Towards next-location prediction for process executions

Andrea Chiorrini*, Claudia Diamantini*, Laura Genga†, Martina Pioli* and Domenico Potena*
*Università Politecnica delle Marche, Ancona, Italy
Email: a.chiorrini@pm.univpm.it, c.diamantini@univpm.it, martinapioli@outlook.com, d.potena@univpm.it
†Eindhoven University of Technology, Eindhoven, The Netherlands
Email: l.genga@tue.nl

***Abstract*—Predictive monitoring of business processes aims at predicting the future of an ongoing process execution. In this work, we focus on the prediction of the next activities to be executed in a running case. However, in contrast with most state-of-the-art approaches, focused on predicting exactly the next activity that will be executed from the current state of the process, we propose an approach aimed at predicting the portion of the process (or "location") that is likely to be executed next. The notion of location allows us to detect activities belonging to the same portion of a control-flow construct (e.g., at the beginning of a parallelism, or at the end of a loop). It provides an abstraction mechanism from the level of the single activity, which can be used to provide the process analyst with an higher-level overview of what can be expected next in the process execution. We validated the approach over a set of real-world datasets comparing and discussing different strategies for training a classifier in returning a location in place of an activity label.**

***Index Terms*—Process Mining; Deep Learning; Predictive Process Monitoring**

I. Introduction

Predictive monitoring of business processes aims at predicting the future of an ongoing process execution. These predictions can pertain to, for instance, the completion time of the current execution, or categorical outcomes representing important properties of the execution or activities that will be executed next [1]. In this work, we focus on the latter. State-of-the-art approaches are focused on predicting which specific process activity will be executed next given the current process execution. This information, however, does not provide the analyst with any insights on the execution context in which the predicted activity is executed. For instance, in the presence of a parallelism it can be argued that a human analyst would like to be informed also on which other activities could occur at that point of the process (and are hence likely to happen shortly after) even though with lower probability. This is particularly true when the chances of execution of the activities are close to each other. In addition, an analyst may want to understand whether the activity to be executed may belong to a cycle, or to know at which point of the process the execution is, e.g. at the beginning or close to the end. If the predictor only returns the activity in output, the only way for the analyst to derive such information consists of a manual inspection of the process model which can be a challenging and error-prone task, since real-life processes can involve complex combination of control-flow constructs with many activities and variants. To tackle this challenge, in this work we introduce an approach that, given an ongoing process instance, aims at predicting the execution context that the process will enter in the next step, hereafter referred to as *location*. Informally, a location is a set of activities with the same structural features, defined on the basis of workflow patterns commonly used in the process modeling domain. We argue that the notion of location provides us with a useful formalism to define activity execution contexts in a rigorous, yet intuitive, way. We provide a precise characterization of the notion of location, and we investigate three possible strategies to predict a location, namely i) an extension of the next-activity prediction problem, where the label of a single activity is predicted and mapped to its location, ii) a direct prediction of the numeric feature vector representing the structural properties of a location, and iii) a direct prediction of the label of the next location. To validate our approach, we carry out a set of experiments on real-world datasets frequently used as benchmarks within the Predictive Process Monitoring (PPM) field. The results obtained show the capability of the approach to support the analyst in the exploration of the process evolution.

The rest of the paper is structured as follows. Section II formally introduces the notion of location; Section III provides the problem statement and the methodology to address it; Section IV describes the experiments designed to assess the validity of the methodology and discusses the results; Section V reviews related work. Finally, Section VI draws some conclusions and highlights possible future work.

II. Defining process locations

Given a reference model for the process under analysis, to determine the location of each process activity we exploit seven model-based features to represent the structural characteristics of process activities, as introduced in [2]. More precisely, let $\mathcal{A}$ denote the set of process activities, we define an embedding $f_e : \mathcal{A} \rightarrow \mathbb{R}^7$ which maps each activity into a vector of seven structural features, each characterizing a specific property of the activity with respect to the morphology of the overall process.

The idea underlying this representation is that activities with similar structural properties will be close to each other in the feature space. The proposed features have been designed taking inspiration from the standard workflow patterns [3], which capture concepts related to well-known control-flow constructs, thus generating an embedding that is understandable for the human analyst. Note that while these features can, in principle, be derived for every process models able to represent the aforementioned control-flow constructs, in this work we focus on workflow nets. The features are described in the following.

1) Path Length: The goal of this feature is to assess the stage of the process in which an activity is executed, defined as the position of the activity from the beginning of the process. Given an activity $a_i \in \mathcal{A}$, we define the longest path $LP(a_i)$ as the maximum number of activities in a path from the beginning of the process to a_i, excluding hidden activities and loops. The Path Length (PL) for a_i is then computed as follows:

$$PL(a_i) = \frac{LP(a_i)}{\max\limits_{a_j \in \mathcal{A}} LP(a_j)} \tag{1}$$

The Path Length assumes values in $]0,1]$, where small values indicate that the activity is located near the beginning of the process and values close to 1 are indicative of activities near the end of the process. For the first activity in the process the value is $\frac{1}{\max\limits_{a_j \in \mathcal{A}} LP(a_j)}$, while for the farthest one it is 1. It should be noted that the farthest activity is not necessarily the last activity in the process, although the two concepts often coincide. This is due to the fact that traversing backloops is not allowed when computing the longest path, with the results that an activity involved in a long loop could lay in a longer path than the final activity. The feature is computed using the longest path so to ensure coherent order among all process activities representation, w.r.t. the process model.

