information

Article

A Deep Learning Approach for Repairing Missing Activity Labels in Event Logs for Process Mining

Yang Lu *, Qifan Chen and Simon K. Poon

School of Computer Science, The University of Sydney, Sydney, NSW 2006, Australia; qche8411@uni.sydney.edu.au (Q.C.); simon.poon@sydney.edu.au (S.K.P.)
* Correspondence: yalu8986@uni.sydney.edu.au

Abstract: Process mining is a relatively new subject that builds a bridge between traditional process modeling and data mining. Process discovery is one of the most critical parts of process mining, which aims at discovering process models automatically from event logs. Like other data mining techniques, the performance of existing process discovery algorithms can be affected when there are missing activity labels in event logs. In this paper, we assume that the control-flow information in event logs could be useful in repairing missing activity labels. We propose an LSTM-based prediction model, which takes both the prefix and suffix sequences of the events with missing activity labels as input to predict missing activity labels. Additional attributes of event logs are also utilized to improve the performance. Our evaluation of several publicly available datasets shows that the proposed method performed consistently better than existing methods in terms of repairing missing activity labels in event logs.

Keywords: process mining; business process management; incomplete event logs; data quality; data management

Citation: Lu, Y.; Chen, Q.; Poon, S.K. A Deep Learning Approach for Repairing Missing Activity Labels in Event Logs for Process Mining. *Information* **2022**, *13*, 234. https://doi.org/10.3390/info13050234

Academic Editor: Kostas Vergidis

Received: 21 March 2022
Accepted: 30 April 2022
Published: 4 May 2022

1. Introduction

Business process management techniques are widely applied in modern information systems, such as financial, production, and hospital systems. Traditionally, process analysts model business processes through knowledge gained from interviews, workshops, or documents [1]. On the one hand, modeling business processes by hand can be cost ineffective and time consuming. On the other hand, involving human beings to model processes can introduce unavoidable biases. Thanks to the large-scale deployment of computer systems, enterprise data have become more accessible. Process mining, a relatively new subject, was introduced to fill the gap between data mining and traditional process modeling. The goal of process mining techniques is to discover process insights directly from the data collected from target organizations [2].

One of the most critical parts of process mining is called process discovery, which aims at discovering a business process model automatically from process data. The datasets used to discover process models are called event logs. Each event log is a collection of traces, and each trace is an ordered sequence of events. Each event contains an activity, timestamp, and other attributes.

Different process discovery algorithms have been proposed in the last decade, and some of them can guarantee the production of accurate process models under certain circumstances [2]. However, like other data mining techniques, the analysis results are heavily related to the quality of the input datasets [3]. Most existing process discovery algorithms assume the event log to be complete, and they may not be able to discover accurate process models when some data in the input event log are missing. Missing data in event logs has been defined as one of the major data quality issues in process mining [4,5]. Several methods were proposed in the field of process mining to repair event logs with

missing data [6–10]. However, none of these methods can accurately repair missing activity labels in event logs when a large number of activity labels are missing.

In this paper, we focus on repairing the missing activity labels in event logs. Inspired by recent research papers that successfully applied deep learning methods to predict the next activities in ongoing traces, we propose an LSTM-based prediction model to predict the missing activity labels. The prediction model takes both the prefix and suffix sequences of the events with missing activity labels as input. In addition, additional attributes of event logs are also utilized to improve the performance.

1.1. Motivating Example

Table 1 shows an example event log L_1, which describes a simple airport process. Each row is an event, which is an execution record of an activity. An event can have multiple attributes. In the example log, each event has an activity label, a resource, and a timestamp. The activity label describes which activity the event recorded, the resource describes the person who performed the event, and the timestamp describes the time when the event was recorded. The event log contains three traces, and each trace is a sequence of events ordered by timestamps. For simplicity, we can write the event log as L_1 = {<Arrive at Airport, Check-in, Security Check, Boarding, Take off>, <Arrive at Airport, Priority Check-in, Priority Security Check, Priority Boarding, Take off>, <Arrive at Airport, Check-in, Security Check, Priority Boarding, Take off>}. The goal of automated process discovery algorithms is to construct a process model that can accurately describe the process behaviors. For example, if we apply the popular algorithm split miner [11] on L_1, we can obtain the process model as shown in Figure 1. It is easy to interpret the process model: some passengers advance through the priority pathways when arriving at the airport, while others advance through the normal pathways. However, passengers with priority tickets can still advance through the normal check-in and security check first, and only advance through the priority boarding in the end.

Assume there is another event log L_2, shown in Table 2, which is the same as L_1 but with missing activity labels; we can write L_2 as L_2 = {<Arrive at Airport, Check-in, Security Check, Boarding, Take off>, <Arrive at Airport, Priority Check-in, Priority Security Check, Priority Boarding, Take off>, <Arrive at Airport, _, Security Check, _, Take off>}. When trying to discover a process model from L_2, we can ignore the missing activity labels (Figure 2) or the whole traces with missing activity labels (Figure 3). None of the process models can accurately describe the process as shown in Figure 1. For example, in Figure 2, passengers can pass the security check without checking in at the airport. In Figure 3, passengers cannot reach priority boarding after a normal security check.

Table 1. An example event log L_1 without missing activity labels.

Event	Trace Id	Activity	Resource	Timestamp
e_1	1	Arrive at Airport	Tom	1/9/2020 12:00:00
e_2	1	Check in	Jack	1/9/2020 12:20:00
e_3	1	Security Check	Thomas	1/9/2020 12:30:00
e_4	1	Boarding	Linda	1/9/2020 14:20:00
e_5	1	Take off	James	1/9/2020 15:00:00
e_6	2	Arrive at Airport	Alice	2/9/2020 12:10:00
e_7	2	Priority Check in	James	2/9/2020 12:20:00
e_8	2	Priority Security Check	Lucas	2/9/2020 13:30:00
e_9	2	Priority Boarding	Linda	2/9/2020 14:20:00
e_{10}	2	Take off	Peter	2/9/2020 15:00:00
e_{11}	3	Arrive at Airport	Steven	2/9/2020 20:00:00
e_{12}	3	Check in	Jack	2/9/2020 20:20:00
e_{13}	3	Security Check	Mark	2/9/2020 20:25:00
e_{14}	3	Priority Boarding	Linda	2/9/2020 21:00:00
e_{15}	3	Take off	Ethan	2/9/2020 21:30:00

Table 2. An example event log L_2 with missing activity labels.

Event	Trace Id	Activity	Resource	Timestamp
e_1	1	Arrive at Airport	Tom	1/9/2020 12:00:00
e_2	1	Check in	Jack	1/9/2020 12:20:00
e_3	1	Security Check	Thomas	1/9/2020 12:30:00
e_4	1	Boarding	Linda	1/9/2020 14:20:00
e_5	1	Take off	James	1/9/2020 15:00:00
e_6	2	Arrive at Airport	Alice	2/9/2020 12:10:00
e_7	2	Priority Check in	James	2/9/2020 12:20:00
e_8	2	Priority Security Check	Lucas	2/9/2020 13:30:00
e_9	2	Priority Boarding	Linda	2/9/2020 14:20:00
e_{10}	2	Take off	Peter	2/9/2020 15:00:00
e_{11}	3	Arrive at Airport	Steven	2/9/2020 20:00:00
e_{12}	3	-	Jack	2/9/2020 20:20:00
e_{13}	3	Security Check	Mark	2/9/2020 20:25:00
e_{14}	3	-	Linda	2/9/2020 21:00:00
e_{15}	3	Take off	Ethan	2/9/2020 21:30:00

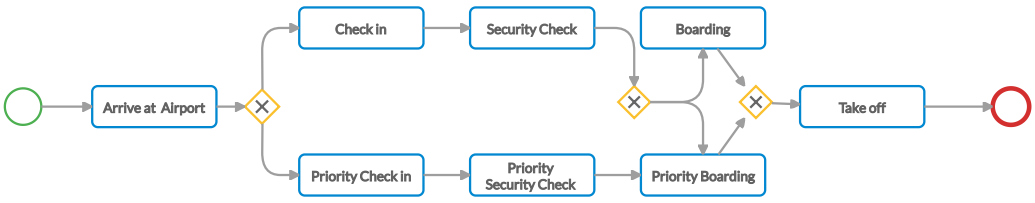

Figure 1. Process model discovered from L_1.

