Remaining Activity Sequence Prediction for Ongoing Process Instances

Xiaoxiao Sun*, Yuke Ying, Siqing Yang and Hujun Shen
School of Computer Science and Technology
Hangzhou Dianzi University
Hangzhou, China
sunxiaoxiao@hdu.edu.cn, yingyuke_hdu@163.com, yangsiqing@hdu.edu.cn, 191050007@hdu.edu.cn

***Abstract*—Remaining activity sequence prediction (i.e., Activity suffix prediction) aims at recommending the most likely future behaviors for ongoing process instances (traces), which enables process managers to rationally allocate resources and detect process deviations in advance. Recently, techniques of neural networks have found promising applications in activity suffix prediction by training a prediction model for next activity and iteratively performing the model to achieve the whole sequence prediction. However, the iterative prediction accumulates the deviations of each iteration and the result also lacks interpretability. In this paper, we propose a novel method to predict activity suffixes from the perspective of control flow and data flow for ongoing traces, where process discovery and trace replay techniques are employed to simulate executions of traces under real conditions and Long Short-Term Memory (LSTM) is applied to characterize the correlation between executed information and future execution. Sequence matching between historical prefix traces and ongoing traces are performed based on the above information to select the optimal-matched (i.e., most similar) activity suffix for ongoing process instances. Experiments on real-life datasets demonstrates that our proposed method outperforms other methods.**

Keywords-activity suffix prediction; process discovery; trace replay; LSTM; sequence matching

I. INTRODUCTION

Business process management (BPM) is a technique that modifies and extends business processes for enterprises by continuously mining, modeling, and monitoring process instances (traces) [1]. As a concrete practice that supports the businesses of enterprises, Process Aware Information System (PAIS) [2] record execution information of process instances (i.e., event logs) such as activity (i.e., event type), timestamp, resources and so on, which is further analyzed for process optimization and improvement. Traditional process mining pays attention to offline analysis like process discovery, conformance checking between logs and process models while recent focus of researchers has gradually turned to online analysis, especially predictive business process monitoring (PBPM). PBPM dedicates to predicting future execution information for ongoing traces such as next activity, remaining time, final outcome, remaining activity sequence (i.e., activity suffix) and so on, which provides reference information for process executors and helps process managers to take effective measures for optimizing process executions.

Among all prediction tasks of PBPM, next activity prediction and remaining time prediction are the most widely studied tasks while activity suffix prediction is seldom considered as a single theme. However, compared to next activity prediction and remaining time prediction, activity suffix prediction provides more extensive future information for both process executors and process managers. Accurate prediction of activity suffix helps process managers to perceive early deviations and resource shortages, which can be prevented by timely and effective measures.

Recently, techniques of neural networks especially Recurrent Neural Network (RNN) [3] and Long-Short Term Memory (LSTM) [4] are widely employed in the fields of PBPM. These works address the tasks of PBPM as regression problems or classification problems and utilize historical traces to train prediction models. However, the diversity of activity suffix categories makes this solution inapplicable. Currently, a variety of researches achieve activity suffix prediction by constructing a prediction model for next activity and iteratively performing the model to predict the whole sequence. These methods, however, are super sensitive to hyper-parameters and each iteration would add deviations to the final result.

Therefore, in this paper, we propose a novel sequence-matching-based approach from the perspective of both control flow and data flow for activity suffix prediction. Summarily, the main contribution of this paper is as follows:

- The techniques of process discovery and trace replay are applied to simulate the real behavioral context of trace executions.
- LSTM is employed to train a prediction model that implicitly characterizes the correlations between executed information and future execution.
- Sequence matching is performed between historical prefix traces and ongoing traces based on the above information to obtain the most similar activity suffixes for ongoing traces.

The rest of this paper is structured as follows: Section II introduces existing works on activity suffix prediction. A detailed description of our approach is presented in Section III. Section IV evaluates the effectiveness of our method and conducts comparisons with the optimal result of other researches on real-life datasets. Eventually, conclusions and future work are demonstrated in Section V.

