Knowledge-Based Systems 254 (2022) 109603

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Multi-task prediction method of business process based on BERT and Transfer Learning

Hang Chen [a,b], Xianwen Fang [a,b,*], Huan Fang [a]

[a] School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China
[b] Anhui Province Engineering Laboratory for Big Data Analysis and Early Warning Technology of Coal Mine Safety, Huainan, China

ARTICLE INFO

Article history:
Received 19 April 2022
Received in revised form 20 July 2022
Accepted 3 August 2022
Available online 6 August 2022

Keywords:
Predictive business process monitoring
Transfer Learning
Transformer
BERT
Masked Activity Model

ABSTRACT

Predictive Business Process Monitoring (PBPM) is one of the essential tasks in Business Process Management (BPM). It aims to predict the future behavior of an ongoing case using completed cases of a process stored in the event log, such as the prediction of the next activity and outcome of the case, etc. Although various deep learning methods have been proposed for PBPM, none of them consider the simultaneous application to multiple predictive tasks. This paper proposes a multi-task prediction method based on BERT and Transfer Learning. First, the method performs the Masked Activity Model (MAM) of a self-supervised pre-training task on many unlabeled traces using BERT (Bidirectional Encoder Representations from Transformers). The pre-training task MAM captures the bidirectional semantic information of the input traces using the bidirectional Transformer structure in BERT. It obtains the long-term dependencies between activities using the Attention mechanism in the Transformer. Then, the universal representation model of the traces is obtained. Finally, two different models are defined for two prediction tasks of the next activity and the outcome of the case, respectively, and the pre-trained model is transferred to the two prediction models for training using the fine-tuning strategy. Experiments evaluation on eleven real-world event logs shows that the performance of the prediction tasks is affected by different masking tactics and masking probabilities in the pre-training task MAM. This method performs well in the next activity prediction task and the case outcome prediction task. It can be applied to several different prediction tasks faster and with more outstanding performance than the direct training method.

1. Introduction

Predictive Business Process Monitoring (PBPM) is one of the important research directions in Business Process Management (BPM). Intend to avoid resource conflicts and execution timeouts during business process cases by predicting the future behavior of running process instances, thus improving the quality and efficiency of business process [1]. PBPM builds prediction models from historical data in the event log to address different prediction tasks, such as predicting the next activity [2–4], predicting the outcome of the case [5–7], predicting the remaining cycle time [8,9], et al.

In recent years, deep learning approaches have been increasingly used to handle various prediction tasks. Evermann et al. [10] have made preliminary attempts to predict the next activity of a running trace using Long Short-Term Memory (LSTM) networks. Tax et al. [2] have extended this research by using one-hot encoding for events and LSTM to go through multiple business processes prediction tasks, including the next activity, the timestamp of the next event, and the remaining time of the case. Camargo et al. [9] improved both of these efforts by encoding the events with an embedding technique that includes the activity attribute and other numerical and categorical attributes, enabling the LSTM to predict the next activity, the timestamp of the next event, and their associated resource. Convolutional Neural Networks (CNN) was first used to predict the next activity by Pasquadibisceglie et al. [4]. Teinemaa et al. [11] started to use LSTM to solve problems of the outcome prediction. Pasquadibisceglie et al. [6] use image processing techniques and CNN to predict the outcome of the ongoing case. These studies applied a deep learning method to a specific prediction task and did not generalize to other prediction tasks. Although literature [2,9] can perform multiple prediction tasks simultaneously (e.g., next activity prediction task, remaining event sequence prediction task, and case remaining time prediction task), to be applied to other prediction tasks (e.g., outcome prediction task) requires reconstructing the prediction model and spending a lot of data and time to train it from scratch.

* Corresponding author at: School of Mathematics and Big Data, Anhui University of Science and Technology, Huainan, China.
E-mail address: xwfang@aust.edu.cn (X. Fang).

https://doi.org/10.1016/j.knosys.2022.109603
0950-7051/

To address this problem, we need a new research approach that can be applied to a variety of prediction tasks rapidly and efficiently. Folino et al. [7] proposed a novel outcome prediction method for the problem of sparse outcome labels for training traces in realistic scenarios, which uses LSTM to learn sufficiently general trace representations from the unlabeled traces. And then, these representations are reused in the outcome prediction model using a fine-tuning strategy, trained with a small number of traces with outcome labels to fit the model quickly. However, the method has only been investigated for the outcome prediction task. The generic trace representations learned through LSTM have not been applied to other prediction tasks, so it cannot demonstrate that the method can be applied to multiple different prediction tasks. Furthermore, long-term dependencies between activities may theoretically be captured via LSTM. However, in practice, LSTM is difficult to train and cannot capture particularly long-term dependencies because the input time step is fixed and the input at the current moment depends on the output at the previous moment.

With the emergence of more and more application scenarios, A large amount of labeled data is required for model training. However, Transfer Learning has attracted more and more attention because labeled data is harder to obtain. Transfer learning is the process of transferring the trained model parameters to the new model to assist the new model in training and fitting quickly. In this paper, we propose a multi-task prediction method based on Transfer Learning by combining the unsupervised pre-training model BERT (**B**idirectional **E**ncoder **R**epresentations from **T**ransformers) [12] with a new pre-training task Masked Activity Model (MAM). The pre-training task MAM captures deeper bidirectional information of the trace by the bidirectional Transformer structure in BERT. Eventually, a sufficiently generalized trace representation is obtained through training. After the completion of the pre-training task MAM, the BERT model was transferred to the next activity prediction model and the outcome prediction model respectively. Then, the next activity prediction model and the outcome prediction model are trained separately using a fine-tuning strategy to finally obtain a prediction model with good performance. In addition, the method processes the input trace as a whole using the Attention mechanism in the Transformer, allowing parallel computation and improving training efficiency. And it sets the distance of each activity in the trace as a constant to capture the long-term dependencies between activities. The main contributions of this paper are as follows:

1. We apply BERT and Transfer Learning to several different PBPM tasks for the first time.
2. For the PBPM task, we propose a new pre-training task MAM to learn a generic representation of traces and explore the impact of masking strategies and masking probabilities on the performance of different prediction tasks on eleven real-world event logs.
3. We compare the results of our method with several state-of-the-art prediction methods for the next activity and outcome, respectively, considering eleven real-world event logs, and evaluate the performance of our method on two prediction tasks of the next activity and outcome.
4. We compare our method with the benchmark method based on eleven real-world event logs to explore the effectiveness of our method in several different PBPM tasks.

The rest of the paper is structured as follows. First, we discuss the related work on PBPM in Section 2. We present the required background for this study in Section 3. The proposed method in this research is detailed in Section 4. Section 5 discusses our experimental setup and the findings of the experiment. Finally, Section 6 summarizes our contributions and discusses and discussing the direction of future work.

2. Related work

Recently, many works have been done on how to improve the accuracy of predicting the future behavior of ongoing cases. Traditional approaches rely on explicit process models in Hidden Markov Models or State Transition Systems. For example, Lakshmanan et al. [13] proposed an instance-specific Hidden Markov Model to predict the likelihood of future behavior of process cases. Letham et al. [14] proposed a generative model based on a Bayesian rule list to assess patient stroke risk. These methods have good prediction results but can only be applied to short and straightforward process cases, and once the process case is too long, the prediction performance decreases. This problem has been improved with the advent of deep learning techniques. Rama-Maneiro et al. [15] conducted a systematic literature review of methods for solving prediction tasks using deep learning techniques. Ten different methods were experimentally evaluated on twelve real-world event logs. Teinemaa et al. [5] systematically described the outcome prediction methods involved in nine real-world event logs covering twenty-four outcome prediction tasks with an experimental evaluation of eleven methods. Due to the interdisciplinary nature of process variant analysis, there is no unified view in this field. Therefore, Taymouri et al. [16] systematically described the research on process variant analysis methods, classified the methods, and identified the gaps in this field.

In terms of the next activity prediction task, Evermann et al. [17] employed embedding techniques to encode categorical and numerical attributes of events and concatenated them together as input to an RNN, inspired by natural language processing (NLP) work. One of the advantages of this paper is that the event logs are mapped as text documents through embedding techniques, where each case is considered a sentence and each event in a case as a word. Tax et al. [2] chose LSTM networks to predict the future behavior of ongoing cases, i.e., the next activity, the timestamp of the next event, and the remaining time of the case. They not only encoded the activity attributes using one-hot encoding but also used the timestamps of the events as input vectors. They then compare the prediction performance of the three different architectures of the prediction models proposed in the paper on two public event logs. The results show that using a shared multi-task layer improves predictive performance.

Both [2,17] employ LSTM to predict the next activity, while Pasquadibisceglie et al. [4] attempted to use CNN to solve the problem of next activity prediction. They adopted image processing technology for the first time, converting the cases in the event log into image structure data. The prefix traces were represented by two-dimensional images composed of activity frequency and time series. However, this method does not fully embed the case into the spatial structure and cannot fully utilize the filtering and pooling operations of CNN during training. Mauro et al. [18] borrowed ideas from Inception [19]. They used an Inception-based CNN architecture to predict the next activity of a running trace, and experiments on three publicly real-world event logs demonstrate that this architecture outperforms the RNN architecture in terms of computational efficiency and prediction accuracy. Taymouri et al. [20] proposed an adversarial training framework based on Generative Adversarial Networks (GANs) to overcome the problem of the next activity prediction due to insufficient training data or poor prediction model architectures. Before the approach proposed in this paper, Bukhsh et al. [21] used the Encoder module of the Transformer model based on the Attention mechanism for the next activity prediction task on nine public real-world event logs and showed good prediction performance. Attributes in an event are not equally important. Jalayer et al. [22] used two Attention mechanisms in LSTM to determine which

attributes are more important at the attribute level and predict the next event at the event level.