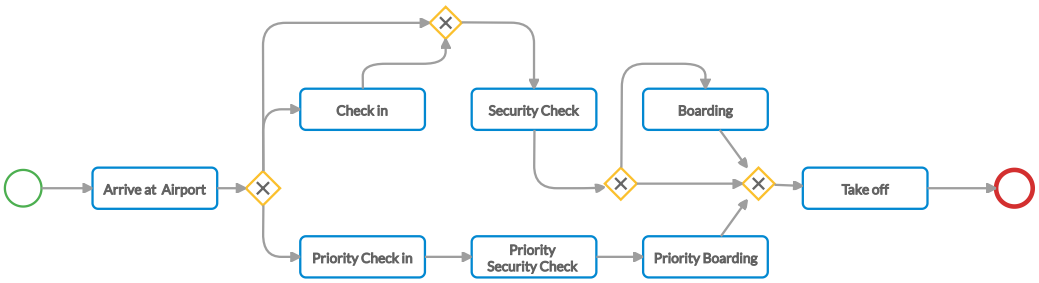

Figure 2. Process model discovered from L_2. Events with missing activity labels are removed.

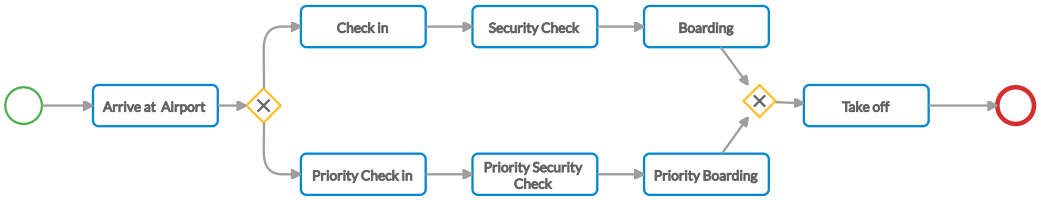

Figure 3. A process model discovered from L_2. Traces with missing activity labels are removed.

The goal of this paper is to propose a method as a data pre-processing tool that can accurately repair the missing activity labels. The repaired event logs can then be used by process discovery algorithms to discover accurate process models.

1.2. Contributions of This Paper

The contributions of this paper include the following:

- To the best of our knowledge, this is the first paper applying artificial neural networks to predict missing activity labels in event logs for process mining.

- An LSTM-based artificial neural network is proposed to repair missing activity labels in event logs.
- Experiments on publicly available datasets under various settings show that our method can accurately repair missing activity labels, even when a large proportion of activity labels were missing in an event log.

The rest of this paper is structured as follows: Section 2 is a literature review of related work. Section 3 introduces some preliminary concepts used in this paper. Our proposed method is presented in Section 4. Section 5 presents the evaluation results. Finally, our paper is concluded in Section 6.

2. Related Work

2.1. Process Discovery Algorithms

Various process discovery algorithms were proposed in the last decade. Alpha algorithms were one of the earliest groups of process discovery algorithms that can construct Petri nets automatically based on the event logs. The original version [12] of the alpha algorithms can guarantee the discovery of certain behaviors in process models when the input event log is noise free and can satisfy certain completeness requirements. However, the original algorithm cannot discover accurate process models with complex behaviors. Later research papers extended the original alpha algorithm to discover short loops [13], invisible tasks [14,15], and non-free-choice behaviors [14,16]. Alpha algorithms produce desirable results on noise-free data, but the performance can be heavily affected when trying to discover process models from real-life event logs. The heuristics miners [17–19] were proposed based on the alpha algorithms to handle noises in event logs.

A challenge for process discovery algorithms was whether the algorithm can guarantee the production of "sound" process models [2], which is the precondition for process models to be used for process simulation or conformance checking (i.e., a group of algorithms to check if the given process model conforms to the input data). None of the algorithms stated above can guarantee the production of sound process models. To solve the problem, inductive miners [20–26] were proposed. Inductive miners always return a process notation called process trees, which can be translated into equivalent block-structured Petri nets [20]. As a result, inductive miners can always discover sound process models. Similar to the alpha algorithms, inductive miners were also shown to discover certain process behaviors when the input event log is complete. Although inductive miners can guarantee the production of sound process models, the behaviors that can be represented by process trees are limited. Recently, the split miner [11] was proposed; it can produce sound process models for most of the time. Instead of process trees, the split miner can discover BPMNs directly. As a result, the split miner can discover more process behaviors.

Besides the process discovery algorithms stated above, there are also other types of algorithms for discovering process models, such as genetic algorithms (e.g., [27,28]), the ILP algorithm (e.g., [29]), and machine-learning-based algorithms (e.g., [30]). However, most of these methods rely on the ordering of events within traces to discover process models. The ordering of events within traces could be incorrect when there are missing activity labels or events [4].

In a nutshell, there are various process discovery algorithms to choose from when discovering process models. However, most process discovery algorithms require the event logs to satisfy a certain degree of completeness requirements. The performance of existing process discovery algorithms can be affected if a large number of activity labels in event logs are missing.

2.2. Missing Data in Event Logs

There are some research papers focusing on handling missing data in event logs for process mining. In [4,5], missing data was defined as one of the data quality issues for event logs. In [6], researchers relied on generalized stochastic Petri nets (GSPNs) and Bayesian network models to repair event logs with missing events. Rogge et al. [6] were the first to

address the missing data issue in process mining. However, when a generalized stochastic Petri net cannot be derived from the event logs (e.g., when a large number of events are missing), the event log may not be accurately repaired. Similar to [6], Song et al. [10] also relied on process models to repair missing events.

In PROELR [7] and SRBA [8], researchers firstly applied trace clustering methods to cluster "complete traces" (i.e., traces without missing activity labels). Each "incomplete trace" (i.e., traces with missing activity labels) would then find the cluster that was closest to it. Finally, the incomplete traces were repaired based on the features of their corresponding trace clusters. Both [7,8] require a large amount of "complete traces" in event logs. As a result, they cannot handle the case when most traces in event logs contain missing activity labels. Furthermore, the performance [7,8] can drop when the event logs contain a large number of missing activity labels.

The MIEC [9] is a multiple-imputation-based method to repair missing data in event logs. Besides repairing missing activity labels, it can also repair all other missing attributes in event logs. The MIEC relied on the dependency relations between event attributes. For example, some activities may always happen on weekends or be performed by a specific group of people. It may not be able to effectively repair event logs when such dependency relations do not exist or the event log contains limited attribute data.

Instead of trying to repair missing data in event logs, Horita et al. [31] applied the decision tree learning algorithm to discover the tendency of missing values in event logs. The output of [31] is a decision tree that indicates the conditions that there is likely to have missing data in event logs (e.g., there is an event with a missing activity label when a certain activity happens before it).

Although a few methods were developed to repair missing data in event logs, a method that is capable of accurately repairing a large amount of missing activity labels is still needed in this field.

2.3. Next Activity Prediction in Event Logs

Recently, artificial neural networks were applied to predict the next events in event logs. The goal of the next activity prediction algorithms is to predict the activity label (or other attributes) of an event in a trace, given its prefix sequence. Different neural networks have been proposed to make the prediction as accurate as possible. For example, Tax et al. [32] and Camargo et al. [33] designed LSTM models for next activity prediction. More specifically, Tax et al. [32] applied one-hot vector encoding to encode all categorical variables, while Camargo et al. [33] applied embedding algorithms to obtain vector representations of categorical variables.

To further improve the accuracy of LSTM-based models, Pasquadibisceglie et al. [34] designed a multi-view LSTM based model that took both control-flow information (i.e., the ordering of activities in traces) and other event log attributes (e.g., the person who performed each activity) as input for next activity prediction. Lin et al. [35] implemented an encoder–decoder structure of LSTMs to predict the next activities. In [35], a customized layer called "modulator" was designed to assign different attributes with different weights. Taymouri et al. [36] combined generative adversarial nets (GANs) with LSTM models to achieve high-accuracy prediction.