2) Optionality: This feature captures the degree of optionality of a specific activity. Optionality is computed by the reciprocal of the number of the maximum alternative branches in which an activity is involved, as follows:

$$Opt(a_i) = 1 - \frac{1}{n_{\#}(a_i)} \tag{2}$$

where $n_{\#}(a_i)$ denotes the number of alternative branches for the activity a_i. The feature assumes values in $[0,1[$, where 0 represents an activity which is not involved in any alternative branches and values close to 1 are related to activities in choice blocks with many possible branches. The feature is not computed for invisible activities.

3) Parallelism: This feature is used to calculate the degree of parallelism of an activity, i.e., how many activities it is in parallel with. It is calculated similarly to the previous feature, evaluating the number of parallel branches in the parallel block containing the activity. The Parallelism for the activity a_i is computed as follows:

$$Par(a_i) = 1 - \frac{1}{n_{\parallel}(a_i)} \tag{3}$$

where $n_{\parallel}(a_i)$ represents the number of parallel branches for the activity a_i. In the case of nested parallel blocks, the maximum number of possible activities in parallel is considered. For example, if an activity a_i is in a block with two parallel branches, which in turn is in the branch of another external block with two parallel branches, for activity a_i there are at most 3 activities in parallel.

The feature assumes values in $[0,1[$, where 0 represents an activity which is not involved in any parallel branch, whilst values close to 1 are related to activities in parallel blocks with many possible branches. The feature is not computed for invisible activities.

4) Parallelism Path Length: This feature is based on the PL feature previously introduced. As PL aims to capture the position of an activity within a process, Parallelism Path Length (PPL) measures the position of an activity within the parallel block in which the activity is located. The formula is almost the same, with the difference that in PPL the starting point is the beginning of the parallel block instead of the beginning of the process. The feature is computed as follows:

$$PPL(a_i) = \frac{LP_{\parallel}(a_i)}{\max\limits_{a_j \in \mathcal{A}} LP_{\parallel}(a_j)} \tag{4}$$

Where, $LP_{\parallel}(a_j)$ is the length of the longest path for the activity a_j within the parallel block. The path takes into account activities in any parallel branch plus the last activity of the parallel block (i.e., the synchronization point); hence, the activity from which the parallelism starts is not considered. For activities that aren't in any parallel branch the value of PPL is equal to 0. The same considerations made for PL apply to this feature, but limited to the parallel block.

5) Self Loopable: This feature evaluates if an activity can be consecutively repeated. Given an activity $a_i \in \mathcal{A}$, the Self Loopable (SL) feature assumes the value 1 if a_i is in a self-loop, 0 otherwise. Activities with self loops are often activities either prone to failures, or difficult to define as completed, or involving some type of periodic supervision. Hence, it is relevant to discriminate such activities from the others, as they likely have different behaviour and role.

6) Long Loopable: This feature, similarly to the previous one, is concerned with representing repeatability of an activity as the effect of the execution of a sub-process. In this case the Long Loopable (LL) feature assumes the value 1, otherwise 0. The same considerations made for a Self Loopable activity also hold here.

7) Not in Model: This is a binary feature equal to one when the activity is not present in the given model, 0 otherwise. This feature represents the fact that when an activity is missing from the process model no structural information is available.

As an example, let us consider the process model in Petri net notation reported in Figure 1. Table I shows for each activity the values for the structural features introduced above.

Activities with identical feature values determine a location. We use dotted squares to represent locations in Figure 1.

Locations L_1 and L_6 correspond to activity A and H, the only activities occurring at the beginning and at the end of the process respectively. From the Table, we can see that the

TABLE I: Structural features computed for the process model in Figure 1.

Activity	Path length	Optionality	Parallelism path length	Parallelism	Self loopable	Long loopable	Not in model	Location label
A	0.25	0.0	0	0.0	0	0	0	L_1
B	0.5	0.0	0.5	0.67	0	0	0	L_2
C	0.5	0.0	0.5	0.67	0	1	0	L_3
F	0.75	0.0	1.0	0.67	0	1	0	L_4
D	0.5	0.0	0.5	0.67	0	0	0	L_2
E	0.75	0.0	1.0	0.67	1	0	0	L_5
G	0.75	0.0	1.0	0.67	1	0	0	L_5
H	1.0	0.0	0.0	0.0	0	0	0	L_6

location provides the analyst with additional insights on these activities. First, we can immediately see that none of them is involved in any parallel, choice, or loop construct, since the corresponding features are zero. From the path length, we also get a good idea on where the activity occurs within the process. L_2 involves B and D, since they both occur at the beginning of the parallelism immediately after A; note that C is instead in a different location, L_3, since it is involved in a loop. From the parallelism value, we can derive that each of these activities (except A and H) occur in parallel to other two (indeed, we have a total of 3 branches); furthermore, they occur halfway in the parallelism, since the parallel branch they belong to only has two activities in total. L_4 and L_5 involve activities occurring at the same position in the process, but that differ because of the presence of a self loop for E and G, while F is involved in a long loop.