Regarding the outcome prediction task, Teinemaa et al. [11] defined the concept of temporal stability for the outcome prediction task of PBPM. They evaluated the temporal stability and accuracy of existing methods. Finally, it was experimentally demonstrated that XGBoost and LSTM methods based on trace indexing patterns exhibit the highest temporal stability. Pasquadibisceglie et al. [6] extended the image representation of prefix trace employed in [4], where a continuum pattern of data appears in adjacent pixels of the image, thus enabling accurate prediction of case outcomes. The above methods use deep learning techniques to conduct experiments, resulting in a black-box model, which makes the prediction results difficult to interpret. Inspired by the field of interpretable artificial intelligence (XAI), Pasquadibisceglie et al. [23] proposed a fully interpretable outcome prediction model that balances prediction performance and interpretability. In addition, Wickramanayake et al. [24] proposed two different prediction models by using different connection modes of feature vectors, namely shared attention-based model and specialized attention-based model. These two models apply interpretability directly to the prediction model through two types of attention, namely event attention and attribute attention, to interpret the prediction results.

Many researchers have attempted to apply different deep learning methods to various prediction tasks. However, these methods are only studied for a particular prediction task with a particular model and cannot be quickly applied to other prediction tasks. Deep learning methods applied to relevant PBPM tasks will inevitably require a large amount of data and training time as more and more PBPM application scenarios emerge. Therefore, a new research approach is now needed that can be quickly and effectively generalized to several different prediction tasks.

3. Background

3.1. Definitions

This section will introduce some basic concepts needed for this paper, defined as follows.

Definition 1 (*Event*)**.** An event is associated with an activity in a business process. In this paper, an event e is denoted by a tuple $e = a, c, t$, where $a \in A$ is the activity associated with the event in execution, $c \in N^+$ is the case ID, and $t \in N^+$ is the timestamp of the event.

Definition 2 (*Trace*)**.** A trace is a finite nonempty sequence $\sigma = \langle e_1, e_2, \ldots, e_{|\sigma|} \rangle$ of events e, where each event occurs only once, i.e., for $\forall 1 \leq i < j \leq |\sigma|$, there is $\sigma(i) \neq \sigma(j)$, $e_i.c = e_j.c$, $e_i.t < e_j.t$.

Definition 3 (*Prefix Trace*)**.** The prefix trace σ^k is a subsequence of the first k events starting from the start event of the trace σ, i.e., $\sigma^k = \langle e_1, e_2, \ldots, e_k \rangle$ $(1 \leq k \leq |\sigma|)$. Thus, a trace is a complete case, including start and end events, while a prefix trace is an ongoing case.

Definition 4 (*Case*)**.** Case c is an execution of the process model. Each event in the event log needs to correspond to a case. In this paper, a case c is denoted by a tuple $c = n, \sigma$, where $n \in N^+$ is the attribute ID of the case, and the trace σ is a mandatory attribute that every case has.

Definition 5 (*Event Log*)**.** The event log is the set L of cases c where each event occurs at most once in the entire event log, i.e., $\forall c_1, c_2 \in L$ with $c_1 \neq c_2$.

Definition 6 (*Multi-task Prediction Problem*)**.** Let $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ be a trace with the outcome label of length n. The multi-task prediction problem lies in predicting the next activity and outcome of the running trace.

The label function y to be the function that maps the trace σ to its outcome label, i.e.:

$$y(\sigma) \in \{0, 1\}$$

The next activity prediction function f_a, which predicts the next activity e' of the prefix trace σ^k, where $k \in [1, n-1]$, i.e.:

$$f_a\left(\sigma^k\right) = e_{k+1}{}'.a$$

The outcome prediction function f_o, which predicts the outcome $y(\sigma)'$ of the prefix trace σ^k, where $k \in [1, n-1]$, i.e.:

$$f_o\left(\sigma^k\right) = y(\sigma)'$$

3.2. Recurrent neural network

When dealing with sequential data such as text, traditional feedforward neural networks do not work well with such data because they cannot read the input order of sequential data and can only consider the input data individually. Recurrent neural networks (RNNs) are similar to feedforward neural networks [25]. They are a particular neural network specializing in processing temporal data structures and can mine the data for temporal and semantic information. RNNs can generate outputs using inputs from the current time step and information from the past. In this way, RNNs can capture the dependencies between input data. However, the RNN cannot capture the long-term dependencies between the input data effectively when the input data is too lengthy. Because the hidden state of the RNN at each moment is not only determined by the input at that moment but also depends on the value of the hidden layer at the previous moment. The RNN will forget the initial moment if the input data is too lengthy, leading to gradient disappearance and explosion during the training phase.

3.3. Long short-term memory

The problem of "long-term dependence" is common when using RNNs, and gradient disappearance and gradient explosion are among the key reasons that plague the training of RNNs. Long Short-Term Memory networks, which are neural networks that can remember long and short-term information [26], address this problem by adding three additional gating units. The input gate is responsible for processing the input at the current sequence position and determining how the information is stored in the cell state. The forgetting gate allows the neural network to forget the unimportant parts of the input data. The output gate is responsible for producing the output at the current moment. It allows the LSTM to better capture long-term dependencies between the input data by controlling which information in the input data is retained over time and which information is forgotten.

3.4. Transformer

RNNs were mainly used to deal with various timing problems before Transformer was proposed, but RNNs are inherently flawed and have a gradient disappearance problem. The main explanation is that the output of the current moment impacts directly on itself in the next moment in their recursive approach. The gradient disappearance problem will occur if the largest eigenvalue in its weight matrix is less than 1. The Transformer was proposed by Vaswani et al. [27] and has been widely used in machine translation and natural language, and recently extended

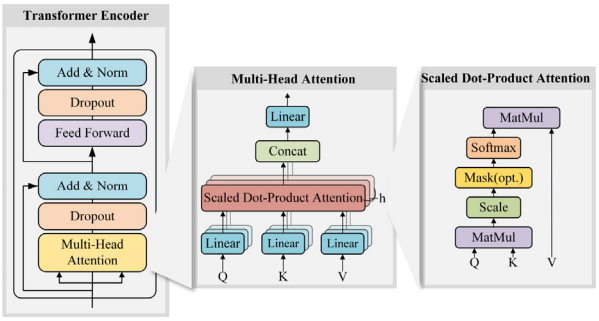

Fig. 1. The model architecture of Transformer Encoder (based on [27]).

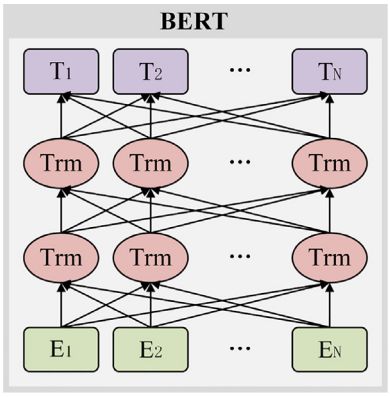

Fig. 2. The model architecture of BERT (based on [12]).

to computer vision [28]. The authors considered that RNN or LSTM could only compute sequentially. The input at the current moment depends on the output at the previous moment, which limits the parallelism capability of the model and does not capture particularly long-term dependencies well. Therefore, Transformer discards the traditional CNN and RNN. The whole network structure consists of the Attention mechanism, which allows the Transformer to reduce the distance between any two activities in the trace to a constant (e.g., one) without relying on past hidden states to capture dependencies on previous data. Instead, it processes the input data in its entirety to allow parallel computation, reduce training time, and reduce performance degradation due to long-term dependencies.

The structure of the Transformer model is similar to that of the Encoder–Decoder model, which is divided into the Encoder module and Decoder module. Because only the Encoder component of the Transformer model is employed in this research, just the Encoder module of the Transformer model is briefly described here, as illustrated in Fig. 1. The Multi-Head Attention mechanism and feedforward neural network are the fundamental components of the Encoder model of Transformer. Multiple Self-Attention Heads are used in the Multi-Head Attention mechanism to calculate the input information in parallel. Each Self-Attention Head focuses on a different part of the input information to finally extract the multiple semantics of the input information. The essence of the Self-Attention mechanism is an addressing process. The Attention Value is calculated by calculating the attention distribution with the vector Key and adding it to the vector Value if given a task-related query vector Query. This process does not need to feed all the input information into the neural network for computation. However, it only selects some task-related information to input into the neural network, which alleviates the complexity of the neural network model.

3.5. BERT

BERT has recently been proposed as an alternative to Word2Vec, which has significantly set records in 11 directions in the field of NLP [12]. Previously, one-way language models (left-to-right or right-to-left) limited the structure of pre-trained models. Therefore, the representational ability of the model is limited, and only one-way contextual information can be obtained. BERT emphasizes the move away from the traditional one-way language model or the shallow splicing of two one-way language models for pre-training as in the past. Instead, a deep bidirectional Transformer model and a new multi-task training target are used to generate deep language representations that can incorporate contextual information.

The model structure of BERT is shown in Fig. 2. The unit sum of three embedding features, namely word embedding, positional embedding, and segmentation embedding, makes up the encoding vector of BERT input. The Encoder module of the Transformer, i.e., Trm in Fig. 2, is used by BERT as the main framework of the algorithm. The Attention mechanism converts the distance between inputs at any position to one. Therefore, BERT's representational ability more thoroughly captures long-term bidirectional dependencies in statements based on the left and right context at all levels.

The essence of BERT is to learn a good feature representation by running a self-supervised learning method base on a massive corpus. The feature representation of BERT could be directly used as the word embedding feature for specific NLP tasks. As a result, BERT is a model that can be transferred to learning for other tasks. Then, the model can be fine-tuned or fixed according to the task and then used as a feature extractor. This process does not necessitate any structural changes to BERT to accomplish a specific task. In this paper, we reformulate the pre-training task Masked Activity Model for the PBPM task, train the BERT model using the historical data in the event log, and transfer it to two prediction tasks of the next activity and outcome, respectively. More details are shown in Section 4.

4. Multi-task prediction method based on transfer learning

Current research work in the field of PBPM has explored the application of various deep learning models to various prediction tasks, such as LSTM [2], GAN [20], and Transformer [21], applied to the next activity prediction task, and CNN [6] and LSTM [11] applied to the outcome prediction task. These methods are studied for a specific model and can be applied only to a specific prediction task and cannot be generalized to other prediction tasks. And it to be applied to other prediction tasks would require rebuild the model and a large amount of data and time for training from scratch. We need a new method that can be generalized to several different prediction tasks to allow the proposed model to be directly applied to prediction tasks without training from scratch. Furthermore, because of the complexity and diversity of process cases in real-world event logs, a cyclic structure is usually present, which leads to overly lengthy cases. Traditional RNNs and LSTMs do not capture particularly long-term dependencies between events well and suffer from gradient disappearance.