Besides LSTM models, other neural network structures were also designed by researchers for next-activity prediction. For example, Pasquadibisceglie et al. [37] converted event logs into 2D representations and designed a neural network model based on a CNN. A stacked autoencoder-based deep learning approach was designed in [38].

The methods stated above can achieve high accuracy. However, as their goal is to predict next activities in ongoing traces, only information in the prefix can be used for prediction. When dealing with missing activity labels in event logs, both prefix and suffix sequences can be used.

3. Preliminaries

3.1. Problem Definition

In this section, we introduce some basic concepts used in this paper.

Definition 1 (Event log, Trace, Activity, Event)**.** *An event log L is a multiset of traces. A trace t, $t \in L$, is an ordered sequence of events. Assuming A is the set of all possible activities, an event e is an execution record of an activity $a \in A$. $\#_n(e)$ denotes the value of attribute n for event e. For example, $\#_{activity}(e)$ refers to the activity label associated with e, and $\#_{timestamp}(e)$ refers to the timestamp of event e.*

Definition 2 (Missing Activity Label)**.** *The activity label of event e is missing if $\#_{activity}(e) = \_$.*

For example, for event log L_2 in Table 2, the activity labels of events e_{12} and e_{14} are missing.

Definition 3 (k-Prefix and k-Suffix of an Event)**.** *Suppose a trace $t \in L$ where $t =< e_1, e_2, e_3, ..., e_n >$. Suppose $i - k \geqslant 0$ and $i + k \leqslant n$; the k-Prefix of event e_i where $e_i \in t$ is the ordered sequence $< e_{i-k}, e_{i-k+1}, ..., e_{i-1} >$, and the k-Suffix of event e_i is the ordered sequence $< e_{i+1}, e_{i+2}, ..., e_{i+k} >$. In this paper, when talking about the prefix and suffix sequences, we refer to the activity sequences. For example, the k-Suffix of event e_i refers to the ordered sequence $< \#_{activity}(e_{i+1}), \#_{activity}(e_{i+2}), ..., \#_{activity}(e_{i+k}) >$.*

In this paper, we focus on repairing events with missing activity labels within event logs.

3.2. Long Short-Term Memory (LSTM)

The method we propose in this paper is based on LSTM [39], which is a common artificial recurrent neural network structure in the deep learning field. LSTM networks are especially suitable for analyzing time-series data and are resistant to the vanishing gradient problem. As mentioned in Section 2, many LSTM-based artificial neural network structures have been proposed by researchers recently to predict next events in ongoing traces.

The definition of an LSTM unit we applied in this paper is presented in the following equations:

$$\mathbf{f}_g^{(t)} = sigmoid(\mathbf{U}_f \mathbf{h}^{(t-1)} + \mathbf{W}_f \mathbf{x}^{(t)} + \mathbf{b}_f) \quad (1)$$

$$\mathbf{i}_g^{(t)} = sigmoid(\mathbf{U}_i \mathbf{h}^{(t-1)} + \mathbf{W}_i \mathbf{x}^{(t)} + \mathbf{b}_i) \quad (2)$$

$$\tilde{\mathbf{c}}^{(t)} = tanh(\mathbf{U}_g \mathbf{h}^{(t-1)} + \mathbf{W}_g \mathbf{x}^{(t)} + \mathbf{b}_g) \quad (3)$$

$$\mathbf{c}^{(t)} = \mathbf{f}_g^{(t)} \circ \mathbf{c}^{(t-1)} + \mathbf{i}_g^{(t)} \circ \tilde{\mathbf{c}}^{(t)} \quad (4)$$

$$\mathbf{o}_g^{(t)} = sigmoid(\mathbf{U}_o \mathbf{h}^{(t-1)} + \mathbf{W}_o \mathbf{x}^{(t)} + \mathbf{b}_o) \quad (5)$$

$$\mathbf{h}^{(t)} = \mathbf{o}_g^{(t)} \circ tanh(\mathbf{c}^{(t)}) \quad (6)$$

$$\forall t \in \{1, 2, \ldots, k\}.$$

In the equations above, $\{\mathbf{U}, \mathbf{W}, \mathbf{b}\}$ are trainable parameters. Each LSTM unit takes a single input vector $\mathbf{x}^{(t)}$. The input vector is passed into different gates that decide how the information will flow into and out of the cell. More specifically, Equation (1) defines the "forget gate", which determines which part of information from the previous cell state to forget. Equation (2) defines the "input gate," which controls the new information to be stored into the memory. Equations (3) and (4) define how the hidden state from the previous LSTM unit $\mathbf{h}^{(t-1)}$ and the new input $\mathbf{x}^{(t)}$ are used to update the cell $\mathbf{c}^{(t)}$. The output gate defined in (5) describes how the information of the cell state $\mathbf{c}^{(t)}$ is used to update the hidden state $\mathbf{h}^{(t)}$, which is passed to the next LSTM unit or subsequent neural network layers. Finally, Equation (6) defines how the output gate is used to update the hidden state $\mathbf{h}^{(t)}$.

4. The Proposed Method

4.1. Data Preprocessing

The core idea of the proposed approach is to use supervised learning approaches to predict the missing activity labels in event logs. In other words, the events without missing activity labels are used to train the prediction model, and the prediction model is then used to predict the missing activity labels in the event log.

Firstly, we need to split the original event log in order to obtain a training dataset, where each sample is labeled. We divide all events in the event log into two sets. The first set $E_{complete}$ contains all events with activity labels, and the second set $E_{missing}$ contains all events with missing activity labels. For example, for the sample log L_2 shown in Table 2, $E_{complete} = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7, e_8, e_9, e_{10}, e_{11}, e_{13}, e_{15}\}$, and $E_{missing} = \{e_{12}, e_{14}\}$. For each event in $E_{missing}$ and $E_{complete}$, we obtain the activity labels of its $k - Prefix$ and $k - Suffix$. For events in $E_{complete}$, we obtain a training dataset where each sample is labeled. The training dataset is then used to train a neural network model that is used to predict the activity labels in $E_{missing}$. Besides activity labels, other attributes of events are also included in our proposed deep learning architecture.

For our example event log L_2, Tables 3 and 4 present the two datasets constructed from $E_{missing}$ and $E_{complete}$ when $k = 3$. For each event, besides its prefix and suffix activity sequences, its resource is also preserved in the dataset. In addition, a special label "Missing" is assigned to the missing activity labels in the prefix and suffix. Using the label "Missing" can let the neural network model know that there is supposed to be an activity label at a specific place. For example, suppose there is a trace $< A, \_, \_ >$ with two missing activity labels in an event log where activity C always happens two activities after activity A. If we represent the last event's prefix as $< A, Missing >$, we can easily know the trace is $< A, \_, C >$.

Table 3. The example dataset constructed from $E_{complete}$.

Event	Resource	Prefix_1	Prefix_2	Prefix_3	Suffix_1	Suffix_2	Suffix_3	Label (Activity)
e_1	Tom				Boarding	Security Check	Check in	Arrive at Airport
e_2	Jack			Arrive at Airport	Take off	Boarding	Security Check	Check in
e_3	Thomas		Arrive at Airport	Check in		Take off	Boarding	Security Check
e_4	Linda	Arrive at Airport	Check in	Security Check			Take off	Boarding
e_5	James	Check in	Security Check	Boarding				Take off
...	...	...	...	...	...	...	...	...
e_{15}	Ethan	Missing	Security Check	Missing				Take off

Table 4. The example dataset constructed from $E_{missing}$.

Event	Resource	Prefix_1	Prefix_2	Prefix_3	Suffix_1	Suffix_2	Suffix_3	Label (Activity)
e_{12}	Jack			Arrive at Airport	Missing	Security Check	Missing	_
e_{14}	Linda	Arrive at Airport	Missing	Security Check			Take off	_

Finally, to feed the dataset into the neural network, the categorical variables have to be transformed into numerical values. As a result, categorical data are passed into an embedding layer first to be transformed into a vector representation. Depending on

the choice of embedding methods, more work could be required to process the data. For example, the categorical values may be required to be represented by non-negative integers [40]. Algorithm 1 presents the steps to construct the dataset from $E_{complete}$. The same steps are applied for construct the dataset from $E_{missing}$.