DOI reference number: 10.18293/SEKE2021-075

II. RELATED WORK

A variety of researches have been put forward to realize activity suffix prediction in the past decade, which are roughly divided into two types according to whether process structures are extracted from event logs, i.e., process-aware methods and non-process-aware methods [5].

Process-aware methods for activity suffix prediction require constructions of process models such as petri net and then apply the models to accomplish prediction. For example, Spoel et al. [6] first mine a causality graph from the event log. Then, they adjust and apply a well-known shortest path algorithm (i.e., Floyd-Warshall algorithm) over the mined causality graphs to find a path whose sum of weights is the least. Similarly, Polato et al. [7] construct a transition system based on the event log and annotate its edges with transition probabilities. Then, they define a cost (i.e., the opposite of the logarithm of the transition probability) between two nodes and apply a shortest path algorithm on annotated transition systems for activity suffix forecasting.

Recently, neural networks especially RNN and LSTM are widely applied to achieve activity suffix prediction, which are typical non-process-aware methods. Tax et al. [8] and Evermanna et al. [9] employ LSTM to forecast activity suffixes for ongoing traces while Lin et al. [11] apply RNN to achieve activity suffix prediction. The similarity of their works is that they all iteratively forecast next activity to realize activity suffix prediction. Specifically, Tax et al. [8] construct a prediction model to forecast both the type of the next event (i.e., next activity) and its timestamp at the same time using a shared LSTM layer. Evermann et al. [9] realize the similar works as [8], however, they encode attributes via embedding space instead of one-hot encoding. Lin et al. [10] propose a RNN-based predictive model called MM-Pred to predict next activities and related attributes, and conduct iteration of the model to obtain the remaining event sequence. In addition, Taymouri et al. [11] present an encoder-decoder architecture grounded on Generative Adversarial Networks (GANs), which generates a sequence of activities and their timestamps in an end-to-end way.

Summarily, most of the current researches adopt neural networks to accomplish activity suffix prediction, which lacks deep mining on event logs as well as interpretation of the predictive results. Therefore, in this paper, we attempt to propose a novel approach to accomplish the task, where process discovery and trace replay techniques are employed to simulate the real trace execution environment and LSTM is applied to characterize the correlation between executed information and future execution of ongoing traces. Eventually, sequence matching is applied to achieve the final prediction based on the above information.

III. APPROACH

A. Preliminaries

1) Event logs

Definition 3.1 (Event; Trace; Event log). An event is one single execution of an activity in different contexts, represented as $e=(c,a,t_s,t_e,r,d_1,...,d_m)$, where $c \in C$ is the case to which the event belongs, $a \in A$ represents the associated activity, $r \in R$ is resources required for execution, t_s and t_e represent the start and end timestamp respectively, and $d_1,...,d_m$ represent the other basic attributes. A trace is a finite ordered sequence of events expressed as $\sigma =< e_1^\sigma, e_2^\sigma, ..., e_{|\sigma|}^\sigma >$, where $|\sigma|$ is the length of σ. An event log L is a collection of multiple traces, which is expressed as $L=\{\sigma_1,\sigma_2,...,\sigma_{|L|}\}$, and $|L|$ denotes the number of traces in L.

Definition 3.2 (Prefix Trace, PT; Suffix Trace, ST). A PT is the first k events of a trace σ, which is denoted as $PT_k(\sigma) =< e_1^\sigma, e_2^\sigma, ..., e_k^\sigma >$. Correspondingly, a ST is the last r events of trace σ and is represented by $ST_r(\sigma) =< e_{|\sigma|-r+1}^\sigma, ..., e_{|\sigma|}^\sigma >$.

Definition 3.3 (Activity Sequence, AS). Given a trace σ, its activity sequence is composed by activities of its events, which is expressed as $AS(\sigma) =< e_1^\sigma.a, e_2^\sigma.a, ..., e_{|\sigma|}^\sigma.a >$.