In this paper, we propose a multi-task prediction method based on BERT and Transfer Learning. Fig. 3 illustrates the architecture of multi-task prediction method and detailed in the followings.

(1) The pre-training task MAM is defined as the source domain task T_S. The pre-training model trained using a large

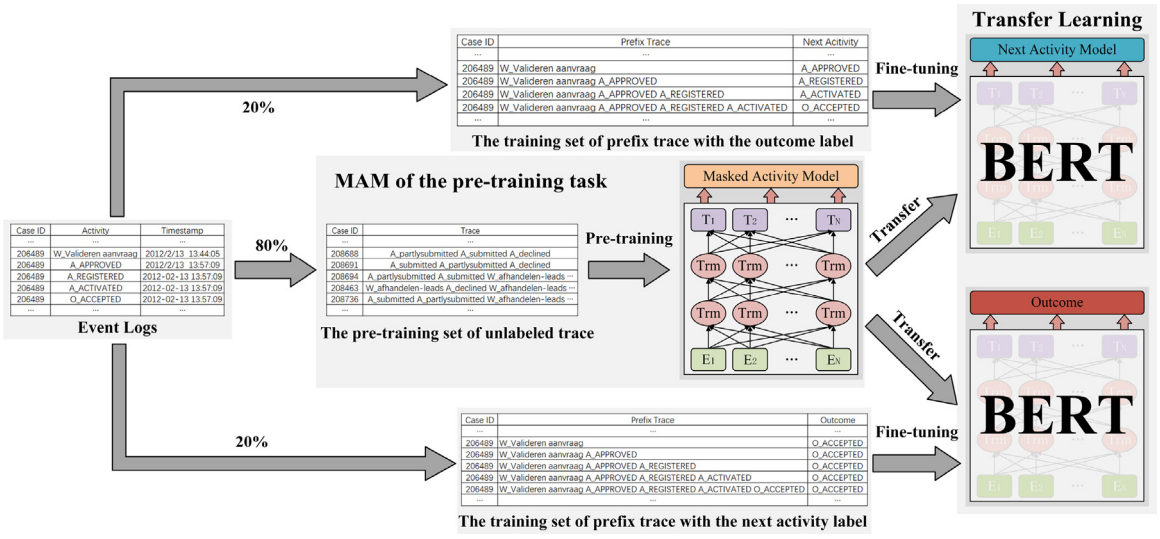

Fig. 3. The architecture of multi-task prediction method.

Table 1
Example of a pre-processing trace for the next activity prediction task in event log BPIC2012.

Case ID	Prefix trace	The next activity
209093	Event1	A_SUBMITTED
209093	Event1, Event2	W_Afhandelen leads
209093	Event1, Event2, Event3	W_Afhandelen leads
209093	Event1, Event2, Event3, Event4	A_DECLINED
209093	Event1, Event2, Event3, Event4, Event5	W_Afhandelen leads

Table 2
Example of pre-processing trace for outcome prediction task in event log BPIC2012.

Case ID	Prefix trace	Outcome
206489	Event1	O_ACCEPTED
206489	Event1, Event2	O_ACCEPTED
206489	Event1, Event2, Event3	O_ACCEPTED
206489	Event1, Event2, Event3, Event4	O_ACCEPTED
206489	Event1, Event2, Event3, Event4, Event5	O_ACCEPTED

amount of historical data is used as the source domain model. The specific expressions are as follows:

$$D_s = \{X_s, Y_s\} \tag{1}$$

where, D_s is the source domain data, i.e., the top 80% of the event logs sorted by time; X_s denotes the source domain samples, i.e., the masked unlabeled prefix traces; denotes the labels corresponding to the source domain samples, i.e., the masked activities.

The pre-training task MAM captures the long-term bi-directional dependencies between activities using the bi-directional Transformer model in BERT, and eventually trains a generalized deep bi-directional trace representation model. The source domain task T_s is represented as follows:

$$P_s = f_s\,(X_s, \theta_s) \tag{2}$$

where, P_s denotes the source domain prediction, i.e., the mask activity prediction; θ_s denotes the source domain model parameters, i.e., the pre-trained model parameters; f_s denotes the function mapping from X_s to P_s.

(2) The next activity prediction task and outcome prediction task are defined as target domain task T_{ta} and T_{to}. The next activity prediction model defined in Section 4.2.1 and the outcome prediction model defined in Section 4.2.2 are taken as the target domain model. The specific expressions are as follows:

$$D_{ta} = \{X_{ta}, Y_{ta}\} \tag{3}$$

$$D_{to} = \{X_{to}, Y_{to}\} \tag{4}$$

where, D_{ta} and D_{to} are the target domain data, i.e., the last 20% of the event logs sorted by time; X_{ta} and Y_{ta} are the target domain samples and the corresponding labels, respectively, as shown in Table 1; X_{to} and Y_{to} are the target domain samples and the corresponding labels, respectively, as shown in Table 2.

The target domain tasks T_{ta} and T_{to} are expressed as follow:

$$P_{ta} = f_{ta}\,(X_{ta}, \theta_{ta}) \tag{5}$$

$$P_{to} = f_{to}\,(X_{to}, \theta_{to}) \tag{6}$$

where, P_{ta} is the target domain prediction value, i.e., the next activity prediction value; θ_{ta} is the target domain model parameter, i.e., the next activity prediction model parameter; f_{ta} is the function mapping from X_{ta} to P_{ta}; P_{to} is the target domain prediction value, i.e., the outcome prediction value; θ_{to} is the target domain model parameter, i.e., the outcome prediction model parameter; f_{to} is the function mapping from X_{to} to P_{to}.

(3) Obtain knowledge related to source domain data D_s and source domain task T_s, and obtain the function mapping f_s. Then, after transferring the mapping f_s to the target domain, the function mappings f_{ta} and f_{to} of the target domain tasks T_{ta} and T_{to} are learned from the target domain data D_{ta} and D_{to}, respectively. The specific expressions are as follows:

$$\theta_s{}' = \arg\min L_s\,(X_s, \theta_s, Y_s) \tag{7}$$

$$\theta_{ta}{}' = \arg\min L_{ta}\,(X_{ta}, \theta_{ta}, Y_{ta}) \tag{8}$$

$$\theta_{to}{}' = \arg\min L_{to}\,(X_{to}, \theta_{to}, Y_{to}) \tag{9}$$

$$\theta_s = \theta_b + \theta_m \tag{10}$$

$$\theta_{ta} = \theta_b + \theta_a \tag{11}$$

$$\theta_{to} = \theta_b + \theta_o \tag{12}$$

where, L_s, L_{ta}, L_{to} are the loss functions of the pre-training task MAM, the next activity prediction task, and the outcome prediction task, respectively; θ_s, θ_{ta}, θ_{to} are the parameters of the source domain model and the target domain model, which have some of the same model parameters θ_b and some of the different model parameters, respectively; θ_m denote the part of the source domain model that is not transferred to the target domain model, i.e., the Mask Activity Model module in Fig. 4; θ_a denote the parameters unique to the target domain model, i.e., the Next Activity module in Fig. 6; θ_o denote the parameters unique to the target domain model, i.e., the Outcome module in Fig. 7.

Training the source domain model eventually yields the optimal model parameters θ_s'. The optimal parameter part θ_b in θ_s' is selected and transferred to the target domain such that the target domain task T_{ta} and T_{to} can be utilized when training the target domain model. Eventually, using the fine-tuning strategy, spending less data and time to train the target domain models (i.e., the next activity prediction model and the outcome prediction model) separately, the optimal model parameters θ_{ta}' and θ_{to}', can be obtained to obtain a prediction model with good performance.

4.1. Pre-training task

BERT is a multi-task model, including two self-supervised tasks in the pre-training phase: Masked Language Model (MLM) and Next Sentence Prediction (NSP). The pre-training task NSP is to determine whether sentence B is sentence A below. The pre-training task MLM masks some random words from the input corpus during training, and then BERT predicts the word by contextual information.

In this paper, we mainly study how to train a model that can extract depth representation of trace through pre-training tasks, regardless of whether the next occurrence trace of trace A is trace B. Therefore, we reformulate the pre-training task Masked Activity Model for the PBPM task in this paper. The structure of the Masked Activity Model of the pre-training task is shown in Fig. 4. Different from the pre-training task of BERT, which uses the unit sum of three embedding features, the input of pre-training task MAM proposed in this paper is the unit sum of two embedding features, namely Activity Embedding, and Position Embedding. The BERT models the position of activities and traces together by inputting two embedding features in MAM, and each activity representation in the current layer is based on all activities in the previous layer using the self-attention mechanism. The result is that it is straightforward to obtain deep representations at the trace level.

In contrast, although the traditional language model Word2Vec also learns semantic knowledge from a large text corpus unsupervised, there is no excellent way to obtain sentence-level representations directly. The general practice is to average all word vectors. It does not equate to sentence-level representations. However, MAM can capture the bidirectional semantic information of the input trace and the long-term dependencies between activities.

This paper proposes a pre-training task MAM for the application scenarios in PBPM. Due to different application scenarios, the traditional training strategy of 80%–10%–10% in pre-training task MLM may not apply to pre-training task MAM proposed in this paper. Therefore, this paper needs to reconsider the training details of the pre-training task MAM in order to find the pre-training strategy that best suits the PBPM task. Fig. 5 shows the traditional 80%–10%–10% masking strategy for MLM and the masking strategy with a uniform "[MASK]" replacement. The traditional 80%–10%–10% masking strategy is that 80% were directly replaced with "[MASK]" after identifying the masked activities, 10% were replaced with other arbitrary activities, and 10% kept the original activities. Random activities are added because the Transformer has to maintain a distributed representation of each input activity; otherwise, the model would habitually remember this "[MASK]" representation.

