Predicting Outcomes of Business Process Executions Based on LSTM Neural Networks and Attention Mechanism

Jiaojiao Wang
Hangzhou Dianzi University https://orcid.org/0000-0003-2294-5577
Dongjin Yu (yudj@hdu.edu.cn)
Hangzhou Dianzi University
Chengfei Liu
Swinburne University of Technology - Hawthorn Campus: Swinburne University of Technology
Xiaoxiao Sun
Hangzhou Dianzi University

Research Article

Keywords: outcome prediction, sequence classication, business process monitoring, LSTM, attention mechanism

Posted Date: April 26th, 2021

DOI: https://doi.org/10.21203/rs.3.rs-260970/v1

Noname manuscript No.
(will be inserted by the editor)

Predicting Outcomes of Business Process Executions based on LSTM Neural Networks and Attention Mechanism

Jiaojiao Wang · Dongjin Yu* · Chengfei Liu · Xiaoxiao Sun

Received: date / Accepted: date

Abstract To effectively predict the outcome of an on-going process instance helps make an early decision, which plays an important role of so-called predictive process monitoring. Existing methods in this field are tailor-made for some empirical operations such as the prefix extraction, clustering and encoding, leading that their relative accuracy is highly sensitive to the dataset. Moreover, they have limitations in real-time prediction applications due to the lengthy prediction time. Since Long Short-term Memory (LSTM) neural network provides a high precision in the prediction of sequential data in several areas, this paper investigates LSTM and its enhancements and proposes three different approaches to build more effective and efficient models for outcome prediction. The first move on enhancement is that we combine the original LSTM network from two directions, forward and backward, to capture more features from the completed cases. The second move on enhancement is that we add attention mechanism after extracting features in the hidden layer of LSTM network to distinct them from their attention weight. A series of extensive experiments are evaluated on twelve real datasets when comparing with other approaches. The results show that our approaches outperform the state-of-the-art ones in terms of prediction effectiveness and time performance.

Keywords outcome prediction · sequence classification · business process monitoring · LSTM · attention mechanism

Jiaojiao Wang
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
Institute of Intelligent Media Technology, Communication University of Zhejiang, Hangzhou, China
E-mail: jjwang_hdu@126.com

Dongjin Yu* (Corresponding author.)
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
E-mail: yudj@hdu.edu.cn

Chengfei Liu
Department of Computer Science and Software Engineering, Swinburne University of Technology, Melbourne, Australia
E-mail: cliu@swin.edu.au

Xiaoxiao Sun
School of Computer Science and Technology, Hangzhou Dianzi University, Hangzhou, China
E-mail: sunxiaoxiao@hdu.edu.cn

1 Introduction

Process mining refers to discover, detect and improve the behavior patterns from event logs produced by Process-aware Information Systems (PAIS) during business process execution (Van der Aalst 2016). Such logs involve some knowledge that can be used to predict the new process execution. For example, on the basis of these historical event logs, we can predict the next activity to be executed (Pauwels and Calders 2020; Tama et al. 2020; Mehdiyev et al. 2020; Tax et al. 2017), the remaining execution time (Sun et al. 2020; Verenich et al. 2019; Tax et al. 2017; Rogge-Solti and Weske 2013), and the final outcome (*positive* vs. *negative*) (Kratsch et al. 2020; Maggi et al. 2014; Metzger et al. 2015) of an on-going process instance (case). The predictive tasks that can monitor the process execution in real-time by predicting its execution information belong to Predictive Process Monitoring (PPM) techniques. In particular, at the process execution (monitoring) stage of a Business Process Management (BPM) lifecycle, to accurately predict the outcome of an on-going process instance can help make decisions at the right time. In this paper, we focus on how to construct a

predictive model to make outcome predictions for these running cases based on their executed events as well as the mined knowledge from historical event log. Usually, the final outcome of a process can be determined by its business goal such as the result of *normal*, *deviant* for a production process, or the result of *accepted*, *declined*, *canceled* for an application process. As for the multi-class outcome prediction task, we can translate it into multiple binary classification tasks (Teinemaa et al. 2019, 2018) so as to compare with other researches (Maggi et al. 2014; Metzger et al. 2015).

Up to now, many approaches based on traditional machine learning techniques have been proposed for the outcome prediction of business processes (Teinemaa et al. 2019, 2018). Among them, the most representative methods are Random Forest (RF) and XGBoost-based. As for a business process, their goal is to construct one or more predictive models (i.e. classifiers) based on the hidden knowledge involved in historical process log, and then predict the outcome of a running case. Here, the running case means that a new process instance is being executed, but not yet completed. Specifically, such methods can be divided into two phases. During the *offline* phase, some customized operations, such as the prefix extraction, clustering, encoding, and classification algorithms, need to be selected empirically for constructing one or more classification models, whereas during the *online* phase, it is necessary to determine which cluster a running case belongs to and then select the corresponding coding strategy and classification model to predict its outcome. On this occasion, an inappropriate selection during the offline phase may lead to unstable performance of the prediction model. In addition, during the online prediction phase, the operations for determining the right predictive model require long time, which limits the efficiency of these methods.

With the development of neural networks, the construction of prediction model becomes more and more intelligent and the effectiveness as well as efficiency of prediction has also been improved significantly. Especially, Recurrent Neural Network (RNN) (Liu et al. 2016) and its improvements have demonstrated their capabilities of classification prediction. To obtain the long context dependency features in a sequence, Long Short-term Memory (LSTM) network (Hochreiter and Schmidhuber 1997; Graves 2012) has been proposed on the basis of RNN and proven to be more efficient in similar predictive tasks (Li et al. 2017; Zhou et al. 2016). Moreover, to enrich the features from two directions (forward and backward), an improvement of Bidirectional LSTM network (BiLSTM) has also been provided for such prediction tasks. Besides, the attention mechanism that originates from human visual system (Rensink 2000) has been applied in some researches of neural networks (Bahdanau et al. 2015; Yang et al. 2016). This mechanism aims at assigning more weight to the most relevant content of the input data instead of focusing on all available information, and is often integrated with RNN, LSTM, BiLSTM, and so on (Li et al. 2017; Zhou et al. 2016; Shang et al. 2015).

Actually, if considering a running case that consists of a sequence of events, we can take the outcome prediction task as a sequential event prediction problem. Motivated by these observations, we leverage neural networks, especially the bidirectional feature extraction and attention mechanism, for the outcome prediction problem, and propose three approaches of Bi-LSTM, Att-LSTM, Att-Bi-LSTM based on the original LSTM network in this paper. Compared with the traditional machine learning techniques, our approaches can automatically build a predictive model without prior empirical knowledge by extracting the features that affect the process outcome from historical process instances. Therefore, our approaches have significant advantages in effectiveness and efficiency in real-time prediction. Among our approaches, the difference lies in the way of feature extraction. The Bi-LSTM approach adopts a single bidirectional feature extraction based on LSTM network, the Att-LSTM approach extracts features based on LSTM network with attention mechanism, and Att-Bi-LSTM approach adopts a hybrid feature extraction of bidirectional LSTM network and attention mechanism. Accordingly, the performance of prediction models that are constructed according to different ways of feature extraction is different. Therefore, in this paper, we study on the prediction effectiveness, time performance, and stability of the above different approaches and compare them with other traditional techniques.

The preliminary work of this paper has been published in (Wang et al. 2019). The major extension includes model construction, experiment, and discussion. Regarding the construction of predictive model, we present two other approaches, Bi-LSTM and Att-LSTM, based on the two enhancements of bidirectional feature extraction and attention mechanism respectively. Moreover, we describe a complete process for the outcome prediction of business processes, including how to train a prediction model offline and how to predict the final outcome of an on-going case online. As for the experiment, we introduce a new indicator of prediction stability to evaluate the performance of our proposed approaches and then compare the influence of bidirectional feature extraction and attention mechanism for the outcome prediction. Finally, we further discuss the

threats to validity of our study in experimental evaluation.

In summary, the contributions of this paper are as follows.

- We propose three approaches, Bi-LSTM, Att-LSTM, and Att-Bi-LSTM, to predict the outcome of a running case, in which we show how to automatically build an effective and efficient prediction model by using single bidirectional feature extraction, single attention mechanism, and a mixture of them on the basis of LSTM network.
- We introduce an indicator of temporal stability in terms of prediction effectiveness to evaluate our approaches and compare them with other approaches based on traditional machine learning techniques.
- We conduct a series of extensive experiments to investigate the effectiveness of the prediction model construction as well as the performance of online outcome prediction.

The rest of paper is organized as follows. In section 2, we briefly discuss the related studies on outcome prediction and deep learning techniques. In section 3, we introduce some basic definitions and give the problem statement. Then the detailed solution and the evaluation of its effectiveness and efficiency are demonstrated in Sections 4 and 5 respectively. Finally, Section 6 discusses the threats to validity of our study, and Section 7 concludes the paper and outlines future work directions.

2 Related Work

2.1 Outcome Prediction of Business processes

The main issue to be addressed in this paper is to predict the outcome class of an on-going case in real time. In fact, the key is to train a prediction model based on the historical cases labeled as the outcome class firstly, and then predict the outcome class of a running case based on this model. Consequently, the solutions can be generally divided into two phases, one for the offline phase where the prediction model is trained, the other for the online phase where the outcome class of a running case can be forecasted based on its executed events and the trained prediction model. In terms of the traditional machine learning techniques, the former phase usually includes four main operations such as the prefix extraction for all historical cases, clustering, encoding, and training of the prediction models. Based on the clusters (buckets) and classifiers constructed in the offline phase, the outcome class of a running case can be predicted at online phase. Until now, many methods have been proposed regarding these operations. For example, Verenich et al. (2016) presented a clustering-classification method, where the extracted prefixes of historical cases are clustered firstly and then a classifier is established for each cluster to predict the outcome. Di Francescomarino et al. (2016) developed a framework by constructing classifier (i.e., prediction model) offline (where each cluster is for each length of prefixes of historical cases) and making prediction online (where a running case is determined to a cluster firstly and then applied with the corresponding classifier). Similarly, De Leoni et al. (2016) implemented a generic framework to obtain a decision tree or a regression tree by correlating business process characteristics and then use this tree to cluster event logs in PPM.

In terms of feature encoding, many methods are proposed to predict the most likely outcome of the ongoing case by extracting features from the historical cases in an event log rather than the events of each case. For example, Leontjeva et al. (2016) proposed two feature encoding methods, one is based on the index of events in a case, and the other combining the index-based encoding method with hidden Markov models (HMMs). Similarly, Verenich et al. (2016) used the index-based encoding method. However, their method can only be used in all traces with the same length because the length of the encoded feature vector increases with each execution event. Besides, the commonly used final state encoding technique based on the last available data attribute after each execution of the activity in case (Maggi et al. 2014; Di Francescomarino et al. 2016), has been proved disadvantageous because it ignores the information of all events that occurred in the past. In addition, Senderovich et al. (2017) investigated a feature encoding method for process cases, which relies on a bi-dimensional state space representation of intra-case dependencies and inter-case dependencies in predicting process monitoring.

In terms of constructing classifier(s), the commonly used methods mainly include decision tree (DT) (De Leoni et al. 2016), random forest (RF) (Bishop 2006, support vector machine (SVM) and gradient boosted trees (XGBoost) (Leontjeva et al. 2016; Friedman 2001). Among them, DT can explain the results well, RF has higher prediction accuracy, while XGBoost usually achieves the higher accuracy than RF. Besides, Teinemaa et al. (2016) proposed another method, which can train a classifier for each possible prediction point such as after the execution of a certain event. Likewise, Conforti et al. (2015) developed an approach to obtain a multi-classification model by using a decision tree method to train a classifier at each decision point of a process. Some process cases include not only the activities them-

selves but also the documents created by these activities. To address the prediction for such this case, Lakshmanan et al. (2010) adopted an ant-colony optimization (ACO) based method to extract a probabilistic activity graph from process traces, and then used this graph to identify key decision points in a given process.

The above mentioned classification approaches and some others such as clustering-based and regression-based (Kang et al. 2012) usually require complex operations to build classification models. Fortunately, deep learning techniques can usually solve sequential data prediction tasks effectively and efficiently by automatically constructing a classifier from historical data. Recently, some approaches based on deep learning techniques have been proposed for the next activity prediction and remaining time prediction tasks in PPM. For example, Tax et al. (2017) used LSTM network to predict the next activity to be executed and its possible timestamp in terms of a running case, which is proven to outperform existing techniques. By viewing the next event prediction task as a multi-classification problem, Mehdiyev et al. (2020) presented a multi-stage deep learning method based on a deep feed forward multilayer neural network. However, to the best of our knowledge, the similar research for predicting the outcome of a process has not yet been proposed. As indicated in (Kratsch et al. 2020; Teinemaa et al. 2019), it is very meaningful to study LSTM for process outcome prediction. Therefore, in this paper, we are dedicated to proposing the approaches based on neural networks for the outcome prediction problem.

2.2 Recurrent Neural Networks and Attention Mechanism

As for an on-going case, its final outcome can be usually predicted based on the executed part of this case and a classification model trained from the completed cases of the event log. Here, each completed historical case consists of a sequence of events and the final outcome of this case can be determined by some features extracted from the execution of these events. In order to obtain a classifier for outcome prediction, these features need to be captured automatically from historical cases. Up to now, there has been no research using neural networks to predict the outcome of a running process case. Fortunately, the essence of outcome-oriented prediction is similar to the text relation classification task in the field of Natural Language Processing (NLP), in which each text consists of a sequence of words and the features extracted from these words can determine the text relation. Regarding the problem of learning hidden features automatically, some Deep Neural Networks (DNN) techniques have been proved to be employed successfully. For instance, in terms of the text relation classification, Zeng et al. (2014) utilized Convolutional Neural Networks (CNN) to extract lexical and sentence level features without complex pre-processing operations. However, considering CNN networks are not suitable for learning long-distance dependency information, Zhang and Wang (2018) employed bidirectional RNN to learn patterns from raw data by extracting the information from the past and future context. To counter the gradient vanishing of RNN networks and the limitation of context scope, LSTM (Hochreiter and Schmidhuber 1997) was introduced for the text relation classification and then some corresponding variants were also proposed. As demonstrated in (Li et al. 2017; Zhou et al. 2016), bidirectional LSTM can not only learn hidden features with both the past and future context information, but also keep long-distance dependency memory. Therefore, in this paper, we focus on the study of bidirectional feature extraction based on LSTM networks for outcome prediction task.

In addition, in some neural networks, the attention mechanism has proven to be able to optimize the extracted features in different fields. For example, in question answering system, Nie et al. (2017) applied the attention mechanism in bidirectional LSTM network to optimize the features by focusing on a certain part of the candidate answer for a question. The experiment results showed that the proposed model outperforms the existing approaches. Similarly, in machine translation task, the attention mechanism is first used in an encoder-decoder framework to optimize the features by using this network to revisit all parts of a source sentence, instead of encoding all features of this sentence (Bahdanau et al. 2015). Especially in the text relation classification, Zhou et al. (2016) employed the attention mechanism in bidirectional LSTM to optimize the features by assigning more weight to the most important semantic features automatically. Likewise, on the basis of CNN, Lin et al. (2016) used a sentence-level attention mechanism to optimize features by reducing the weights of multiple noisy instances dynamically for distant supervised relation extraction. The experimental results showed that the model can achieve significant and consistent improvements as compared with the state-of-the-art methods. Besides, the attention-based models have also been applied to various areas such as the semantic relation extraction (Geng et al. 2020), image classification (Mnih et al. 2014; Zhu et al. 2020), speech recognition (Chorowski et al. 2015), and image caption generation (Xu et al. 2015). In a word, the attention mechanism is effective at extracting the most

distinguishing features. Hence, in this paper, we focus on the application of attention mechanism for the outcome prediction so as to identify the features that have a decisive effect on the outcome of a case.

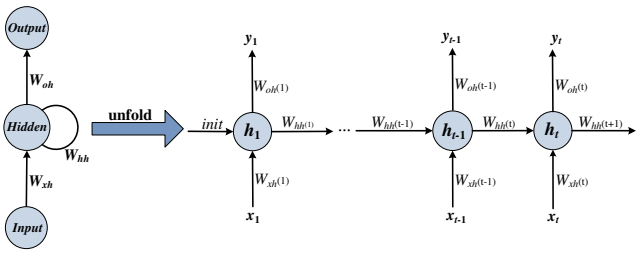
3 Preliminaries and Problem Statement

For ease of understanding, in this section, we will give the background knowledge and some basic concepts as well as the detailed statements regarding the outcome prediction problem.

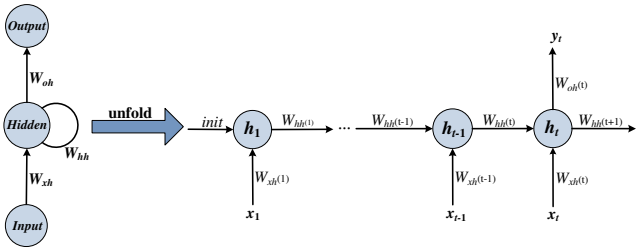
3.1 Recurrent Neural Networks and Attention Mechanism

Artificial Neural Network (ANN) is a mathematical model that simulates the processing mechanism of the human brain's nervous system for complex information based on the basic principles of neural network in biology. It is characterized by parallel distributed processing ability, high fault tolerance, intelligence and self-learning ability (Hopfield 1982). Especially with the development of deep learning technology, neural networks are widely used in the field of predictive classification. In general, a neural network consists of a layer of input cells, multiple layers of hidden cells, and a layer of output cells. Cells in each layer are connected by weighted connections to cells in previous and following layers in various forms, depending on the architecture of neural networks. The output of each cell represents a specific "activation" function of the weighted sum of its input.