Definition 3.4 (Event encoding; Encoded matrix). An event encoding is a function $f: e \to D^m$ that transforms the attribute values of event e into a numerical vector (one-hot encoding for category attributes and normalization for numeric attributes), where m denotes the dimension of the encoded vector. Then, for each trace σ, we integrate encoded vectors of its events by time order and obtain an encoded matrix expressed as $EM(\sigma) = [f(e_1^\sigma), f(e_2^\sigma), ..., f(e_{|\sigma|}^\sigma)]$.

2) Petri net

Definition 3.5 (Petri Net). A petri net is an explicit representation of an event log consisting of nodes (places and transitions) and direct arcs. Each place holds a non-negative integer number of tokens, which can be transferred according to firing rules (Definition 3.7). The number of tokens in place p_i is expressed as $\beta(p_i)$. A petri net is defined as a six-tuple, i.e., $pn=(P,T,F,A,\pi,M)$, where:

- $P=\{p_0,p_1,...,p_{|P|-1}\}$ is a finite and non-empty set of places.
- $T=\{t_0,t_1,...,t_{|T|-1}\}$ is a finite and non-empty set of transitions. Transitions in petri net are associated with activities of an event log by a function π, that is, $a=\pi(t)$, $t \in T$, $a \in A \bigcup \{\tau\}$, where A is the activity set of an event log and $\{\tau\}$ represent non-observable activities. Transitions interrelated to non-observable activities are hidden (invisible) transitions.
- $F \subseteq (P \times T) \bigcup (T \times P)$ is the set of directed arcs connecting places and transitions.
- M is the marking that represents the state (the token distribution of places) of the petri net and is denoted as $M=[\beta(p_0),\beta(p_1),...,\beta(p_{|P|-1})]$, where $\beta(p_i)$ can be expressed as $M[i]$, $0 \le i < |P|$.

Definition 3.6 (Input Set, Output Set). Given a node $x \in P \bigcup T$, its input set is denoted as $\bullet x = \{y \mid y \in P \bigcup T \wedge (y,x) \in F\}$ and its output set is represented by $x \bullet = \{y \mid y \in P \bigcup T \wedge (x,y) \in F\}$.

Definition 3.7 (Firing rules). A transition $t \in T$ is enabled iff $\forall p_i \in \bullet t : \beta(p_i) > 0$. Besides, when the transition t is enabled, it can be fired and current marking M converts into a new marking M_t, where $M_t[i]$ is calculated as:

$$M_t[i] = \begin{cases} M[i]-1, p_i \in \bullet t \\ M[i]+1, p_i \in t \bullet \\ M[i], otherwise \end{cases} \quad (1)$$

B. Sequence-matching-based activity suffix prediction

After presenting basic concepts and definitions of this paper, this section introduces the procedure of sequence-matching-based activity suffix prediction, which is divided into three parts, i.e., behavioral context replay, data context prediction and sequence matching.

1) behavioral context replay

Trace replay is a technique that executes traces of an event log on a process model to measure the conformance between the event log and the model [1]. In this paper, inspired by Theis et al. [12], we develop a new application of trace replay to simulate the real-life environment of process executions, i.e., behavioral context replay. Since rare behaviors cannot be characterized by the process model, we adjust the firing rules in Definition 3.7 to guarantee that all the transitions related to activities of traces to be enabled. Specifically, when a transition t is not enabled, we first obtain its input set and find places with token missing. Then, to fill requirements of tokens in these places, we further enable some hidden transitions that connects these places with other places which hold tokens. If t still cannot reach the enabled state by the above operations, we manually add tokens to these places to fulfill the firing requirement of transition t.

In detail, we first conduct process discovery on an historical event log to obtain a petri net using Inductive Miner (IM), which is easy to operate and friendly to implement trace replay [13]. Then, for each trace σ in the event log, we replay it on the obtained petri net according to the adjusted firing rules to simulate its execution. Specifically, during the replay of σ, whenever a transition t_i related to activity a_i is fired, we update the token value of each place and acquire a new marking M_{t_i}. When the trace ends its replay, we integrate all markings and obtain the behavioral context information of σ, which is denoted as $BehavContext(\sigma) = [M_{t_0}, M_{t_1}, ..., M_{t_{|\sigma|-1}}]$. The whole process of behavioral context replay is illustrated in Figure 1.