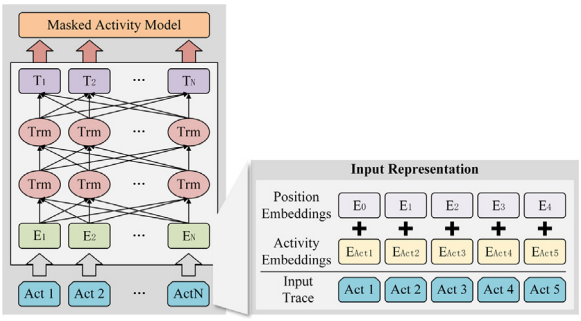

Fig. 4. The structure of Masked Activity Model of pre-training task.

4.2. Fine-tuning a multi-task prediction method

It can be applied to different PBPM prediction tasks after completing the pre-training task MAM on BERT using historical data from the event log. Different layers of the BERT model contain different information. Lower layers may contain more general semantic information, while higher layers contain more task-specific semantic information. After completing the pre-training task MAM, this paper transfers the BERT model to the next activity prediction model and the outcome prediction model. Fine-tuning strategies are applied to each of these two prediction models.

4.2.1. Next activity prediction task

Next activity prediction is a PBPM task that addresses the problem of how to accurately and efficiently predict the next activity of a running trace. This paper uses a fine-tuning strategy to transfer the BERT model after the pre-training task MAM is completed to the next activity prediction model for training, as shown in Fig. 6.

First, the task requires extracting traces that satisfy the business process from the historical data in the event log. For instance, a trace in event log BPIC 2012 is as follows:

⟨Event1, Event2, Event3, Event4, Event5, Event6⟩

where Event1 is event A_PARTLYSUBMITTED, Event2 is event A_SUBMITTED, Event3 is event W_Afhandelen leads, Event4 is event W_Afhandelen leads, Event5 is event A_DECLINED, and Event6 is event W_Afhandelen leads. Second, extracting prefix traces of these traces and marking the next activities of these prefix traces. The trace details are shown in Table 1. Finally, the padding technique is used to extract marker samples of the same length from prefix traces with a length equal to the maximum length of the traces in the event log.

First, the BERT model receives the preprocessed labeled samples. Next, each activity is transformed into an embedding vector

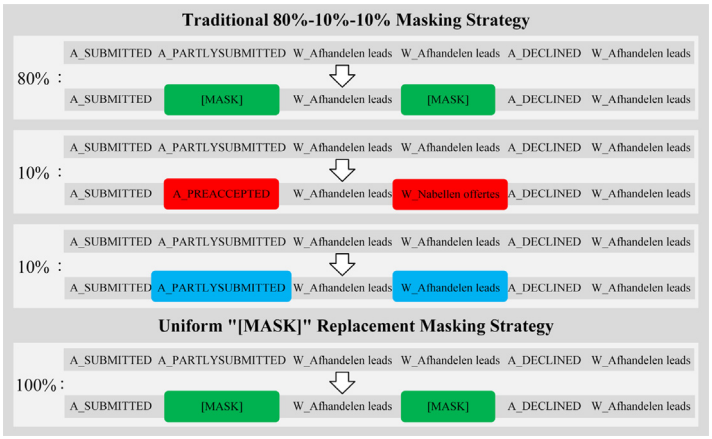

Fig. 5. MAM masking strategy for pre-training tasks.

using activity embedding, and the location information of each activity is encoded into a feature vector using location embedding. Then, the sum of the two is fed into a bidirectional Transformer model to learn a deep trace representation incorporating contextual information. Finally, the output of the BERT model is passed to the next activity prediction module.

The next activity prediction module consists of the GlobalAveragePooling layer, the Dropout layer, and the Softmax layer. Since the fully connected layer leads to a surge in network parameters and a significant increase in computational cost, this paper uses the GlobalAveragePooling layer instead of the fully connected layer, which dramatically reduces the number of network parameters and lowers the computational cost. The Dropout layer is used to temporarily discard the neural network units from the network with a set probability during the training process to prevent overfitting. The final output is sent to the Softmax layer. The probability of the next occurrence of different activities is calculated using the Softmax activation function. The activity with the highest probability is selected as the final output, i.e., the next activity of the running trace. The Softmax activation function is defined as follows:

$$y_i = soft\max(a)_i = \frac{\exp(a_i)}{\sum_j \exp(a_i)} \tag{13}$$

The backpropagation algorithm performs iterations to calculate the loss function multiclassification cross-entropy. The gradient descent algorithm is used to find the optimal weights and deviations of the prediction model, resulting in a prediction model that can accurately and efficiently predict the next activity of the running trace.

4.2.2. Outcome prediction task

The outcome prediction task plays an essential role in PBPM, as it demonstrates how to accurately predict the outcome of ongoing cases earlier, thereby reducing resource waste. This paper also uses a fine-tuning strategy to transfer the BERT model after the pre-training task is completed to the outcome prediction model for training, as shown in Fig. 7.

This task entails extracting traces that satisfy the business process from historical data in the event log, as described in the literature [5]. For instance, a trace in event log BPIC 2012 is as follows:

⟨Event1, Event2, Event3, Event4, Event5⟩

where Event1 is event W_Valideren aanvraag, Event2 is event A_APPROVED, Event3 is event A_REGISTERED, Event4 is event A_ACTIVATED, and Event5 is event O_ACCEPTED. Then extract

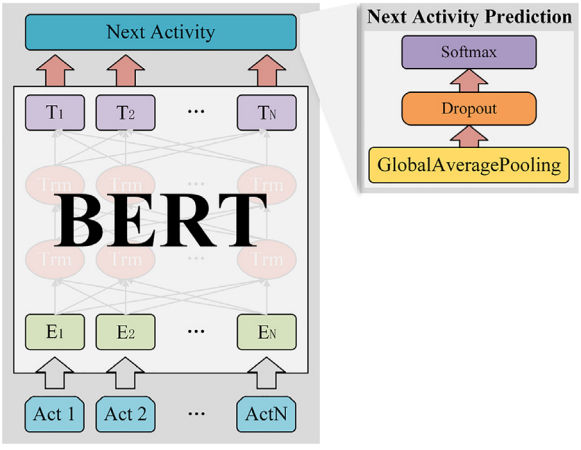

Fig. 6. The structure of next activity prediction model.

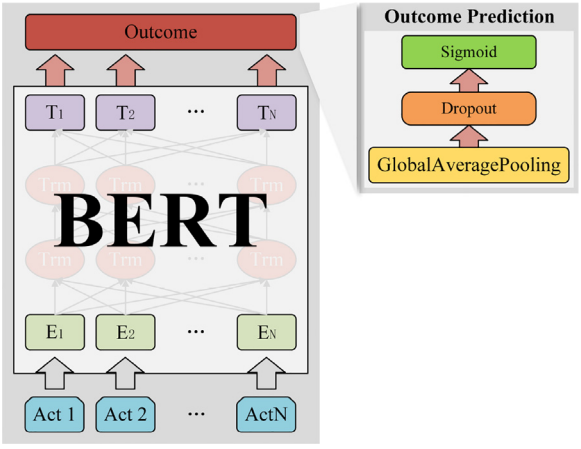

Fig. 7. The structure of outcome prediction model.

the prefix traces of these traces and mark the outcomes of these prefix traces. The trace details are shown in Table 2. The padding technique extracts marked samples of the same length from the prefix traces, whose length is equal to the maximum length of the traces in the event logs.

The outcome prediction model is similar to the next activity prediction model, and only the respective modules are different. The outcome prediction module finally goes through the Sigmoid

layer to calculate the probability of different outcomes occurring. The outcome with the highest probability is selected as the final output, i.e., the outcome of the ongoing case. The Sigmoid activation function is defined as follows:

$$y_i = \text{sigmoid}(a)_i = \frac{1}{1 + \exp(a_i)} \tag{14}$$

A backpropagation algorithm executes iterations to calculate the loss function binary cross-entropy. The gradient descent algorithm is used to find the optimal weights and deviations of the prediction model, finally obtaining a prediction model that can predict the outcome of the ongoing case earlier and more accurately.

5. Experiments

In this paper, a series of experiments will be conducted on eleven real-world event logs to verify the feasibility of the multi-task prediction method. This section first describes the real-world event logs, experimental setup, and evaluation metrics used to evaluate the proposed method. Then explore the effects of different training strategies in the pre-training task MAM on fine-tuning different prediction tasks. Finally, compare the experimental results with related methods to assess the effectiveness of the proposed method.

5.1. Event logs

The datasets used for the experiments in this paper are all from the publicly available event log dataset from the 4TU Center for Research Data,[1] where the outcome prediction dataset is referenced in the literature [5]. An overview of these datasets is provided in Table 3.

The event log Helpdesk[2] contains an Italian software company's helpdesk ticket management process, with a short case and a small number of activities.

The event log BPI Challenge 2012[3] contains a personal loan or overdraft application process in a Dutch financial institution. The application process consists of three sub-flows, namely the state of the trace application (A), the state of the work item associated with the application in the trace (W), and the state of the trace offer (O). In addition, the W process contains three different lifecycle transitions, i.e., start, scheduling, and completion, while the A and O processes contain only the events of lifecycle transition completion. Similar to other research methods, BPIC2012 can be considered multiple subsets with different scenarios. Therefore, to make the next activity prediction method more convincing, BPIC2012W and BPIC2012CW subsets are added, BPIC2012W contains only W processes, and BPIC2012CW contains only the events in which the lifecycle variation is completed in W processes. For the outcome prediction task, this paper refers to the literature [5] to mark the case results in the log in various ways, i.e., whether the application is accepted, rejected, or canceled. The BPIC2012 log is divided into three binary outcome prediction tasks based on the marked cases: BPIC2012_1, BPIC2012_2, and BPIC2012_3.

The event log Sepsis[4] documents the process cases of patients presenting with sepsis cases in a hospital in the Netherlands. Each case in the log collects the process of events from the patient's registration in the emergency room to discharge from the hospital. This paper uses Sepsis as a dataset to study the next activity prediction method. For the outcome prediction task, this paper also refers to the literature [5], where three different case markers are defined in this log: Sepsis_1, Sepsis_2, Sepsis_3.

