Liu *et al. Journal of Cloud Computing* (2025) 14:36
https://doi.org/10.1186/s13677-025-00763-8

Journal of Cloud Computing

RESEARCH Open Access

Comparative evaluation of encoding techniques for workflow process remaining time prediction for cloud applications

Cong Liu[1,2], Wenjuan Liu[1], Na Guo[1*], Rongjia Song[3], Yan Gu[4], Long Cheng[4] and Qingtian Zeng[5]

Abstract
With the increasing adoption of cloud infrastructure for workflow process deployment, the workflow process remaining time prediction has received more attention due to its significance in improving response time and optimizing resource allocation. Event encoding techniques, by capturing the temporal dynamics and contextual dependencies of process events, improve the accuracy of workflow process remaining time predictions. However, there remains a lack of comprehensive empirical evaluation to analyze the impact of various event encoding techniques on prediction accuracy. To fill this gap, this paper conducts an extensive experimental evaluation of five state-of-the-art event encoding techniques, including One-Hot, Skip-Gram, CBOW (Continuous Bag-of-Word), FastText and GloVe, across nine prediction models based on LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), and QRNN (Quasi-Recurrent Neural Network). The evaluation utilizes eight real-world event logs to assess the accuracy of workflow remaining time predictions. The experimental results demonstrate that the GloVe encoding technique consistently yields superior prediction accuracy across the majority of prediction models and event logs.

Keywords Process mining, Remaining time prediction, Encoding techniques, Empirical evaluation

Introduction

In the wave of digital transformation, cloud computing has revolutionized enterprise workflow management with its flexibility and scalability. By migrating workflow activities and data processing to the cloud, enterprises have achieved efficient resource integration and utilization, but they are also faced with the challenge of managing and optimizing these cloud workflows, [23]. In this context, process mining has emerged as a powerful data analysis technique. By deeply analyzing event logs, process mining reveals process structures, behavioral patterns, and potential issues, providing a scientific basis for process optimization, risk control, and efficiency improvement. Particularly in cloud environments, where workflows are highly dynamic and span multiple organizations, the application of process mining can not only

*Correspondence:
Na Guo
guona_7@163.com
[1]School of Computer Science and Technology, Shandong University of Technology, Zibo 255000, China
[2]NOVA Information Management School, Nova University of Lisbon, Lisbon 1070-312, Portugal
[3]School of Management, Hangzhou Dianzi University, Hangzhou 210018, China
[4]School of Control and Computer Engineering, North China Electric Power University, Beijing 100096, China
[5]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao 266950, China

help enterprises better understand and monitor the state of their cloud workflows but also support continuous process optimization and innovation, [9].

Predictive process monitoring (PPM) is known as one of the most important challenging tasks in the process mining area, which aims to predict and analyze the future executing status of a process instance, e.g., the next activity, the remaining time, and the process outcome, [27, 28]. In general, PPM techniques use historical data collected by process-aware information systems to effectively avoid potential risks including deviations or missing bottlenecks, [29]. As a critical task in PPM, the main goal of remaining time prediction is to estimate the remaining execution time of running process instances [3]. Given that cloud services rely heavily on immediate response and efficient resource allocation, precise prediction of process remaining time can support enterprises to conduct efficient resource scheduling. Moreover, this capability enhances service quality and strengthens market competitiveness by enabling precise prediction of customer demand.

Traditional model-based remaining time prediction techniques typically consist of two consecutive steps. First, process models (e.g., transition systems and stochastic Petri nets) enriched with performance information is first constructed from scratch or discovered from event logs, [26, 31]. Then the remaining time of a running process instance is given with respect to the model. In recent years, machine learning and deep learning techniques have been increasingly applied in data-based remaining time prediction to achieve better accuracy [5, 6]. Therefore, discrete events need to be transformed into real vectors that can be used by deep learning models. As indicated in [19], effective event encoding techniques may have a positive effect on the prediction accuracy of the remaining time by representing the correlation between events.

Applying cloud computing techniques in the process remaining time prediction task mainly includes the following two challenges: 1) cross-organizational and cross-regional process data transmission faces the problem of privacy leakage; 2) there is a specific temporal order and correlation among events in a process instance, which may affect the remaining execution time of the process instance. To tackle this challenge, event encoding technique, which contains more event information and can effectively express context, is introduced. This paper experimentally evaluates the effect of five state-of-the-art encoding techniques, including One-Hot, Skip-Gram, CBOW (Continuous Bag-of-Word), FastText and GloVe, together with nine prediction models based on LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), and QRNN (Quasi-Recurrent Neural Network), on the remaining time prediction using eight real-life event logs. The evaluation results can provide valuable insights and guidance for researchers and practitioners to choose the most appropriate encoding technique in the remaining time prediction.