Algorithm 1: Constructing the Training Dataset from $E_{complete}$

Input: $E_{complete}$, k, Attributes
// k refers to the length prefix/suffix. Attributes refers to the list of additional attributes used to repair missing activity labels
DatasetComplete ← []
for *e in* $E_{complete}$ **do**
  new_record ← []
  for *attr in Attributes* **do**
    new_record.add($\#_{attr}(e)$)
    // Add additional attributes into the dataset
  end
  prefix_events ← getPrefix(e, k)
  for e_{prefix} *in prefix_events* **do**
    if $\#_{activity}(e_{prefix})$ *is missing* **then**
      new_record.add("Missing")
      // a special label ‘‘Missing’’ is assigned to the missing activity labels in the prefix
      else
        new_record.add($\#_{activity}(e_{prefix})$)
      end
    end
  end
  suffix_events ← getSuffix(e, k)
  for e_{suffix} *in suffix_events* **do**
    if $\#_{activity}(e_{suffix})$ *is missing* **then**
      new_record.add("Missing")
      // a special label ‘‘Missing’’ is assigned to the missing activity labels in the suffix
      else
        new_record.add($\#_{activity}(e_{suffix})$)
      end
    end
  end
  new_record.add($\#_{activity}(e)$)
  DatasetComplete.add(new_record)
end
Output: DatasetComplete

4.2. The Deep Learning Architecture

The overall architecture of our proposed neural network is presented in Figure 4. The LSTM models for the prefix and suffix sequences are established separately. The deep learning architecture contains two LSTM models. One LSTM model handles the prefix sequence, and the other handles the suffix sequence.

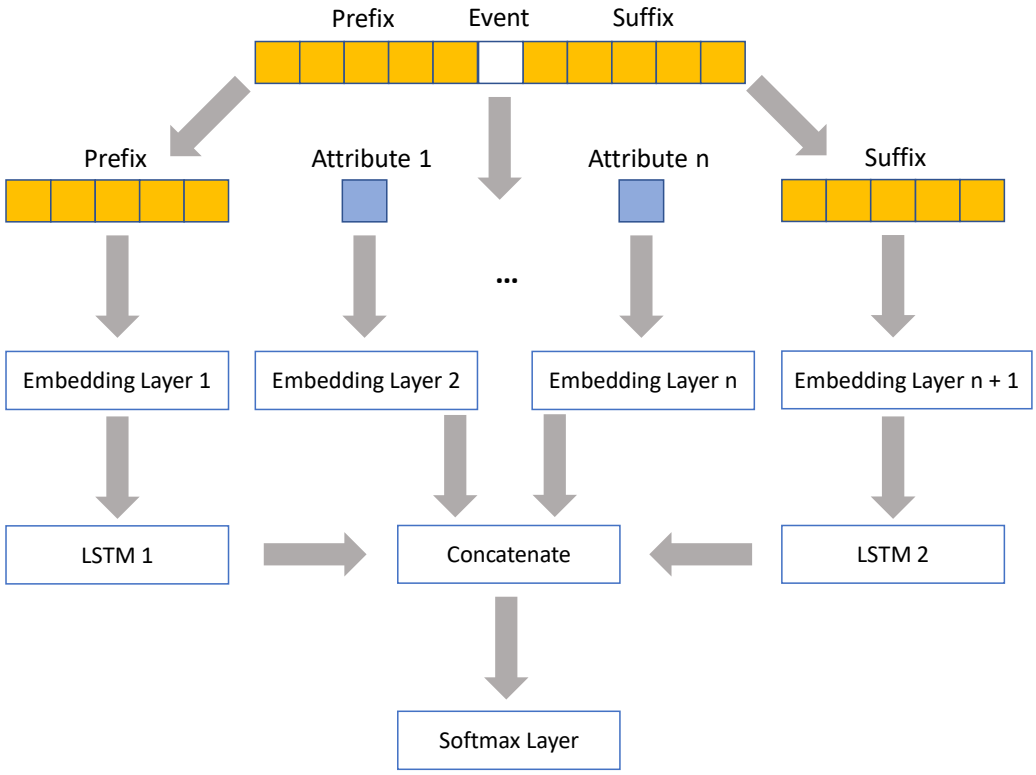

Figure 4. Architecture of our proposed neural network.

For example, Figure 5 shows the unfolded LSTM network for the prefix of e_5 in L_2. The prefix of e_5 is presented in Table 3. The LSTM network captures the temporal information for the prefix of e_5 and outputs a hidden representation $\mathbf{h}^{(t)}$, which is a fixed-size vector. When predicting next activities in ongoing traces, the vector could be passed directly into a dense layer to make the prediction [32,33]. However, when predicting the missing activity label of an event, we can use the temporal information from both its prefix and suffix. As a result, another LSTM, that captures the temporal information of the suffix sequences of events, is also included in the deep learning architecture.

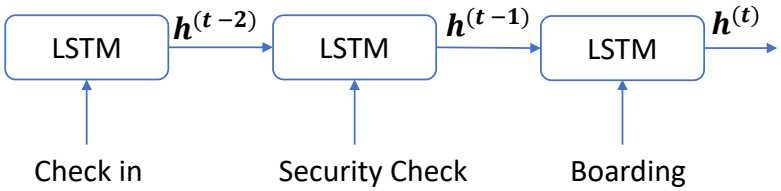

Figure 5. Unfolded LSTM network for the prefix of e_5, the activities are in vector representations.

Besides the temporal information of prefix and suffix sequences, the known attribute values of the events are also used by our proposed method. As shown in Figure 4, these attribute values are also passed into embedding layers to be transformed into vector representations. It has to be noted that embedding layers are only needed for categorical variables.

A concatenation layer is used to combine all of the vector representations we obtained. The output vector of the concatenation layer contains the temporal information of the input events' prefix sequences and suffix sequences as well as the information of its attribute values.

Finally, the output of the concatenation layer is fed into a dense layer that uses the softmax activation function:

$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \tag{7}$$

The output of the softmax function is a vector that contains the probability of different activity labels. The activity label with the highest probability is selected to repair the event. To train the neural network model, the backpropagation algorithm is used to find the optimal trainable parameters. In addition, cross-entropy is used as the loss function.

5. Evaluation

To prove that our proposed method can accurately repair missing activity labels in event logs, a large number of experiments were conducted. Overall, we performed two groups of experiments. The first group of experiments compared the performance of our proposed method with existing methods to repair missing activity labels in event logs. The second group of experiments performed further analysis to prove the effectiveness of our method.

We implemented our approach in Python 3.7.1 based on Tensorflow 2.7.0. For the embedding layers, we used the built-in embedding layer in Keras, which requires the categorical variables to be transformed into non-negative integers. In all our experiments, the prefix and suffix lengths were set to five. Zero padding was added if the length of the prefix/suffix was shorter than five (e.g., the event is at the beginning of a trace). In addition, only resources were used as additional attributes to repair missing activities in the experiments of this paper. The dimensions for the embedding layers of the prefix and suffix were set to 100, and the dimension for the embedding layer of resources was set to 16. Probabilistic dropouts of 0.2 were also applied to the outputs of the embedding layers. Moreover, batch normalization was also added to the output vectors of the concatenation layer. Both LSTM networks in our proposed structure contained two layers (32 neurons in the first layers, and 16 neurons in the second layers). During the training process of the model, the training dataset was shuffled first, and 20% of the training dataset was used as the validation set. To minimize the loss, we used the Nadam optimizer. The maximum number of epochs was 100 (Early stop was set to 10 epochs), the batch size was set to 32, and the learning rate was set to 0.002.

To evaluate the performance of our method, we applied the same evaluation matrix as found in [7–9], i.e., the success rate. It measures the proportion of missing activity labels repaired successfully to the total number of missing activity labels. Equation (8) defines the success rate, where m is the number of activity labels that are repaired successfully, and n is the total number of missing activity labels.

$$\text{Success Rate} = \frac{m}{n} \tag{8}$$

The experiments utilized several publicly available datasets. In total, our evaluation was based on six publicly available event logs:

- The Production Process Log (https://doi.org/10.4121/uuid:68726926-5ac5-4fab-b873-ee76ea412399 (accessed 31 December 2021)): An event log of a factory's production process.
- Hospital Billing (https://doi.org/10.4121/uuid:76c46b83-c930-4798-a1c9-4be94dfeb741 (accessed 31 December 2021)): An event log extracted from a regional hospital's ERP system. It contains the processes used to bill bundled packages of medical services.
- BPI Challenge 2012 (https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f (accessed 31 December 2021)): An event log containing a loan application process in a Dutch financial institute.