As for the outcome prediction problem, an on-going case consisting of a sequence of executed events can be viewed as the input of a neural network, while the outcome class label can be viewed as the output of this neural network. In this way, we can construct a neural network with specific structure, called classifier, by training some executed traces with labeled outcome class in an event log. In general, neural networks with special structures can memorize information from input at early moment (viewing the order of sequential data as different moments). RNN is such a special neural network (Liu et al. 2016), which is composed of a series of repeating modules, where the repeating module is a single hidden layer with typical activation functions such as *tanh* and *sigmoid*. The general RNN is shown in Fig. 1(a) and its variant with multiple inputs and single output is shown in Fig. 1(b). The latter is more applicable to the outcome prediction problem. Here, x_t, y_t and h_t denote the values of the *input* layer, the *output* layer, and the *hidden* layer at moment t respectively. Besides, h_{t-1} records the state of the previous output



(a) a general RNN



(b) a variant RNN

Fig. 1 Two structures of a general RNN and its variant

at moment $t-1$. $W_{xh}(t)$, $W_{hh}(t)$, and $W_{oh}(t)$ denote the input-hidden weight matrix, the hidden-hidden weight matrix, and the hidden-output weight matrix respectively. The state h_t of hidden layer and the output y_t can be calculated by:

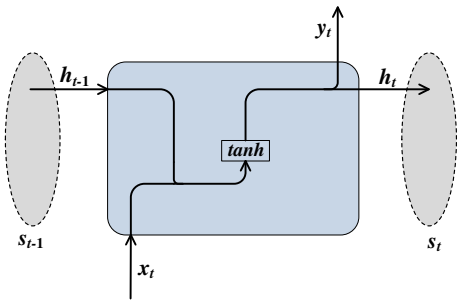
$$y_t = F_2(W_{oh}(t) \cdot h_t + b_o) \tag{1}$$

$$h_t = F_1(W_{xh}(t) \cdot x_t + W_{hh}(t-1) \cdot h_{t-1} + b_h) \tag{2}$$

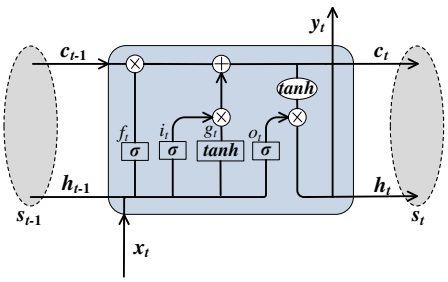
where F_1 and F_2 represent the activation functions in hidden layer and output layer, respectively. Usually, F_1 denotes *tanh* function and F_2 denotes *softmax* function. Based on these equations, we can find that the input information at early time will be lossy when transmitting to the output layer by $h_t = W_{hh}(t-1) \cdot h_{t-1}$. When the hidden state information h_0 of the start moment is transmitted to the moment t with $h_t = W_{hh}^t \cdot h_0$, the matrix W_{hh} needs to be multiplied many times. In this case, we can get $h_t = Q\Lambda^t Q^T \cdot h_0$ if W_{hh} can be eigenvalue decomposed. If the eigenvalue $\Lambda < 1$, the result of h_t diminishes to 0 (called *gradient diminishing*). If $\Lambda > 1$, it expands to infinity (called *gradient explosion*). Hence, RNN cannot maintain the memory of input that is longer or far away from the output.

As a variant of RNN, LSTM (Hochreiter and Schmidhuber 1997) can learn long-term dependencies by replacing the hidden layer with a memory cell c_t to tackle with the gradient vanishing issue. Fig. 2 shows a module of general RNNs and a memory cell for LSTM respectively. In this memory cell, there are three gates

including input gate i_t that determines how much information can flow into this cell, forget gate f_t determines how much information can be forgotten in this cell and output gate o_t determines how much information can be outputted from this cell. Each gate represents a way to control the data information.



(a) a module of general RNN



(b) a cell for Long Short-term Memory (LSTM)

Fig. 2 The neural unit of RNN and LSTM networks respectively

In Fig. 2(b), h_{t-1} and c_{t-1} denote the output and the state of the memory cell at $t-1$ moment, respectively. Based on them, we first take the input of x_t and h_{t-1} into consideration and determine what the information to throw away from the cell state with a *sigmoid* activation function (i.e., σ in Fig. 2(b)). Then, we need to determine what information to store in the cell state by the input gate and the updated cell state. As for the input gate, i_t can decide which state to be updated. Afterwards, a new candidate value g_t can be obtained by a *tanh* activation with the input x_t and h_{t-1}. Based on the i_t and g_t, we can update the old cell state c_{t-1} to a new cell state c_t according to the information to be forgotten and the new candidate values. Finally, the information to be outputted can be determined by running the previous information with a *tanh* activation and then putting the cell state c_i in a *sigmoid* activation. In this way, the LSTM units can keep the information of the previous state (i.e., c_{t-1} and h_{t-1}) and memorize the extracted features of the current data input of x_t. Thus, it can be used for sequential data prediction and usually achieves good performance.

Inspired by human cognition, the attention mechanism was first presented by Minh et al. and then applied to the image field (Mnih et al. 2014). It has a significant optimization effect on the traditional neural networks. In general, human attention refers to the fact that human beings generally do not regard all things as a whole but tend to selectively acquire some important parts of the observed things according to their needs when observing external things. Such a mechanism can greatly improve the efficiency and accuracy of visual information processing. Similarly, when people observe an image, they do not actually get the pixels of each position of the whole image at one time. Instead, they focus on specific parts of the image according to their needs. When learning from a part of an image, in order to simplify the task, the current state will process the pixels of the focused part that determined by previous learning and the current input image rather than all the pixels of this image. So the attention mechanism can extract more critical information and gain popularity in DNNs, especially for RNNs and CNNs (Wu et al. 2018). In general, the obtained features from the input of a neural network have different influence on the output. The attention mechanism is used to assign the different attention (weight) to all features (local parts) based on their different effect on the output and then comprise the global representation of these features. In this case, the higher the weight of a feature, the more important it is for the output. Therefore, using the attention mechanism in neural networks can make more accurate judgment by giving the networks with the ability to distinguish the different significance of features.

3.2 Basic Concepts

In terms of a business process, the executed process instances of an event log are required for the outcome prediction of an on-going case. The event log is composed of many completed cases, each of which consists of a series of event records. In general, each event record has some attributes, such as the event class attribute (i.e., *activity name*) specifies which activity the event refers to, the *timestamp* attribute specifies when the event starts and ends, and the *case ID* attribute specifies which case of the process generates the event. Usually, these three attributes exist in all event records. In addition, event records may also have other attributes. If the value may change with the event, such an attribute is often called *event attribute*. For instance, the amount of a bill in order-payment process can be

recorded as an *event attribute* represented by the activity *Create Bill*. In contrast, another kind of attribute, called *case attribute*, which value can be shared by all events in a case. For example, in this order-payment process, the *customer ID* is a typical *case attribute* because all events in a case share this attribute. In other words, the value of this attribute for all events in this case is identical. Different from the *event attribute*, such an attribute is static and its value does not change throughout the lifetime of a case.

Definition 3.1 (Event, Attribute). An *event* (record) e can be represented as a tuple $e = (a, c, t_{start}, t_{end}, d_1, ..., d_m)$, in which a is the process activity *attribute* related to this event, c is the case id *attribute*, t_{start} and t_{end} are the *attributes* of start timestamp and end timestamp, and $d_1, ..., d_m$ are a list of additional attributes (where $\forall i \in [1, m], d_i \in \mathcal{D}_i$ with $\mathcal{D}_i$ being the domain of such an attribute value). Here, a, t_{start}, and t_{end} belong to the *event attribute*, c is the *case attribute*. In terms of an event log, the collection of all events is recorded as A.

One execution of a process is usually called process instance or case, which consists of a series of events. For a certain process instance, the sequence of involved events can form a *trace*.

Definition 3.2 (Trace, Prefix trace). Each case corresponds to a *trace* consisting of a non-empty finite sequence of events $\sigma = \langle e_1, e_2, ..., e_{|\sigma|} \rangle$ such that $\forall i, j \in [1, |\sigma|], e_i \in A, e_j \in A, e_i.c = e_j.c$. In terms of an event log, the collection of all traces is denoted as S. As for a trace σ, a *prefix trace* of length $l (l \leq |\sigma|)$ can be defined as $\sigma^l = \langle e_1, e_2, ..., e_l \rangle$, which indicates the first l executed events of this case. In an event log, all prefix traces with length l can be included in S^l.

As a running example, we consider a simple log of an insurance claim process as shown in Table 1. Here, we obtain a trace $\sigma_1 = \langle (a1, 1, 2010/08/01\ 10:26:00, 2010/08/01\ 11:06:00, (resource, r_1), (amount, 2000)), (a2, 1, 2010/08/02\ 12:12:00, 2010/08/03\ 14:38:00, (resource, r_3), (amount, 2000)), \ldots, (a8, 1, 2010/08/14\ 14:20:00, 2010/08/14\ 14:30:00, (resource, r_1), (amount, 2000)) \rangle$ where the first event with *case ID* of 1 indicates the *register* event occurs at 2010/08/01 10:26:00. This trace corresponds to a process instance (i.e. case) with *case ID* of 1. We also find that the *resource* attribute of each event is different, thus called *event attribute*, while the attributes of *amount* and *case ID*, called *case attribute*, are identical in the whole case. Generally, the value of *event attributes* or *case attributes* may be numeric data, categorical data, or text data. For instance, the value of *activity* attribute and *resource* attribute in Table 1 is categorical data while the value of *amount* attribute is numeric data. Different types of data will differ in processing, which will be discussed in Section 5.

The outcome-oriented prediction for a given running case aims at predicting its outcome class, which indicates the final outcome according to some business goals. In order to construct a classifier (i.e., classification model) from the event log of a business process, each completed case needs to be labeled with its outcome class firstly.

Definition 3.3 (Outcome class labeling). A single outcome class label $y(\sigma)$ with domain of $\{0, 1\}$ can be assigned to trace σ for binary classification, in which 1 denotes that the final outcome of this case is consistent with the goal of business process, while 0 is the opposite.

After labeling the outcome class, it is necessary to encode these historical traces of an event log for training a classification model automatically.

Definition 3.4 (Event encoding, Trace encoding). *Event encoding* $f : A \rightarrow \mathbb{R}^p$ maps the attribute values of each event e into a vector with fixed length, in which p denotes the total dimension of the encoded vectors of all attribute values for a given event. Based on it, *trace encoding* $g : S \rightarrow X_{|S| \times p}$ maps trace σ into a matrix where $|S|$ denotes the maximum length of a trace. In particular, the operation of zero-padding is required when the length of a trace is less than the maximum.

After case encoding, we take these matrices as inputs of neural networks so as to construct a classification model regarding the outcome prediction automatically. Actually, the essence of a classification model reveals a mapping relationship between the encoded vector of a case and its outcome class label.

Definition 3.5 (Prediction model). A *prediction model* (i.e., classification model or classifier) $y : S \rightarrow \{0, 1\} (\sigma \in S, \forall e \in \sigma, e \rightarrow \mathbb{R}^p)$ estimates the outcome class label of an encoded (prefix) trace where 1 denotes that the final outcome is consistent with the goal of business process, while 0 is the opposite.

According to the constructed classification model, the outcome class of an on-going case can be predicted by taking the encoded matrix of a case that consists of the encoded vector of executed events and zero-padding events as the input of this model.

3.3 Problem Definition

In this paper, the problem to be solved is the outcome prediction of a case at run-time, which involves training a classifier from the historical event log for a business process at offline phase. In this phase, the executed cases with labeled outcome class in an event log

Table 1 Event log of an insurance claim process

case ID	start timestamp	end timestamp	activity	resource	amount
1	2010/08/01 10:26:00	2010/08/01 11:06:00	register/ a_1	Bob/ r_1	2000
1	2010/08/02 12:12:00	2010/08/03 14:38:00	high insurance check/ a_2	Alice/ r_3	2000
1	2010/08/03 09:24:00	2010/08/09 15:21:00	high medical check/ a_3	Alice/ r_3	2000
1	2010/08/11 10:49:00	2010/08/12 11:21:00	contact hospital/ a_4	Alice/ r_3	2000
1	2010/08/12 12:27:00	2010/08/12 12:33:00	decide/ a_5	Wil/ r_4	2000
1	2010/08/14 11:11:00	2010/08/14 11:39:00	prepare notification/ a_6	Bob/ r_1	2000
1	2010/08/14 12:20:00	2010/08/14 12:31:00	send notification by email a_7	Bob/ r_1	2000
1	2010/08/14 14:20:00	2010/08/14 14:30:00	archive/ a_8	Bob/ r_1	2000
2	2010/08/01 15:25:00	2010/08/01 16:10:00	register/ a_1	Anita/ r_2	500
.....	.....	.....	.....	.....	.....

is required. Based on the constructed classifier, the outcome class label of this running case can be predicted at online phase. In other words, the problem can be formalized as follows.

Input: the event log $L = \{\sigma_1, \sigma_2, ..., \sigma_s\}$ of s cases with outcome class labels;

an on-going case to be predicted $\sigma^{'} = \langle e_1, e_2, ..., e_{|\sigma^{'}|}\rangle$;

Middle result: a classification model M;

Output: the predictive outcome class label of $\sigma^{'}$.

As shown in Fig. 3, the input of this problem is an event log, which is composed of some historical cases $\sigma_1, \sigma_2, ..., \sigma_s$ with labeled outcome classes such as *Yes* (denoted as 1) and *No* (denoted as 0). Based on this labeled log, a classification model can be trained by neural networks at offline phase. Then, at online phase, the outcome prediction, i.e. the outcome class *Yes* or *No*, of the given case $\sigma^{'}$ can be determined based on this model as well as the execution of $\sigma^{'}$.

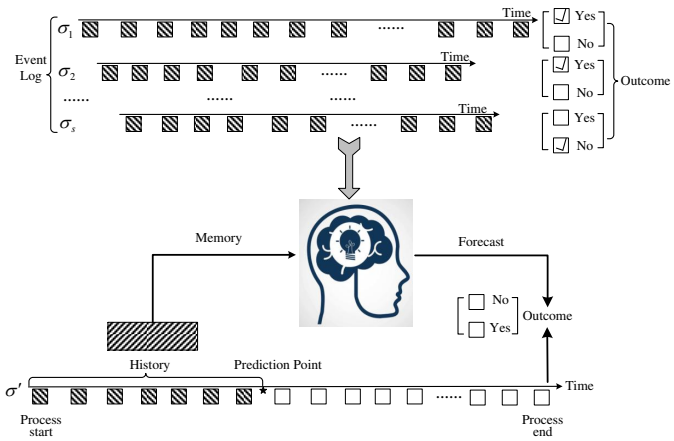


Fig. 3 Outcome prediction of an on-going case in a business process.

4 The Approaches

The key to predicting the outcome prediction of an on-going case is to construct an effective classifier based on the historical cases in event log. Such a classifier reveals the relationship between a case and its outcome class, and requires the ability to learn the long-term dependencies among a sequence, because each input historical case is a sequence of events when training this classifier. Based on the above description, thus, we can employ a neural network based on LSTM to obtain a classifier for the outcome prediction problem. The entire process is shown in Algorithm 1.

Algorithm 1 contains two parts, i.e., the *offline* phase when a classifier is trained from the historical cases of an event log (lines 1-8), and the *online* phase when the outcome of an on-going case is predicted according to this classifier (line 9). At the *offline* phase, given a random initial value for all parameters of the neural network (model), we can compute the output of this model for each trace (case) in an event log and then measure the loss between this output value and the labeled outcome class (lines 1-3). Based on this loss, we update all the weights and biases of this model by using the error Back-Propagation (BP) algorithm to propagate the error to the neural network (line 4). The above operations are repeated for each case until the end condition is satisfied (lines 5-6). At last, we obtain a classifier for an event log based on the final determined parameters. At the *online* phase, the outcome class of an on-going case can be predicted according to the execution of this case.

4.1 Offline Training

As described above, the LSTM network can automatically capture the decisive hidden features from the historical cases in event log. During the offline phase of training a classifier, some different approaches based on the original LSTM network are available for discovering the relationship between cases and outcome class. Thus,

Algorithm 1 : Outcome prediction with neural networks

Input: the outcome class labeled event log $L = \{\langle \sigma_1, y_1 \rangle, \langle \sigma_2, y_2 \rangle, \cdots, \langle \sigma_s, y_s \rangle\}(y_t \in \{0, 1\})$
an on-going case σ'
Output: the predicted outcome class $\hat{y'}$ of case σ'

Randomly initialize all weights W and biases b of a neural network;
for $t = 1, 2, \cdots, s$ **do**
  Compute the predicted outcome $\hat{y_t}$ of σ_t based on the current values of W, b and Eq. (10) (in Section 4.1.1) or Eq. (19) (in Section 4.1.3);
  Update all weights W and biases b of a neural network according to Eq. (10) or Eq. (19);
  if the loss function $Loss(\hat{y_t}, y_t)$ (in Eq. (11) of Section 4.1.1) is convergence **then**
    break;
  end if
end for
Compute the predicted outcome class $\hat{y'}$ of case σ' based on the final trained W, b and Eq. (10) or Eq. (19);
return $\hat{y'}$;

in this paper, we first propose an approach based on the original LSTM network, called LSTM, to construct a classifier for the outcome prediction problem. However, the LSTM network only considers the relationship between the preceding events and the current event, but ignores the events that occur after this event when extracting features. Especially, the outcome of a case is not only related to the events of this case, but also related to the context information of these events, especially for some complex processes. Usually, the more complex a business process is, the more complex the decisive factors of its final outcome are. Furthermore, the original LSTM network considers all the features have the same influence on the outcome when transforming the extracted features into the output of the classification result. In fact, each feature extracted from cases has different influence on the outcome. Therefore, we make two enhancements based on the original LSTM network.