To measure the behavioral context consistency between two traces, we further introduce a definition named trace behavioral similarity (TBS), whose mathematical expression is illustrated as (2).

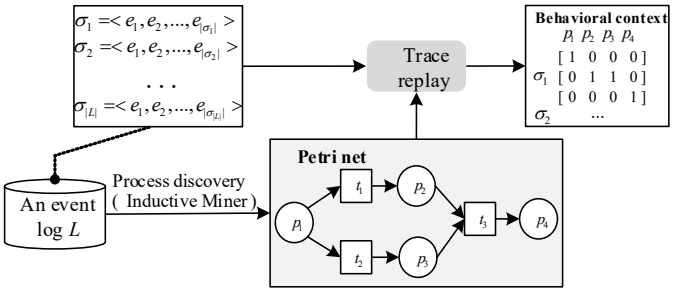

Figure 1. The whole process of behavioral context replay.

Definition 3.8 (Trace Behavioral Similarity, TBS). Given two traces $\sigma_1 =< e_1^{\sigma_1}, e_2^{\sigma_1}, ..., e_{|\sigma_1|}^{\sigma_1} >$ and $\sigma_2 =< e_1^{\sigma_2}, e_2^{\sigma_2}, ..., e_{|\sigma_2|}^{\sigma_2} >$, their TBS is defined as:

$$TBS(\sigma_1, \sigma_2) = \frac{\sum_{0 \le i < \max(|\sigma_1|,|\sigma_2|), 0 \le j < |P|} Eq_{ij}}{|P| * \max(|\sigma_1|, |\sigma_2|)} \quad (2)$$

$$Eq_{ij} = \begin{cases} 1, BehavContext(\sigma_1)_{ij} = 0 \bigcap BehavContext(\sigma_2)_{ij} = 0 \\ 1 - \frac{|BehavContext(\sigma_1)_{ij} - BehavContext(\sigma_2)_{ij}|}{\max(BehavContext(\sigma_1)_{ij}, BehavContext(\sigma_2)_{ij})}, otherwise \end{cases} \quad (3)$$

Where Eq_{ij} represents the token equivalence of place p_j after the i-th activity is executed. The average equivalence of $BehavContext(\sigma_1)$ and $BehavContext(\sigma_2)$ is calculated as $TBS(\sigma_1, \sigma_2)$. Besides, if the dimension of $BehavContext(\sigma_1)$ and $BehavContext(\sigma_2)$ is not consistent, we stuff the one with smaller dimension using padding vectors, i.e., vectors filled with 0.

2) Data context prediction

During the execution of process instances, a variety of data information is produced and recorded as attributes (i.e., resource, cost and so on) in event logs, which is collectively considered as data context in this paper. Data context characterizes the variations of essential attributes, which has significant influence on future execution. In this section, we attempt to mine the correlation between executed information and future execution, and use LSTM model to predict the future data context of ongoing traces.

In detail, we first split historical traces in event logs into PTs and STs. Then, several time-related features are added including *year*, *month*, *day*, *weekday*, *hour* and *duration* for the purpose of enriching information. Subsequently, we perform event encoding on PTs and STs, whose encoded matrixes are considered as the input and training target of LSTM respectively to learn their correlation. Meanwhile, to reduce memory consumption during the training, we conduct dimensionality reduction on suffix matrixes using a popular technique named Uniform Manifold Approximation Projection (UMAP) [14] before training, which is used to deal with high-dimensional data. The matrix of ST *st* after dimensionality reduction is denoted as $EM_{UMAP}(st)$. After training, the correlation between executed data information and future execution is implicitly expressed in the prediction model. For an ongoing trace, we import its encoded matrix to the prediction

model and the output is the predicted data context we need for further sequence matching.

3) Sequence matching

After introducing the behavioral context replay and data context prediction, we further describe the procedure of sequence matching in this section, which is divided into four steps:

Step 1: To better simulate real-life executions, we sort traces in event logs by time and take the first 70% of traces as training set and the remaining 30% of traces as testing set. We further divide the traces in training set and testing set into PTs and STs, where PTs and STs of training set are employed for sequence matching while PTs of testing set are considered as ongoing traces and STs are utilized for evaluation.