We argue that the notion of location in the context of predicting process monitoring is an abstraction that can provide the process analyst with a higher-level overview of what can be expected next in the process execution. For instance, after the execution of A, traditional next-activity prediction techniques would return either B, C or D, whilst using the notion of location, either L_2 or L_3 is reported. This higher-level information allows the analyst to grasp important insights on the execution context of the ongoing process instance which can, in turn, help in getting decisions regarding the next steps. Keeping the notion of location in predicting the next execution steps in the process model in Figure 1, after one of the activities in L_2 or L_3 has been executed, the process can either stay in the same location (which suggests that multiple parallel branches are usually initiated before proceeding within a specific branch), or move towards L_4 or L_5, suggesting whether the process moves towards a potential repetition of multiple/single activities respectively.

III. Methodology

In this work, we introduce an extension of the next-activity prediction problem, hereafter referred to as *next-location* problem. In particular, we propose a robust approach that exploits both information regarding the process structure, extracted from a process model describing the prescribed process behaviors, and knowledge regarding the sequential execution order for the prediction. The proposed approach involves three main steps: i) *extraction* of location features, ii) *data encoding*, which transforms the event log in a set of prefix traces labeled with the corresponding location, and iii) *classifier training* to perform the prediction. The extraction of location features has already been discussed in Section II. Hence, in the following subsections we first give a formal definition of the next-location problem statement, then we delve into the data encoding step. We do not discuss the classifier architecture and training, since off-the-shelf techniques can be applied.

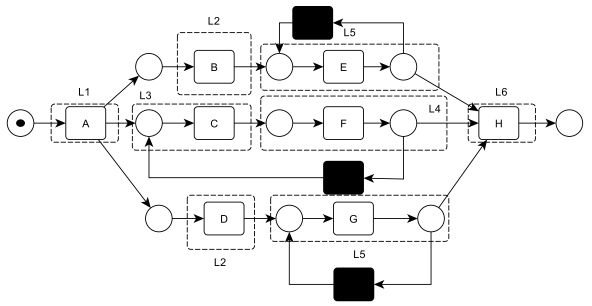


Fig. 1: Example process model with activity locations

A. Problem statement

In our setting, we assume the availability of an *event log* tracking the executions of the process under analysis and of a process model describing the prescribed process behaviors, either provided by domain experts or mined with process discovery techniques. Note that, within the context of this work, we assume that the provided model is able to properly represent (at least) the core process behaviors. Clearly, a poor quality model will heavily impact the performance of the classification, since the control-flow information used to determine the different locations would not correspond to the real process behaviors. We plan to investigate how to improve the robustness of the method w.r.t. the model in input in future work. An event log consists in a collection of *traces*, i.e., sequences of *events*, each corresponding to the execution of a process activity. Formally, given the set $\mathcal{A}$ of process activities, the set $\mathcal{C}$ of process execution identifiers, the set T of all timestamps and the set of integer numbers $\mathbb{N}^+$, an event $e = (a, c, i, t) \in \mathcal{A} \times \mathcal{C} \times \mathbb{J} \times T$ is a tuple consisting of an executed activity $a \in \mathcal{A}$, a case identifier $c \in C$ a number $i \in \mathbb{J} \subseteq \mathbb{N}^+$, and a timestamp $t \in T$. $\mathcal{E}$ represents the set of events. A case corresponds to a single process execution; the number i identifies the position of the event within the sequence of events occurred within a case. Here we introduce also the *projection* operator $\pi_{Att}(x)$, which is used to build the projection of a tuple x on a subsets of its attributes Att. For instance, given an event e we can define the projection $\pi_{\mathcal{A},\mathcal{C},\mathbb{J}}(e_i) = (a, c, i)$. An event trace $\sigma \in \mathcal{E}^*$ is a sequence of events with the same case id. From each trace in the event log, we can derive several *prefix traces*, where a prefix of length k of a trace $\sigma = \langle e_1, e_2, \dots e_n \rangle \in \mathcal{E}^*$, is a trace $p_k(\sigma) = \langle e_1, e_2, \dots e_k \rangle \in \mathcal{E}^*$ where $k \leq n$. Let Π be the set of all prefix traces extracted from L, $\Theta \subset \mathbb{R}^7$ be the set of locations extracted by using the features defined in Section II, let S be a set of location labels, and let $f_m : \Theta \rightarrow S$ be a function mapping each location to a location label.

Finally, the *next-location* prediction problem can be defined as the one of learning a mapping function able to label each prefix trace with the process location of the next state of the process.

B. Data Encoding

Given the the set Π of m trace prefixes that can be generated from L, the goal of this step consists in determining a function mapping each prefix to its vectorial representation. Formally, let V be a vector of size j, we have to define the function $f_v : \Pi \rightarrow V$. In this work, we investigate three strategies to generate the prediction, which require three different kinds of encoding. The first one corresponds to an enrichment of the output of a next-activity classifier, hereafter referred to as *enriched next-activity prediction* (ENAP). The construction of the prefix set for this strategy is the same as in traditional next-activity prediction; namely, we build the dataset $P = \{(f_v(p_i), a_i)\}_{i=2}^m$ where p_i is a prefix of length k of a trace $\sigma \in L$ and a_i corresponds to the next activity of the partial execution described by p_i. Once an activity is returned by the classifier, the function f_m is applied to return in output its corresponding location label. The second strategy aims at returning, given the current state of the process execution, directly the process location label, without first predicting the next activity. We refer to this strategy as *next location label prediction* (NLLP). To support this strategy, we build the dataset $P = \{(f_v(p_i), s_i)\}_{i=2}^m$ where p_i is a prefix of length k of a trace $\sigma \in L$ and s_i corresponds to the label of the next location of the partial execution described by p_i. Finally, the third strategy corresponds to the prediction of the next location feature vector, instead of its label. In practice this strategy corresponds to a regression problem, hereafter referred to as *direct next-location prediction* (DNLP). For implementing this strategy, we need to build a dataset $P = \{(f_v(p_i)), \theta_i\}_{i=2}^m$, where p_i is a prefix of length k of a trace $\sigma \in L$ and $\theta_i \in \Theta$ corresponds to the next location of the partial execution described by p_i.