On the one hand, in order to capture more context features, we propose an approach, called Bi-LSTM, based on the adapted bidirectional LSTM network, which is combined with the original LSTM network forward and backward. On the other hand, in order to distinct and optimize the extracted features, we propose an approach, called Att-LSTM, which adds the attention mechanism in the original LSTM network. In addition, on the basis of these enhancements, we propose a hybrid approach, called Att-Bi-LSTM, to combine the advantages of the bidirectional LSTM network and attention mechanism. In this case, the constructed classifier can not only capture rich features but also optimize them further. The four approaches mentioned above can be used to construct an effective outcome prediction model for predicting the outcome of a process. They can be divided into three parts: the vectorization of input events in a case, the feature extraction from these events of a case, and the classification of the output. The details are as follows.

4.1.1 LSTM approach

Here, we describe how to use the original LSTM network to train a classifier from an event log by taking a completed case in event log L for example. At first, the vectorization representation of executed cases are obtained by encoding the attributes of the events with different ways according to their value types. Then, the LSTM network is used to extract key features from these encoded cases according to the fact that the outcome of a case is determined by some certain events as well as their attributes. Finally, the probability of outcome class is calculated based on the extracted feature vectors. The architecture of LSTM approach for process outcome prediction is shown in Fig. 4, which can be divided into 4 layers.

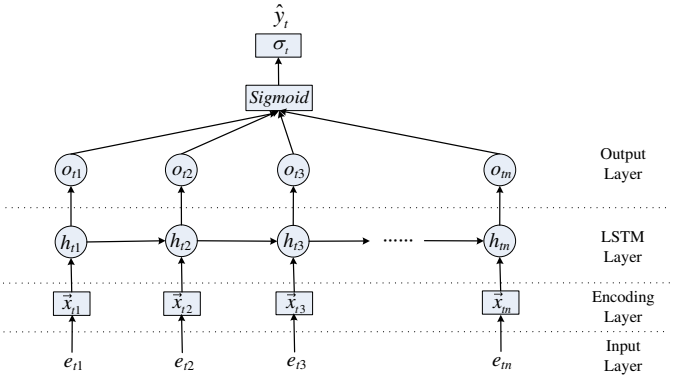


Fig. 4 The architecture of LSTM approach for process outcome prediction.

Input layer. In this layer, an event log $L = \{\sigma_1, \sigma_2, ..., \sigma_s\}$ with s cases are viewed as the input of LSTM neural network for training a classifier. The t-th trace in L is denoted as $\sigma_t = \langle e_{t1}, e_{t2}, \ldots, e_{tn}\rangle (n = |\sigma_t|)$, in which $e_{ti} (i \in \{1, 2, \cdots, n\})$ is the i-th event (record) in this trace. In addition, each trace is labeled with its outcome class according to the business goal. Such a trace can be used for training a classifier once.

Encoding Layer. In this layer, all event attributes in trace can be encoded according to the type of attribute value. If its type is numerical, the attribute value can be normalized according to the range of all possible values for this attribute in event log L. If the type is categorical, *one-hot* encoding method can be used to convert the value to vector of 0 and 1. According to these rules, we obtain the vector of each event in all traces by mapping it into a p-dimension vector expressed as $\vec{x}_{ti} = [x_{ti,1}, x_{ti,2}, \ldots, x_{ti,p}]$, in which p is the total dimension of the encoded vector of all attributes for an event, determined by the record attributes in an event log. Afterwards, the encoded sequence of vectors $\vec{x}_{t1}, \vec{x}_{t2}, \cdots, \vec{x}_{tn}$ can be fed into the next layer for feature extraction.

Feature Extraction Layer (LSTM Layer). For t-th trace in this event log, the input of this layer is a sequence of event vectors $\vec{x}_{t1}, \vec{x}_{t2}, \ldots, \vec{x}_{tn}$. Then, we obtain the outputs $h_{t1}, h_{t2}, \ldots, h_{tn}$ of this (hidden) layer for each event, in which the output $h_{ti}(\vec{h}_{ti})$ of the i-th event can be calculated based on the output state $h_{t,i-1}$ of the previous event $\vec{x}_{t,i-1}$ as well as the current input event vector $\vec{x}_{ti}$. The obtained hidden feature vector $h_{ti}(\vec{h}_{ti})$ denotes the learned information between the current event and its previous events, which can be defined as:

$$o_{ti}(h_{ti}) = LSTM(\vec{x}_{ti}, h_{t,i-1}), i \in [1, n] \tag{3}$$

Each cell of this layer is equipped with three gates to control the flow of information as shown in Fig. 2(b). One is the input gate $input_{ti}$ (similar to i_t in Fig. 2(b)) which determines how much information can flow into the memory cell, one is the forget gate $forget_{ti}$ (similar to f_t in Fig. 2(b)) which determines how much information is forgotten, and the other is the output gate $output_{ti}$ (similar to o_t in Fig. 2(b)) which determines how much information outputs from the current state cell c_{ti} (similar to c_t in Fig. 2(b)). For each cell of this layer, let $h_{t,i-1}$ and $c_{t,i-1}$ be the vectors of output and cell state separately from the prior unfold cell on the same level. In order to extract the features from the current event e_{ti}, we take the input $\vec{x}_{ti}$ of the current state and the previous state information of $h_{t,i-1}$ and $c_{t,i-1}$ into consideration. At first, we determine what information (i.e., the features) should be thrown away from $\vec{x}_{ti}$ and $h_{t,i-1}$ by the *sigmoid* activation function of F_1.

$$forget_{ti} = F_1(W_f \cdot [h_{t,i-1}, x_{ti}] + b_f) \tag{4}$$

where W_f and b_f are trainable parameters of weights and biases. Then, we determine the information to be stored by using the input gate and an update operation. By using the input gate, we decide which state to be updated by Eq. (5).

$$input_{ti} = F_1(W_i \cdot [h_{t,i-1}, x_{ti}] + b_i) \tag{5}$$

where F_1 is the *sigmoid* activation function, W_i and b_i are also the parameters to be trained. Afterwards, a new candidate value g_{ti} (similar to g_t in Fig. 2(b)) needs to be computed first by:

$$g_{ti} = F_2(W_g \cdot [h_{t,i-1}, x_{ti}] + b_g) \tag{6}$$

where F_2 is *tanh* activation function, W_g and b_g are the corresponding parameters to be trained. Once $input_{ti}$ and g_{ti} are obtained, we update the previous state $c_{t,i-1}$ to a new state c_{ti} by:

$$c_{ti} = forget_{ti} * c_{t,i-1} + input_{ti} * g_{ti} \tag{7}$$

where the left item represents the information which needs to forget from the previous state $c_{t,i-1}$ while the right item represents the new update information to the current cell state c_{ti}. At last, what information of this cell state outputs to the final output can be computed by:

$$output_{ti} = F_1(W_o \cdot [h_{t,i-1}, x_{ti}] + b_o) \tag{8}$$

where F_1 is the *sigmoid* activation function, W_o and b_o are the trainable parameters. Then the final output of this cell state o_{ti} to the following layer and h_{ti} to the subsequent cell can be obtained by running the the information of c_{ti} with the *tanh* activation function firstly and then multiplying with the information of this cell state provided as output:

$$o_{ti}(h_{ti}) = output_{ti} * F_2(c_{ti}) \tag{9}$$

Output Layer. Let $o_t = [o_{t1} \oplus o_{t2} \oplus \cdots \oplus o_{tn}]$ ($[\oplus]$ represents vector concatenation) be the input of *sigmoid* function to estimate the outcome class of trace σ_t. The computed probability $\hat{y}_t$ of the outcome class of 1 (i.e. positive) for this trace can be calculated as Eq. (10):

$$\hat{y_t} = sigmoid(W_c o_t + b_c) \tag{10}$$

where W_c and b_c are the trainable parameters in this layer and the value of $\hat{y_t}$ is in the range of $0 - 1$.

At last, we use a binary cross-entropy function to measure the loss of this classifier based on the labeled outcome class y_t and the predicted probability $\hat{y_t}$ (i.e. actual result from neural networks) for trace σ_t.

$$Loss(\hat{y_t}, y_t) = -(y_t log\hat{y_t} + (1 - y_t)log(1 - \hat{y_t})) \quad (11)$$

In order to determine the parameters that minimize the loss function, we use some different optimization BP (Back-Propagation) algorithms such as RMSProp (Root Mean Square Prop) (Tieleman and Hinton 2012) and Adam (Adaptive Moment Estimation) (Kingma and Ba 2015) in training. Based on these optimized gradient descent algorithms, the above parameters of all W and b in the LSTM network can be adjusted by the propagation of the loss $Loss(\hat{y_t}, y_t)$ (i.e., error). In this way, all cases in event log L are processed until a distinct set of parameters are determined with the minimum sum of loss from these cases. Therefore, a classification model for an event log can be constructed based on LSTM network with these determined parameters.

4.1.2 Bi-LSTM approach

In different business processes, the relationship between cases and their outcomes can be diverse. In general, the outcome of a case is not only related to the events in this case, but also related to its content, especially for some long cases. If the contextual information between these events can be captured, the extracted feature can be further enriched to make the trained classifier more effective. As the first enhancement of LSTM approach, Bi-LSTM adds a backward propagation layer of the original LSTM network. In terms of the events sequence of a case, the forward propagation layer (i.e. the LSTM Layer in LSTM network) obtains the relationship information between the preceding events and the current event, while the backward propagation layer obtains the relationship information between the current event and the subsequent events. In order to extract the hidden features from the events in a case by Bi-LSTM, we obtain the contextual information for each event by combining the outputs of the forward and backward hidden layers where events are fed forward and backward simultaneously. Similar to LSTM, the Bi-LSTM approach also has four layers, as shown in Fig. 5. The difference between them exists in the event feature extraction layer (LSTM Layer vs. BiLSTM Layer). Here, we only describe the BiLSTM Layer.

Feature Extraction Layer (BiLSTM Layer). In this (hidden) layer, each unit is combined by the original LSTM network from two directions (i.e., forward and backward) as shown in Fig. 5. Taking the sequence of event vectors $\vec{x}_{t1}, \vec{x}_{t2}, \ldots, \vec{x}_{tn}$ as the input of this

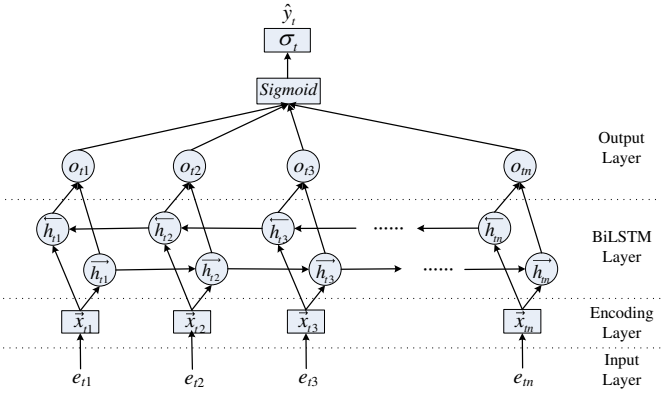


Fig. 5 The architecture of Bi-LSTM approach for process outcome prediction.

layer, we obtain the output $o_{t1}(h_{t1}), o_{t2}(h_{t2}), \ldots, o_{tn}(h_{tn})$ of them by updating $\overrightarrow{h_{ti}}$ and $\overleftarrow{h_{ti}}$ forward and backward simultaneously based on the current event e_{ti} and the output state $\overrightarrow{h_{t,i-1}}$ of the previous event $e_{t,i-1}$ (forward) or the output state $\overleftarrow{h_{t,i+1}}$ of the next event $e_{t,i+1}$ (backward), respectively. The obtained vector $o_{ti} \in R^p$ for event e_{ti} denotes the learned hidden information (i.e., features) based on the current event e_{ti} and the hidden information from the previous event in the same level, which can be calculated by:

$$\overrightarrow{h_{ti}} = LSTM_{forward}(\vec{x}_{ti}, \overrightarrow{h_{t,i-1}}), i \in [1, n] \quad (12)$$

$$\overleftarrow{h_{ti}} = LSTM_{backward}(\vec{x}_{ti}, \overleftarrow{h_{t,i+1}}), i \in [1, n] \quad (13)$$

$$o_{ti}(h_{ti}) = [\overrightarrow{h_{ti}} \oplus \overleftarrow{h_{ti}}] \quad (14)$$

where $\overrightarrow{h_{ti}}$ denotes the output of the cell based on the vector $\vec{x}_{ti}$ and the output $\overrightarrow{h_{t,i-1}}$ of the previous cell forward, and $\overleftarrow{h_{ti}}$ denotes the output of the cell based on the vector $\vec{x}_{ti}$ and the output $\overleftarrow{h_{t,i+1}}$ of the previous cell backward. $\overrightarrow{h_{ti}}$ and $\overleftarrow{h_{ti}}$ can be calculated according to the Eqs. (4)-(9). After that, the obtained $o_{t1}, o_{t2}, \cdots, o_{tn}$ can also be as the input of Output Layer for calculating the estimated probability. Thus, a classification model can be obtained based on the bidirectional LSTM network with the determined parameters.

4.1.3 Att-LSTM approach

The outcome of a case is related not only to some features that have a decisive effect on the outcome, but also to their influence degree. Although the LSTM approach can capture the decisive features from historical cases, it cannot figure out the extent of their influence.

As the second enhancement of LSTM approach, Att-LSTM approach adds additional attention mechanism layer base on LSTM network, which can distinct the weight of each feature. The architecture of Att-LSTM approach is shown in Fig. 6, which has five layers, one more Attention Layer than LSTM approach. Besides, the Output Layer is also different from that in LSTM approach. Here, we only describe the Attention Layer and Output Layer.

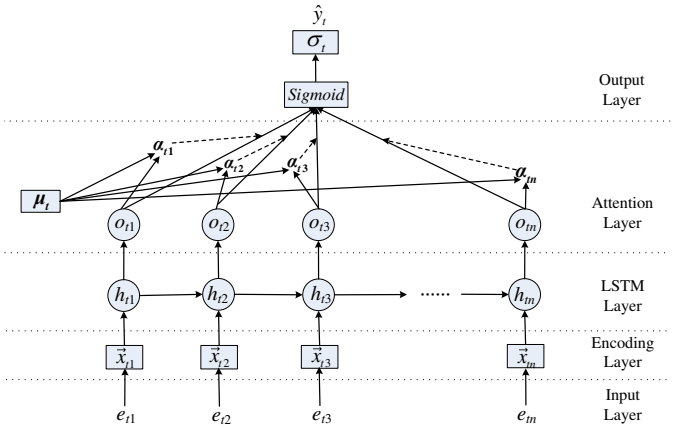


Fig. 6 The architecture of Att-LSTM approach for process outcome prediction.

Attention Layer. After extracting features from the previous hidden layer (LSTM Layer), we obtain the output o_{ti} for each event vector $\vec{x}_{ti}$. As for each hidden output o_{ti}, we firstly compute its attention score u_{ti} for the outcome class. Then, we translate each attention score to probability distribution by the *softmax* activation function. Finally, a weighted sum of these hidden outputs, i.e. a global representation of these features, is obtained based on their probability distributions. Fig. 7 shows how the attention mechanism works. On the basis of the outputs $o_{t1}, o_{t2}, \ldots, o_{tn}$ from the previous hidden layer, the final global representation v_t of them can be computed with the attention mechanism. In this figure, $o_t \in R^{p \times n}$ denotes the matrix of $[o_{t1}, o_{t2}, \ldots, o_{tn}]$, in which p is the dimension of each vector o_{ti} and n is the length of this vector (i.e., the event number of case σ_t). In this layer, each vector o_{ti} can be nonlinearly transformed to its implicit representation u_{ti} by:

$$u_{ti} = tanh(W_h o_{ti} + b_h) \tag{15}$$

where u_{ti} denotes the attention score of feature o_{ti}, W_h and b_h are the parameters of this layer. Similarly, we obtain the matrix $u_t = [u_{t1}, u_{t2}, \cdots, u_{tn}]$ that corresponds to the hidden feature output $o_t = [o_{t1}, o_{t2}, \cdots, o_{tn}]$. In order to obtain the weight of each feature, we then use the *softmax* activation function to normalize the point multiplication of u_t and the initialized attention distribution probability μ_t by:

$$\alpha_t = softmax(\mu_t^T u_t) \tag{16}$$

Here, μ_t is a parameter vector obtained by training (μ_t^T is its transpose), and each $\alpha_{ti} \in \alpha_t$ denotes the attention weight of each feature vector o_{ti}, which can be computed by:

$$\alpha_{ti} = \frac{exp(\mu_t^T u_{ti})}{\sum_{i=1}^{n} exp(\mu_t^T u_{ti})} \tag{17}$$

Finally, as shown in Fig. 7, the final output v_t denotes the global representation of these features o_t, which can be computed as a weighted sum of them by:

$$v_t = \sum_{i=1}^{n} \alpha_{ti} o_{ti} \tag{18}$$

The obtained v_t involves some features that are significantly correlated with the outcome class (label) for trace σ_t.

Output Layer. The obtained v_t can be the input of the *sigmoid* function to estimate the outcome class (label) of trace σ_t. The estimated probability $\hat{y}_t$ of the outcome class (label) of 1 for trace σ_t can be calculated as follow:

$$\hat{y}_t = sigmoid(W_c v_t + b_c) \tag{19}$$

where W_c and b_c are the trainable parameters in this layer and the value of $\hat{y}_t$ is in the range of $(0, 1)$. Similarly, we also use a binary cross-entropy function to measure the loss of this model and then use the optimized BP algorithms to adjust the relevant parameters. Finally, a classification model can be obtained based on the determined parameters of this Att-LSTM network.