- Sepsis Log (https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460 (accessed 31 December 2021)): An event log containing processes to deal with sepsis patients in a hospital.
- Helpdesk (https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb (accessed 31 December 2021)): An event log that describes the ticketing management process in a software company in Italy.
- BPIC 2013 Incidents (https://doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee (accessed 31 December 2021)): An event log of the incident management process in Volvo IT.

The details of all used datasets are presented in Table 5. For the "Hospital Billing" event log, we filtered out all traces with only one or two events. The "Hospital Billing" event log was filtered in the same way as in [8]. As shown in Figure 6, we firstly randomly deleted a number of activity labels from these event logs. Two datasets were then constructed. The dataset constructed from $E_{complete}$ was used to train the neural network model, and the dataset constructed from $E_{missing}$ was used to evaluate the model and calculate success rates.

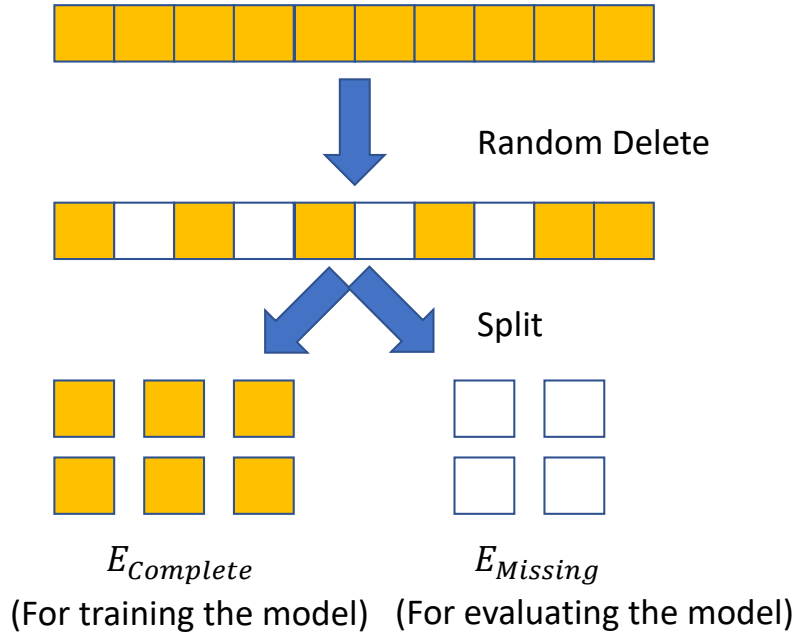

Figure 6. Pre-processing publicly available datasets to evaluate our method.

Table 5. Characteristics of the used publicly available datasets.

Dataset	Number of Traces	Number of Events	Number of Activities	Number of Resources
Production Process Log	225	4544	55	31
Hospital Billing (Filtered)	69,252	412,236	18	1105
BPI Chanllenge 2012	13,087	262,200	24	69
Sepsis Log	1050	15,214	16	26
Helpdesk	4580	21,348	14	22
BPIC 2013 Incidents	7554	65,533	13	1440

The settings based on those datasets were slightly different in different experiments, which are explained in detail in the following subsections.

5.1. Comparing with Existing Methods

In the first group of experiments, we compared the performance of our proposed method with [7–9]. Since all of these methods were evaluated using publicly available

datasets, we can compare our results directly with their results using the same datasets and under the same settings.

We firstly compared our method with PROELR [7] and SRBA [8]. In [8], both methods were evaluated using the "Hospital Billing Log" and the "BPI Challenge 2012" logs with randomly deleted activity labels. Since both methods rely on trace clustering algorithms, only a small portion of activity labels can be deleted. In addition, only one activity was deleted in each trace.

Following the data preparation methods in [8], we deleted activity labels of 100, 150, 200, and 300 events from the "Hospital Billing Log" and "BPI Challenge 2012" logs, respectively, and only one activity label was allowed to be removed from each trace. For each number of missing activity labels, we repeated the same procedure 10 times and reported the average. For example, when removing 100 activity labels from the "Hospital Billing Log", we obtained 10 different event logs with 100 missing activity labels, and the success rate reported is the average success rate among the 10 logs.

The results for comparing our method with ROELR and SRBA are presented in Table 6 and Figure 7, where the success rates of ROELR and SRBA were referenced directly from [8]. The success rates of our methods are all above 0.99, which indicates that almost all missing activity labels can be successfully repaired. The success rates are around 0.8 for SRBA and 0.4–0.7 for PROELR, which are lower than our method.

Next, we compared our method with the MIEC [9]. The MIEC was evaluated by the "Production Process Log" in [9]. To obtain event logs with missing activity labels, the different proportions of activity labels were randomly removed from the log. There were no limits on the number of activity labels deleted in each trace.

Following [9], we deleted 15%, 20%, 25%, and 30% of the activity labels from the "Production Process Log" to obtain event logs with missing activity labels. The evaluation results are presented in Table 7 and Figure 7. As in the previous experiment, the results of the MIEC were referenced from [9] directly, and all the success rates of our method are the average of 10 repeats. Both methods can achieve high success rates when only a small proportion of activity labels are missing in the event log. However, the performance of the MIEC drops when the number of missing activity labels increases. The success rates drop by around 5% when the number of missing activity labels increases by 5%, and only 78.8% of the missing activity labels can be repaired successfully when 30% of the activity labels are removed. Compared to our method, the success rates remain stable when the number of missing activity labels increases. Around 94% of the missing activity labels can be repaired successfully at all different levels of missing activity labels.

Table 6. Comparison of our method with PROELR [7] and SRBA [8].

Dataset	Number of Missing Activity Labels	PROELR [7]	SRBA [8]	Our Method
Hospital Billing	100	0.644	0.816	**0.995**
	150	0.650	0.805	**0.991**
	200	0.635	0.811	**0.991**
	300	0.668	0.825	**0.993**
BPI Chanllenge 2012	100	0.441	0.800	**0.993**
	150	0.463	0.790	**0.996**
	200	0.432	0.772	**0.992**
	300	0.438	0.760	**0.994**

Table 7. Comparison of our method with the MIEC [9].

Dataset	Number of Missing Activity Labels	MIEC [9]	Our Method
Production Process Log	15% (681)	0.925	**0.946**
	20% (908)	0.876	**0.938**
	25% (1817)	0.837	**0.937**
	30% (1363)	0.788	**0.938**

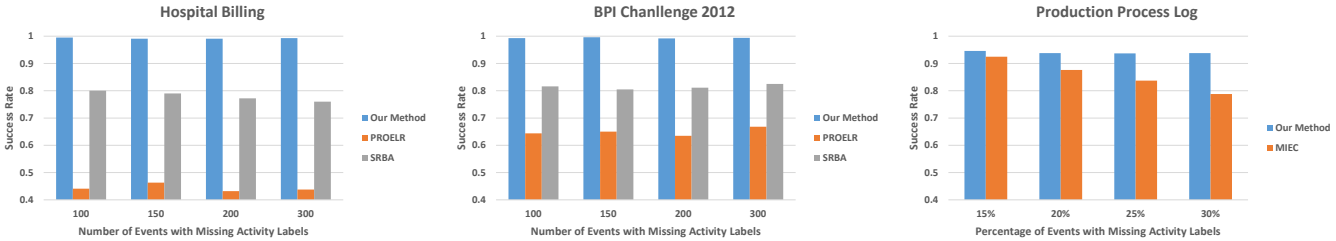

Figure 7. Comparison of our method with others.