In all the discussed strategies, the feature vector V can be computed in three different ways; namely, one can generate the one-hot encoding of the process activities, consider their corresponding location, or the combination of the two.

IV. Experiments

This section describes the experiments we carried out on a set of real-world datasets to assess the performance of our approach and to provide some examples on the kind of predictions that are returned.

In particular, we are interested in answering the following questions: *RQ1: How does the direct location prediction performs w.r.t. the location prediction via next-activity prediction? RQ2: What is the impact of the use of the location features on the prediction performance?*

In the following, we first provide a description of the experimental setup and the selected datasets; then, we discuss the results obtained.

A. Experimental setup

In order to answer RQ1, we recall that the prediction problem may be approached in one of the following way:

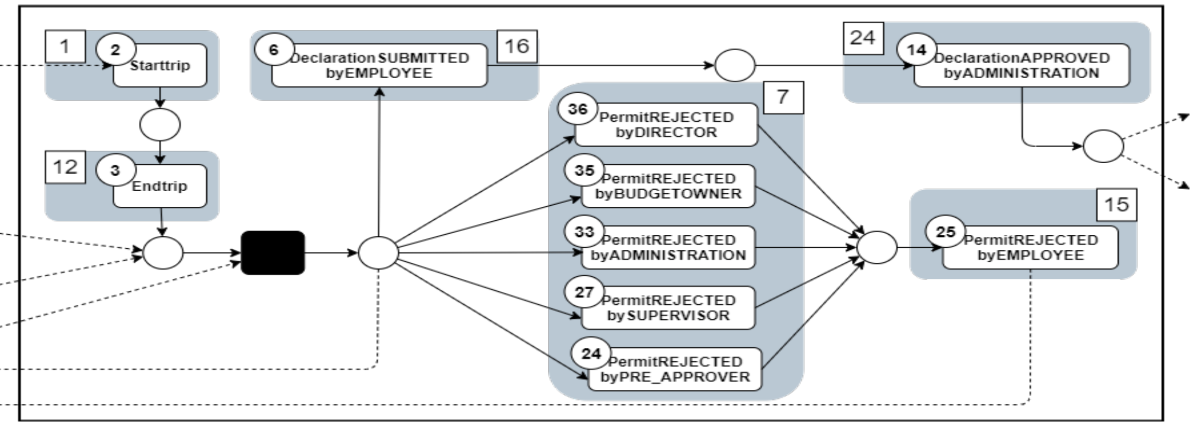
- Predict the next activity label, then associate it to the location it belongs (**ENAP**),
- Predict the next location label (**NLLP**),
- Predict the next location (**DNLP**).

To answer RQ2, we consider 3 possible representation for the input data encoding:

- only the one-hot encode of each activity label (**OHE**);
- only the location of each activity (**ENC**);
- a combination of the two above (**ALL**).

We performed the experiments for 4 combinations of prediction problem and input representation. In particular: $ENAP + OHE$, which implements the classification of the next activity followed by the mapping to the location; $DNLP + ENC$, which determines the location as a regression problem, returning in output the set of location features; $NLLP + ALL$, which exploits both the location features and the activity label, returning in output the location label; finally, $NLLP + OHE$, which only exploits the activity label and returns in output the location label. All the experiments have been performed using tensorflow 2.5 [4] and python 3.8.10 on a machine with a Intel(R) Core(TM) i7-8700K CPU @ 3.70GHz, 32GB on RAM and two GPUs GeForce GTX 1080 with 6GB of dedicated memory each.

We selected for each approach the same neural network architecture: two LSTM layers each with a dropout and a BatchNormalization layer after them and, finally, a fully connected output layer. We also used the same optimizer, Adam [5] and we used as loss function a mean absolute error for the ENC approach and a categorical cross entropy for all the other approaches. For any unexplained parameter the default value has been used. We also run the tree-structured Parzen estimator (TPE) hyper-optimization algorithm [6] with 30 as maximum number of iterations. In particular, the optimization space was composed of the learning rate in the range $[10^{-6}, 10^{-2}]$ and of the LSTM units and dropout, respectively in the set $\{32, 64, 128, 256, 512\}$ and in the range $[0, 1]$, for both the 1st and 2nd layer (separately), for a total of 5 hyper-optimized parameters. The process models were extracted using the



1
2 Starttrip
12
3 Endtrip
6 DeclarationSUBMITTED byEMPLOYEE
16
7
36 PermitREJECTED byDIRECTOR
35 PermitREJECTED byBUDGETOWNER
33 PermitREJECTED byADMINISTRATION
27 PermitREJECTED bySUPERVISOR
24 PermitREJECTED byPRE_APPROVER
24
14 DeclarationAPPROVED byADMINISTRATION
15
25 PermitREJECTED byEMPLOYEE