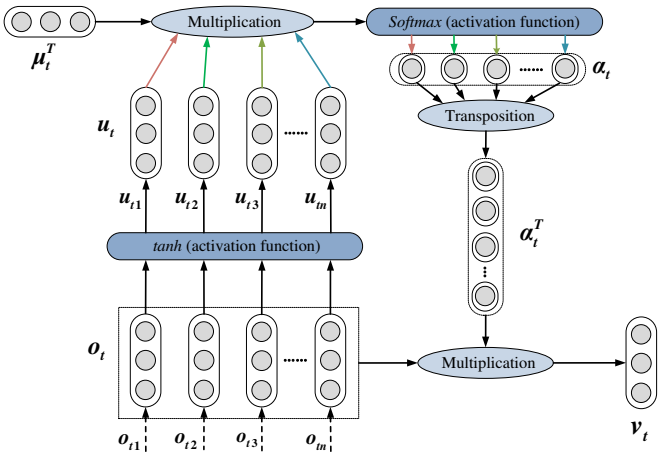


Fig. 7 The process of attention mechanism.

4.1.4 Att-Bi-LSTM approach

In order to construct a more effective classifier, we combine the above two enhancements and propose the Att-Bi-LSTM approach, which can not only capture more features but also distinct their extent to influence the outcome for a case. As shown in Fig. 8, Att-Bi-LSTM also has five layers, in which the BiLSTM Layer is the same as that of the Bi-LSTM approach, the Attention layer and the Output layer are the same as that of the Att-LSTM approach, and the first two layers are identical to that of LSTM (Bi-LSTM or Att-LSTM) approach. In this case, we first obtain the predicted probability $\hat{y}_t$ of the outcome class for trace σ_t. After that, we also use the binary cross-entropy function to measure the loss between the actual output $\hat{y}_t$ from the network and the expected (labeled) output y_t for trace σ_t. By updating all involved parameters with BP algorithms for all traces, the classifier can be obtained by combining this network and the final determined parameters.

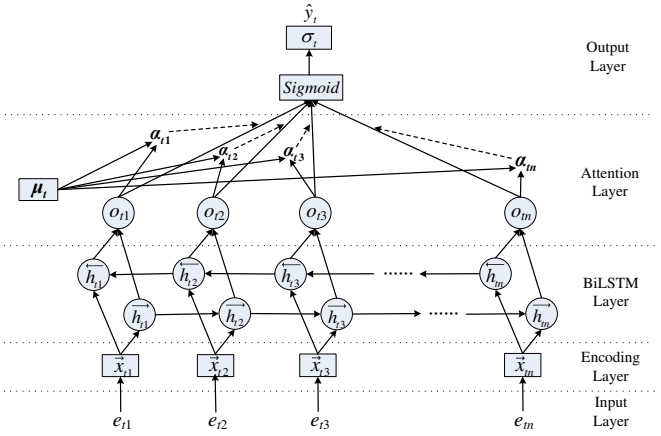


Fig. 8 The structure of Att-Bi-LSTM approach for process outcome prediction.

4.2 Online Predicting

In order to predict the outcome of an on-going case, we need to use the classifier trained at the offline phase based on the execution events of this case. For an on-going case, there should be some already executed events and each of them has some attributes. Taking the encoded vectors of these events and attributes as the input of the trained classifier, the probability of outcome class for this running case can be calculated. In general, an on-going case keeps changing with the arrival of new events during its running process. Therefore, the predicted results (i.e., the probabilities of getting a positive result) of each phase for an on-going case are not completely consistent because their inputs of the classifier are different. Under such circumstance, the prefix trace is used to simulate an executing case. Especially in the evaluation experiments of Section 5, we first extract the distinct cases from an event log and then obtain all prefix traces with different lengths. Taking these prefix traces as the input of the trained classifiers for different approaches, we then predict the outcome class to evaluate the prediction accuracy of these approaches. As for each distinct case in an event log, the accuracy of its prediction can be calculated based on the predictive results of extracted prefix traces with different lengths. The detail will be discussed in the next section.

5 Evaluation

5.1 Experimental Settings

In this section, to evaluate the effectiveness and efficiency of our proposed approaches, i.e., Bi-LSTM, Att-LSTM and Att-Bi-LSTM, we compare them with LSTM and the conventional machine learning methods of XGBoost (Friedman 2001) and RF (Random Forest) (Bishop 2006). Here, the RF and XGBoost are selected because they are more effective than other traditional learning methods (Teinemaa et al. 2019; Fernández-Delgado et al. 2014). Moreover, the RF (i.e. RF-based) and XGBoost (i.e. XGBoost-based) respectively denote a series of approaches that utilize the Random Forest algorithm and the XGBoost algorithm to build one or more classifiers but with the different encoding methods. To make the comparison experiment more convincing, we apply the above six approaches to twelve real datasets that are from different business processes. Besides, we select the prediction accuracy, earliness, stability, and time performance for a more comprehensive comparison. The approaches are implemented in Python by using the scikit-learn library, and ran on the server with six cores of E5-2620 2.00GHz, 64GB memory, and Windows 7.

5.1.1 Datasets

The datasets used in our experiment are originated from six real-life event logs of *BPIC_2012*, *BPIC_2017*, *Sepsis_cases*, *Production*, *Road Traffic Fines* and *Hospital Billing*, which are all from the public 4TU Centre for Research Data [1]. They are selected because they contain both case attributes (static) and event attributes (dynamic). Moreover, the outcome of cases in these logs

[1] https://researchdata.4tu.nl/home/

can be derived easily, and the order between the events in these logs is clearly defined without ambiguity.

BPIC_2012. This event log was derived from the Business Process Intelligence Challenge in 2012 (BPIC 2012), which records the execution history of a loan application process in a Dutch financial institution. In order to facilitate classification, we define the class labels based on the final outcome of each case, i.e. whether the loan application is accepted, rejected, or cancelled. For such an event log to be classified, it could be thought of as a multi-class classification problem. In order to facilitate comparison with the existing methods, we transform this event log into three event logs of *bpic_2012_accepted*, *bpic_2012_declined*, and *bpic_2012_cancelled* with independent binary classification problems.

BPIC_2017. This event log originates from the same loan application process in BPIC_2012 and then has been improved by cleaning noise data. Similarly, we define three separate binary classification problems with event logs of *bpic2017_accepted*, *bpic2017_declined* and *bpic2017_cancelled* from the event log *BPIC_2017*.

Sepsis_cases. This log comes from a hospital in Dutch and records the diagnostic trajectories of patients with life-threatening sepsis when they enter the hospital. According to the conditions after patients left, such as returning to emergency within 28 days, being admitted to intensive care, and being discharged from the hospital, the log was classified into three event logs of *sepsis_cases_1*, *sepsis_cases_2*, and *sepsis_cases_3* for three separate binary classification problems. Similarly, we can obtain three datasets from the log *Sepsis_cases*.

Production. The log is derived from a manufacturing process that records the information referred to the activities, performers and/or machines involved in producing goods. The outcome of this event log is labeled according to whether or not the rejected products exist. If there is no rejected products in this manufacturing process, the outcome of this case is labeled as normal. Otherwise it is labeled as abnormal.

Road Traffic Fines. The log comes from a police station in Italy. It mainly includes activities of paying fines as well as some information related to individual cases, such as the reason and the total amount paid for each fine. In terms of this log, we label these cases according to whether the fine is repaid in full or is sent for credit collection.

Hospital Billing. The log is derived from an ERP system of a hospital, where each case represents an execution of a billing procedure for medical services. We define two labels based on whether they were reedited or not.

Thus, we have collected 12 datasets, in total.

5.1.2 Preprocessing

The preprocessing includes deleting noise data, filling in attributes for aligning all attributes of historical cases, encoding and extending additional attributes, reducing feature spaces for avoiding the dimension explosion, and under-sampling for sample imbalance, illustrated as follows.

Delete noise records. All cases are checked if they are finished by comparing the last event of each case with a specific end event. If the case doesn't end with the specific event, it can be viewed as a incomplete case and should be deleted. In addition, some other methods are used here to check whether a historical case is completely executed, by aligning the cases with the corresponding process model.

Fill attribute values. In general, there are very few data missing because the event log was automatically generated by information systems. However, that may be for the case due to the following two reasons. First, events usually only records some attributes when a particular event changes. Therefore, in order to determine the value of this attribute when an event occurs, we need to search the (prefix) trace for the latest event where the value of the problematic attribute has changed (if no change point is found, search for the first event). For instance, the resource name involved in the execution of an activity in a case is usually only logged when the resource has changed due to a previously occurring event. In this case, we need to search for the most recent previous event under the same circumstances and record the resource of that event as the resource of the current event. Secondly, different activities can produce different types of data. For example, in a loan application process, the order information for each customer only exists after the order event is created. Similarly, during ticket processing, the payment amount is only available after a payment event has occurred. The absence of these forms is called legal missing data or missing data out of range (Schafer and Graham 2002). In this experiment, padding is done by adding additional attributes. If an event does not have this attribute, the attribute has a value of 0.

Extend additional attributes. In order to improve the prediction accuracy of a classification model, we extend some new attributes in these datasets based on the original attributes. As for the *timestamp* attribute in the above datasets, some other new attributes corresponding to an event can be calculated such as *hour*, *weekday*, *month*, *time since case start* and *time since last event*. These attributes are extracted from inter-case, which can help predict whether the outcome of a running process violates according to (Senderovich

et al. 2017; Conforti et al. 2015). Besides, the waiting time of an event has a lot to do with all the active cases since the event has a great impact on the outcome of the case if the outcome is determined by the waiting time or the satisfaction of the customer. Therefore, we add an attribute *open cases* to record the on-going cases that occur at the same time as an event. In addition, a case attribute *event number* has been added to record how many events has been performed in this case for the current event.

Reduce feature space. To avoid the dimension explosion after attribute encoding, we compress the spatial domain of some attribute values before training a classifier. For instance, regarding the categorical attribute of events in historical cases, we divide its possible values into several intervals if there are many. In addition, as for the sparse values of a categorical attribute, we group them together and classify as *other*.

Under-sample for sample imbalance. Very long traces can decrease the performance of classifiers (Teinemaa et al. 2018). To avoid the sample imbalance, we truncate the traces over the *truncated length*. As for each dataset, the *truncated length* is determined independently where 90% of the traces in the minority outcome class (i.e., positive *vs.* negative) have already completed. In other words, we choose the outcome class with less traces first and then decide the length of traces by grouping them in ascending order based on their length and finding the point of 90%. These truncated sequences are not available anymore for training or evaluation because these few long traces can decrease the performance of classifiers.

Table 2 gives the statistics of the datasets after preprocessing. As it indicates, the size of datasets varies significantly ranging from 220 (*production*) to 129,615 (*traffic_fines*). Meanwhile, the ratio of positive class is also different. For example, the most imbalanced one is the *hospital_billing* dataset where only 5% of cases are labeled as positive, whereas the classes are almost balanced in the *production*, *bpic2012_accepted*, *bpic2012_cancelled*, and *traffic_fines* datasets. Additionally, in terms of the length of trace, the most heterogeneous is the *sepsis_cases_3* dataset, where the longest case consists of 185 events but the shortest one consists of only 4 events. After the under-sampling operations, we obtain the truncated length for each dataset. For the datasets originated from *BPIC_2012* and *BPIC_2017* event logs, the truncated length are determined because the signal starts to converge around 40 and 20 events (Teinemaa et al. 2018), respectively.

5.1.3 Evaluation metrics

In predictive process monitoring, a good prediction should be *accurate* in the *early* stage of an on-going case. In addition, the prediction should be made in a continuous way for a running case when each new event arrives. Thus, it is very important to make the stable and consistent predictions. Therefore, we select *accuracy*, *earliness*, *stability* to measure the effectiveness of our proposed approaches, and *execution time* to measure their efficiency.

Accuracy. In general, the output of a classification model usually gives the probability of a class, not a certain class. Therefore, to determine the class of prediction result, a threshold needs to be set manually first and then the probability can be converted into a certain class. Only when the prediction probability of a prediction sample is greater than this threshold, the sample will be labeled as positive. Otherwise it will be labeled as negative. Thus, the value of threshold greatly affects the calculation of accuracy. Here, we use the *Area Under ROC Curve* (AUC) as the measurement of prediction accuracy based on (Bradley 1997). Each point on the ROC curve represents a pair of FPR (False Positive Rate, on the X-axis coordinate) and TPR (True Positive Rate, on the Y-axis coordinate) based on a given threshold. The AUC can be used to measure the quality of the classifier. Even if the sample distribution is not uniform, the accuracy of classification model measured by AUC can remain unbiased.

Earliness. The outcome prediction of a running case is a continuous prediction problem, in which the same case has many predictive stages formed by an incoming event. Hence, if the accuracy of a classification model can reach a certain level at early stage, this model is considered to have a good performance at prediction. Inspired by (Leontjeva et al. 2016), we keep applying the trained classifiers from the above six approaches to the subset of prefix traces with different lengths until the outcome prediction of a running case reaches an acceptable level of accuracy. The minimum prefix length at which the classifier reaches a specified accuracy threshold can be used to measure its *earliness*, indicated as Eq. (20). The smaller the value of earliness is, the more predictive the prediction model is.

$$Earliness = \frac{1}{N}\sum_{i=1}^{N}\frac{l_i}{\mathcal{L}} \times 100 \tag{20}$$

where N is the number of different prefix lengths of the test set, $\mathcal{L}$ denotes the maximum length of these cases in test set, and l_i indicates the length of prefix traces at which their obtained average AUC from a classifier reaches a certain threshold δ. In particular,

Table 2 Statistics of the datasets used in the experiments

Dataset	#Traces	Positive class(%)	Length range	Med length	Trunc. length	#Event classes	#Events	#Dynamic Attrs	#Static Attrs
bpic2012_accepted	4,685	48	[15,175]	35	40	36	155,783	10	1
bpic2012_declined	4,685	17	[15,175]	35	40	36	155,783	10	1
bpic2012_cancelled	4,685	35	[15,175]	35	40	36	155,783	10	1
bpic2017_accepted	31,413	41	[10,180]	35	20	26	624,352	20	3
bpic2017_declined	31,413	12	[10,180]	35	20	26	624,352	20	3
bpic2017_cancelled	31,413	47	[10,180]	35	20	26	624,352	20	3
sepsis_cases_1	782	14	[5,185]	14	29	15	12,189	13	24
sepsis_cases_2	782	14	[4,60]	13	13	15	9,178	13	24
sepsis_cases_3	782	14	[4,185]	13	22	15	11,056	13	24
production	220	53	[1,78]	9	8	26	2,275	15	3
traffic_fines	129,615	46	[2,20]	4	10	10	460,462	14	4
hospital_billing	77,525	5	[2,217]	6	23	17	404,721	21	1

if the AUC of outcome prediction for a running case has never reached the threshold all the time, we will set its earliness to 100, meaning the worst earliness.

Stability. A classification model is considered as stable if its outputs from successive prefix traces of the same case (i.e. the different prediction stages of the same case) is similar. Inspired by (Teinemaa et al. 2018), we define the temporal stability of a classification model as one minus the average absolute difference between any two consecutive prediction scores:

$$Stability = 1 - \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|\sigma_i|-1}\sum_{t=2}^{|\sigma_i|}|\hat{y}_t^i - \hat{y}_{t-1}^i| \qquad (21)$$

where N is the number of traces to be predicted, $|\sigma_i|$ denotes the length of trace σ_i, and $|\hat{y}_t^i - \hat{y}_{t-1}^i|$ gives the difference between two successive predictions, $\hat{y}_t^i$ and $\hat{y}_{t-1}^i$, of σ_i. Here, $\hat{y}_t^i$ denotes the prediction after t-th event occurred in the i-th case, and $\hat{y}_{t-1}^i$ denotes the prediction after $(t-1)$-th event occurred in the i-th case. As shown in Eq. (21), in order to eliminate the bias of prediction against long traces, this metric evaluates the average absolute difference between successive prediction accuracy in each case and then computes the average of them.

Running time. A prediction is of no practical significance, if the time required for prediction is close to the remaining execution time of a running case. Here, we select two time metrics, i.e., *offline time* and *online time*, to evaluate the time performance of the above approaches. *Offline time* refers to the total time required to build a prediction model from a historical event log, and *online time* refers to the average time to predict the outcome of a running case.

5.1.4 Parameter settings

To simulate the outcome prediction in real scenarios, we divide each dataset into 80% training set and 20% test set based on the temporal order. In other words, all cases involved in an event log are sorted according to their start time. The first 80% of them are used for training, and the remaining 20% are used for testing. At the same time, we remove the cases where the time of some events in training set might overlap with the time of that in test set to avoid the interference in the evaluation process. In order to find the best performance of each approach under hyper-parameters optimization, each training set is further randomly divided into 80% training data and 20% validation data. Similarly, the training data and validation data are composed of some historical cases. Specifically, we train the classifiers by applying LSTM-based (i.e., LSTM, Bi-LSTM, Att-LSTM and Att-Bi-LSTM) approaches to the training data with different parameter settings firstly and then choose the one that best fits the validation data. Afterwards, we determine the values of hyperparameters by using random search method (Friedman 2001) rather than grid search because the latter is unavailable when many parameters are involved in the LSTM-based approaches (Teinemaa et al. 2018). The detailed hyperparameters as well as their distributions used in optimization are shown in Table 3. Regarding our proposed approaches based on neural networks, the hyperparameters are mainly involved in the number of hidden layers, the number of units in these hidden layers, the initial learning rate, batch size, dropout, and the used optimization algorithm in gradient descent algorithm. Based on them, 16 parameter combinations for each approach are determined randomly and the number of epoch for these LSTM-based approaches is fixed to 50. Similarly, as for the approaches of RF and XGBoost, the Tree-structured Parzen Estimator (TPE) algorithm (Bergstra et al. 2011) can be used for each combination of a bucketing method and a sequence encoding method in each dataset respectively. In terms of these parameter combinations, we choose one of them with the highest AUC in validation data based on (Teinemaa et al. 2019). Besides, in order to calculate the earliness, we set the threshold of AUC

Table 3 Hyperparameters as well as their distributions used in optimization of LSTM-based approaches

Approach	Parameters	Distribution	Values
	the number of hidden layers (n_layer)	Categorical	$x \in \{1, 2, 3\}$
LSTM	the number of units in hidden layer (n_hidden)	Log-uniform integer	$x \in [10, 150]$
Bi-LSTM	the initial learning rate ($learning\_rate$)	Log-uniform	$x \in [0.000001, 0.0001]$
Att-LSTM	batch size ($batch$)	Categorical	$x \in \{8, 16, 32, 64\}$
Att-Bi-LSTM	dropout	Uniform	$x \in [0, 0.3]$
	optimizer	Categorical	$x \in \{rmsprop, adam\}$

$\delta = 0.7$ (where the earliness of different approaches can make the biggest difference) within a range of [0.5,1] increasing by 0.01.