Step 2: Subsequently, we perform process discovery on training set using IM and obtain a petri net. For each PT pt in training set and testing set, we replay it on the petri net and acquire its behavioral context information $BehavContext(pt)$.

Step 3: Traces in training set is applied to train the prediction model for data context as mentioned above. Then, for each PT pt in testing set, we import its encoded matrix into the prediction model and obtain its predictive data context, which is denoted as $DataContext(pt)$.

Step 4: After step 2~3, each PT pt in testing set is associated to $BehavContext(pt)$ and $DataContext(pt)$. Then, we perform sequence matching between PTs in testing set and PTs in training set. Specifically, for each PT pt in testing set, we traverse PTs in training set and select PTs with the highest TBS as pt. Then, we further calculate the Euclidean Distance (ED) between the data suffix matrixes of selected PTs and $DataContext(pt)$ to select the most similar PT as pt. Finally, the activity suffix of the selected PT is considered as the predictive activity suffix of pt. Algorithm 1 illustrates the procedure of Step 4.

Algorithm 1: The procedure of Step 4.

INPUT: 1. Training set $S_{training}$; 2. A PT pt in the testing set;

OUTPUT: The predictive activity suffix of pt, $PAS(pt)$;

BEGIN

01: $maxTBS \leftarrow 0$; $minDist \leftarrow \infty$; $PAS(pt) \leftarrow null$;

02: **FOREACH** trace σ in $S_{training}$ **DO**:

03: **FOREACH** i in $range(1, |\sigma|+1)$ **DO**:

04: $curTBS \leftarrow TBS(pt, PT_i(\sigma))$;

05: **IF** $curTBS > maxTBS$ **DO**:

06: $maxTBS \leftarrow curTBS$; $PAS(pt) \leftarrow AS(TS_{|\sigma|-i}(\sigma))$;

07: $minDist \leftarrow ED(EM_{UMAP}(TS_{|\sigma|-i}(\sigma)), DataContext(pt))$;

08: **ELIF** $curTBS == maxTBS$ **DO**:

09: $curDist \leftarrow ED(EM_{UMAP}(TS_{|\sigma|-i}(\sigma)), DataContext(pt))$;

10: **IF** $curDist < minDist$ **DO**:

11: $minDist \leftarrow curDist$; $PAS(pt) \leftarrow AS(TS_{|\sigma|-i}(\sigma))$;

12: **RETURN** $PAS(pt)$;

END

IV. EVALUATION

A. Datasets

In order to verify the effectiveness of our proposed approach, we perform evaluation using four real-life datasets, which can be download from 4TU Centre (https://data.4tu.nl/). The concrete description of datasets is present below and characteristics of datasets is shown in Table I, where "*#Trace*" and "*#Event*" indicate the total count of traces and events in the datasets separately, "*#Activity*" and "*#AS*" denote the number of the different activities and activity sequences in the datasets respectively, and "*#Avg. length*" represents the average length of traces in dataset.

***Helpdesk*:** This dataset contains events from a ticketing management process of the help desk of an Italian software company. All cases in the log start with the insertion of a new ticket into the ticketing management system and end when the issue is resolved and the ticket is closed.
(https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb)

Sepsis: This real-life event log contains events of sepsis cases from a hospital, which were recorded by the Enterprise Resource Planning (ERP) system.
(https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460)

BPIC2012W_Complete: BPIC2012 dataset is an event log taken from a Dutch Financial Institute and represents the process of an application process for a personal loan or overdraft within a global financing organization, which can be split into three sub-processes, i.e., the application itself (BPIC2012A), the work items belonging to applications (BPIC2012W) and the offer (BPIC2012O). In this paper, events with the transition lifestyle of "completed" in BPIC2012W are employed to conduct experiments, which is called BPIC2012W_Complete.
(https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f)

BPIC2012W_Deduplication: Since BPIC2012W_Complete contains a lot of self-loops, i.e., some activities are continuously executed several times, we further perform experiments on BPIC2012W_Complete without self-loops, which preserves the first loop of traces and removes the others. The processed dataset is named as BPIC2012W_Deduplication.