[1] https://data.4tu.nl/.

[2] https://doi.org/10.4121/uuid:0c60edf1-6f83-4e75-9367-4c63b3e9d5bb.

[3] https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f.

[4] https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460.

5.2. Experimental setup

We implemented the methods in this paper on Python 3.8 using the Tensorflow2 and Keras library. All experiments in this paper are performed on Windows 10, Intel(R) Core (TM) i7-11800H CPU, GeForce RTX 3070 GPU, and 16 GB RAM.

First, the cases in the event log are arranged chronologically to simulate the prediction model using historical cases to predict the future behavior of ongoing cases. Then, the event logs are divided into 80% and 20% in chronological order for pre-training and fine-tuning, respectively, because this paper requires a pre-training task MAM for the BERT model. The outcome prediction method only retains the event logs required by fine-tuning because it can be directly applied to the BERT model after the completion of pre-training, without the need for pre-training tasks. Finally, the datasets utilized for fine-tuning are separated into 80% and 20% for training and testing in chronological sequence, with 10% of the data in the training set used for validation in the training phase.

Table 4 reports the hyperparameters and the corresponding values for the multi-task prediction method. In the pre-training task MAM, the model was optimized using the Adam optimizer [29]. The backpropagation training was applied with early stopping to avoid overfitting. The training phase is stopped when the loss value of 10 consecutive epochs on the validation set is less than 0.001. The model input sequence length is the maximum case length in the event log. In addition, the masking strategy and the masking probability will be investigated in the experiments.

This paper focuses on the feasibility of using fine-tuning strategies to transfer the BERT model after completing the pre-training task MAM to the prediction model of the next activity and outcome, respectively. Due to time constraints, the effect of different fine-tuning strategies on the performance of different prediction tasks is not investigated, so only the number of epochs and the learning rate need to be set. The model is optimized using the Adam optimizer. The backpropagation training was applied with early stopping to avoid overfitting. The training phase is stopped when the loss value of 10 consecutive epochs on the validation set is less than 0.003. In this paper, we did not use the hyperparametric optimization algorithm to further improve the model, which will be investigated in our future work.

5.3. Evaluation metrics

This experiment assesses the prediction performance of the comparative approaches by calculating standard multiclass metrics. The specific evaluation metrics are shown below.

$$Accuracy = \frac{1}{n}\sum_{i=1}^{m} tp_i \tag{15}$$

$$Precision = \frac{1}{m}\sum_{i=1}^{m} \frac{tp_i}{tp_i + fp_i} \tag{16}$$

$$Recall = \frac{1}{m}\sum_{i=1}^{m} \frac{tp_i}{tp_i + fn_i} \tag{17}$$

$$F - score = \frac{1}{m}\sum_{i=1}^{m} 2 * \frac{P_i * R_i}{P_i + R_i} \tag{18}$$

AUC (Area Under the Curve) measures the average area under the ROC curve for each activity. The False Positive Rate (FPR) is on the X-axis, and the True Positive Rate (TPR) is on the Y-axis in the ROC space of the curve. The main advantage of AUC over

Table 3
Information of event log.

Event log	# Events	# Activities	# Traces	Min length	Max length	Mean length
Helpdesk	21 348	14	4580	2	15	4.67
BPIC2012	262 200	24	13 087	3	175	20.04
BPIC2012W	170 107	7	9658	2	156	17.61
BPIC2012CW	72 413	6	9658	1	74	7.50
BPIC2012_1	186 693	24	4685	15	175	39.85
BPIC2012_ 2	186 693	24	4685	15	175	39.85
BPIC2012_ 3	186 693	24	4685	15	175	39.85
Sepsis	15 214	16	1049	3	185	14.48
Sepsis_1	13 120	15	782	5	185	16.78
Sepsis_2	10 924	15	782	4	60	13.97
Sepsis_3	12 463	15	782	4	185	15.94

Table 4
Details of the hyperparameters of the multi-task prediction method.

Module	Hyperparameter	Values
Pre-training task MAM	Epochs	50
	Batch size	32
	Number of Transformer layers	5
	Number of attention heads	8
	Embedding size	32
	Learning rate	0.0005
	Dropout	0.1
Next activity prediction model	Epochs	50
	Learning rate	0.001
Outcome prediction model	Epochs	50
	Learning rate	0.001

accuracy is that the ROC curve remains constant even when the sample is unevenly distributed [30].

$$FPR = \frac{fp_i}{fp_i + tn_i} \tag{19}$$

$$TPR = \frac{tp_i}{tp_i + fn_i} \tag{20}$$

The n appearing in the above formula is the total number of predictions, n_i is the number of events of activity i, m is the number of different activity types, tp_i is positive sample executed activity i predicted by the model as positive class, fp_i is negative sample unexecuted activity i predicted by the model as positive class, tn_i is negative sample unexecuted activity i predicted by the model as negative class, and fn_i is positive sample execute activity i predicted by the model as negative class, P_i is Precision of activity i, and R_i is Recall of activity i.

5.4. Results and discussion

This section will be divided into four parts to verify the feasibility of the multi-task prediction method. The first part explores the effect of the masking strategy and masking probability in the pre-training task MAM on fine-tuning different prediction tasks. The second part will be compared with the state-of-the-art next activity prediction method. The third part will compare state-of-the-art outcome prediction methods. The fourth part will be compared with the results obtained by directly training with the next activity prediction model and the outcome prediction model proposed in this paper.

5.4.1. Effects of masking strategy and masking probability

This section aims to use the multi-task prediction method to compare the effects of a traditional 80%–10%–10% masking strategy and a masking strategy with a uniform "[MASK]" replacement combined with different masking probabilities on the performance of different prediction tasks under the same experimental setup. Specifically, only the masking strategy and the masking probability were changed, while the other experimental parameters of the pre-training task MAM and the experimental parameters of the fine-tuning strategy in the multi-task prediction method were not changed. The objective is to find the best pre-training strategy for the PBPM task.

The Accuracy/AUC and F-score of the next activity prediction task and the outcome prediction task are compared in Figs. 8 and 9. From the comparative analysis, it is clear that the next activity prediction task and the outcome prediction task usually achieve better results at a mask probability of 40%. The prediction results show an upward trend when starting from a mask probability of 15% and reaching the highest point at a mask probability of 40% when the best prediction results are achieved. When the mask probability rises to 50%, the prediction results are affected and show a downward trend. Therefore, compared with the traditional default of 15% mask probability, 40% mask probability can make PBPM tasks achieve better prediction results overall.

For the masking strategy, it is clear from the analysis in Figs. 8 and 9 that the traditional 80%–10%–10% training strategy may be more suitable for the PBPM task than the uniform "[MASK]" replacement strategy. Because some activities that are not "seen" will appear in the model when the fine-tuning strategy is used to transfer the BERT model to the next activity prediction model and the outcome prediction model for training if an activity is entirely replaced by "[MASK]". It can make the prediction model poorly predict the activity, resulting in reduced predictive performance.

5.4.2. Next activity prediction

In order to evaluate the feasibility of the multi-task prediction method for PBPM tasks, the seven most advanced next activity prediction methods in the PBPM field were selected and compared with the multi-task prediction method. The multi-task prediction method was tested under the experimental settings of 40% masking probability, 80%–10%–10% masking strategy, and other hyperparameters unchanged. There are the following methods:

(1) LSTM [2]: Tax et al. proposed the next activity prediction method based on one-hot coding and LSTM.
(2) SAE [3]: Evermann et al. proposed the multi-stage next activity prediction method based on feature preprocessing and autoencoder.
(3) CNN [4]: Pasquadibisceglie et al. proposed the next activity prediction method based on image coding and CNN.
(4) LSTM [9]: Camargo et al. proposed the next activity prediction method based on data processing and LSTM.
(5) LSTM [17]: Evermann et al. proposed the next activity prediction method based on Embedding and RNN.
(6) CNN [18]: Mauro et al. proposed the next activity prediction method based on the Inception module.
(7) Trans [21]: Bukhsh et al. proposed the next activity prediction method based on the Attention mechanism.

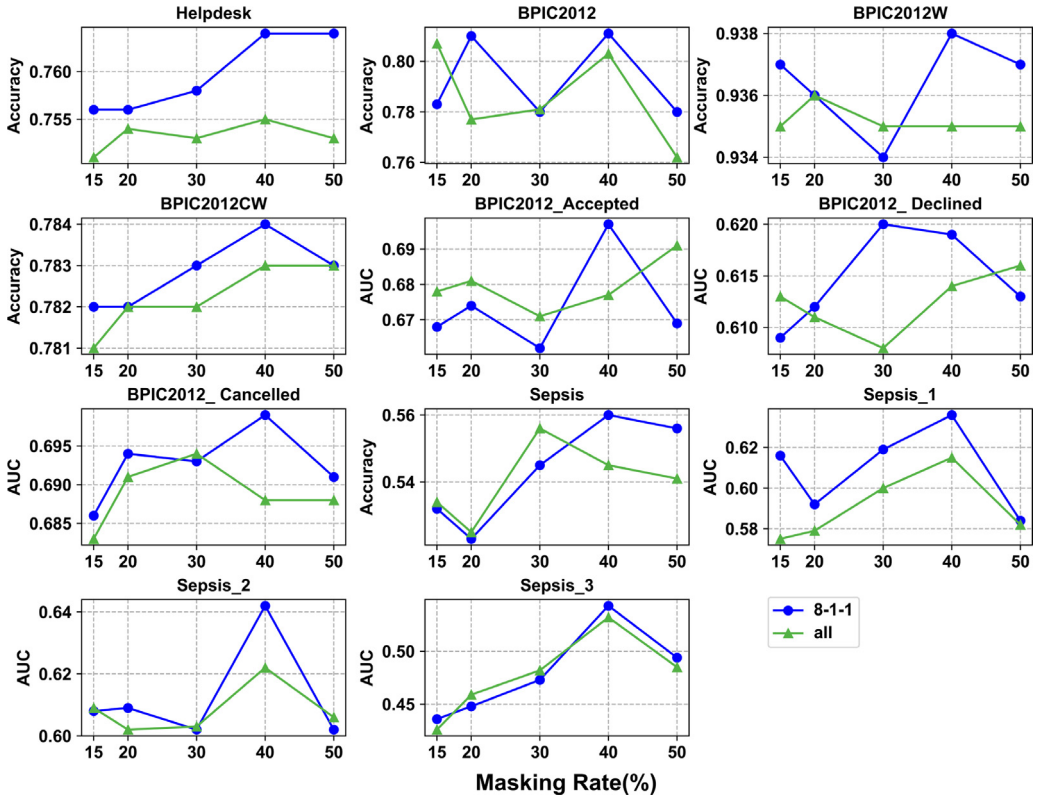

Fig. 8. Effect of masking strategy and masking probability on Accuracy/AUC of different prediction tasks.