5.2 Experimental Results

In order to evaluate the effectiveness and efficiency of the above approaches, we apply them to the training set of each dataset for building classification models. In this offline training phase, for each dataset, we construct a prediction model from training data first and then optimize all the hyperparameters based on the validation data. In the online phase, in order to simulate the on-going cases, we first extract all the prefix traces with different length from each complete trace in test set and then make prediction for each of them. After that, for each dataset, we calculate the prediction accuracy based on AUC for each prefix trace independently with different approaches. Based on them, we finally compute the earliness, temporal stability, offline time, and online time of predictions.

5.2.1 Accuracy comparison

To compare the accuracy of prediction, Table 4 shows the overall AUC as well as the standard deviation for each dataset and the mean overall AUC for different approaches. Here, the overall AUC of RF and XGBoost denote the best one that the they can obtain among the RF-based and the XGBoost-based approaches, respectively. The overall AUC for a dataset is a weighted average of the AUC calculated from all prefix traces with different lengths where the weights are assigned according to the number of prefix traces. For example, given m different lengths $\{l_1, l_2, \ldots, l_m\}$ of prefix traces and the set of prefix traces $L^{l_i} = \{\sigma_1, \sigma_2, \ldots, \sigma_{n_i}\}$ of each length l_i in a dataset, the AUC value of each prefix trace σ_j, i.e., $\sigma_j.score$, can be obtained firstly when predicting with a classifier. Then the overall AUC can be calculated by:

$$Overall\_AUC = \frac{1}{m} \times \sum_{i=1}^{m} (\frac{1}{n_i} \times \sum_{j=1}^{n_i} \sigma_j.score) \qquad (22)$$

The overall AUC is different from the average AUC of all prefix traces. The benefit of this weighting method is that the overall AUC is affected by different lengths of prefix traces equally in the testing set rather than being biased longer prefix traces. As Table 4 indicates, Att-Bi-LSTM achieves the best performance with the highest overall AUC in 11 out of 12 datasets. In terms of the mean overall AUC of all datasets, Att-Bi-LSTM performs best, followed by Bi-LSTM. Besides, the approaches of LSTM, Att-LSTM and XGBoost perform similar while RF the worst. Thus, we can conclude that the enhancement of bidirectional feature extraction in Bi-LSTM is more effective than that of attention mechanism in Att-LSTM, and the prediction accuracy of Att-Bi-LSTM, which combines both, has been greatly improved.

For further comparison, Fig. 9 presents the AUC values of test prefix traces (i.e., running cases) with different lengths, with each subfigure showing the detailed AUC values for each dataset. In this way, the number of cases with a certain length of prefix is monotonically decreasing with the increasing of prefix length. In the datasets of *bpic2012_cancelled* and *sepsis_case_2*, the prediction accuracy in terms of AUC has a downward change as the prefix length reaches at a certain length, such as the 35 and 10 respectively. Generally, the prediction accuracy should increase with the prefix length because the classifier could obtain more useful information. Similar to (Teinemaa et al. 2019, 2018), the reason may lie in that many cases in these datasets are relatively short and are finished with a short (prefix) length. Therefore, the accuracy of prediction decreases as the prefix length increases to a certain extent.

As shown in Fig. 9, as for the datasets of *bpic2012* variants, *bpic2017* variants, and *hospital_billing*, we find that the AUC values of different approaches keep increasing normally as the prefix length increases but increase dramatically after the prefix length reaches at a specific value. Furthermore, in terms of the datasets of *sepsis_cases* variants, the AUC values of different approaches are similar when the prefix length is small, but differ significantly once the length reaches at a specific value. This may indicate the occurrence of a certain key activity in business process (not the last

Table 4 The overall AUC comparison of different approaches on different datasets

Dataset	RF[1]	XGBoost[1]	LSTM	Bi-LSTM	Att-LSTM	Att-Bi-LSTM
bpic2012_accepted	0.69± 0.11	0.70± 0.11	0.69± 0.13	0.69± 0.14	0.69± 0.14	**0.71± 0.14**
bpic2012_declined	0.60± 0.06	0.62± 0.07	0.63± 0.10	0.63± 0.10	0.62± 0.10	**0.64± 0.10**
bpic2012_cancelled	0.70± 0.16	0.70± 0.15	0.71± 0.15	0.72± 0.16	0.70± 0.15	**0.73± 0.15**
bpic2017_accepted	0.83± 0.17	**0.84± 0.16**	0.83± 0.16	0.83± 0.17	0.83± 0.16	**0.84± 0.16**
bpic2017_declined	0.81± 0.15	0.81± 0.15	0.81± 0.15	**0.83± 0.15**	**0.83± 0.15**	**0.83± 0.15**
bpic2017_cancelled	0.80± 0.17	0.80± 0.17	0.81± 0.17	0.81± 0.17	0.80± 0.16	**0.82± 0.17**
sepsis_cases_1	0.69± 0.15	0.63± 0.14	0.67± 0.10	0.67± 0.09	0.65± 0.11	**0.69± 0.09**
sepsis_cases_2	0.80± 0.06	0.85± 0.11	0.86± 0.06	0.86± 0.05	0.86± 0.05	**0.87± 0.05**
sepsis_cases_3	0.72± 0.04	**0.73± 0.11**	0.64± 0.07	0.72± 0.02	0.69± 0.04	0.72± 0.02
production	0.76± 0.12	0.73± 0.11	0.78± 0.17	0.81± 0.15	0.81± 0.14	**0.84± 0.13**
traffic_fines	0.67± 0.14	0.67± 0.15	0.70± 0.11	0.70± 0.12	0.69± 0.12	**0.73± 0.12**
hospital_billing	0.72± 0.05	0.76± 0.04	0.75± 0.05	0.77± 0.06	0.75± 0.05	**0.78± 0.04**
Mean	0.73	0.74	0.74	0.75	0.74	**0.77**

[1] The results are from (Teinemaa et al. 2019)

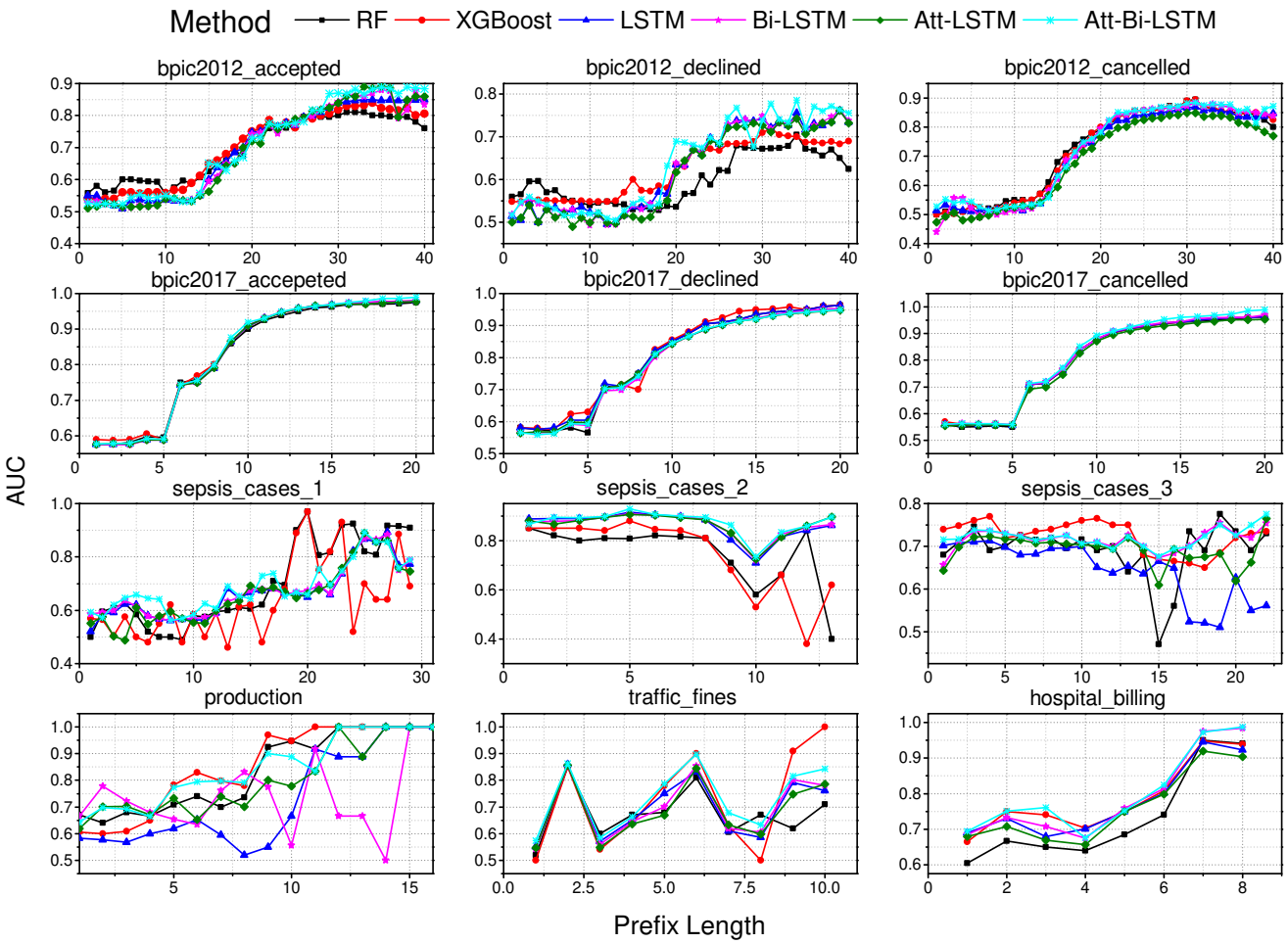


Fig. 9 The AUC comparison of outcome prediction with different approaches on different datasets.

executed event) has a decisive effect on the outcome. Among our proposed approaches, with the increase of prefix length, the prediction accuracy AUC of Att-Bi-LSTM increases more obvious than other approaches on most datasets. Nevertheless, the advantage of prediction accuracy gradually disappears as the prefix length increases to a certain value on the datasets of *sepsis_cases_1* and *traffic_fines*. The reason may be that the key events that affect the process outcome occur earlier in process execution.

5.2.2 Earliness comparison

Table 5 shows the earliness of all approaches on each dataset, which varies dramatically. As defined in Eq. (20), the value of earliness ranges between $[0, 100]$. Particularly, we find that the earliness of both XGBoost for *sepsis_cases_2* dataset and LSTM for *sepsis_cases_3* dataset is up to 100, which means that the on-going case only can be predicted accurately when it is almost finished. Furthermore, as shown in Fig. 9, we discover that the AUC value of XGBoost for *sepsis_cases_2* dataset reduces greatly with the prefix length increasing and fails to reach the AUC threshold of 0.7 finally. Similarly, the AUC of LSTM on the dataset of *sepsis_cases_3* also decreases and fails to reach 0.7 when the prefix length reaches the maximum. However, on the *sepsis_cases_2* dataset, the earliness of other approaches is relatively small. In addition, it is obvious that the earlinesses of LSTM-based approaches are similar especially on some datasets like *bpic2012_accepted*, *bpic2012_cancelled*, *bpic2017_accepted*, and *sepsis_cases_2*. In terms of the average earliness, the enhancement of attention mechanism in Att-LSTM is proved to be more effective than that of bidirectional feature extraction in Bi-LSTM, and both of them performs better than the basic LSTM. Finally, Att-Bi-LSTM combined by these two enhancements can achieve the best earliness, then followed by Att-LSTM and Bi-LSTM, while RF has the worst earliness.

5.2.3 Stability comparison

To evaluate the effectiveness of the above approaches further, Table 6 gives their temporal stabilities on each dataset. To compare with our proposed approaches, we first show the range of stability for RF (i.e. RF-based) and XGBoost (i.e. XGBoost-based) approaches respectively since the stability of these approaches based on the same classification algorithm (RF or XGBoost) varies significantly. As for our proposed LSTM-based approaches, we compute their stabilities according to Eq. (21) based on the prediction accuracy of AUC. As shown in Table 6, in most cases, the stability of XGBoost-based approaches varies but the achieved maximum can be the best among all approaches. Especially on the dataset of *sepsis_cases_1*, the stability of XGBoost-based approaches reaches 1, which means that the difference of each two successive predictions of an on-going case may always keep the same although it is not the usual situation. To compare two enhancements of feature extraction (i.e. the bidirectional feature extraction and the attention mechanism for feature extraction), we find that Bi-LSTM achieves the better stability on 9 out of 12 datasets. Moreover, in terms of the mean stability of these 12 datasets, Bi-LSTM has a better stability than that of Att-LSTM. Among these LSTM-based approaches, Att-Bi-LSTM combined with these two enhancements of feature extraction performs best in terms of mean stability. However, on 6 out of 12 datasets, the prediction of Att-Bi-LSTM is not stable than that of Att-LSTM or Bi-LSTM, which can also be found in Fig. 9.

5.2.4 Time performance comparison

In order to evaluate the efficiency of the above approaches, Table 7 shows the *offline_total* (in seconds) and the *online_avg* (in milliseconds) by making predictions for all prefix traces extracted from the test set of each dataset. Here, the *offline_total* denotes the time required to train a classifier while the *online_avg* denotes the average of time required to predict all test samples (i.e., prefix traces). Table 7 also gives the number of test samples for each dataset.

For a fair comparison, we present the range of required time for *offline_total* and *online_avg*, for RF and XGBoost respectively since the time of these two approaches varies significantly based on the selected operations. As shown in Table 7, we find that the *online_avg* of LSTM-based approaches keeps within 10 milliseconds (most of them are within 5 milliseconds) and is significantly less than that of RF and XGBoost. Meanwhile, the *offline_total* of our proposed approaches is mostly within the ranges of RF and XGBoost. These results show that LSTM-based approaches have obvious advantages in real-time prediction. In addition, we find that the two enhancements of bidirectional feature extraction and attention mechanism make Bi-LSTM, At-LSTM and Att-Bi-LSTM less efficient in terms of *offline_total* and *online_avg*, compared with the basic LSTM approach. However, compared with the improved accuracy of prediction, the small difference of time performance can be neglected.

