}, \texttt{F}, \texttt{F} \rangle \mid \texttt{C} \Rightarrow \texttt{R})$ along with the traces in the running example Table 1 following matching results would be observed. While r_1 results in a full match for t_1, t_4 and a partial match for t_1, t_2, t_3, t_4; r_2 fully matches t_1, t_4 and partially also matches t_1, t_4. Accordingly the confidence of r_1 is $2/4 = 0.5$ while for r_2 it becomes $2/2 = 1$ such that r_2 is assumed as representing more significant behaviour/predictions.

Finally, the outlined SPR matching (evaluation, resp.) and expansion steps are combined to mine SPRs and create the prediction model M, cf. Algorithm 1. For this, first an initial preliminary prediction model with minimal SPRs is created. Subsequently, the preliminary rules in M are expanded in an iterative manner to increase, iteration by iteration, their respective confidence. Finally, all identified potential SPRs are evaluated based on a final confidence evaluation step. The latter ensures that each $r \in M$ complies to the minimal user chosen confidence threshold $minc \in [0, 1]$. Hereby, a prediction model M is created which can in return be utilized to predict upcoming activity executions for novel ongoing execution traces p of P based on significant historic behaviour in L.

Optimization: For the sake of understandability and simplicity the outlined Algorithm 1 and concepts, cf. Sect. 2, do not reflect a number of optimization

```
Algorithm sprMine(historic traces $t \in L$, minimum SPR confidence $minc \in [0,1]$)
    Result: final prediction model $M$ (SPR collection, resp.)
    $A := \{e.ea | e \in t; t \in L\}$ // all activities in $L$
    $M_{init} := \{(\varnothing, a, a') | a, a' \in A\}$ // initial set of minimal SPRs to expand, Fig. 1 ②
    $M_{ext} := M_{init}$ // SPR set for the expansion process, start with $M_{init}$
    $M := \varnothing$ // the prediction model, a set of SPRs; initially empty
    do
        // create all possible SPR expansions, exploit previous iteration, Fig. 1 ③
        $M_{ext} := \{(r.pre \oplus a, r.pre, r.fut) | r \in M_{ext}, a \in A\}$
        // stop condition, $M_{ext}$'s SPRs shall full match to at least a single $t \in L$
        $M_{ext} := \{r \in M_{ext} | count(r, L) > 0\}$
        $M := M \cup \{r \in M_{ext} | conf(r, L) > minc\}$ // evaluating $M_{ext}$ to enforce the
         min. confidence requirement, Fig. 1 ④, cf. Def. 2
    while $|M_{ext}| > 0$ // expand SPRs till no new behaviour can be learned
    return $M$
```
Algorithm 1: Mines SPRs for a given log L of historic executions.

strategies which were applied by its public prototypical implementation used throughout the evaluation, cf. Sect. 4. For example, instead of iterating over all traces in L throughout each iterative step the implementation holds a list of relevant (i.e., fully matching traces) for each rule. When extending a rule solely this list is analyzed and updated. Further optimizations are applied when generation the initial (M_{init}) and extended (M_{ext}) version of the SPRs. For example, instead of generating all possible (potential) rules only direct successors in L taken into consideration to restrict the generated rules to behaviour which is at least once observed in L – reducing the SPR evaluation efforts. Hereby, this work builds upon optimization strategies proposed for association rules by [25], enabling to reduce the impact of a potential state explosions.

3.2 SPR based Predictions & Fallbacks

After finalizing the prediction model M the activity to be executed next can be predicted for novel ongoing execution traces p where $p \notin L$, cf. Algorithm 2. For this the most recently executed activity ($p^l.ea$) is utilized to identify all *relevant* SPRs in M based on a *rough* and a *fine* filter. The *rough* filter assumes a SPR as relevant if its present activity value (i.e., *r.pre*) is equal to the most recently executed activity in p (i.e., $p^l.ea$) *and* if *r.pas* could be observed beforehand (i.e., before p^l) in p's execution events. We assume that under this circumstances a rule is most likely related to the current execution state of p and its direct future.