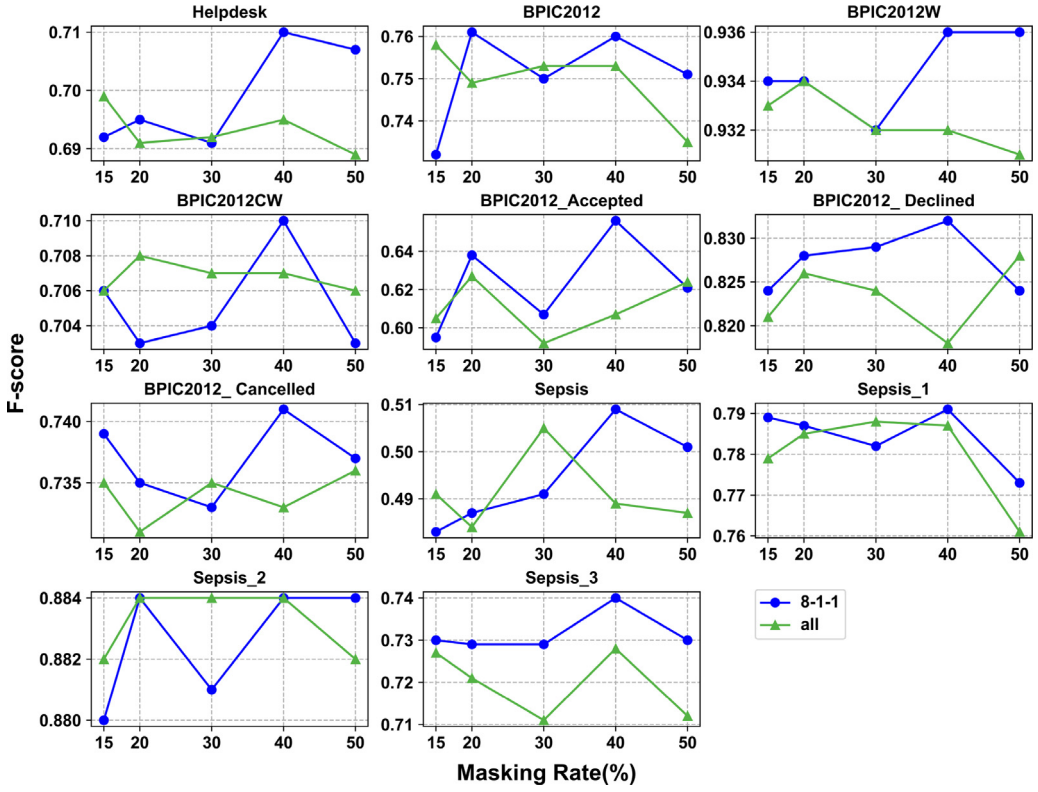

Fig. 9. Effect of masking strategy and masking probability on F-score of different prediction tasks.

Table 5 reports the hyperparameters and the corresponding values for the seven next activity prediction methods. CNN [18] optimized the hyperparameters using the Tree-structured Parzen Estimator (TPE) algorithm [31]. The remaining six methods are

Table 5
Details of hyperparameters for the seven next activity prediction methods.

Method	Hyperparameter	Values
LSTM [2]	Epochs	500
	Learning rate	0.002
	Number of layers	3
SAE [3]	Epochs	100
	Learning rate	0.005
	Number of layers	6
CNN [4]	Epochs	500
	Learning rate	0.0002
	Batch size	128
LSTM [9]	Epochs	200
	Batch size	32
LSTM [17]	Epochs	100
	Number of training epochs	100
	Base learning rate	1.00
CNN [18]	Number of layers	{1, 2, 3}
	Batch size	$\{2^9, 2^{10}\}$
	Learning rate	[0.00001, 0.01]
Trans [21]	Epochs	100
	Learning rate	0.001
	Number of attention heads	4

the same as this paper and do not use the hyperparameter optimization algorithm. More experimental details can be found in [2–4,9,17,18,21].

In order to compare with related methods and avoid the effect of uneven distribution of the number of events, accuracy, precision, recall, and F-score are calculated in a weighted form. Table 6 reports the prediction results of these methods in five real-world event logs. As shown in Table 6, the multi-task prediction method outperforms the other prediction methods in four of the five event logs. While the accuracy in Helpdesk is not as good as the autoencoder-based prediction method [3] and the Inception-based prediction method [18], the F-score performs as well as the prediction method [3]. Thus, the conventional model can capture the dependencies between events for event logs with simple processes. The multi-task prediction method achieves optimal results on BPIC2012, BPIC2012W, and BPIC2012CW for all evaluation metrics except for the poor accuracy on BPIC2012 compared to the LSTM-based prediction method [17]. It is worth noting that the multi-task prediction method has excellent performance results in BPIC2012W, and the Transformer-based prediction method [21] also performs well. However, the LSTM-based prediction method [17] has a big gap between the above two methods. It suggests that using the attention mechanism can better predict the next activity in BPIC2012W. The prediction method [17] has the highest accuracy in BPIC2012. However, it does not perform well in accuracy and precision in BPIC2012W and BPIC2012CW, thus indicating that no prediction method can perform equally well in different business process scenarios. However, the fine-tuned next activity prediction model performs well in five event logs, which is sufficient to show that the proposed multi-task prediction method can be effectively applied to the next activity prediction task in this paper.

5.4.3. Outcome prediction

In order to further explore the applicability of the multi-task prediction method to different PBPM tasks, the seven most advanced outcome prediction methods in the PBPM field are selected and compared with multi-task prediction methods. The multi-task prediction method was tested under the experimental settings of 40% masking probability, 80%–10%–10% masking strategy, and other hyperparameters unchanged. There are the following methods:

Table 6
Results obtained compared to the next activity prediction method.

Event log	Method	Accuracy	Precision	Recall	F-score
Helpdesk	Ours	0.764	**0.723**	0.764	**0.711**
	LSTM [2]	0.712	–	–	–
	SAE [3]	0.782	0.632	**0.781**	**0.711**
	CNN [4]	0.739	–	–	–
	CNN [18]	**0.785**	–	–	–
BPIC2012	Ours	**0.811**	0.841	**0.811**	**0.761**
	LSTM [9]	0.786	–	–	–
	LSTM [17]	–	**0.859**	–	–
	CNN [18]	0.789	–	–	–
BPIC2012W	Ours	**0.939**	**0.946**	**0.939**	**0.936**
	LSTM [17]	0.818	0.832	–	–
	Trans [21]	0.915	–	–	0.910
BPIC2012CW	Ours	**0.786**	**0.666**	**0.786**	**0.710**
	LSTM [2]	0.760	–	–	–
	CNN [4]	0.782	–	–	–
	LSTM [9]	0.778	–	–	–
	LSTM [17]	0.711	0.658	–	–
Sepsis	Ours	**0.561**	**0.515**	**0.561**	**0.509**

Table 7
Details of hyperparameters for the seven outcome prediction methods.

Method	Hyperparameter	Values
SVM [5]	Penalty parameter of the error term	$[2^{-15}, 2^{15}]$
	Kernel coefficient	$[2^{-15}, 2^{15}]$
LR [5]	Inverse of regularization strength	$[2^{-15}, 2^{15}]$
RF [5]	Max features	[0, 1]
XGB [5]	Learning rate	[0, 1]
	Subsample	[0.5, 1]
	Max tree depth	[4, 30]
	Colsample bytree	[0.5, 1]
	Min child weight	[1, 6]
ORANGE [6]	Learning rate	[0.00001, 0.0001]
	Number of layers	[1, 2]
	Batch size	$[2^6, 2^{10}]$
LSTM [11]	Number of hidden layers	{1, 2, 3}
	Number of units in hidden layers	[10, 150]
	Initial learning rate	[0.000001, 0.01]
	Batch size	{8, 16, 32, 64}
	Dropout	[0, 0.3]
	Optimizer	{RMSProp, NAdam}
Trans [21]	Learning rate	{0.0001, 0.01}
	Batch size	[128, 256, 512]

(a) SVM [5]: Teinemaa et al. proposed the outcome prediction method based on Support Vector Machines.
(b) LR [5]: Teinemaa et al. proposed the outcome prediction method based on Logistic Regression.
(c) RF [5]: Teinemaa et al. proposed the outcome prediction method based on Random Forest.
(d) XGB [5]: Teinemaa et al. proposed the outcome prediction method based on Gradient Boosted Trees.
(e) ORANGE [6]: Pasquadibisceglie et al. proposed the outcome prediction method based on image processing and CNN.
(f) LSTM [11]: Teinemaa et al. proposed the outcome prediction method based on feature index extraction and LSTM.
(g) FOX [23]: Pasquadibisceglie et al. proposed the fully interpretable outcome prediction method based on neuro-fuzzy networks.

Table 7 reports the hyperparameters and corresponding values for the seven outcome prediction methods. SVM [5], LR [5], RF [5], XGB [5], ORANGE [6], and FOX [23] all optimize the hyperparameters using the TPE algorithm. LSTM [11] optimizes the hyperparameters using the random search algorithm [32]. More experimental details can be found in [5,6,11,23].

Table 8
Results obtained compared to the AUC of the outcome prediction method.