Fig. 2: Excerpt from the International Declaration process model

TABLE II: Structural features computed for the activities of process model in Figure 2

Activity	Path length	Optionality	Parallelism path length	Parallelism	Self loopable	Long loopable	Not in model	Location label
2	0.5556	0.6667	0.9	0.5	0	1	0	**1**
3	0.6111	0.6667	1.0	0.5	0	1	0	**12**
33	0.6667	0.8333	0.0	0.0	0	1	0	**7**
36	0.6667	0.8333	0.0	0.0	0	1	0	**7**
24	0.6667	0.8333	0.0	0.0	0	1	0	**7**
27	0.6667	0.8333	0.0	0.0	0	1	0	**7**
35	0.6667	0.8333	0.0	0.0	0	1	0	**7**
6	0.6667	0.0	0.0	0.0	0	0	0	**16**
14	0.7222	0.0	0.0	0.0	0	0	0	**24**
25	0.7222	0.5	0.0	0.0	0	1	0	**15**

Infrequent Inductive Miner [7] with the lowest noise threshold that granted at least a fitness of 90%.

All the above have been performed in a 3-fold fashion division of the dataset where a 20% of the training dataset have been used as validation set.

B. Dataset

For our experiments, we selected some of the benchmark datasets commonly used in the literature. Table III provides the list of datasets together with some of their characteristics.

The *Helpdesk* dataset[1] comes from a ticketing management process of the help desk of an Italian software company.

The *BPI12* dataset[2] is taken from a Dutch Financial Institute. The process represents a personal loan or overdraft applications within a global financing organization. As usually done in literature, we retained only the completed events.

The *BPI20* dataset[3] is taken from the reimbursement process at TU/e. The data is split into travel permits and several request types from which we selected three dataset. The first one is the *"Requests for Payment"* (RfP) sub-log, where cost declarations are referred to expenses that should not be related to trips, e.g. representation costs, or hardware purchased for work, etc. The second one is the *Travel Permit* (TP), which includes all related events of prepaid travel cost declarations and travel declarations. The third one is *Prepaid Travel cost* (PC), which contains the events related to prepaid travel costs.

[1]https://data.mendeley.com/datasets/39bp3vv62t/1

[2]https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204

[3]https://data.4tu.nl/collections/BPI_Challenge_2020/5065541

TABLE III: Overview of benchmark dataset. $|\sigma|$ is used to represent the trace length.

Dataset	N. traces	N. events	N. activities	Min $\|\sigma\|$	Max $\|\sigma\|$	Avg $\|\sigma\|$
Helpdesk	3804	13710	9	1	14	3
BPI12	13087	262200	23	3	175	38
BPI20RfP	6886	50568	21	1	20	7
BPI20TP	7065	86581	51	3	90	12
BPI20PC	2099	22444	31	3	23	11
BPI20ID	6449	85049	36	5	29	13

The fourth one is *International Declaration* (ID), which is about events related to declarations for international trips.

C. Results and discussion

1) Example of predicted output: To provide an example of the output returned by our approach, in Figure 2 we draw a portion of the model from the International Declaration dataset. Table II shows the location features and location labels for the activities in the Figure. Locations are also represented in the Figure as grey rectangles.

Let us now discuss the predicted output obtained by the different encodings when the process reaches the *Endtrip* activity. Using *DNLP*, the predictor outputs a location feature vector describing the properties of the next location. The most similar to this one among the all available true feature vectors will be considered the predicted one. In our example, the predicted output was $[0.6567, 0.8463, 0.02, 0.09, 0.01, 0.98, 0.01]$; the most similar location feature vector is $[0.6667, 0.8333, 0.0, 0.0, 0, 1, 0]$, whose label is 7. Using *ENAP*, the predictor first outputs a distribution of probabilities for each process activity; in this case, the predictor returned 36, which is the most likely activity in the set $\{6, 20, 24, 27, 33, 35, 36\}$. That output is then converted in its corresponding location, i.e., 7. This is an example of how it is possible to predict the next location from the next activity. Using *NLLP*, the output will be a distribution of probabilities of the next location classes; in this case, 7 was the most likely location label from set $\{5, 7, 16\}$. We remark that this prediction is performed without the intermediate activity-location mapping used in *ENAP*. As regards the interpretation of the outcome, the user can determine the level of abstraction she prefers, i.e., whether to have in output information only on the location or also on the activity, choosing the appropriate format for the input and the output. In some cases, it might be enough to know that the permit is likely to be rejected by some authority, even without knowing the precise type of rejection. Also, as discussed for the toy example in Section II, the analysis of the location feature equips the analyst with a good overview of the current execution context with respect the overall model, which can be quite challenging to derive manually, given the size of the model. For instance, the analyst is immediately aware that activities of location 7 can lead to work repetition, since they are part of a long loop. This makes sense, since it is reasonable to assume that if a permit is rejected, additional actions have to be taken. On the other hand, activities of locations 16 and 5 do not belong to any cycle and they are closer to the end of the process, as shown by the path length. Summarizing, the used notion of location successfully manage to group together all the activities connected to a rejection from a human controller that lead to potentially restart the procedure business process. We would like to highlight once more that the user can derive such information from our output directly, without having to manually inspect the process model.