TABLE I. CHARACTERISTICS OF DATASETS

Dataset	*#Trace*	*#Event*	*#Activity*	*#AS*	*#Avg. length*
Helpdesk	4580	21348	14	226	4.66
Sepsis	1050	15214	16	895	13.64
BPIC2012W_Complete	9658	72413	6	2263	7.50
BPIC2012W_Deduplication	9658	29410	6	71	3.05

B. Evaluation metrics

Similar as references [8], [10] and [11], we employ Demerau-Levinstain similarity (DLS, $DLS \in [0,1]$) to measure the similarity of the true activity suffix (TAS) and the predictive activity suffix (PAS). The mathematical representation of DLS is illustrated as (4), where $DL(PAS, TAS)$ is the Demerau-

Levinstain distance between PAS and TAS, and $PAS.length$ and $TAS.length$ represent the length of PAS and TAS respectively. Demerau-Levinstain distance is the minimum number of single-character editions (i.e., insertion, deletion, substitution, and transposition) required to transform one sequence into another.

$$DLS(PAS,TAS) = 1 - \frac{DL(PAS,TAS)}{\max(PAS.length, TAS.length)} \quad (4)$$

C. Experimental setup

Our experiments were run on a 10 core Intel(R) Core (TM) i9-7900X CPU @ 3.30GHz with 64 GB RAM. The approach was implemented in Python 3.6, Keras 2.2.4 with Tensorflow 1.15.0 backend and Pm4py 2.1.0, using CUDA 10.1 and UMAP 0.5.1. The hyper-parameters of LSTM and UMAP are shown in Table II, where *maxLen* is the length of the longest trace in the log.

TABLE II. HYPER-PARAMETERS OF LSTM AND UMAP

Hyper-parameter	*Value*	*Hyper-parameter*	*Value*
LSTM layers	1	Epoch	200
LSTM units	50	Dropout	0.5
Optimizer	Adam	n_neighbors (UMAP)	5
Batch size	128	min_dist (UMAP)	0.3
Learning rate	0.001	n_components (UMAP)	*maxLen*
Loss	mse		

D. Result

Table III summarizes the performance of our method on four datasets in terms of the average DLS. We further analyze the performances in three specific prefix lengths as *short* PTs, *medium* PTs and *long* PTs. As shown in the table, we calculate the average DLS of PTs whose length is more than 2, 4 and 6 for Helpdesk and BPIC2012W_Deduplication while calculate average DLS of PTs whose length is more than 2, 5 and 10 for the other two datasets since the sequence length of Helpdesk and BPIC2012W_Deduplication is relatively shorter than the other two. Besides, *All* represents the average DLS of all PTs in the event log. From the table, we notice that Helpdesk achieve the best DLS, i.e., 84.01%, while BPIC2012W_Complete demonstrates relatively poor performance.

TABLE III. THE AVERAGE DLS OF OUR METHOD ON FOUR DATASETS

Dataset	*DLS*			
	>= 2	*>=4(5)*	*>=6(10)*	*All*
Helpdesk	0.8585	0.8946	0.7970	0.8401
Sepsis	0.3402	0.3369	0.3152	0.3428
BPIC2012W_Complete	0.2936	0.3014	0.3028	0.2821
BPIC2012W_Deduplication	0.4575	0.4650	0.4748	0.4013

Furthermore, to explore the reason for the difference of performances among datasets, we introduce a definition named Coincidence Degree (CD).

Definition 4.1 (Coincidence Degree, CD). The CD of traces in an event log is defined as (5), where $\#AS$ and $\#Trace$ mean the number of different activity sequences and the total count of traces in the log, respectively. The trace behavior of event logs with low CD are highly variable, which improves the difficulty of sequence matching.

$$CD = 1 - \frac{\#AS}{\#Trace} \quad (5)$$

We analyze the correlation between CD and DLS for four datasets, which is shown in Figure 2. In general, our method demonstrates better performance in datasets with high CD while performs relatively poor in datasets with low CD. For example, the CD of Helpdesk is high and its average DLS is correspondingly high while Sepsis demonstrates an opposite situation. However, we notice that the CD of BPIC2012W_Complete is high while its average DLS is low. As mentioned above, this dataset contains a lot of self-loops, which causes our approach to predict overly long sequences of the same activity. From the figure, we conclude that the average DLS of BPIC2012W_Deduplication improves a lot compared to BPIC2012W_Complete, which demonstrates that the self-loops have an adverse effect on our prediction.