Finally, the confidence of each SPR in the roughly filtered representation of M is determined to identify the SPR (rule resp.) with maximum confidence, resulting in a *fine* filter. Hereby, the assumption is that the most confident applicable rule has the highest probability to represent global significant behaviour relevant for p's current execution state and its to be predicted future. If the rough filter removes all rules from M the proposed technique is found to be not applicable for p's current state. The latter can occur if the behaviour in p is too novel, unique, or varying (concept drift) and such could not be mined from L.

Throughout the evaluation we found that about 40% of all activity predictions were affected by this, see Section 4 for a detailed discussion. Accordingly, we propose to apply an alternative local prediction technique in such cases which is not or less affected by such execution behaviour, such as, the technique presented in [5]. Given such a unique execution behaviour the latter technique can still be applied as it solely relies on trace similarities and probability distributions. Such similarities might become low or the probability distributions become less significant but nevertheless some future can and will still be predictable.

```
Algorithm sprPredictAct(ongoing trace p, SPR prediction model M, historic traces L)
    Result: predicted activity to be executed next $a$ for the ongoing trace $p$
    $a := \varnothing$ // by default no activity could be predicted
    $M' := \{r \in M | r.pre = p^l.ea\}$ // filter for relevance by recently executed activity
    if $M' \neq \varnothing$ // only if relevant SPRs are available then
        $mcr := \{r \in M' | conf(r, L) > conf(r', L), r' \in M\}^0$ // filter, max confidence
        $a := mcr.fut$ // r.fut is the most probable activity
    else
        // get $a$ by applying a fallback technique, such as, [5]
    return $a$ // the activity to be predicted next (SPR or fallback)
```
Algorithm 2: Prediction for trace p and prediction model M.

For example, imagine that L consists of the traces t_1 to t_3 given in the running example, cf. Table 1. When assuming t_4 as the currently ongoing trace p which should be predicted upon and that p's most recently observed execution event is $t_4.e_5$ (i.e., the execution of activity C). Firstly, a number of rules will be generated and stored in M. Hereby, M will, inter alia, contain, $r_1 := (\langle \mathtt{A} \rangle \mid \mathtt{E} \Rightarrow \mathtt{E})$ with a confidence c of $0.\dot{3}$, $r_2 := (\langle \mathtt{E}, \mathtt{F} \rangle \mid \mathtt{C} \Rightarrow \mathtt{W})$ with $c := 0.\dot{6}$ and $r_3 := (\langle \mathtt{E}, \mathtt{F}, \mathtt{F} \rangle \mid \mathtt{C} \Rightarrow \mathtt{R})$ with $c := 1$. Secondly, the rules in M are roughly filtered based on *r.pre* (which should be equal to p's most recent activity C) and *r.pas* which should be observed in p's execution events before p^l. Accordingly, only r_2 and r_3 will be taken into consideration for the next fine granular filtering step.

Fine granular filtering identifies the SPR in M with the highest confidence. Given that r_2 has a confidence of $c := 0.\dot{6}$ while r_3 has a confidence of $c := 1$ it is determined that r_3 should be applied for the final prediction step. That final step exploits the information given in r_3 on the related future behaviour (i.e., $r_3.fut$) to predict the direct successor activity for p^l (cf., $t_4.e_5$) correctly as R, cf. Table 1. Demonstrating an advantage of the proposed significant global behaviour based prediction technique over local techniques. Hence, a local prediction technique, such as, [5], could come to the conclusion that, based on L, C is followed by W with a likelihood of $0.\dot{6}$ while R follows on C with a likelihood of $0.\dot{3}$. In return *incorrectly* predicting W as $t_4.e_5$'s most likely direct successor, cf. Table 1.

The proposed global technique predicts based on the SPR with the highest confidence. Alternatives, which were inspired by *bagging* and *boosting* were also explored, cf. [3]. These include, for example, to take the activity with has, on average, the highest relative confidence based on all appropriate SPRs or to choose the future activity which is backed up by the most SPRs. However, throughout the evaluation no consistent tendencies towards one of the explored alternatives were recognized. Hence, the described most simple approach was applied.

3.3 Prediction of Temporal Behaviour

The proposed prediction technique is capable of predicting upcoming activities based on unique global execution behaviour. This also paves a foundation to improve the temporal behaviour prediction technique presented in [5]. The latter is capable of predicting the occurrence timestamp of the next activity execution.