Event log	SVM [5]	LR [5]	RF [5]	XGB [5]	LSTM [11]	ORANGE [6]	FOX [23]	Ours
BPIC2012_1	0.63	0.65	0.69	**0.70**	0.62	0.67	0.64	**0.70**
BPIC2012_2	0.55	0.59	0.61	0.57	0.60	0.61	0.60	**0.62**
BPIC2012_3	**0.70**	0.69	**0.70**	0.69	**0.70**	**0.70**	0.69	**0.70**
Sepsis_1	0.49	0.57	0.41	0.33	0.52	0.57	0.58	**0.64**
Sepsis_2	0.82	**0.86**	0.79	0.85	0.73	**0.86**	0.73	0.64
Sepsis_3	0.72	**0.73**	0.66	0.72	0.61	**0.73**	0.68	0.54

Table 9
Results obtained compared to the F-score of the outcome prediction method.

Event log	SVM [5]	LR [5]	RF [5]	XGB [5]	LSTM [11]	ORANGE [6]	Ours
BPIC2012_1	0.38	0.53	0.64	0.59	0.60	**0.66**	**0.66**
BPIC2012_2	0.00	0.13	0.17	0.16	0.15	0.21	**0.83**
BPIC2012_3	0.20	0.30	0.42	0.36	0.31	0.46	**0.74**
Sepsis_1	0.00	0.09	0.00	0.00	0.05	0.11	**0.79**
Sepsis_2	0.00	0.47	0.39	0.42	0.37	0.50	**0.88**
Sepsis_3	0.00	0.37	0.34	0.35	0.87	**0.89**	0.74

In order to compare with related methods and avoid the effect of uneven distribution of the number of events, F-score and AUC are calculated in a weighted form. Tables 8 and 9 report the AUC and F-score of the multi-task prediction method and other outcome prediction methods on the six real-world event logs. The results show that the multi-task prediction method generally outperforms (or performs the same as) the other prediction methods on most event logs, F-score performance on BPIC2012_2, BPIC2012_3, Sepsis_1, and Sepsis_2 was outstanding, AUC performed best on BPIC2012_1, BPIC2012_2, BPIC2012_3, and Sepsis_1. In AUC, ORANGE performs well on Sepsis_2 and Sepsis_3, LR performs well on Sepsis_2 and Sepsis_3, and SVM, RF, LSTM, and GRANGE perform well on BPIC2012_3. In terms of F-score, ORANGE performs well on BPIC2012_1 and Sepsis_3, and LSTM likewise performs well on Sepsis_3. It is worth noting that no prediction method can achieve optimal performance on F-score on all event logs, except for multi-task prediction methods. Although the multi-task prediction method does not perform as well as other prediction methods on individual event logs, the overall performance is better than other prediction methods. It is sufficient to demonstrate that the multi-task prediction method in this paper can be effectively applied to outcome prediction methods.

In the section, we use the Friedman test [33] to rank these methods to statistically test whether these methods are statistically significant. It is a non-parametric test that is typically used to compare the performance of multiple methods on multiple datasets [34]. The ranking of each method's performance is obtained based on each method's performance results on each dataset. Finally, the summation is averaged to obtain the average ordinal value of each method. Since the prediction method FOX [23] does not provide F-score-related data, only the relevant AUC ranking is performed for FOX. Under the original hypothesis that the performance of each method is the same, the AUC and F-score of these methods on each data set are tested using the Friedman test. The original hypothesis is rejected if the test result is greater than the critical value of 0.05, i.e., the performance of each method is not the same. Since the original hypothesis is rejected, the critical value domain for the difference in mean ordinal values is calculated using the Nemenyi test to compare whether the performance of each method is significantly different.

The final test results yield a Friedman test plot, as shown in Fig. 10. The results show that ORANGE has the best performance in terms of AUC, probably because ORANGE goes through a complex preprocessing stage to model the possible correlation patterns between trace features. The multi-task prediction method performs in the second position, outperforming FOX, LSTM, XGB, RF, LR, and SVM, with LSTM performing the worst. The multi-task prediction method does not go through a complex preprocessing stage. The pretraining task MAM simply uses unlabeled traces as input. In terms of F-score, the multi-task prediction method outperforms the other outcome prediction methods, with ORANGE performing in second place and SVM performing the worst. In addition, there is a significant difference in performance between the multi-task prediction methods and SVM. Consistent with the conclusions drawn in Tables 8 and 9, the overall performance of the multi-task prediction methods is good and can be effectively applied to the outcome prediction task.

5.4.4. Comparison with benchmark method

This section compares the multi-task prediction method with the benchmark method and explores the effectiveness of the multi-task prediction method when applied to several different PBPM tasks. The benchmark method is described below.

Benchmark method: Unlike the training step of the multi-task prediction method, the benchmark method does not perform the pre-training task MAM. In other words, instead of transferring the BERT model after the pre-training task is completed to the prediction model and then training it, the benchmark method directly trains the next activity prediction model defined in Section 4.2.1 and the outcome prediction model defined in Section 4.2.2, i.e., model weight initialization. In addition, the event log is divided into 80% and 20% in chronological order for training and testing, respectively.

Both the multi-task prediction method and the benchmark method use the same experimental setup and experiment on the same experimental equipment.

Table 10 reports the prediction results of the multi-task prediction method and the benchmark method on all event logs. The results show that the multi-task prediction method outperforms the benchmark method on most of the event logs. The benchmark method outperforms the multi-task prediction method on Sepsis. The multi-task prediction method outperforms the benchmark method on the next event prediction task, i.e., on Helpdesk, BPIC2012, BPIC2012W, and BPIC2012CW, except for the F-score on BPIC12CW, which is worse than the benchmark method. The multi-task prediction method also outperforms the benchmark method in the outcome prediction task, i.e., it outperforms the benchmark method on BPIC2012_1, BPIC2012_2, BPIC2012_3, Sepsis_1, Sepsis_2, and Sepsis_3, except for the AUC on BPIC2012_3 and Sepsis_3 and the F-score on Sepsis_1 is worse than the benchmark method.

As shown in Fig. 11, the time spent by the multi-task prediction method for model fine-tuning on all event logs is much

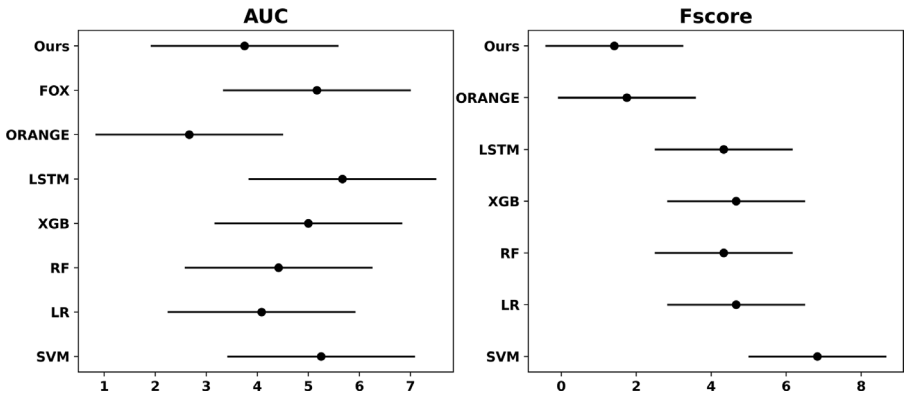

Fig. 10. Friedman plots on AUC and F-score for outcome prediction methods.

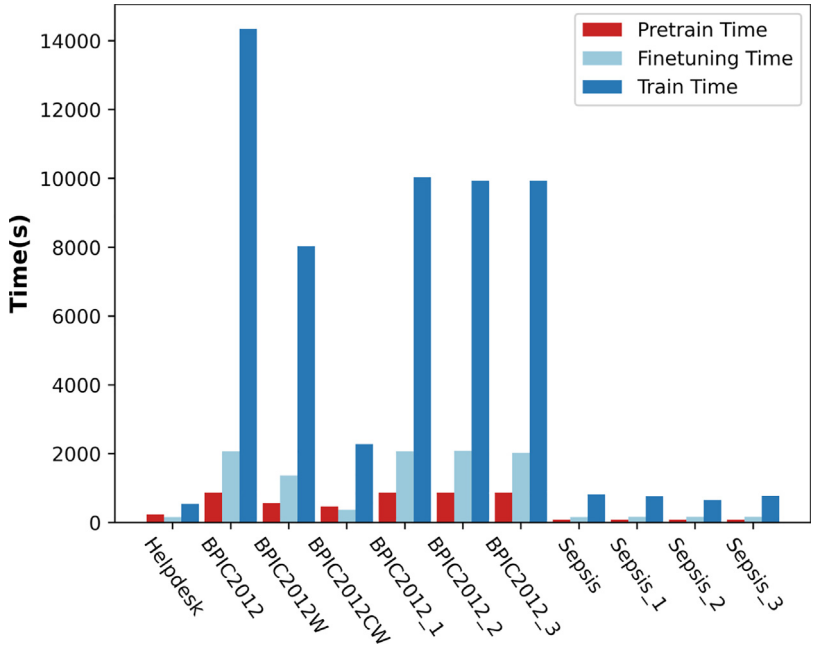

Fig. 11. Comparison of time spent (in seconds) on all event logs by multi-task prediction methods and benchmark methods.

Table 10
Comparison of multi-task prediction methods and benchmark methods for prediction results on all event logs.