5.2. Further Analysis of Our Proposed Method

To further analyze the performance of our proposed method, we evaluated our method with more event logs and missing activity labels. Besides running the experiments on our method, several baselines were also implemented to prove the effectiveness of our method:

- Prefix Only: A LSTM-based prediction model that predicts the missing activity label of an event using only its prefix sequence.
- Suffix Only: A LSTM-based prediction model that predicts the missing activity label of an event using only its suffix sequence.
- Our Method (Without Resources): Our proposed model that uses only prefix and suffix sequences of an event to predict its missing activity label.

In total, all six event logs were used to evaluate our method in this section. For each event log, 10%, 20%, 30%, and 40% of the activity labels were randomly deleted to create missing activity labels. For "Hospital Billing" and "BPI Challenge 2012" logs, instead of deleting a small number of activity labels, a large proportion of activity labels were removed. The evaluation results are presented in Table 8 where all success rates were averaged by 10 repeats. The success rates of our method are much higher than using only prefix or suffix sequences to predict the missing activity labels, whether the additional attributes (i.e., resources) are used or not. The results prove that, when repairing missing activity labels of events, using the information from both their prefix and suffix sequences can significantly improve the success rates.

It is also interesting to notice that, except for the production process log, although using additional attributes (i.e., resources) can improve the success rates, applying our method without additional attributes can also obtain high success rates. Our results indicate that our method can be used to repair missing activity labels when additional attributes are not available in event logs.

Figure 8 shows the success rates of our method each round separately in different event logs when different numbers of activity labels are missing. Overall, the success rates of our method are stable in different rounds without huge fluctuations. Although the success rates become lower when more activity labels are missing, the drops are slight. On average, the success rate drops by only 0.04 when the number of missing activity labels increases from 10% to 40%. These results suggest that our method can accurately repair event logs with a large proportion of missing activity labels.

Table 9 shows the average time for our method to repair the missing activity labels for each event log. The time includes the time to train the prediction model and repair the missing activity labels. All our code was run on Google Colab Pro with Nvidia Tesla P100

GPU. It is interesting to notice that our method takes less time when there are more missing activity labels. The datasets for training the model become smaller when there are more missing activity labels.

Finally, to show how different process discovery algorithms can handle the repaired event logs, we selected the event logs the with highest success rates (Hospital Billing) and event logs with lowest success rates (Sepsis Log) for additional experiments. We used their original two event logs, the event logs with missing activity labels (20 event logs in total) and the repaired event logs (20 event logs in total) to discover process models, and we used all the discovered process models to perform conformance checking against the original event logs. The conformance checking was based on Variants.ALIGNMENT_BASED (fitness) and Variants.ALIGN_ETCONFORMANCE (precision) tools in the PM4PY [41] framework. Table 10 shows the average f-scores of the discovered process models using the inductive miner infrequent algorithm [21]. Table 11 shows the average f-scores of the discovered process models using the split miner algorithm [11]. Default settings were applied to all process discovery algorithms. A higher f-score indicates the discovered process model can represent the process behaviors of the original event log more accurately. As shown in both tables, the average f-scores for the event logs with missing activity labels and repaired event logs drop when there are more missing activity labels. However, the f-scores for process models of repaired event logs drop slower and remain closer to the process models of the original event logs. The results indicate that the proposed method can help with improving the quality of discovered process models when there are missing activity labels in event logs.

Table 8. Further analysis of our method.

Dataset	Number of Missing Activity Labels	Prefix Only	Suffix Only	Our Method (Without Resources)	Our Method
Hospital Billing	10% (41,223)	0.878 ± 0.002	0.885 ± 0.002	0.986 ± 0.001	$\mathbf{0.990 \pm 0.001}$
	20% (82,447)	0.865 ± 0.003	0.880 ± 0.002	0.983 ± 0.002	$\mathbf{0.988 \pm 0.001}$
	30% (123,670)	0.852 ± 0.002	0.875 ± 0.003	0.980 ± 0.001	$\mathbf{0.985 \pm 0.001}$
	40% (164,894)	0.836 ± 0.002	0.868 ± 0.001	0.976 ± 0.001	$\mathbf{0.983 \pm 0.001}$
BPI Chanllenge 2012	10% (26,220)	0.828 ± 0.096	0.809 ± 0.091	0.975 ± 0.046	$\mathbf{0.983 \pm 0.002}$
	20% (52,440)	0.823 ± 0.002	0.795 ± 0.002	0.968 ± 0.002	$\mathbf{0.971 \pm 0.002}$
	30% (78,660)	0.802 ± 0.003	0.768 ± 0.004	0.952 ± 0.004	$\mathbf{0.957 \pm 0.003}$
	40% (104,880)	0.778 ± 0.002	0.735 ± 0.004	0.932 ± 0.002	$\mathbf{0.942 \pm 0.002}$
Sepsis Log	10% (1521)	0.634 ± 0.020	0.588 ± 0.017	0.819 ± 0.022	$\mathbf{0.888 \pm 0.016}$
	20% (3042)	0.602 ± 0.016	0.545 ± 0.013	0.763 ± 0.009	$\mathbf{0.846 \pm 0.016}$
	30% (4564)	0.575 ± 0.017	0.515 ± 0.010	0.712 ± 0.017	$\mathbf{0.812 \pm 0.012}$
	40% (6085)	0.547 ± 0.012	0.481 ± 0.011	0.660 ± 0.012	$\mathbf{0.779 \pm 0.012}$
Helpdesk	10% (2134)	0.822 ± 0.014	0.868 ± 0.009	0.937 ± 0.009	$\mathbf{0.943 \pm 0.013}$
	20% (4269)	0.807 ± 0.014	0.861 ± 0.008	0.936 ± 0.007	$\mathbf{0.941 \pm 0.005}$
	30% (6404)	0.795 ± 0.008	0.853 ± 0.006	0.929 ± 0.007	$\mathbf{0.934 \pm 0.003}$
	40% (8539)	0.786 ± 0.005	0.847 ± 0.008	0.922 ± 0.005	$\mathbf{0.923 \pm 0.005}$
BPIC 2013 Incidents	10% (6553)	0.662 ± 0.009	0.712 ± 0.007	0.831 ± 0.006	$\mathbf{0.873 \pm 0.007}$
	20% (13,106)	0.650 ± 0.007	0.703 ± 0.007	0.823 ± 0.003	$\mathbf{0.864 \pm 0.004}$
	30% (19,659)	0.634 ± 0.005	0.690 ± 0.005	0.816 ± 0.006	$\mathbf{0.855 \pm 0.005}$
	40% (26,213)	0.617 ± 0.008	0.679 ± 0.004	0.807 ± 0.005	$\mathbf{0.843 \pm 0.006}$
Production Process Log	10% (454)	0.481 ± 0.041	0.520 ± 0.053	0.568 ± 0.045	$\mathbf{0.944 \pm 0.014}$
	20% (908)	0.461 ± 0.018	0.507 ± 0.034	0.546 ± 0.029	$\mathbf{0.938 \pm 0.017}$
	30% (1363)	0.430 ± 0.020	0.478 ± 0.027	0.514 ± 0.020	$\mathbf{0.936 \pm 0.013}$
	40% (1817)	0.416 ± 0.025	0.461 ± 0.022	0.500 ± 0.025	$\mathbf{0.932 \pm 0.010}$

Table 9. Average time used to repair each log.