For this, [5] is extended by adding a preliminary filtering step. It filters the traces in L such that $LC \subseteq L$ only retains traces $t \in L$ for which an immediately followed by relation between the most recently executed activity in the ongoing trace p, i.e., p^l, and its predicted successive activity, cf. Algorithm 3, can be observed. Overall, this reduces the noise in $LC \subseteq L$ which the local prediction technique has to cope with – improving the achieved temporal behaviour (i.e., next execution event timestamp) prediction quality, as shown in Section 4.

```
Algorithm tempHistFilter(ongoing trace $p$, predicted activity $a$, historic traces $L$)
    Result: $LC$, a less noisy representation of $L$, cleaned based on the predicted activity $a$
    $LC := \varnothing$ // cleaned representation of $L$
    foreach $t \in L$ // each trace is independently analyzed do
        for $histTraceIndex = 0$ to $|t| - 1$ // check each activity do
            // $t$'s events to evaluate for their direct successive activity relation
            $a_1 := t_{histTraceIndex}.ea$, $a_2 := t_{histTraceIndex+1}.ea$
            // between the most recent activity in $p$ and its predicted next activity
            if $a_1 = p^l.ea \land a_2 = a$ then
                $LC := LC \cup \{t\}$ break // preserve $t \in L$ only if deemed relevant
    return $LC$
```
Algorithm 3: Preparing L for temporal behaviour prediction.

This is achieved by the proposed flexible combination of two specialized techniques. Enabling that improvements and advantages gained in one technique can be integrated into existing alternative solutions to improve their prediction quality as well. For example, assume the first three traces in the running example, cf. Table 1, as L and also assume the most recently executed activity as `C` for the ongoing trace p ($t_4.e_5$, resp.). Without the proposed noise reduction approach the next activity execution timespan between $t_4.e_5$ and $t_4.e_6$ would be predicted based on all three traces in L as 6 (resulting in a timestamp of 36) by [5].

This is because 6 is the most frequently observed timespan for C and an arbitrary successive activity execution based on the non filtered L. When applying the proposed filtering approach, cf. Algorithm 3, $LC \subseteq L$ is reduced to only hold trace t_2 as it is the only trace for which `C` is directly followed by `R`. Hereby, the predicted timestamp becomes 39 as the timespan between `C` and `R` based on t_2 is 9; reducing the timestamp prediction error for this example from 2 to 1.

4 Evaluation

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

LoGo, namely: BPI Challenge 2012[2] (BPIC) and Helpdesk[3]. Both were chosen as they are the *primary evaluation data source* for a range of existing state-of-the-art approaches: [1, 7, 13, 6, 23, 5, 2]. This enables to compare the proposed approach[4] with diverse alternative techniques, such as, neural network or probability based prediction approaches with varying focus on local/global behaviour.
BPIC 2012 log: The BPIC 2012 log is provided by the Business Process Intelligence Challenge (BPIC) 2012. It contains traces generated by the execution of a finance product application process. This process consists of one manually and two automatically executed subprocesses: 1) application state tracking (automatic); 2) handling of application related work items (*manual*); and 3) offer state tracking (automatic). The comparison approaches, such as, [7, 23, 5], are only interested in the prediction of manually performed events. Accordingly, this and the comparison work narrow down the events in the log to become comparable. Overall 9,657 traces with 72,410 execution events were retained.
Helpdesk log: This log is provided by a software company and contains execution traces generated by a support-ticket management process. The log holds 3,804 traces which consist of 13,710 execution events. We assume the helpdesk log as being more *challenging* than the BPIC log. This is because the structural and temporal fluctuation along with the number of activities is higher while the amount of traces, from which behaviour can be learned, is lower.
Comparison approaches: The proposed approach is compared with nine alternative process execution behaviour prediction approaches, see [1, 7, 13, 6, 23, 5]. These apply a number of techniques, such as, finite state automata, histogram like prediction models or neural networks. The latter, either use Recurrent Neural Networks (RNN) – which incorporate feedback channels between the neurons a network is composed of – or Long Short-Term Memory (LSTM) based neural networks. LSTM based neural networks were found to deliver consistent high quality results by adding the capability to "memorize" previous states, cf. [23].