2) Analysis of prediction performance: Table IV reports the results obtained over the selected datasets. The $DNLP + ENC$ approach is the one obtaining the worst results. This was expected, since it cannot leverage information related to the specific activities, in contrast with the the other approaches. Instead, there is no clear winner between the two location prediction strategies, since both obtain very similar results. Also regarding the use of the location features no clear trend can be identified. Indeed, $NLLP + ALL$ leads to some improvements w.r.t. $NLLP + OHE$ for the permit and the international datasets; however, it obtains worse results for the request dataset and, in particular, for the prepaid dataset, suggesting that in those experiments the use of the structural features do not provide useful information for the classifier and, instead, it likely introduced some noise. To shed some light on the reasons underlying these results, we delved into the mined process models to understand how locations were defined. In particular, we computed the ratio between the number of the activities in the process model and the number of identified locations. In this regards, it is worth noting that for the prepaid dataset, where $NLLP + ALL$ obtained the worst results, we have 24 locations for 30 activity type; this suggests that the introduction of the location features does not bring much abstraction, supporting the hypothesis that they likely introduce more noise than useful information. The other datasets show a lower ratio, which indicates that more activities could be grouped in the same locations, thus presenting a more favorable condition for the application of our approach. Note however that the mapping between the activities and the locations is not the only factor affecting the classification performance. For instance, helpdesk also presents a high activity-location ratio, without resulting in a reduction of performance. This may depend on the fact that helpdesk is an easier problem, having a very small number of activities and quite a linear structure.

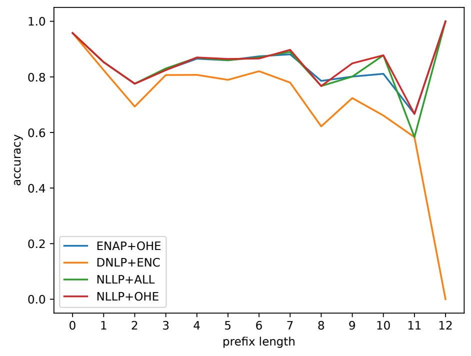
TABLE IV: Classification results from the tested encodings. We report the accuracy (**acc**) and the f1 score (**fsc**) for each fold, together with the average results

		ENAP+OHE		DNLP+ENC		NLLP+ALL		NLLP+OHE	
Dataset	**fold**	**acc**	**fsc**	**acc**	**fsc**	**acc**	**fsc**	**acc**	**fsc**
Helpdesk	0	0.85	0.85	0.81	0.80	0.86	0.85	0.86	0.85
	1	0.85	0.85	0.81	0.80	0.86	0.85	0.86	0.85
	2	0.86	0.85	0.84	0.83	0.86	0.85	0.85	0.85
	avg	**0.85**	**0.85**	**0.82**	**0.81**	**0.86**	**0.85**	**0.86**	**0.85**
BPI20PC	0	0.56	0.58	0.42	0.44	0.53	0.55	0.56	0.58
	1	0.88	0.86	0.69	0.69	0.88	0.86	0.88	0.86
	2	0.89	0.88	0.62	0.60	0.77	0.71	0.88	0.86
	avg	**0.78**	**0.78**	**0.58**	**0.57**	**0.73**	**0.70**	**0.77**	**0.77**
BPI20RfP	0	0.90	0.88	0.86	0.83	0.91	0.87	0.90	0.88
	1	0.90	0.86	0.89	0.95	0.89	0.86	0.90	0.86
	2	0.82	0.79	0.71	0.68	0.81	0.79	0.85	0.82
	avg	**0.87**	**0.84**	**0.82**	**0.79**	**0.87**	**0.84**	**0.88**	**0.85**
BPI20TP	0	0.74	0.72	0.60	0.58	0.75	0.73	0.75	0.72
	1	0.84	0.82	0.80	0.77	0.84	0.83	0.84	0.82
	2	0.83	0.82	0.76	0.73	0.84	0.82	0.83	0.80
	avg	**0.80**	**0.79**	**0.72**	**0.70**	**0.81**	**0.79**	**0.80**	**0.78**
BPI12	0	0.82	0.80	0.72	0.70	0.82	0.80	0.82	0.80
	1	0.82	0.80	0.68	0.67	0.82	0.80	0.82	0.80
	2	0.82	0.80	0.73	0.71	0.82	0.80	0.82	0.80
	avg	**0.82**	**0.80**	**0.71**	**0.69**	**0.82**	**0.80**	**0.82**	**0.80**
BPI20ID	0	0.69	0.68	0.41	0.41	0.73	0.72	0.72	0.71
	1	0.87	0.86	0.42	0.39	0.89	0.88	0.87	0.87
	2	0.86	0.83	0.67	0.65	0.88	0.86	0.86	0.86
	avg	**0.80**	**0.79**	**0.50**	**0.48**	**0.83**	**0.82**	**0.82**	**0.81**

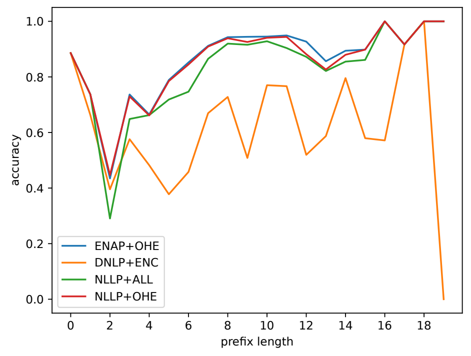
Finally, we analyze the performance with respect to prefix lengths, to study the stability of the outcome. Results are reported in Figure 3. Overall, the figures confirm the trends we already observed; the various approaches perform very close to each other for most prefix lengths, even though the performance of $DNLP + ENC$ are consistently lower than the others. $NLLP + ALL$ performs quite close to $NLLP + OHE$ in most cases, even though some peaks in performance decrease occur for some prefix length, in particular for the BPI20RfP dataset. BPI20TP and BPI12 show quite unstable results for all the encodings for long prefixes, which is reasonable, since few samples are available for training.