Dataset	Number of Missing Activity Labels	Average Time Used by Our Method to Repair Each Log
Hospital Billing	10% (41,223) 20% (82,447) 30% (123,670) 40% (164,894)	2190.91 s 1463.12 s 1217.85 s 1093.86 s
BPI Challenge 2012	10% (26,220) 20% (52,440) 30% (78,660) 40% (104,880)	1772.21 s 1109.62 s 1000.51 s 899.21 s
Sepsis Log	10% (1521) 20% (3042) 30% (4564) 40% (6085)	140.69 s 89.99 s 75.65 s 69.66 s
Helpdesk	10% (2134) 20% (4269) 30% (6404) 40% (8539)	140.40 s 89.33 s 80.30 s 72.96 s
BPIC 2013 Incidents	10% (6553) 20% (13,106) 30% (19,659) 40% (26,213)	218.59 s 170.65 s 157.66 s 150.10 s
Production Process Log	10% (454) 20% (908) 30% (1363) 40% (1817)	40.40 s 36.38 s 32.52 s 29.56 s

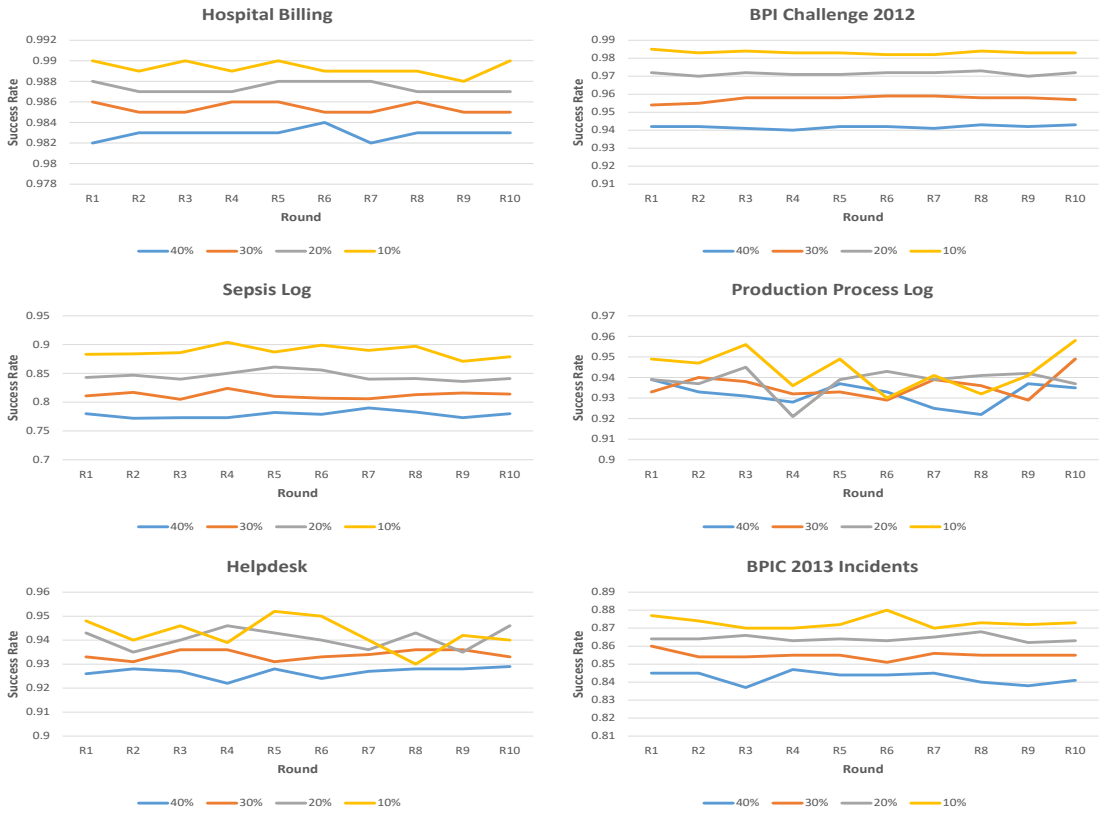

Figure 8. Successful rates of our method on different datasets in different rounds.

Table 10. F-scores of process models discovered by the inductive miner (infrequent) from the original, problem and repaired event logs. All conformance checking was conducted between discovered process models and the original event logs.

Dataset	Number of Missing Activity Labels	F-Score (the Original Log)	Average F-Score (the Logs with Missing Activity Labels)	Average F-Score (the Repaired Logs)
Hospital Billing	10% (41,223)	0.75	0.67	0.74
	20% (82,447)	0.75	0.59	0.71
	30% (123,670)	0.75	0.57	0.71
	40% (164,894)	0.75	0.55	0.70
Sepsis Log	10% (26,220)	0.77	0.62	0.72
	20% (52,440)	0.77	0.61	0.68
	30% (78,660)	0.77	0.59	0.66
	40% (104,880)	0.77	0.58	0.65

Table 11. F-scores of process models discovered by the split miner from the original, problem and repaired event logs. All conformance checking was conducted between discovered process models and the original event logs.

Dataset	Number of Missing Activity Labels	F-Score (the Original Log)	Average F-Score (the Logs with Missing Activity Labels)	Average F-Score (the Repaired Logs)
Hospital Billing	10% (41,223)	0.95	0.86	0.94
	20% (82,447)	0.95	0.85	0.94
	30% (123,670)	0.95	0.81	0.93
	40% (164,894)	0.95	0.80	0.93
Sepsis Log	10% (26,220)	0.86	0.76	0.80
	20% (52,440)	0.86	0.74	0.78
	30% (78,660)	0.86	0.72	0.77
	40% (104,880)	0.86	0.68	0.76

6. Conclusions

In this paper, we proposed a deep learning method to repair missing activity labels in event logs. The method was inspired by recent research papers in the field of process mining that designed artificial neural network models to predict the next activities in ongoing traces. Different from algorithms that predict next activities, to repair the missing activity labels of events, our method uses both their prefix and suffix sequences. The success rates of our method are much higher compared to existing methods in terms of repairing missing activity labels. Additional attributes in the event log can also be utilized to improve the success rates of our method.

Comparing with existing methods, our method can accurately repair the missing activity labels when a large portion of activity labels are missing. In addition, a high success rate can be achieved when only control-flow information is provided to our method, which indicates that our method can be applied to a wide range of event logs. However, since the proposed method uses neural networks to predict missing activity labels, there can be higher requirements for computer hardware to execute the method. For example, a GPU could be required to achieve good performance. In addition, the method requires many parameters from users, which may impact the performance of the method. A future study on how these parameters can impact the proposed method is needed.

It has to be noted that, like other methods used to repair missing activity labels in event logs, such as those found in [7–9], we assumed that we know the exact locations of the missing values. Although our method cannot be applied directly to event logs when the locations of missing values are unknown, our method can be used together with anomaly detection algorithms, such as [42]. For example, a missing event may exist between two events when the direct succession relation between two consecutive events is identified as an anomaly.

Future work includes the following aspects: Firstly, besides repairing activity labels, we also plan to expand our method to repair other attributes in the event logs (e.g., resources and timestamps). Secondly, besides resources, we also aim at evaluating our method using

other additional attributes. Thirdly, we plan to investigate how different parameters settings can impact the method. Finally, we plan to investigate the feasibility of applying our method in online settings.