4.1 Metrics and Evaluation

This work applies the same metrics and evaluation concepts as *previous work*, such as, [23, 5], to archive *comparability*. First, the *Mean Absolute Error* (MAE) measure enables to analyze the temporal behaviour prediction quality. For this, the difference between the real observed temporal behaviour in the logs and the predicted timestamps is aggregated. Here, MAE was chosen as it is less affected by unusual short/large inter event timespans (outliers), than alternatives, such as, Mean Square Error, cf. [23]. Secondly, we apply the activity prediction *accuracy*; using the percentage of correctly predicted next activity executions.

Before each evaluation run the log traces were chronologically ordered based on their first event's timestamp (ascending). Enabling to separate them into training (first 2/3 of the traces) and test data (remaining 1/3). Subsequently,

[2] DOI: 10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[3] DOI: 10.17632/39bp3vv62t.1
[4] Public prototypical implementation: `https://github.com/KristofGit/LoGo`

all possible sub traces, $t_{[1,n]}$ where $2 \leq n < |t| - 1$, are generated from the test data and the $(n+1)$th event (activity and timestamp) is predicted. The minimal sub traces length of ≥ 2 provides a sufficient behaviour base for each prediction task. We are aware that this can potentially result in using information from the *future*, if the training data contains long running traces. However, for the sake of *comparability* with existing work the described approach was applied.

4.2 Evaluation Results

Primary tests were applied to identify an appropriate minimum SPR confidence *minc* value for each log and prediction task. For this, potential confidence values, ranging between 0.7 to 0.9, were analyzed. Lower/higher values were not taken into consideration as significant overfitting/underfitting was observed for them.

The achieved evaluation results are summarized in Table 2 (event timestamp prediction) and 3 (activity prediction) – the best result is marked in bold. Overall, the proposed SPR based prediction approach outperforms the state-of-the-art comparison approaches. Throughout the evaluation 64% (BPIC 2012) and 58% (Helpdesk) of the events could be predicted based on SPRs alone, without the need of applying the proposed fallback mechanism. In addition, we compared the correctness of the local prediction approach (fallback, resp. [5]) when the SPR based prediction was applicable. In such cases the SPR based approach predicted 84% of all activities correctly while the fallback approach only achieved 77%.

Accordingly, it can be concluded that the exploitation of global behaviour improves the quality of the prediction results compared to approaches which mainly focus on local behaviour. Nevertheless, given that the proposed fallback mechanism was required regularly it can be assumed that the recorded execution behaviour fluctuates significantly, e.g., because of process drift, cf. [7]. A manual and process mining based inspection of the logs confirmed this observations – which are likely intensified by the long time spans recorded in each log. Future work will explore advanced means to represent global behaviour to address this situation based on clustering, filtering techniques, and extended rule formalisms.

Table 2. Evaluation Results: Execution Event Timestamp Prediction MAE

	Timestamp, Mean Absolute Error (MAE) in Days						
	Proposed SPR	Similarity Histogram Probability [5]	Set abstraction Probability [1]	Bag abstraction Probability [1]	Sequence abstraction Probability [1]	LSTM Neural Network [23]	Recurring Neural Network [23]
Helpdesk	**3.37**	3.54	5.83	5.74	5.67	3.75	3.98
BPIC 2012	**1.53**	1.54	1.97	1.97	1.91	1.56	N.A.[5]

The evaluation shows that the proposed approach is feasible and outperforms a range of comparison approaches. In addition, we found that the generated prediction model can easily be updated if new traces become available or the

Table 3. Evaluation Results: Execution Event Activity Prediction Accuracy

	Activity, Prediction Accuracy						
	Proposed SPR	LSTM Neural Network [7]	Similarity Histogram Probability [5]	LSTM Neural Network [13]	Finite automaton Probability [6]	LSTM Neural Network [23]	Recurring Neural Network [23]
Helpdesk	**0.80**	0.78	0.77	N.A.[5]	N.A.[5]	0.71	0.66
BPIC 2012	**0.78**	0.77	0.77	0.62	0.72	0.76	N.A.[5]

process model is changed. This is because the prediction model is composed of independent rules which have no relation to each other, such that, existing (e.g., outdated) rules can easily be removed while new rules can be added. This property also results in M becoming more transparent and explainable.
Baseline: An additional baseline approach was evaluated. For activity prediction it determines and predicts the most frequent activity for each trace index. Event timestamps are predicted based on the average execution duration between two successive activity executions. The baseline approach achieved an activity accuracy of 0.49 along with a MAE of 1.61 for the BPIC 2012 log.