Event log	Multi-task prediction method			Benchmark method		
	Accuracy	F-score	AUC	Accuracy	F-score	AUC
Helpdesk	**0.764**	**0.710**	–	0.699	0.629	–
BPIC2012	**0.811**	**0.760**	–	0.775	0.733	–
BPIC2012W	**0.939**	**0.936**	–	0.885	0.881	–
BPIC2012CW	**0.786**	0.710	–	0.761	**0.721**	–
BPIC2012_1	–	**0.66**	**0.70**	–	0.62	0.69
BPIC2012_ 2	–	**0.83**	**0.62**	–	0.74	0.61
BPIC2012_ 3	–	**0.74**	0.70	–	0.72	**0.72**
Sepsis	0.561	0.515	–	**0.602**	**0.567**	–
Sepsis_1	–	0.79	**0.64**	–	**0.80**	0.50
Sepsis_2	–	**0.88**	**0.64**	–	0.84	0.46
Sepsis_3	–	**0.74**	0.54	–	0.73	**0.63**

less than the time spent by the benchmark method for direct model training. Even when the time spent on the pre-training task MAM and the time spent on model fine-tuning are added together, they are far less than the time spent directly training the model. It is pronounced on the event logs with large data volumes, i.e., BPIC2012, BPIC2012W, and BPIC2012CW. Specifically, the benchmark method spent 14,349 s on BPIC2012, while the multi-task prediction method took 867 s to complete the pre-training task and 2065 s to perform the fine-tuning training. The combined time of the two methods was only 2932 s, an 80% reduction in training time. Since the prediction task of the next activity and outcome applied by the multi-task prediction method share a BERT model after the completion of MAM. So BPIC2012, BPIC2012_1, BPIC2012_2, BPIC2012_3, and Sepsis, Sepsis_1, Sepsis_2, Sepsis_3 all require only one pre-training task MAM completion time spent, which dramatically reduces the

training time. Although the multi-task prediction method does not perform as well as the benchmark method in individual event logs, the overall performance is better than the benchmark method, and the time spent to train the model is much less than the benchmark method. Therefore, it can be demonstrated that the proposed multi-task prediction method can be quickly applied to different prediction tasks with good prediction performance by only spending a short time to complete the pre-training task.

6. Conclusion

In this paper, we propose a multi-task prediction method based on Transfer Learning, which utilizes the BERT model to pre-training task MAM on many unlabeled traces to learn a generic representation of the traces. By exploring the influence of masking strategy and masking probability in the pre-training task MAM on the next activity prediction task and outcome prediction task, it is found that masking probability of 40% and masking strategy of 80%–10%–10% are most suitable for the PBPM task.

Different from traditional deep learning methods that spend a lot of time and resources on training, the multi-task prediction methods can transfer the BERT model to the next activity prediction model and the outcome prediction model and spend fewer data and time on training, allowing for fast and efficient application to the next activity prediction task and the outcome prediction task. The experimental results show that the multi-task prediction method outperforms other comparative prediction methods and benchmark methods in most event logs. Specifically, the multi-task prediction method achieved an accuracy of 0.939 for the next activity prediction task on BPIC2012W and an F-score of 0.88 for the outcome prediction task on Sepsis_2. In addition, the time spent for training the multi-task prediction method is much less than that of the benchmark method. Specifically, the multi-task prediction method took 80% less time to train on BPIC2012 than the benchmark method.

Although this paper demonstrates the effectiveness of the multi-task prediction method, there are still some limitations. In this paper, only activities are considered the input of the pre-training task MAM, and more case attributes and event attributes, such as timestamps and resources, are omitted. Therefore, the multi-task prediction method does not apply to the timestamp prediction task and the remaining time prediction task. In future work, our goal is to design a new pre-training task with additional inputs to the pre-training task so that the multi-task prediction method can be applied to more prediction tasks concerning time. The second future work addresses the uneven distribution of activity and case outcomes in the dataset during the training process. We plan to use generators in GAN to generate new traces to increase the number of less distributed activities and case outcomes and balance the data distribution.

CRediT authorship contribution statement

Hang Chen: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data curation, Visualization, Writing – original draft. **Xianwen Fang:** Conceptualization, Validation, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Huan Fang:** Validation, Formal analysis, Investigation, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation, China (No. 61572035, 61402011, 61902002), Key Research and Development Program of Anhui Province, China (2022a05020005), the Leading Backbone Talent Project in Anhui Province, China (2020-1-12), the Open Project Program of the Key Laboratory of Embedded System and Service Computing of Ministry of Education, China (No. ESSCKF2021-05), and the Graduate School Level Innovation Fund of Anhui University of Science and Technology, China (No. 2021CX2119).

References

[1] Fabrizio Maria Maggi, Chiara Di Francescomarino, Marlon Dumas, Chiara Ghidini, Predictive monitoring of business processes, in: International Conference on Advanced Information Systems Engineering, Springer, 2014, pp. 457–472.
[2] Niek Tax, Ilya Verenich, Marcello La Rosa, Marlon Dumas, Predictive business process monitoring with LSTM neural networks, in: International Conference on Advanced Information Systems Engineering, Springer, 2017, pp. 477–492.
[3] Nijat Mehdiyev, Joerg Evermann, Peter Fettke, A novel business process prediction model using a deep learning method, Bus. Inf. Syst. Eng. 62 (2) (2020) 143–157.
[4] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, Donato Malerba, Using convolutional neural networks for predictive process analytics, in: 2019 International Conference on Process Mining (ICPM), IEEE, 2019, pp. 129–136.
[5] Irene Teinemaa, Marlon Dumas, Marcello La Rosa, Fabrizio Maria Maggi, Outcome-oriented predictive process monitoring: Review and benchmark, ACM Trans. Knowl. Discov. Data (TKDD) 13 (2) (2019) 1–57.
[6] Vincenzo Pasquadibisceglie, Annalisa Appice, Giovanna Castellano, Donato Malerba, Giuseppe Modugno, ORANGE: outcome-oriented predictive process monitoring based on image encoding and CNNs, IEEE Access 8 (2020) 184073–184086.
[7] Francesco Folino, Gianluigi Folino, Massimo Guarascio, Luigi Pontieri, Learning effective neural nets for outcome prediction from partially labelled log data, in: 2019 IEEE 31st International Conference on Tools with Artificial Intelligence (ICTAI), IEEE, 2019, pp. 1396–1400.
[8] Hans Weytjens, Jochen De Weerdt, Learning uncertainty with artificial neural networks for improved remaining time prediction of business processes, in: International Conference on Business Process Management, Springer, 2021, pp. 141–157.
[9] Manuel Camargo, Marlon Dumas, Oscar González-Rojas, Learning accurate LSTM models of business processes, in: International Conference on Business Process Management, Springer, 2019, pp. 286–302.
[10] Joerg Evermann, Jana-Rebecca Rehse, Peter Fettke, A deep learning approach for predicting process behaviour at runtime, in: International Conference on Business Process Management, Springer, 2016, pp. 327–338.
[11] Irene Teinemaa, Marlon Dumas, Anna Leontjeva, Fabrizio Maria Maggi, Temporal stability in predictive process monitoring, Data Min. Knowl. Discov. 32 (5) (2018) 1306–1338.
[12] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, 2018, arXiv preprint arXiv:1810.04805.
[13] Geetika T Lakshmanan, Davood Shamsi, Yurdaer N Doganata, Merve Unuvar, Rania Khalaf, A markov prediction model for data-driven semi-structured business processes, Knowl. Inf. Syst. 42 (1) (2015) 97–126.
[14] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, David Madigan, Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model, Ann. Appl. Stat. 9 (3) (2015) 1350–1371.
[15] Efrén Rama-Maneiro, Juan Vidal, Manuel Lama, Deep learning for predictive business process monitoring: Review and benchmark, IEEE Trans. Serv. Comput. (2021).
[16] Farbod Taymouri, Marcello La Rosa, Marlon Dumas, Fabrizio Maria Maggi, Business process variant analysis: Survey and classification, Knowl.-Based Syst. 211 (2021) 106557.
[17] Joerg Evermann, Jana-Rebecca Rehse, Peter Fettke, Predicting process behaviour using deep learning, Decis. Support Syst. 100 (2017) 129–140.
[18] Nicola Di Mauro, Annalisa Appice, Teresa Basile, Activity prediction of business process instances with inception CNN models, in: International Conference of the Italian Association for Artificial Intelligence, Springer, 2019, pp. 348–361.
[19] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[20] Farbod Taymouri, Marcello La Rosa, Sarah Erfani, Zahra Dasht Bozorgi, Ilya Verenich, Predictive business process monitoring via generative adversarial nets: the case of next event prediction, in: International Conference on Business Process Management, Springer, 2020, pp. 237–256.
[21] Zaharah A. Bukhsh, Aaqib Saeed, Remco M. Dijkman, ProcessTransformer: Predictive business process monitoring with transformer network, 2021, arXiv preprint arXiv:2104.00721.
[22] Abdulrahman Jalayer, Mohsen Kahani, Asef Pourmasoumi, Amin Beheshti, HAM-Net: Predictive business process monitoring with a hierarchical attention mechanism, Knowl.-Based Syst. 236 (2022) 107722.
[23] Vincenzo Pasquadibisceglie, Giovanna Castellano, Annalisa Appice, Donato Malerba, FOX: a neuro-fuzzy model for process outcome prediction and explanation, in: 2021 3rd International Conference on Process Mining (ICPM), IEEE, 2021, pp. 112–119.
[24] Bemali Wickramanayake, Zhipeng He, Chun Ouyang, Catarina Moreira, Yue Xu, Renuka Sindhgatta, Building interpretable models for business process prediction using shared and specialised attention mechanisms, Knowl.-Based Syst. 248 (2022) 108773.
[25] John J. Hopfield, Neural networks and physical systems with emergent collective computational abilities, Proc. Natl. Acad. Sci. 79 (8) (1982) 2554–2558.
[26] Sepp Hochreiter, Jürgen Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).
[28] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko, End-to-end object detection with transformers, in: European Conference on Computer Vision, Springer, 2020, pp. 213–229.
[29] Diederik P. Kingma, Jimmy Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.
[30] Andrew P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognit. 30 (7) (1997) 1145–1159.
[31] James Bergstra, Rémi Bardenet, Yoshua Bengio, Balázs Kégl, Algorithms for hyper-parameter optimization, Adv. Neural Inf. Process. Syst. 24 (2011).
[32] James Bergstra, Yoshua Bengio, Random search for hyper-parameter optimization, J. Mach. Learn. Res. 13 (2) (2012).
[33] Milton Friedman, The use of ranks to avoid the assumption of normality implicit in the analysis of variance, J. Amer. Statist. Assoc. 32 (200) (1937) 675–701.
[34] Janez Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.