V. Related work

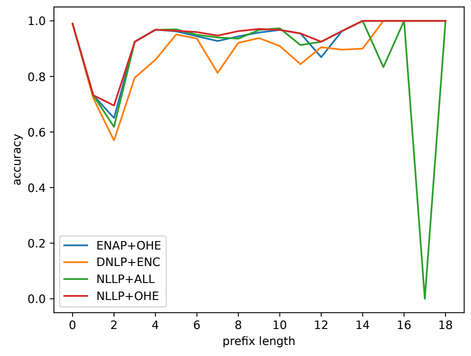
"Predictive monitoring of business processes aims at predicting the future of an ongoing (uncomplete) process execution" [8]. It has made its appearance as a process mining



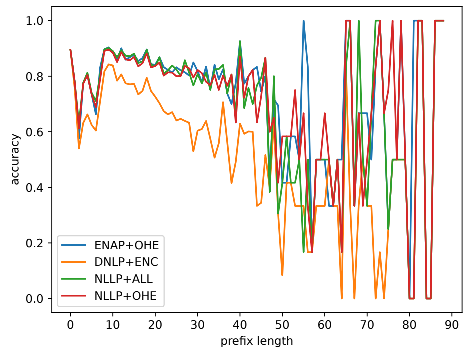
(a) Helpdesk



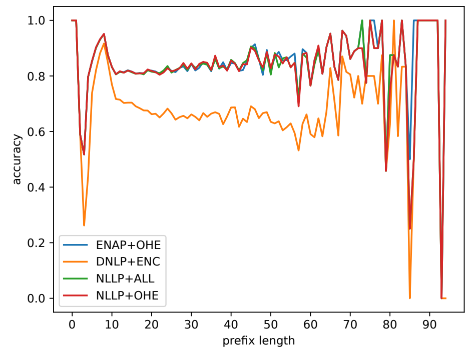
(b) BPI20PC



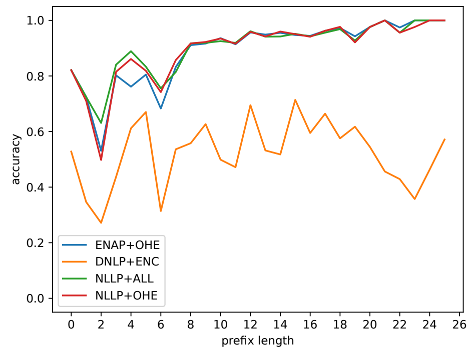
(c) BPI20RfP



(d) BPI20TP



(e) BPI12



(f) BPI20ID

Fig. 3: Accuracy across temporal windows.

task during first decade of the 2000s [9], [10] and is receiving increasing attention in the latest years, as confirmed by the number of recent surveys [1], [11]–[13]. The task can be specialized on the basis of the type of prediction performed.

In particular, three kinds of predictions can be recognized from the literature [1], [11]: prediction of (typically continuous) measures of interest like the remaining execution time, overall duration, or cost of an ongoing case [10], [14], prediction of categorical values like the final outcome or class of risk of a case [11], [15], and predictions related to the sequence of next activities that will be performed [16]. Many different machine learning approaches have been proposed to perform such predictions [17]–[21], but lately a renewed attention is being devoted on leveraging business process model information to perform such predictions. Some approaches rely on the definition of a process model from data, that is then used in combination with machine learning algorithms such as, e.g., decision trees,or regression (e.g., [16], [22], [23]). Recent approaches integrate structural information within a deep learning architecture (e.g., [24], [25]). Very recently [2] explored an alternative approach to representing process structure by introducing a set of features that describe the context and role of an activity inside a process, thus embedding the process structure in the activity representation. To the best of our knowledge this is the first work addressing a problem like the next location prediction.

VI. Conclusion and future work

In the present work, we proposed an approach aimed at predicting the location of a process that is likely to be executed next. The notion of location provides an abstraction mechanism from the level of the single activity, which can be used to supply the process analyst an higher-level overview of what can be expected next in the process execution. We used seven structural features representing common control-flow constructs to define the notion of location. We provided three different formulations for the next-location prediction problem, with different assumptions on the format of the input and the output. We tested our approach on a dataset involving five real-world event logs. The obtained results shown little differences among the tested formulations, which obtained in any case good prediction performance. We also discussed an example of output, to better highlight differences in the output generation obtained by the different formulations and to show how the location features can be used to support the analyst in grasping relevant insights on the execution context.

In future work, we plan to extend our experimental setup to generalize the results obtained. Next, we plan to develop log metrics to use in combination with the structural features to be able to identify process activities without using the one-hot encoding notion. Finally, we plan to investigate applications on our approach for supporting process deviation analysis.