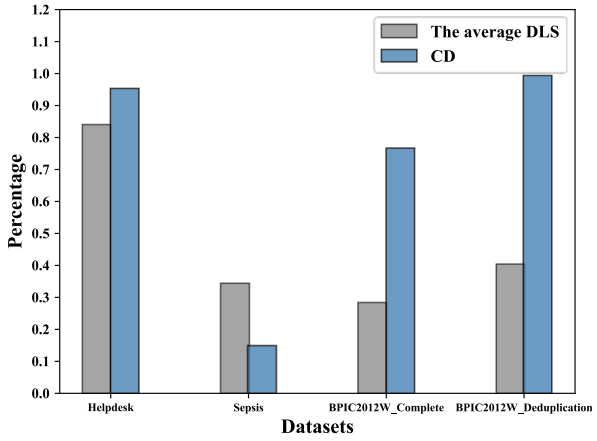

Figure 2. The variation trends of DLS and CD in different datasets.

Furthermore, to explore the performance of our method on PTs with different length, we analyze the variations of average DLS at each prefix length in Figure 3, where the blue polyline and the columnar in gray represent the average DLS and the sample proportion at current prefix length, respectively. As shown in the figure, with the increase of prefix length, the sample proportion gradually decreases and even reaches 0 at some long prefix length. We also notice that with the increase of prefix length, the average DLS first shows a trend of slowly rise since PTs with medium length carry more information than PTs with short length but possess a similar sample proportion, which corresponds to a better sequence prediction result. Gradually, the polylines demonstrate dramatic changes with the further increase of prefix length in all datasets. The reason is that although PTs with long length carry more information, the relatively small number of samples raises the bar of finding historical PTs with consistent behaviors.

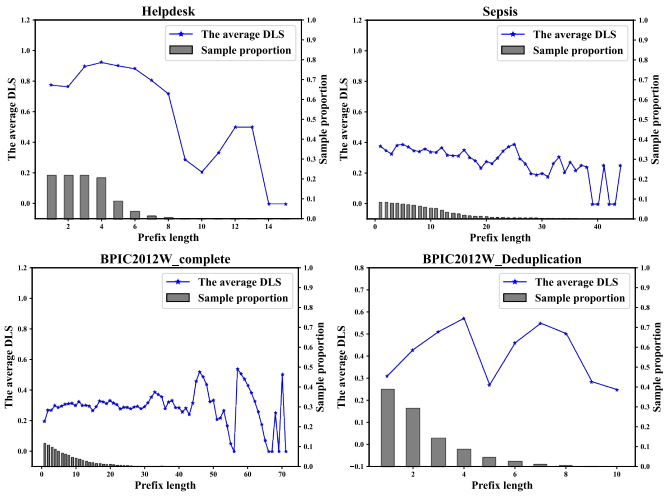

Figure 3. The average DLS at different prefix lengths of four datasets.

E. Comparison with other methods

The comparison of our method with the results of other researches in terms of DLS is shown in Table IV. In this paper, we conduct comparison on Helpdesk, BPI2012W_Complete and BPIC2012W_Deduplication datasets since only their prediction results are reported in references as Tax et al. [8], Evermann et al. [9], Lin et al. [10], and Taymouri et al. [11], which all employ neural networks to achieve activity suffix prediction. In addition, since reference [8], [9], [10] and [11] only report prediction results of PTs at certain prefix lengths, we also calculate the average DLS of the corresponding length of our method to make comparisons. The result shows that our method improves the average DLS over [8], [10] and [11] by 9.16%, 1.74% and 2.06 % in Helpdesk respectively, and outperforms [9] and [11] by 0.44% and 2.74% in BPIC2012W_Complete, respectively. The result of [8], however, surpasses our result by 5.97% in BPIC2012W_Complete. As for BPIC2012W_Deduplication, [8] only improves the average DLS by 4.04% compared to BPIC2012W_Complete while our method has a significant improvement of 16.39%. Besides, our method outperforms [8] by 6.38% in terms of average DLS on BPIC2012W_Deduplication.