6 Threats to Validity

In this section, we discuss some threats that may affect the validity of our study. One potential threat may come from the selected case attributes, event attributes, as well as the encoding ways of them when training classifiers. These selected attributes may interfere with the outcome of a case when training. Therefore, we extended some additional attributes of event logs as much as possible in our experiment so as to enhance event logs for training classifiers. Moreover,

Table 5 The earliness comparison of different approaches

Dataset	RF	XGBoost	LSTM	Bi-LSTM	Att-LSTM	Att-Bi-LSTM
bpic2012_accepted	50	48	50	50	50	50
bpic2012_declined	85	75	65	65	65	70
bpic2012_cancelled	50	48	50	50	50	50
bpic2017_accepted	30	30	30	30	30	30
bpic2017_declined	30	40	30	40	30	30
bpic2017_cancelled	30	30	30	30	40	30
sepsis_cases_1	62	81	79	79	79	69
sepsis_cases_2	50	100	8	8	8	8
sepsis_cases_3	58	48	100	48	58	48
production	41	31	69	55	29	31
traffic_fines	60	53	53	53	57	53
hospital_billing	75	25	38	44	44	44
Mean	52	51	50	46	45	**43**

Table 6 The temporal stability comparison of different approaches

Dataset	RF[1]	XGBoost[1]	LSTM	Bi-LSTM	Att-LSTM	Att-Bi-LSTM
bpic2012_accepted	[0.947,0.978]	[0.965,0.992]	0.972	0.972	0.974	0.971
bpic2012_declined	[0.967,0.993]	[0.978,0.999]	0.983	0.983	0.988	0.986
bpic2012_cancelled	[0.953,0.982]	[0.964,0.982]	0.976	0.976	0.976	0.976
bpic2017_accepted	[0.959,0.976]	[0.957,0.979]	0.965	0.970	0.966	0.968
bpic2017_declined	[0.983,0.993]	[0.978,0.995]	0.986	0.987	0.987	0.989
bpic2017_cancelled	[0.960,0.975]	[0.951,0.977]	0.971	0.971	0.966	0.970
sepsis_cases_1	[0.971,0.999]	[0.975,1.000]	0.995	0.999	0.999	0.994
sepsis_cases_2	[0.978,0.995]	[0.970,0.999]	0.993	0.996	0.995	0.997
sepsis_cases_3	[0.951,0.992]	[0.971,0.995]	0.991	0.991	0.989	0.993
production	[0.893,0.952]	[0.912,0.962]	0.955	0.971	0.967	0.975
traffic_fines	[0.662,0.773]	[0.706,0.762]	0.760	0.760	0.757	0.771
hospital_billing	[0.978,0.987]	[0.976,0.995]	0.989	0.990	0.991	0.988
Mean	-	-	0.961	0.964	0.963	**0.965**

[1] The results are from (Teinemaa et al. 2019)

Table 7 The comparison of execution time of different approaches

Dataset/#Test samples	RF[1]		XGBoost[1]		LSTM	
	offline_total(s)	online_avg(ms)	offline_total(s)	online_avg(ms)	offline_total(s)	online_avg(ms)
bpic2012_accepted/30,637	[29,7198]	[11,506]	[30,2638]	[4,143]	6,902	3
bpic2012_declined/30,637	[30,8060]	[10,509]	[28,585]	[4,403]	333	1
bpic2012_cancelled/30,637	[28,4076]	[10,513]	[30,2816]	[4,156]	838	3
bpic2017_accepted/124,815	[136,25283]	[24,1784]	[125,19582]	[14,1601]	19,917	2
bpic2017_declined/124,815	[132,22481]	[22,1792]	[125,15822]	[17,1488]	2,537	2
bpic2017_cancelled/124,815	[117,19949]	[13,1679]	[125,17385]	[15,1480]	1,023	1
sepsis_cases_1/2,493	[3,165]	[36,255]	[3,93]	[26,61]	62	2
sepsis_cases_2/2,493	[1,81]	[46,307]	[1,25]	[31,83]	26	2
sepsis_cases_3/2,493	[2,138]	[39,272]	[2,81]	[27,68]	107	1
production/286	[1,27]	[41,338]	[1,14]	[23,62]	12	2
traffic_fines/88,530	[424,2553]	[86,560]	[401,2895]	[59,158]	1,023	1
hospital_billing/73,528	[363,131454]	[309,930]	[316,21000]	[104,960]	7,985	5
Mean	-	-	-	-	3,397	2

Dataset/#Test samples	Bi-LSTM		Att-LSTM		Att-Bi-LSTM	
	offline_total(s)	online_avg(ms)	offline_total(s)	online_avg(ms)	offline_total(s)	online_avg(ms)
bpic2012_accepted/30,637	2,775	4	5,659	6	6,268	10
bpic2012_declined/30,637	694	1	523	1	741	2
bpic2012_cancelled/30,637	2,687	2	2,104	2	4,982	3
bpic2017_accepted/124,815	20,352	3	23,473	2	18,943	3
bpic2017_declined/124,815	10,524	2	12,498	2	11,830	2
bpic2017_cancelled/124,815	11,129	2	4,062	2	3,844	2
sepsis_cases_1/2,493	133	2	201	5	650	7
sepsis_cases_2/2,493	26	1	26	1	28	1
sepsis_cases_3/2,493	66	2	630	8	667	8
production/286	21	3	19	3	25	3
traffic_fines/88,530	1,867	2	1,453	2	2,363	2
hospital_billing/73,528	24,277	9	25,229	8	25,225	10
Mean	6,213	3	6,323	4	6,297	4

[1] The results are from (Teinemaa et al. 2019)

some available encoding ways based on aggregation involved in training a classifier may be lossy, which may cause some information to be discarded. However, in our experiment, only the RF and XGBoost approaches use such coding ways by some aggregation functions. Similarly, they also utilize the coding ways that encode each case based on their last events. However, the coding strategies employed in LSTM-based approaches are lossless, such as the one-hot coding. In addition, we choose AUC as the accuracy evaluation criteria to deal with the problem of sample imbalance. However, the measurement of AUC may also introduce some other unknown bias that we have not found it yet. However, up to now, the threats from the other existing accuracy measurements are much more serious than the AUC.

Besides, some other threats of our study about effectiveness may result from the incompleteness of the experiment. Firstly, we only tried one attention mechanism, while many other attention mechanism variants have been proposed such as the multi-dimensional attention, soft attention *vs.* hard attention, and global attention *vs.* local attention. In our opinion, the basic attention mechanism has been evaluated more effective while its variant only could be better. Secondly, although the hyperparameters are optimized using a state-of-the-art hyperparameter optimization technique, it is possible that there exists a different optimization algorithm that can be used to find better parameter settings. In our experiment, we have conducted an additional validation experiment to determine a optimized hyper-parameter, which has reduced the threat to some extent. Finally, the generalization of our study may have some limitations because the experiments were conducted only on 12 datasets from six events logs. However, this is usually inevitable for all research.

7 Conclusion and Future Work

In this paper, we proposed three approaches (Bi-LSTM, Att-LSTM, and Att-Bi-LSTM) based on the two enhancements of LSTM feature extraction (bidirectional feature extraction and attention mechanism) to predict the outcome of on-going process instances. In particular, the proposed Bi-LSTM approach adopts the bidirectional feature extraction enhancement by combining the original LSTM network from two directions, i.e., forward and backward. The proposed Att-LSTM approach utilizes attention mechanism to optimize and enhance the features captured by LSTM network. Finally, Att-Bi-LSTM was proposed by combining these two enhancements of feature extraction mentioned above. To investigate the effectiveness of these enhancements and compare with the conventional machine learning techniques, we conducted a series of extensive experiments on twelve real datasets and then analyzed the experimental results in terms of accuracy, earliness, stability, and time performance. The experiments demonstrated our proposed LSTM-based approaches can predict the outcome of running cases more effectively and efficiently than the other approaches because they can assign the different weights to bidirectionally captured features that affect the outcome.

According to the characteristics of enhancement strategies, in the near future, we will study the "concept drift" in event logs, meaning features that affect the outcome of the process change over time. Besides, we plan to investigate the incremental outcome prediction based on neural networks to avoid the duplicate offline training of the samples, i.e. the historical process instances, with the continual execution of a process. Last but not least, we plan to further study the outcome prediction task that can provide the interpretable guidances.

Acknowledgment

The work is supported by Natural Science Foundation of China (No. 62002316), Zhejiang Provincial Natural Science Foundation (No. LQ20F020017), the Key Science and Technology Project of Zhejiang (No. 2017C01010) and ARC Linkage Project of Australia (No. LP150100892).

Declarations

Funding

The work is supported by Natural Science Foundation of China (No. 62002316), Zhejiang Provincial Natural Science Foundation (No. LQ20F020017), the Key Science and Technology Project of Zhejiang (No. 2017C01010) and ARC Linkage Project of Australia (No. LP150100892).

Conflicts of interest/Competing interests

The authors declare that they have no conflict of interest.

Availability of data and material

The initial event logs are originated from https://researchdata.4tu.nl/home/ and the preprocessed datasets for analyses are available in https://github.com/jiaojiaowang1992/result_prediction/tree/main/data.

Code availability

The code is available in https://github.com/jiaojiaowang1992/result_prediction/tree/main/code.

Authors' contributions

All authors contributed to the study conception and design. Material preparation, data collection and analysis were performed by Jiaojiao Wang and Xiaoxiao Sun. The first draft of the manuscript was written by Jiaojiao Wang and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

References

Van der Aalst, W., 2016. Data science in action, in: Process Mining. Springer, pp. 3–23. https://doi.org/10.1007/978-3-662-49851-4_1.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural machine translation by jointly learning to align and translate, in: Proceedings of the 3rd International Conference on Learning Representations(ICLR). https://arxiv.org/pdf/1409.0473.pdf.

Bergstra, J.S., Bardenet, R., Bengio, Y., Kégl, B., 2011. Algorithms for hyper-parameter optimization, in: Advances in neural information processing systems 24 (NIPS 2011), pp. 2546–2554. http://papers.nips.cc/paper/4443-algorithms-for-hyper-parameter-optimization.pdf.

Bishop, C.M., 2006. Pattern recognition and machine learning. springer.

Bradley, A.P., 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. Pattern recognition 30, 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2.

Chorowski, J.K., Bahdanau, D., Serdyuk, D., Cho, K., Bengio, Y., 2015. Attention-based models for speech recognition, in: Advances in Neural Information Processing Systems 28 (NIPS 2015), pp. 577–585. http://papers.nips.cc/paper/5847-attention-based-models-for-speech-recognition.pdf.

Conforti, R., de Leoni, M., La Rosa, M., van der Aalst, W.M., ter Hofstede, A.H., 2015. A recommendation system for predicting risks across multiple business process instances. Decision Support Systems 69, 1–19. https://doi.org/10.1016/j.dss.2014.10.006.

De Leoni, M., van der Aalst, W.M., Dees, M., 2016. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Information Systems 56, 235–257. https://doi.org/10.1016/j.is.2015.07.003.

Di Francescomarino, C., Dumas, M., Maggi, F.M., Teinemaa, I., 2016. Clustering-based predictive process monitoring. IEEE transactions on services computing 12, 896–909. https://doi.org/10.1109/TSC.2016.2645153.

Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D., 2014. Do we need hundreds of classifiers to solve real world classification problems? The Journal of Machine Learning Research 15, 3133–3181. https://doi.org/10.5555/2627435.2697065.

Friedman, J.H., 2001. Greedy function approximation: a gradient boosting machine. Annals of statistics 29, 1189–1232. https://www.jstor.org/stable/2699986.

Geng, Z., Chen, G., Han, Y., Lu, G., Li, F., 2020. Semantic relation extraction using sequential and tree-structured lstm with attention. Information Sciences 509, 183–192. https://doi.org/10.1016/j.ins.2019.09.006.

Graves, A., 2012. Long short-term memory, in: Supervised sequence labelling with recurrent neural networks. Springer, pp. 37–45. https://doi.org/10.1007/978-3-642-24797-2_4.

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural computation 9, 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the national academy of sciences 79, 2554–2558. https://doi.org/10.1073/pnas.79.8.2554.

Kang, B., Kim, D., Kang, S.H., 2012. Real-time business process monitoring method for prediction of abnormal termination using knni-based lof prediction. Expert Systems with Applications 39, 6061–6068. https://doi.org/10.1016/j.eswa.2011.12.007.

Kingma, D.P., Ba, J., 2015. Adam: A method for stochastic optimization, in: Proceddings of the 3rd International Conference on Learning Representations (ICLR 2015). https://arxiv.org/pdf/1412.6980.pdf.

Kratsch, W., Manderscheid, J., Röglinger, M., Seyfried, J., 2020. Machine learning in business process monitoring: A comparison of deep learning and classical approaches used for outcome prediction. Business & Information Systems Engineering , 1–16https://doi.org/10.1007/s12599-020-00645-0.

Lakshmanan, G.T., Duan, S., Keyser, P.T., Curbera, F., Khalaf, R., 2010. Predictive analytics for semi-structured case oriented business processes, in: Proceddings of the 7th International Conference on Business Process Management, Springer. pp. 640–651. https://doi.org/10.1007/978-3-642-20511-8_59.

Leontjeva, A., Conforti, R., Di Francescomarino, C., Dumas, M., Maggi, F.M., 2016. Complex symbolic sequence encodings for predictive monitoring of business processes, in: Proceddings of the 13rd International Conference on Business Process Management, Springer. pp. 297–313. https://doi.org/10.1007/978-3-319-23063-4_21.

Li, L., Nie, Y., Han, W., Huang, J., 2017. A multi-attention-based bidirectional long short-term memory network for relation extraction, in: Proceddings of the 24th International Conference on Neural Information Processing, Springer. pp. 216–227. https://doi.org/10.1007/978-3-319-70139-4_22.

Lin, Y., Shen, S., Liu, Z., Luan, H., Sun, M., 2016. Neural relation extraction with selective attention over instances, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 2124–2133. https://doi.org/10.18653/v1/P16-1200.

Liu, P., Qiu, X., Huang, X., 2016. Recurrent neural network for text classification with multi-task learning, in: Proceedings of the 25th International Joint Conference on Artificial Intelligence, pp. 2873–2879. https://www.ijcai.org/Proceedings/16/Papers/408.pdf.

Maggi, F.M., Di Francescomarino, C., Dumas, M., Ghidini, C., 2014. Predictive monitoring of business processes, in: International conference on advanced information systems engineering, Springer. pp. 457–472. https://doi.org/10.1007/978-3-319-07881-6_31.

Mehdiyev, N., Evermann, J., Fettke, P., 2020. A novel business process prediction model using a deep learning method. Business & information systems engineering 62, 143–157.

Metzger, A., Leitner, P., Ivanović, D., Schmieders, E., Franklin, R., Carro, M., Dustdar, S., Pohl, K., 2015. Comparing and combining predictive business process monitoring techniques. IEEE Transactions on Systems, Man, and Cybernetics: Systems 45, 276–290. https://doi.org/10.1109/TSMC.2014.2347265.

Mnih, V., Heess, N., Graves, A., et al., 2014. Recurrent models of visual attention, in: Advances in neural information processing systems 27 (NIPS 2014), pp. 2204–2212. http://papers.nips.cc/paper/5542-recurrent-models-of-visual-attention.pdf.

Nie, Y.p., Han, Y., Huang, J.m., Jiao, B., Li, A.p., 2017. Attention-based encoder-decoder model for answer selection in question answering. Frontiers of Information Technology & Electronic Engineering 18, 535–544. https://doi.org/10.1631/FITEE.1601232.

Pauwels, S., Calders, T., 2020. Bayesian network based predictions of business processes, in: International Conference on Business Process Management, Springer. pp. 159–175. https://doi.org/10.1007/978-3-030-58638-6_10.

Rensink, R.A., 2000. The dynamic representation of scenes. Visual cognition 7, 17–42. https://doi.org/10.1080/135062800394667.

Rogge-Solti, A., Weske, M., 2013. Prediction of remaining service execution time using stochastic petri nets with arbitrary firing delays, in: International Conference on Service-Oriented Computing, Springer. pp. 389–403. https://doi.org/10.1007/978-3-642-45005-1_27.

Schafer, J.L., Graham, J.W., 2002. Missing data: our view of the state of the art. Psychological methods 7, 147–177. https://doi.org/10.1037/1082-989X.7.2.147.

Senderovich, A., Di Francescomarino, C., Ghidini, C., Jorbina, K., Maggi, F.M., 2017. Intra and inter-case features in predictive process monitoring: a tale of two dimensions, in: Proceddings of the 14th International Conference on Business Process Management, Springer. pp. 306–323. https://doi.org/10.1007/978-3-319-65000-5_18.

Shang, L., Lu, Z., Li, H., 2015. Neural responding machine for short-text conversation, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pp. 1577–1586. https://doi.org/10.3115/v1/P15-1152.

Sun, X., Hou, W., Ying, Y., Yu, D., 2020. Remaining time prediction of business processes based on multilayer machine learning, in: 2020 IEEE International Conference on Web Services (ICWS), IEEE. pp. 554–558. https://doi.org/10.1109/ICWS49710.2020.00080.

Tama, B.A., Comuzzi, M., Ko, J., 2020. An empirical investigation of different classifiers, encoding, and ensemble schemes for next event prediction using business process event logs. ACM Transactions on Intelligent Systems and Technology (TIST) 11, 1–34.

Tax, N., Verenich, I., La Rosa, M., Dumas, M., 2017. Predictive business process monitoring with lstm neural networks, in: International Confer-

ence on Advanced Information Systems Engineering, Springer. pp. 477–492. https://doi.org/10.1007/978-3-319-59536-8_30.

Teinemaa, I., Dumas, M., Leontjeva, A., Maggi, F.M., 2018. Temporal stability in predictive process monitoring. Data Mining and Knowledge Discovery 32, 1306–1338. https://doi.org/10.1007/s10618-018-0575-9.

Teinemaa, I., Dumas, M., Maggi, F.M., Di Francescomarino, C., 2016. Predictive business process monitoring with structured and unstructured data, in: Proceddings of the 13rd International Conference on Business Process Management, Springer. pp. 401–417. https://doi.org/10.1007/978-3-319-45348-4_23.

Teinemaa, I., Dumas, M., Rosa, M.L., Maggi, F.M., 2019. Outcome-oriented predictive process monitoring: Review and benchmark. ACM Transactions on Knowledge Discovery from Data (TKDD) 13, 1–57. https://doi.org/10.1145/3301300.

Tieleman, T., Hinton, G., 2012. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 4, 26–31.

Verenich, I., Dumas, M., La Rosa, M., Maggi, F.M., Di Francescomarino, C., 2016. Complex symbolic sequence clustering and multiple classifiers for predictive process monitoring, in: Proceddings of the 13rd International Conference on Business Process Management, Springer. pp. 218–229. https://doi.org/10.1007/978-3-319-42887-1_18.

Verenich, I., Dumas, M., Rosa, M.L., Maggi, F.M., Teinemaa, I., 2019. Survey and cross-benchmark comparison of remaining time prediction methods in business process monitoring. ACM Transactions on Intelligent Systems and Technology 10, 1–34. https://doi.org/10.1145/3331449.

Wang, J., Yu, D., Liu, C., Sun, X., 2019. Outcome-oriented predictive process monitoring with attention-based bidirectional lstm neural networks, in: 2019 IEEE International Conference on Web Services (ICWS), pp. 360–367. https://doi.org/10.1109/ICWS.2019.00065.

Wu, Y., Mao, H., Yi, Z., 2018. Audio classification using attention-augmented convolutional neural network. Knowledge-Based Systems 161, 90–100. https://doi.org/10.1016/j.knosys.2018.07.033.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., Bengio, Y., 2015. Show, attend and tell: Neural image caption generation with visual attention, in: Proceedings of the 32nd International conference on machine learning, pp. 2048–2057. http://proceedings.mlr.press/v37/xuc15.pdf.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., Hovy, E., 2016. Hierarchical attention networks for document classification, in: Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 1480–1489. https://doi.org/10.18653/v1/N16-1174.