5 Related Work

Existing prediction work can be assigned into four main categories: *a*) predicting the next event [23]; *b*) estimating remaining execution times [23]; *c*) classifying and predicting instance outcomes [9]; and *d*) predicting risks which could hinder successful instance completions [4]. For this work, we assume *a*) as most relevant.

Related work seems not to take the differences between local vs. global prediction techniques into account *explicitly*. When analyzing the applied fundamental techniques, such as, neural networks, (hidden) markov models, support vector machines, and state automata they, by design, apply a fuzzy mixture of local/global prediction techniques/models [24, 14, 15, 8, 15, 23, 16, 12]. Hence, they aggregate all behaviour in L into a single prediction model using a single technique which is neither specifically optimized for local nor global prediction. This results in requiring the underling core technology to determine which kind of local/global execution behaviour is relevant for each individual event prediction task.

For this, recent work [7, 23], seems to apply a distinct focus on neural network based techniques. Hereby, commonly advanced LSTM networks with multiple hidden layers and up to hundreds of neurons are applied. Such approaches achieve top end prediction quality, see Section 4, but imply significant hardware and computation time requirements throughout the prediction model generation/learning phase. For example, [19] used servers with 128GB of memory and multiple high end GPUs. While [23] stated the individual timespan required for a single training iteration as between "*15 and 90 seconds per training iteration*" [23, p. 483] – while frequently hundreds to thousands of iterations are required.

Such lengthy and computation intense training cycles can harden a prediction approaches' application. Especially for flexible, changing (concept drift), and non centralized application scenarios, cf. [27, 11]. Such are, for example, observed throughout the application of processes in the Internet of Things [27] or during process executions in the Cloud [11]. Given that each change can result in the need to update all prediction models such volatile scenarios can trigger frequent and lengthy training phases for the neural network based approaches.

Accordingly, we assume LoGos capability to train explainable prediction models within minutes on single-core processors, as advantageous. LoGo gains these advantages by being highly specialized on the data (process execution traces) and the task at hand (activity and temporal prediction). In comparison alternative techniques, such as, neural networks have broader applications (e.g., they support also failure prediction) while these flexibility seems to be connected with higher computational requirements and less explainable models, cf. [17, 2].

6 Discussion and Outlook

This paper focuses on two challenges *a*) to provide a global prediction approach; *b*) which can be combined with existing local prediction approaches. We conclude that the proposed approach LoGo was able to meet both challenges. Further, this work evaluates the impact of global behaviour on predictions and compares with and even outperforms a number of related state-of-the-art prediction approaches.

Applying the proposed separation of local and global prediction capabilities into two distinct mixable techniques/models enables to choose the most appropriate one for each upcoming prediction task (similar to ensemble learning, cf. [10]). Compared to alternative prediction approaches these separation also results in simpler prediction models, enabling to gain explainable model capabilities, cf. [20]. As prediction models can have a significant impact on an organization we see this as a significant aspect to leverage a decision maker's trust into such predictions and prediction results, cf. [17]. Finally, the task specialization (i.e., process instance activity and timestamp prediction) of the proposed techniques reduces its hardware requirements and complexity in comparison to general purpose techniques, such as, neural networks [2]. Overall, we assume that this eases the proposed approaches' application on today's fluctuating processes.

Future work will *a*) explore advanced representations of global behaviour; and *b*) widen the global behaviour which is taken into account. Hereby, especially the representation of the past behaviour is of relevance as it can become more or less precise. In this work we applied sequential rules for this, as they are well known and generic such that SPRs will less likely struggle with *overfitting*.