References

[1] C. Di Francescomarino, C. Ghidini, F. M. Maggi, and F. Milani, “Predictive process monitoring methods: Which one suits me best?” in *Business Process Management*, 2018, pp. 462–479.

[2] A. Chiorrini, C. Diamantini, L. Genga, M. Pioli, and D. Potena, “Embedding process structure in activities for process mapping and comparison (under publication),” in *New Trends in Database and Information Systems - ADBIS 2022 Short Papers*, ser. Communications in Computer and Information Science, 2022.

[3] W. M. van Der Aalst, A. H. Ter Hofstede, B. Kiepuszewski, and A. P. Barros, “Workflow patterns,” *Distributed and parallel databases*, vol. 14, no. 1, pp. 5–51, 2003.

[4] M. Abadi, A. Agarwal, O. Vinyals, X. Zheng, and et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[5] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2017.

[6] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, “Algorithms for hyper-parameter optimization,” *Advances in neural information processing systems*, vol. 24, 2011.

[7] S. J. Leemans, D. Fahland, and W. M. Van Der Aalst, “Discovering block-structured process models from event logs containing infrequent behaviour,” in *International conference on business process management*. Springer, 2013, pp. 66–78.

[8] C. D. Francescomarino, *Predictive Business Process Monitoring*. Cham: Springer International Publishing, 2019, pp. 1271–1280. [Online]. Available: https://doi.org/10.1007/978-3-319-77525-8_105

[9] M. Castellanos, N. Salazar, F. Casati, U. Dayal, and M.-C. Shan, “Predictive business operations management,” *International Journal of Computational Science and Engineering*, vol. 2, no. 5-6, pp. 292–301, 2006.

[10] W. Van Der Aalst, M. Pesic, and M. Song, “Beyond process mining: From the past to present and future,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 6051 LNCS, pp. 38–52, 2010.

[11] I. Teinemaa, M. Dumas, M. Rosa, and F. Maggi, “Outcome-oriented predictive process monitoring: Review and benchmark,” *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 2, 2019.

[12] A. Marquez-Chamorro, M. Resinas, and A. Ruiz-Cortes, “Predictive monitoring of business processes: A survey,” *IEEE Transactions on Services Computing*, vol. 11, no. 6, pp. 962–977, 2018.

[13] E. Rama-Maneiro, J. Vidal, and M. Lama, “Deep learning for predictive business process monitoring: Review and benchmark,” *IEEE Transactions on Services Computing*, 2021.

[14] W. Van Der Aalst, M. Schonenberg, and M. Song, “Time prediction based on process mining,” *Information Systems*, vol. 36, no. 2, pp. 450–475, 2011.

[15] J. Becker, D. Breuker, P. Delfmann, and M. Matzner, “Designing and implementing a framework for event-based predictive modelling of business processes,” vol. P-234, 2014, pp. 71–84.

[16] G. Lakshmanan, D. Shamsi, Y. Doganata, M. Unuvar, and R. Khalaf, “A markov prediction model for data-driven semi-structured business processes,” *Knowledge and Information Systems*, vol. 42, no. 1, pp. 97–126, 2015.

[17] V. Pasquadibisceglie, A. Appice, G. Castellano, and D. Malerba, “Predictive process mining meets computer vision,” in *”Business Process Management Forum (BPM’20), Lecture Notes in Business Information Processing*, vol. 392, 2020, pp. 176–192.

[18] N. Tax, I. Verenich, M. La Rosa, and M. Dumas, “Predictive business process monitoring with lstm neural networks,” in *Advanced Information Systems Engineering. CAiSE 2017. Lecture Notes in Computer Science, vol 10253*, 2017, pp. 477–492.

[19] M. Camargo, M. Dumas, and O. González-Rojas, “Learning accurate LSTM models of business processes,” in *Proceedings of the 17th International Conference on Business Process Management (BPM’19), Lecture Notes in Computer Science, 11675*, 2019, pp. 286–302.

[20] F. Taymouri, M. L. Rosa, S. Erfani, Z. D. Bozorgi, and I. Verenich, “Predictive business process monitoring via generative adversarial nets: The case of next event prediction,” in *Business Process Management*, D. Fahland, C. Ghidini, J. Becker, and M. Dumas, Eds. Cham: Springer International Publishing, 2020, pp. 237–256.

[21] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, “A preliminary study on the application of reinforcement learning for predictive process monitoring,” in *Proceedings of 2nd International Conference on Process Mining (ICPM20), Lecture Notes in Business Information Processing*, 2020.

[22] M. Polato, A. Sperduti, A. Burattin, and M. De Leoni, “Time and activity sequence prediction of business process instances,” *Computing*, vol. 100, no. 9, pp. 1005–1031, 2018.

[23] J. Becker, D. Breuker, P. Delfmann, and M. Matzner, “Designing and implementing a framework for event-based predictive modelling of business processes,” *Enterprise modelling and information systems architectures-EMISA 2014*, 2014.

[24] A. Chiorrini, C. Diamantini, A. Mircoli, and D. Potena, “Exploiting instance graphs and graph neural networks for next activity prediction,” in *Process Mining Workshops, Lecture Notes in Business Information Processing*, vol. 433, 2021.

[25] I. Venugopal, J. Tollich, M. Fairbank, and A. Scherp, “A comparison of deep learning methods for analysing and predicting business processes,” in *Proceedings of International Joint Conference on Neural Networks, IJCNN*, 2021.