Zeng, D., Liu, K., Lai, S., Zhou, G., Zhao, J., et al., 2014. Relation classification via convolutional deep neural network, in: Proceedings of the 25th International Conference on Computational Linguistics: Technical Papers, pp. 2335–2344. https://doi.org/10.1.1.656.5606.

Zhang, D., Wang, D., 2018. Relation classification via recurrent neural network. Big Data Mining and Analytics 1, 234–244. https://doi.org/10.26599/BDMA.2018.9020022.

Zhou, P., Shi, W., Tian, J., Qi, Z., Li, B., Hao, H., Xu, B., 2016. Attention-based bidirectional long short-term memory networks for relation classification, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pp. 207–212. https://doi.org/10.18653/v1/P16-2034.

Zhu, X., Li, Z., Li, X., Li, S., Dai, F., 2020. Attention-aware perceptual enhancement nets for low-resolution image classification. Information Sciences 515, 233–247. https://doi.org/10.1016/j.ins.2019.12.013.

Figures

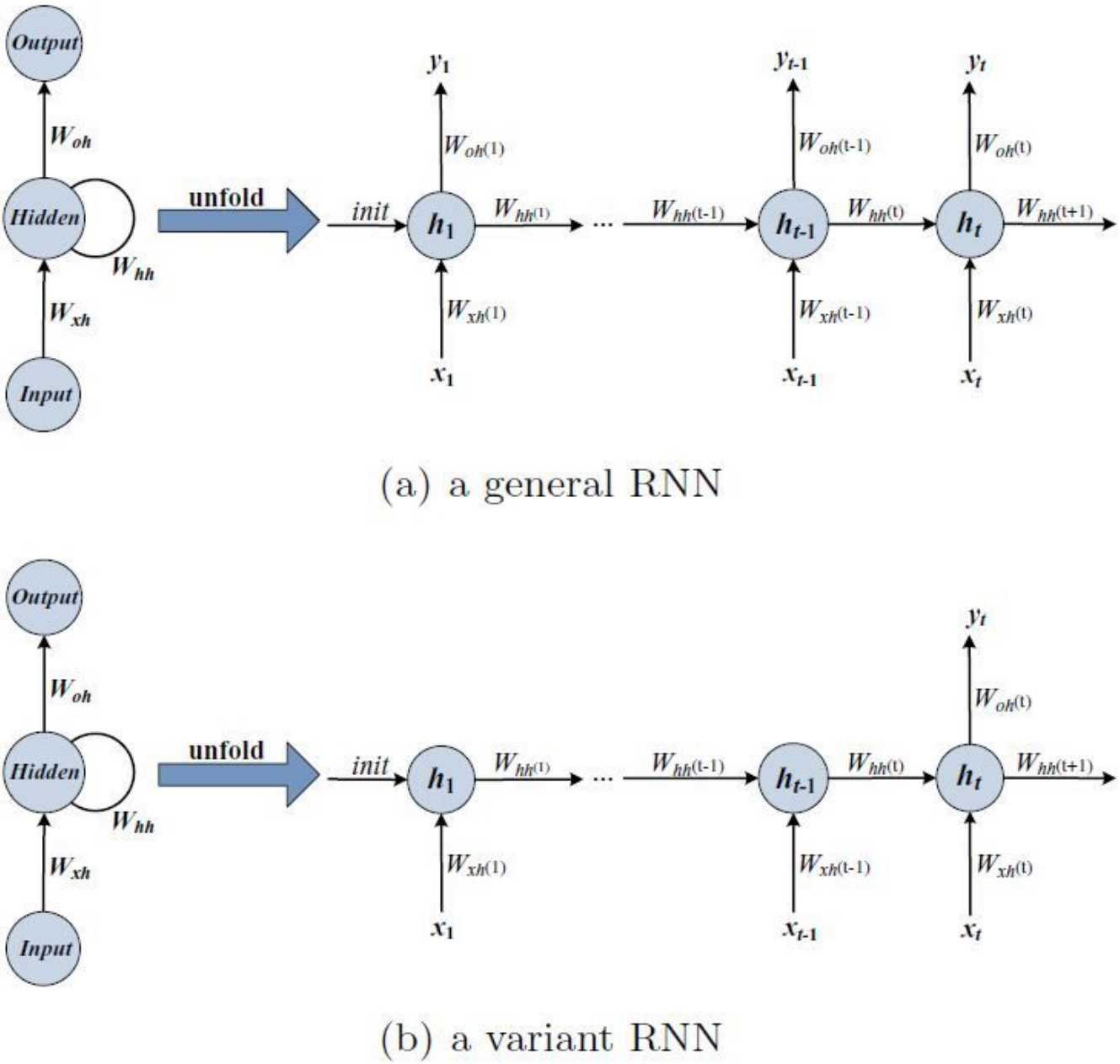


Figure 1

Two structures of a general RNN and its variant

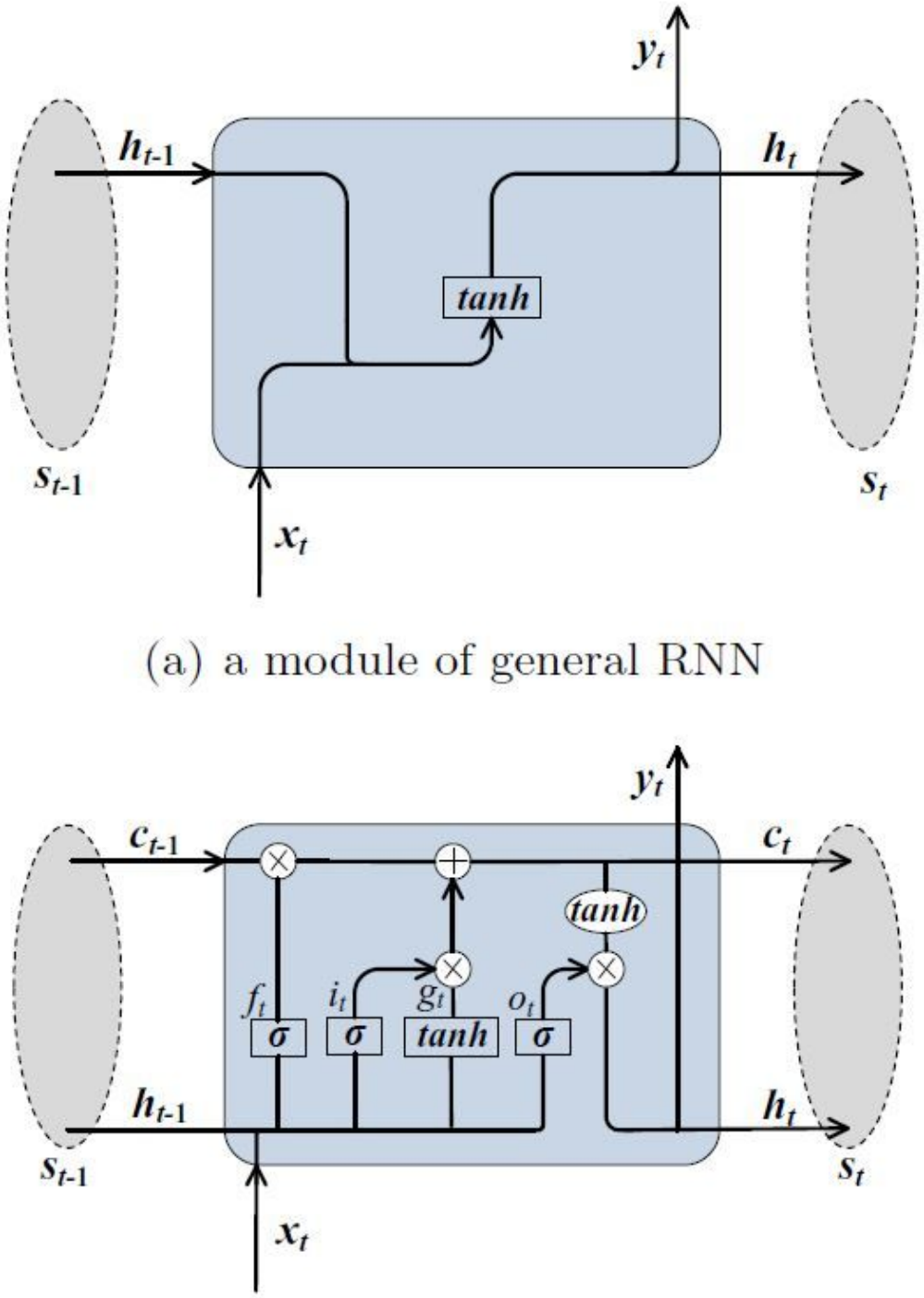


Figure 2

The neural unit of RNN and LSTM networks respectively

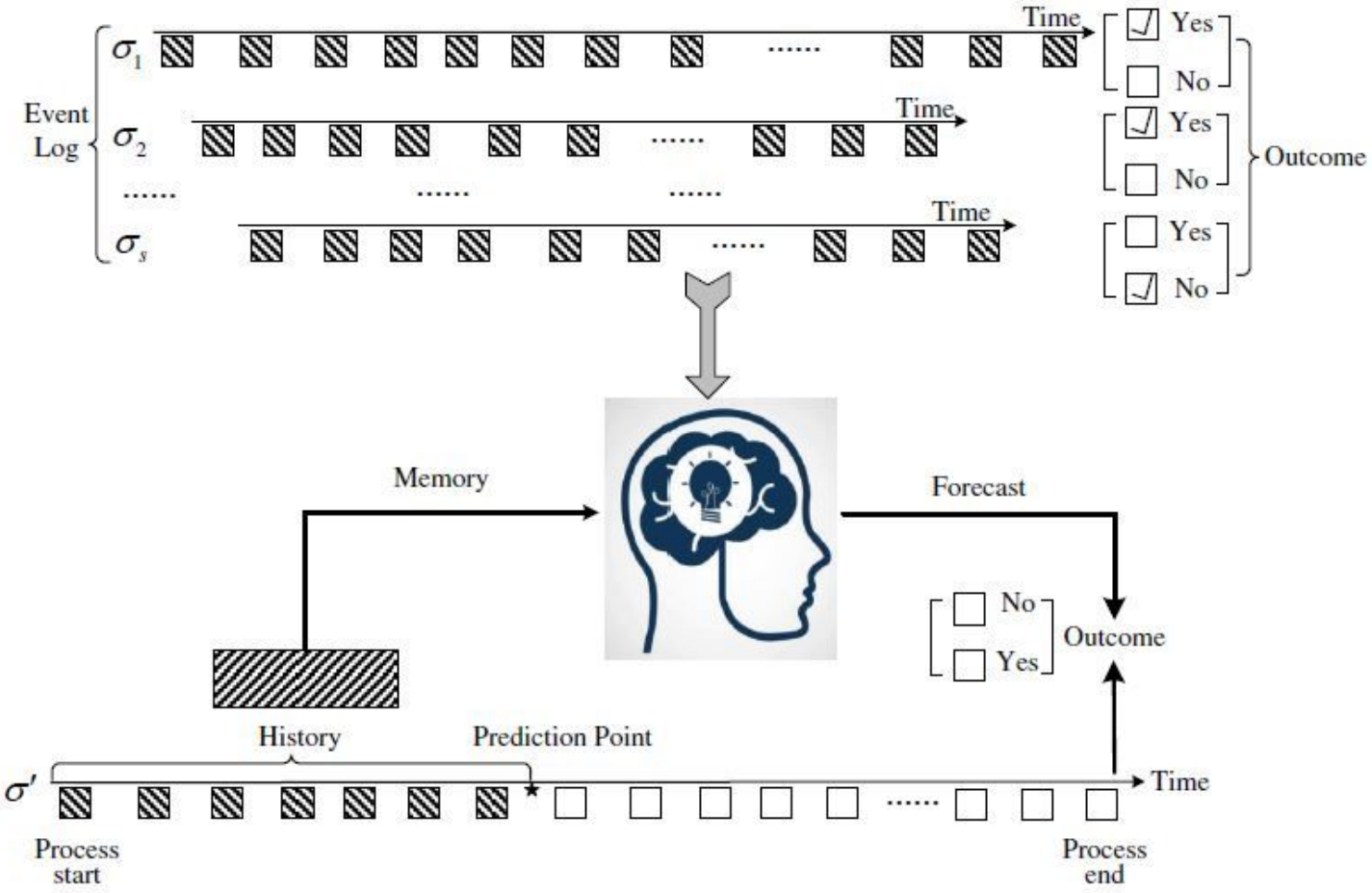


Figure 3

Outcome prediction of an on-going case in a business process.

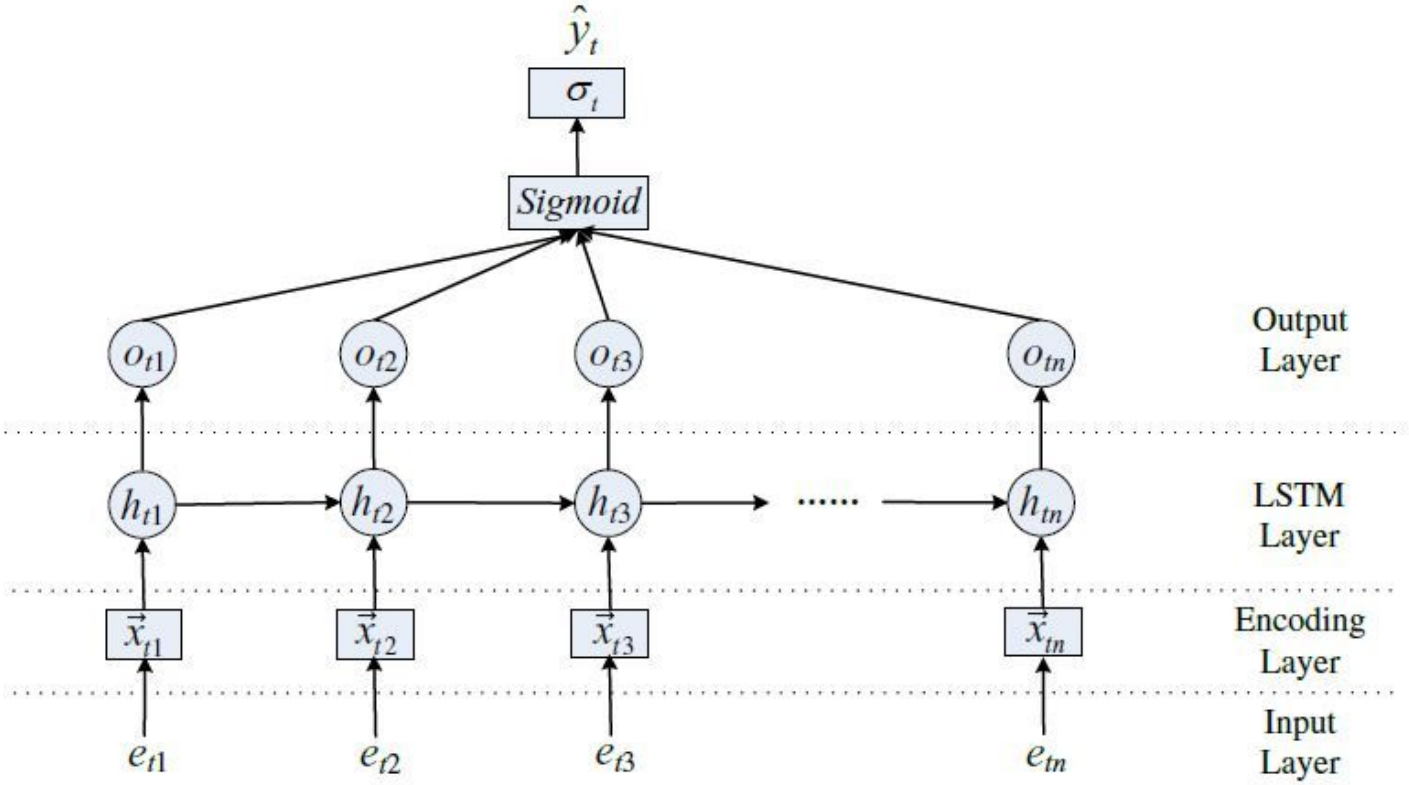


Figure 4

The architecture of LSTM approach for process outcome prediction.