References

1. Van der Aalst, W.M., Schonenberg, M.H., Song, M.: Time prediction based on process mining. Information systems 36(2), 450–475 (2011)
2. Appice, A., Mauro, N.D., Malerba, D.: Leveraging shallow machine learning to predict business process behavior. In: Services Computing, SCC 2019, Milan, Italy, July 8-13, 2019. pp. 184–188 (2019)

3. Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. Machine learning 36(1-2), 105–139 (1999)
4. van Beest, N.R., Weber, I.: Behavioral classification of business process executions at runtime. In: Business Process Management. pp. 339–353. Springer (2016)
5. Böhmer, K., Rinderle-Ma, S.: Probability based heuristic for predictive business process monitoring. In: OTM COOPIS. pp. 78–96. Springer (2018)
6. Breuker, D., Matzner, M., Delfmann, P., Becker, J.: Comprehensible predictive models for business processes. MIS Quarterly 40(4), 1009–1034 (2016)
7. Camargo, M., Dumas, M., González-Rojas, O.: Learning accurate lstm models of business processes. In: Business Process Management. pp. 286–302. Springer (2019)
8. Ceci, M., et al.: Completion time and next activity prediction of processes using sequential pattern mining. In: Discovery Science. pp. 49–61. Springer (2014)
9. Conforti, R., et al.: Prism–a predictive risk monitoring approach for business processes. In: Business Process Management. pp. 383–400. Springer (2016)
10. Dietterich, T.G.: Ensemble methods in machine learning. In: International workshop on multiple classifier systems. pp. 1–15. Springer (2000)
11. Euting, S., Janiesch, C., Fischer, R., Tai, S., Weber, I.: Scalable business process execution in the cloud. In: Cloud Engineering. pp. 175–184. IEEE (2014)
12. Evermann, J., Rehse, J.R., Fettke, P.: Predicting process behaviour using deep learning. Decision Support Systems 100, 129–140 (2017)
13. Evermann, J., et al.: A deep learning approach for predicting process behaviour at runtime. In: Business Process Management. pp. 327–338. Springerg (2016)
14. Ferilli, S., et al.: Extenaded process models for activity prediction. In: Methodologies for Intelligent Systems. pp. 368–377. Springer (2017)
15. Francescomarino, C.D., et al.: Predictive process monitoring methods: Which one suits me best? In: Business Process Management. pp. 77–93 (2018)
16. Francescomarino, D., et al.: An eye into the future: Leveraging a-priori knowledge in predictive business process monitoring. In: BPM. pp. 252–268. Springer (2017)
17. Ghorbani, A., Abid, A., Zou, J.: Interpretation of neural networks is fragile. In: Artificial Intelligence. vol. 33, pp. 3681–3688 (2019)
18. Greco, G., Guzzo, A., Pontieri, L.: Mining taxonomies of process models. Data & Knowledge Engineering 67(1), 74–102 (2008)
19. Lin, L., Wen, L., Wang, J.: Mm-pred: A deep predictive model for multi-attribute event sequence. In: Data Mining. pp. 118–126. SIAM (2019)
20. Lipton, Z.C.: The mythos of model interpretability. Queue 16(3), 30:31–30:57 (2018)
21. Ly, L.T., Maggi, F.M., et al.: Compliance monitoring in business processes: Functionalities, application, and tool-support. Inf. Syst. 54, 209–234 (2015)
22. Mehdiyev, N., et al.: A multi-stage deep learning approach for business process event prediction. In: Business Informatics. vol. 1, pp. 119–128. IEEE (2017)
23. Niek, T., et al.: Predictive business process monitoring with lstm neural networks. In: Advanced Information Systems Engineering. pp. 477–492. Springer (2017)
24. Pandey, S., Nepal, S., Chen, S.: A test-bed for the evaluation of business process prediction techniques. In: Collaborative Computing. pp. 382–391. IEEE (2011)
25. Pei, J., et al.: Mining sequential patterns by pattern-growth: The prefixspan approach. Knowledge and data engineering 16(11), 1424–1440 (2004)
26. Sheikh, L.M., Tanveer, B., Hamdani, M.: Interesting measures for mining association rules. In: Multitopic Conference. pp. 641–644. IEEE (2004)
27. Žliobaitė, I., Pechenizkiy, M., Gama, J.: An overview of concept drift applications. In: Big data analysis: new algorithms for a new society, pp. 91–114. Springer (2016)