TABLE IV. COMPARISON WITH STATE-OF-THE-ART METHODS

Implementation	*Dataset*				
	Helpdesk		*BPIC2012W Complete*		*BPIC2012W Deduplication*
	>=2	*>=3*	*>=2*	*>=5*	*>=2*
Our method	**0.8585**	**0.8946**	0.2936	**0.3014**	**0.4575**
Tax et al.[8]	0.7669	-	**0.3533**		0.3937
Evermann et al.[9]	-	-	-	0.2970	-
Lin et al.[10]	-	0.8740	-	-	-
Taymouri et al.[11]	0.8411	-	0.2662	-	-

Note: "-" represents that the corresponding result of the dataset is not reported in the reference

V. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a method to address the problem of activity suffix prediction, where process discovery and trace replay techniques are employed to simulate executions of traces under real conditions and LSTM is applied to predict and characterize future data context of ongoing process instances. Besides, the above information is eventually applied to perform sequence matching between historical PTs and the current traces. The result of our method outperforms the best result of most methods.

Since our work only cope with the problem of activity suffix prediction, we plan to make suffix predictions related to other execution status such as resource, time and so on in the future, which would provide more reference information for both process executors and process managers.

ACKNOWLEDGMENT

This work is supported by Natural Science Foundation of China (No.61472112), Natural Science Foundation of Zhejiang Province (No.LQ20F020017) and the Key Science and Technology Project of Zhejiang (No.2017C01010).

REFERENCES

[1] Van Der Aalst, W. (2011). Process mining: discovery, conformance and enhancement of business processes (Vol. 2). Heidelberg: Springer.

[2] M. Dumas, W.M. Van der Aalst, and A.H. Ter Hofstede. Process-aware information systems: bridging people and software through process technology. John Wiley & Sons, 2005.

[3] R.J. Williams, and D. Zipser, (1998). A learning algorithm for continually running fully recurrent neural networks. Neural Computation, 1(2).

[4] Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

[5] Marquez-Chamorro A E, Resinas M , Ruiz-Cortes A . Predictive monitoring of business processes: a survey[J]. IEEE Transactions on Services Computing, 2017:1-1.

[6] Spoel S V D, Keulen M V, Amrit C . Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital[C]// International Symposium on Data-Driven Process Discovery and Analysis. Springer Berlin Heidelberg, 2012.

[7] Polato M, Sperduti A , Burattin A , et al. Time and Activity Sequence Prediction of Business Process Instances[J]. Computing, 2016.

[8] N. Tax, I. Verenich, M.L. Rosa, and M. Dumas, (2017). Predictive business process monitoring with LSTMs. Proceedings of the Twenty-Sixth Benelux Conference on Machine Learning (BENELEARN).

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

[10] L. Lin, L. Wen and J. Wang, (2019). MM-Pred: A Deep Predictive Model for Multi-attribute Event Sequence. SIAM International Conference on Data Mining (SDM19).

[11] Taymouri, Farbod, and Marcello La Rosa. "Encoder-Decoder Generative Adversarial Nets for Suffix Generation and Remaining Time Predication of Business Process Models." arXiv preprint arXiv:2007.16030 (2020).

[12] Theis, J., & Darabi, H. (2019). Decay replay mining to predict next process events. IEEE Access, 7, 119787-119803.

[13] Leemans S J J, Fahland D , Aalst W M P . Discovering Block-Structured Process Models from Event Logs - A Constructive Approach[C]// Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings. Springer-Verlag, 2013.

[14] Mcinnes L , Healy J . UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction[J]. The Journal of Open Source Software, 2018, 3(29):861.