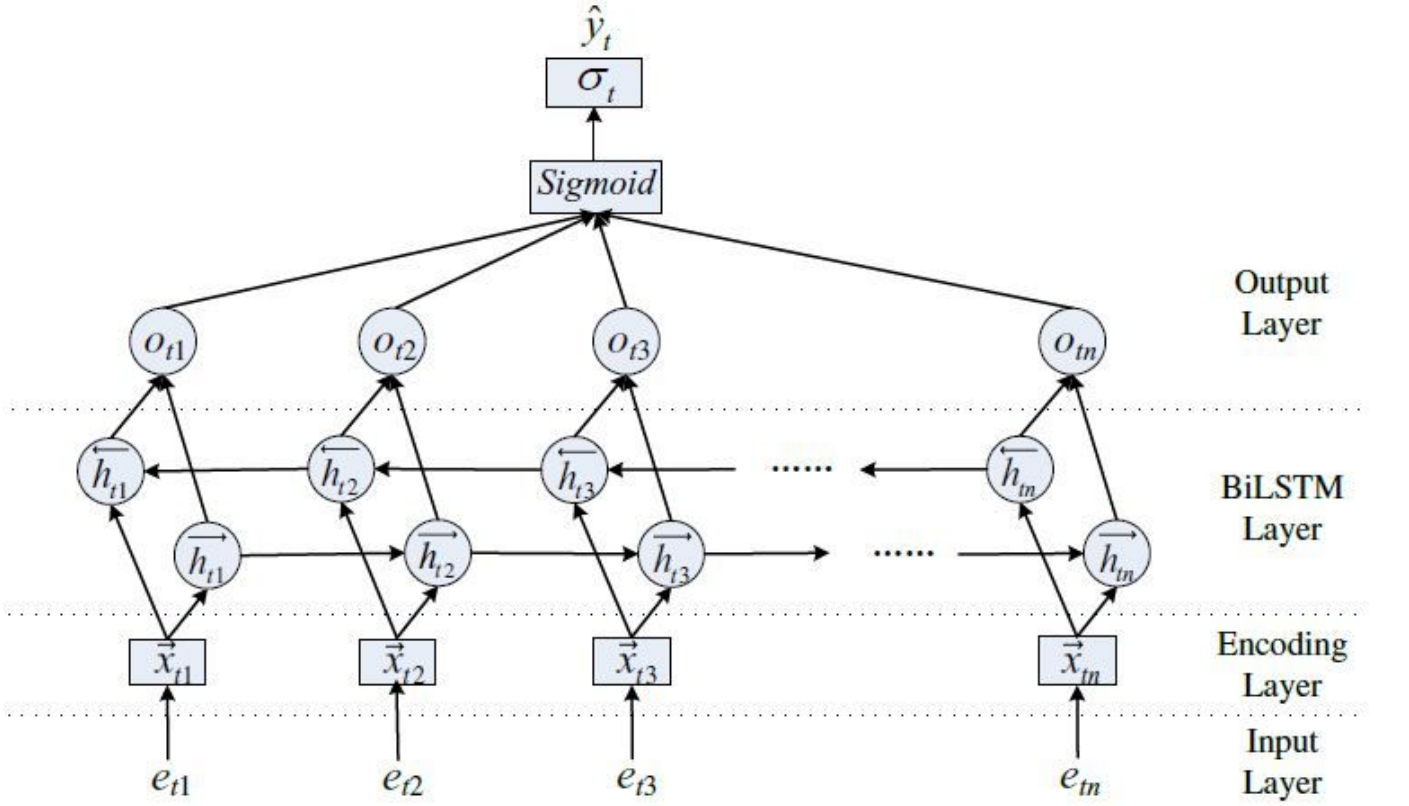


Figure 5

The architecture of Bi-LSTM approach for process outcome prediction.

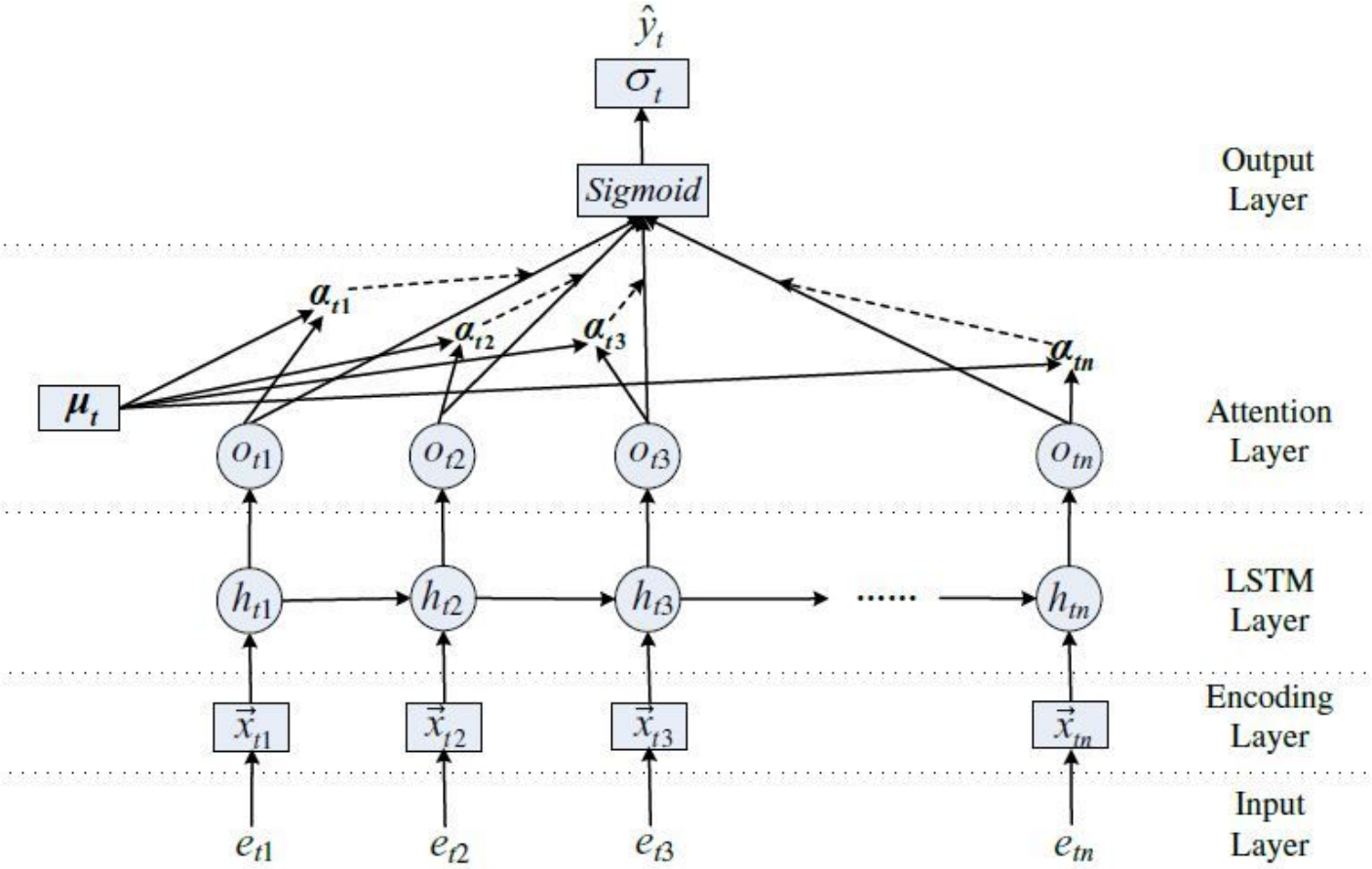


Figure 6

The architecture of Att-LSTM approach for process outcome prediction.

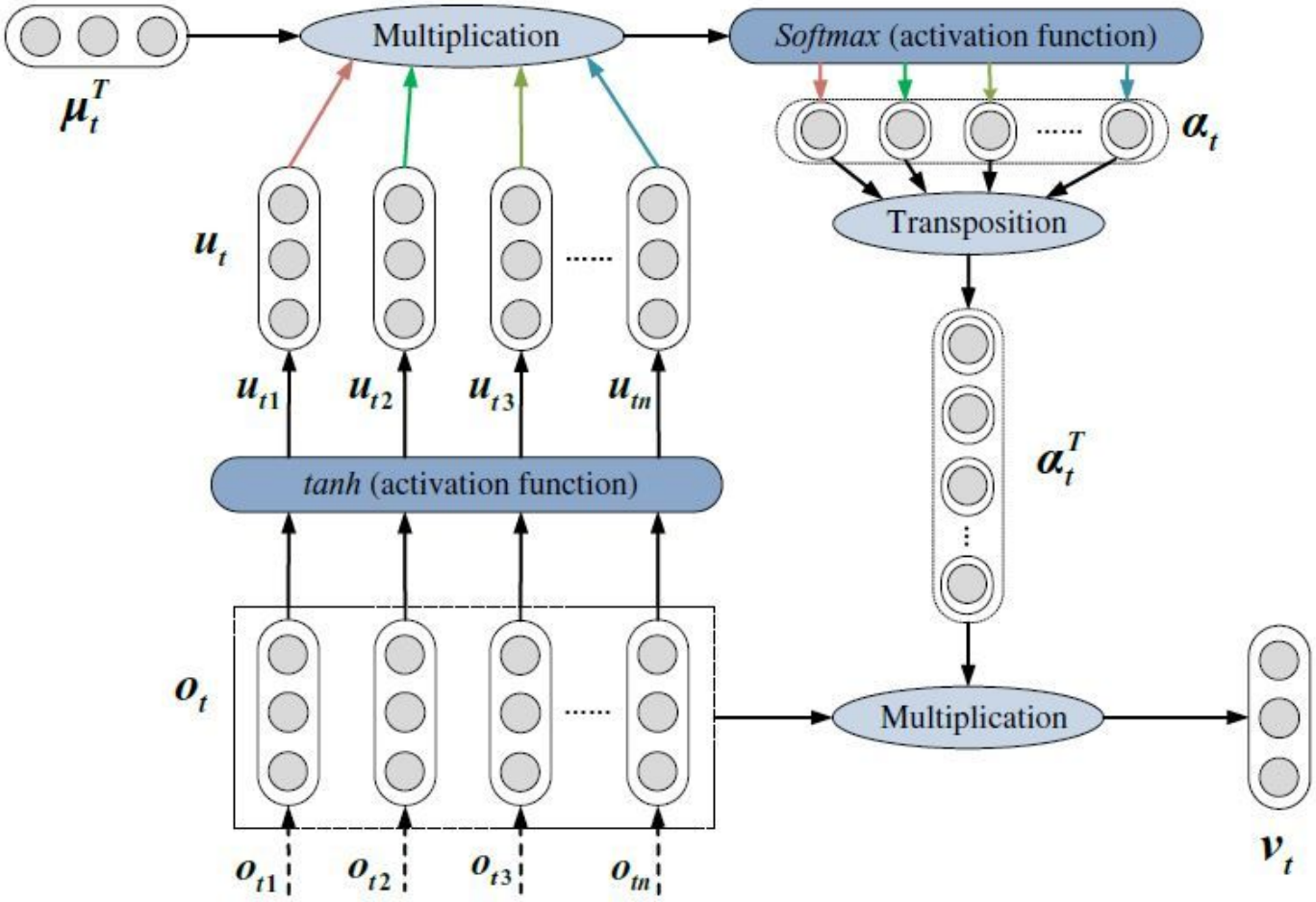


Figure 7

The process of attention mechanism.

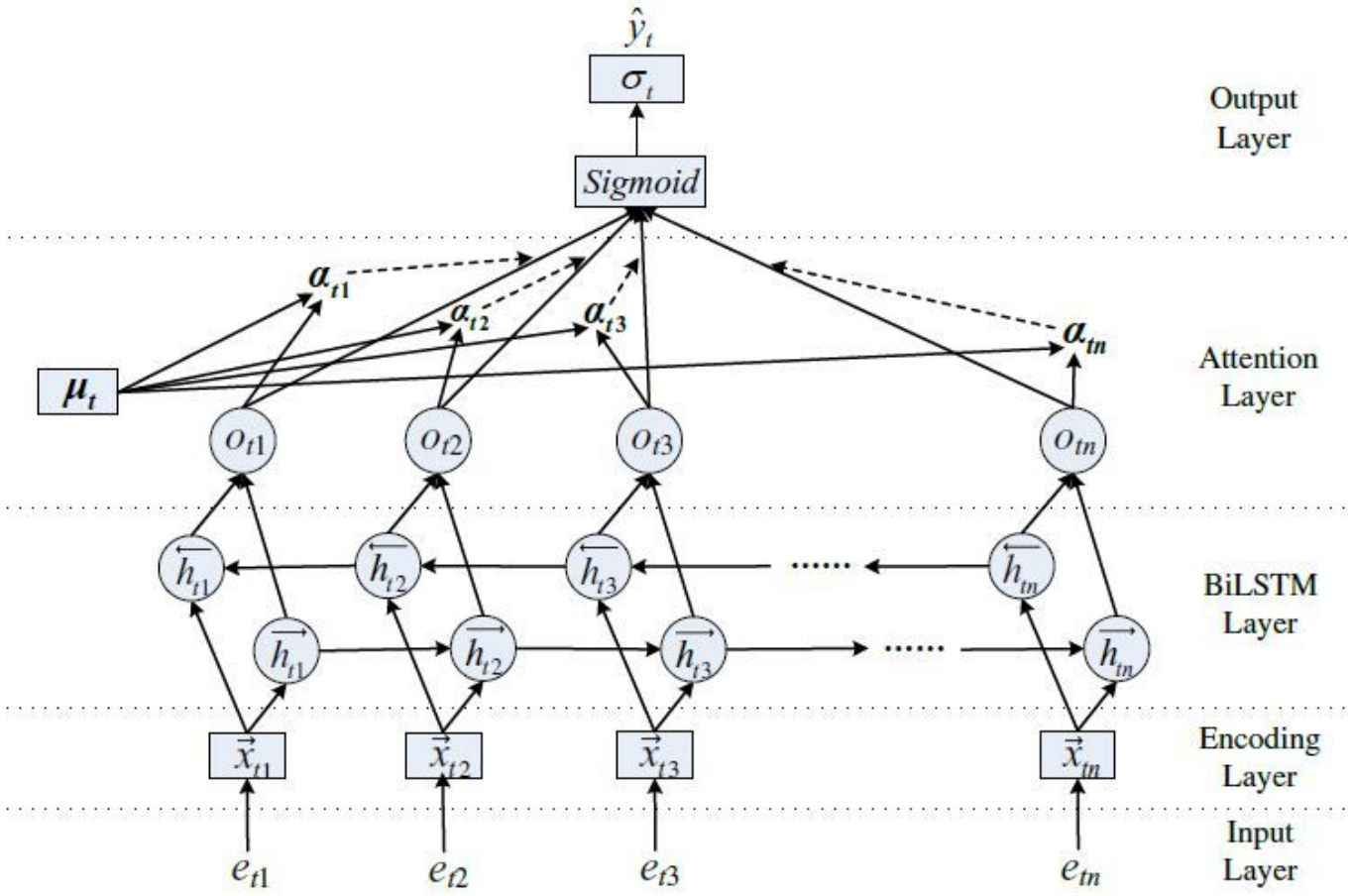


Figure 8

The structure of Att-Bi-LSTM approach for process outcome prediction.

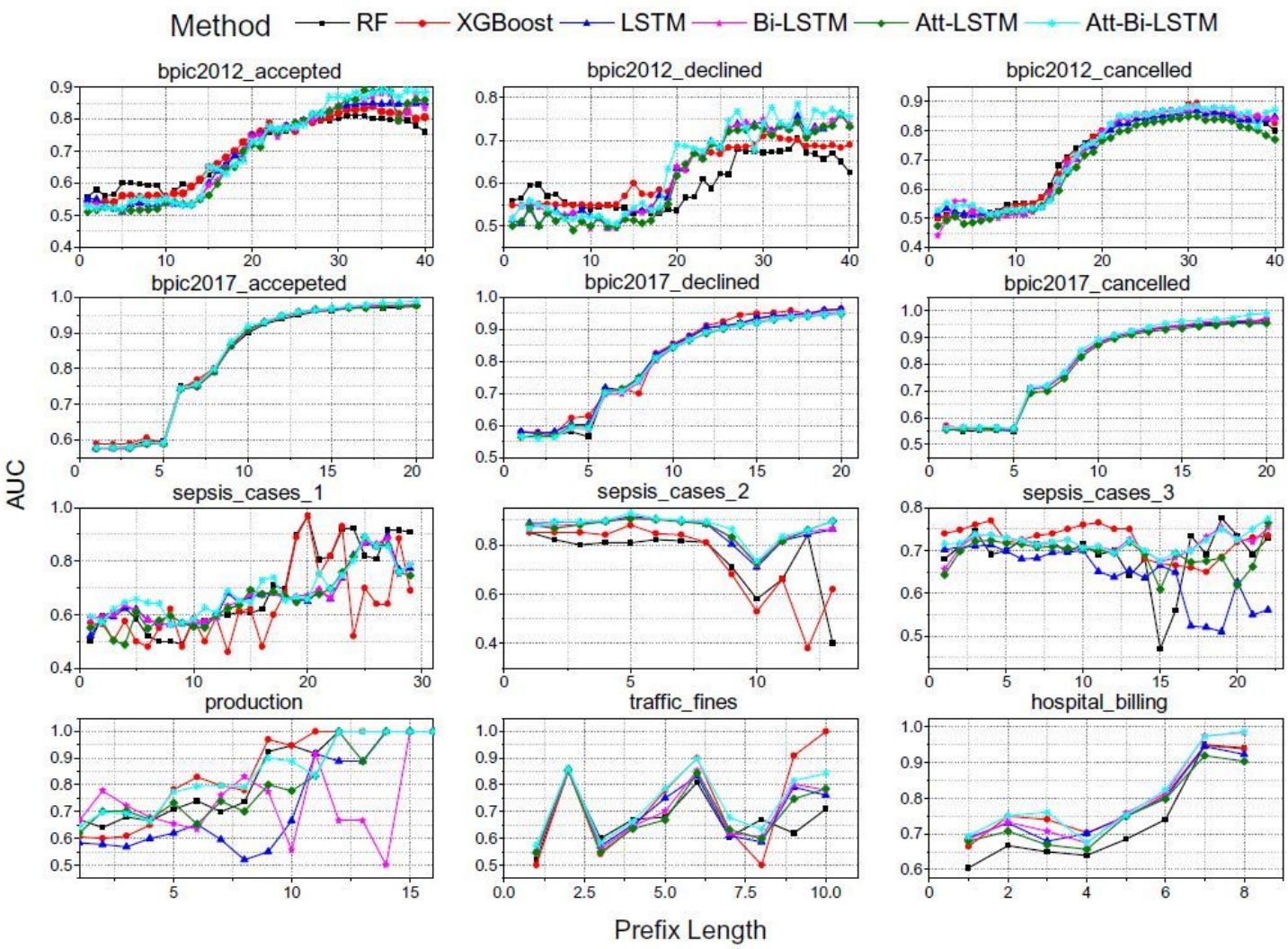


Figure 9

The AUC comparison of outcome prediction with different approaches on different datasets.