Author Contributions: Conceptualization and methodology, Y.L., Q.C., S.K.P.; Development, Y.L.; Validation, Y.L., Q.C., S.K.P.; Writing—original draft preparation: Y.L; Writing—review and editing: Y.L., Q.C., S.K.P.; Supervision: S.K.P. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets used to evaluate the proposed method are publicly-available; please refer to notes for links to access the datasets.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Dumas, M.; La Rosa, M.; Mendling, J.; Reijers, H.A. *Fundamentals of Business Process Management*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 1.
2. Van der Aalst, W. *Process Mining*; Springer: Berlin/Heidelberg, Germany, 2016.
3. Cai, L.; Zhu, Y. The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. *Data Sci. J.* **2015**, *14*, 2. [CrossRef]
4. Suriadi, S.; Andrews, R.; ter Hofstede, A.; Wynn, M. Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs. *Inf. Syst.* **2017**, *64*, 132–150. [CrossRef]
5. Jagadeesh Chandra Bose, R.; Mans, R.; van der Aalst, W.M. Wanna improve process mining results?: it's high time we consider data quality issues seriously. *BPM Rep.* **2013**, *1302*, 127–134.
6. Rogge-Solti, A.; Mans, R.S.; van der Aalst, W.M.; Weske, M. *Repairing Event Logs Using Stochastic Process Models*; Universitätsverlag Potsdam: Potsdam, Germany, 2013; Volume 78.
7. Xu, J.; Liu, J. A profile clustering based event logs repairing approach for process mining. *IEEE Access* **2019**, *7*, 17872–17881. [CrossRef]
8. Liu, J.; Xu, J.; Zhang, R.; Reiff-Marganiec, S. A repairing missing activities approach with succession relation for event logs. *Knowl. Inf. Syst.* **2021**, *63*, 477–495. [CrossRef]
9. Sim, S.; Bae, H.; Choi, Y. Likelihood-based multiple imputation by event chain methodology for repair of imperfect event logs with missing data. In Proceedings of the 2019 International Conference on Process Mining (ICPM), IEEE, Aachen, Germany, 24–26 June 2019; pp. 9–16.
10. Song, W.; Xia, X.; Jacobsen, H.A.; Zhang, P.; Hu, H. Heuristic recovery of missing events in process logs. In Proceedings of the 2015 IEEE International Conference on Web Services, IEEE, New York, NY, USA, 27 June–2 July 2015; pp. 105–112.
11. Augusto, A.; Conforti, R.; Dumas, M.; La Rosa, M.; Polyvyanyy, A. Split miner: automated discovery of accurate and simple business process models from event logs. *Knowl. Inf. Syst.* **2019**, *59*, 251–284. [CrossRef]
12. Van der Aalst, W.; Weijters, T.; Maruster, L. Workflow mining: Discovering process models from event logs. *IEEE Trans. Knowl. Data Eng.* **2004**, *16*, 1128–1142. [CrossRef]
13. De Medeiros, A.A.; van Dongen, B.F.; Van der Aalst, W.M.; Weijters, A. *Process Mining: Extending the α-Algorithm to Mine Short Loops*; Technische Universiteit Eindhoven: Eindhoven, The Netherlands, 2004.
14. Guo, Q.; Wen, L.; Wang, J.; Yan, Z.; Philip, S.Y. Mining invisible tasks in non-free-choice constructs. In Proceedings of the International Conference on Business Process Management, Rio de Janeiro, Brazil, 18–22 September 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 109–125.
15. Wen, L.; Wang, J.; van der Aalst, W.M.; Huang, B.; Sun, J. Mining process models with prime invisible tasks. *Data Knowl. Eng.* **2010**, *69*, 999–1021. [CrossRef]
16. Wen, L.; Van Der Aalst, W.M.; Wang, J.; Sun, J. Mining process models with non-free-choice constructs. *Data Min. Knowl. Discov.* **2007**, *15*, 145–180. [CrossRef]
17. Weijters, A.; Ribeiro, J.T.S. Flexible heuristics miner (FHM). In Proceedings of the 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, Paris, France, 11–15 April 2011; pp. 310–317.
18. Weijters, A.; van Der Aalst, W.M.; De Medeiros, A.A. Process mining with the heuristics miner-algorithm. *Tech. Univ. Eindh. Tech. Rep. WP* **2006**, *166*, 1–34.
19. vanden Broucke, S.K.; De Weerdt, J. Fodina: a robust and flexible heuristic process discovery technique. *Decis. Support Syst.* **2017**, *100*, 109–118. [CrossRef]

20. Leemans, S.J.; Fahland, D.; van der Aalst, W.M. Discovering block-structured process models from event logs-a constructive approach. In Proceedings of the International Conference on Applications and Theory of Petri Nets and Concurrency, Milan, Italy, 24–28 June 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 311–329.
21. Leemans, S.J.; Fahland, D.; van der Aalst, W.M. Discovering block-structured process models from event logs containing infrequent behaviour. In Proceedings of the International Conference on Business Process Management, Beijing, China, 26–30 August 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 66–78.
22. Leemans, S.J.; Fahland, D.; van der Aalst, W.M. Discovering block-structured process models from incomplete event logs. In Proceedings of the International Conference on Applications and Theory of Petri Nets and Concurrency, Tunis, Tunisia, 23–27 June 2014; Springer: Berlin/Heidelberg, Germany, 2014; pp. 91–110.
23. Leemans, S.J.; Fahland, D.; van der Aalst, W.M. Scalable process discovery and conformance checking. *Softw. Syst. Model.* **2018**, *17*, 599–631. [CrossRef] [PubMed]
24. Leemans, S.J.; Fahland, D.; van der Aalst, W.M. Using life cycle information in process discovery. In Proceedings of the International Conference on Business Process Management, Rio de Janeiro, Brazil, 18–22 September 2016; Springer: Berlin/Heidelberg, Germany, 2016; pp. 204–217.
25. Leemans, M.; van der Aalst, W.M. Modeling and discovering cancelation behavior. In Proceedings of the OTM Confederated International Conferences "On the Move to Meaningful Internet Systems", Rhodes, Greece, 23–27 October 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 93–113.
26. Lu, Y.; Chen, Q.; Poon, S. A Novel Approach to Discover Switch Behaviours in Process Mining. In Proceedings of the International Conference on Process Mining, Padua, Italy, 4–9 October 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 57–68.
27. Van der Aalst, W.M.; De Medeiros, A.A.; Weijters, A.J. Genetic process mining. In Proceedings of the International Conference on Application and Theory of Petri Nets, Miami, FL, USA, 20–25 June 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 48–69.
28. Buijs, J.C.; van Dongen, B.F.; van der Aalst, W.M. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *Int. J. Coop. Inf. Syst.* **2014**, *23*, 1440001. [CrossRef]
29. Van der Werf, J.M.E.; van Dongen, B.F.; Hurkens, C.A.; Serebrenik, A. Process discovery using integer linear programming. In Proceedings of the International Conference on Applications and Theory of Petri Nets, Xi'an, China, 23–27 June 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 368–387.
30. Sommers, D.; Menkovski, V.; Fahland, D. Process discovery using graph neural networks. In Proceedings of the 2021 3rd International Conference on Process Mining (ICPM), IEEE, Eindhoven, The Netherlands, 31 October–4 November 2021; pp. 40–47.
31. Horita, H.; Kurihashi, Y.; Miyamori, N. Extraction of missing tendency using decision tree learning in business process event log. *Data* **2020**, *5*, 82. [CrossRef]
32. Tax, N.; Verenich, I.; La Rosa, M.; Dumas, M. Predictive business process monitoring with LSTM neural networks. In Proceedings of the International Conference on Advanced Information Systems Engineering, Essen, Germany, 12–16 June 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 477–492.
33. Camargo, M.; Dumas, M.; González-Rojas, O. Learning accurate LSTM models of business processes. In Proceedings of the International Conference on Business Process Management, Vienna, Austria, 1–6 September 2019; Springer: Berlin/Heidelberg, Germany, 2019; pp. 286–302.
34. Pasquadibisceglie, V.; Appice, A.; Castellano, G.; Malerba, D. A multi-view deep learning approach for predictive business process monitoring. *IEEE Trans. Serv. Comput.* **2021**. [CrossRef]
35. Lin, L.; Wen, L.; Wang, J. Mm-pred: A deep predictive model for multi-attribute event sequence. In Proceedings of the 2019 SIAM International Conference on Data Mining, SIAM, Calgary, AB, Canada, 2–4 May 2019; pp. 118–126.
36. Taymouri, F.; La Rosa, M.; Erfani, S.; Bozorgi, Z.D.; Verenich, I. Predictive business process monitoring via generative adversarial nets: The case of next event prediction. In Proceedings of the International Conference on Business Process Management, Seville, Spain, 13–18 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 237–256.
37. Pasquadibisceglie, V.; Appice, A.; Castellano, G.; Malerba, D. Using convolutional neural networks for predictive process analytics. In Proceedings of the 2019 International Conference on Process Mining (ICPM), IEEE, Aachen, Germany, 24–26 June 2019; pp. 129–136.
38. Mehdiyev, N.; Evermann, J.; Fettke, P. A novel business process prediction model using a deep learning method. *Bus. Inf. Syst. Eng.* **2020**, *62*, 143–157. [CrossRef]
39. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
40. Guo, C.; Berkhahn, F. Entity embeddings of categorical variables. *arXiv* **2016**, arXiv:1604.06737.
41. Berti, A.; Van Zelst, S. J.; van der Aalst, W. Process Mining for Python (PM4Py): Bridging the Gap Between Process- and Data Science. *arXiv* **2019**, arXiv:1905.06169.
42. Huo, S.; Völzer, H.; Reddy, P.; Agarwal, P.; Isahagian, V.; Muthusamy, V. *Graph Autoencoders for Business Process Anomaly Detection*; Polyvyanyy, A., Wynn, M.T., Van Looy, A., Reichert, M., Eds.; Business Process Management; Springer International Publishing: Cham, Switzerland, 2021; pp. 417–433.