In general, the main contributions of this paper are summarized as follows:

- This paper reveals that event encoding techniques can substantially enhance the accuracy of workflow remaining time predictions by capturing complex temporal dynamics and contextual interdependencies inherent in event sequences.
- This paper presents an extensive experimental evaluation of five state-of-the-art event encoding techniques, which have been comprehensively assessed across nine various prediction models using eight real-world event logs, demonstrating their comparative effectiveness in various scenarios.
- The experiments demonstrate that the GloVe encoding technique consistently yields superior prediction accuracy, providing valuable insights and guidance for researchers and practitioners to choose the most appropriate encoding technique for workflow remaining time prediction.

The rest of this paper is organized as follows. Related work section introduces a review of the related work. Basic concepts section describes the basic concepts. Workflow process remaining time prediction model section details state-of-the-art remaining time prediction models. Event encoding techniques section illustrates event encoding techniques. Experimental evaluation section conducts empirically experiments using real-life event logs. Conclusion section summarizes this paper and presents further research directions.

Related work

This section briefly reviews existing work on workflow process remaining time prediction and encoding techniques.

Workflow process remaining time prediction

The existing remaining time prediction approaches can be divided into three categories, including model-driven, data-driven, and model-data-driven.

Model-driven approaches construct process models by event logs or domain experts to predict the process remaining time. For example, van der Aalst et al. propose a remaining time prediction method based on a transition system, which aims to mine the transition system model from event logs to efficiently record the different possible states of process instances, [29]. Then, the time in each state is marked to predict the remaining time. Rogge-Solti et al. mine stochastic Petri nets from event

logs, which are used to simulate the currently executing process instances for predicting remaining time [25]. The prediction results of these model-driven approaches are relatively easier to understand, but their prediction accuracy needs to be improved.

Data-driven approaches use machine learning techniques to predict the remaining time. Verenich et al. firstly cluster all process instances, and construct the remaining time prediction model according to the clustering results [30]. Van Dongen et al. predict the process remaining time by establishing a regression model [8]. Folino et al. combine data and clustering to further enhance the model-building ability for remaining time prediction [10]. The prediction accuracy of these data-driven approaches is higher than that of model-driven approaches. At present, new techniques in deep learning are increasingly applied to predictive process monitoring, which is the mainstream research direction. Tax et al. apply an LSTM neural network for business process prediction, thereby improving prediction accuracy [27]. Bukhsh et al. use the Transformer model for remaining time prediction, incorporating both activity and time attributes as inputs, which demonstrates the advantages of this advanced model [2]. However, most of these methods tend to learn highly complex models, which are often difficult to interpret. To address this issue, Guo et al. introduce a cascade prediction framework that enhances remaining time prediction by establishing correlations between input features and their impact on prediction outcomes [12]. Additionally, Cao et al. develop a remaining time prediction model based on gated RNNs, which improves model interpretability by constructing a reachability graph [4].

Model-data-driven approaches combine data with the process model and use machine learning to improve remaining time prediction accuracy, which address the drawbacks of relying excessively on either data or the process model alone. Polato et al. construct a prime Bayesian classification model for process states in a perceptually transition system to predict remaining time, [24]. Verenich et al. use a process tree to describe the workflow process, and train regression models at each process node to predict the remaining time, [30]. However, model-data-driven approaches usually complicate the overall prediction algorithm.

Encoding

Encoding techniques can be effectively applied to PPM to further analyze the event log. Encoding aims to convert discrete events into low-dimensional real number vectors, which can be used as inputs to deep learning models. Leontjeva et al. propose a complex sequence encoder based on indexing and Hidden Markov, which encodes the activity sequence in a process trace as a symbol sequence to improve the prediction accuracy, [18]. As the developing of natural language processing, Koninck et al. apply Word2vec and Doc2vec to process feature learning, [17], and Hake et al. combine Word2vec with neural networks to label each process model node by word embedding, [13]. Recently, Goyal et al. propose a graphic information encoding technique, which use graphs to represent process models, [11]. More specifically, it replace nodes and edges in the graph with activities and direct following relationships in the workflow process. In addition, graphic encoding can provide new possibilities for analyzing processes, such as capturing graphical structure and discovering similarities between different process models. Although various encoding techniques are applied to PPM, it is particularly important to choose the appropriate encoding technique for different event logs and deep models. However, there is no systematic empirical evaluation to quantitatively evaluate the influence of different encoding techniques on the remaining time prediction accuracy.

Basic concepts

Cloud computing

Cloud computing has emerged as a prominent computing paradigm, leveraging virtualization technologies to provide scalable and flexible resources on a pay-as-you-go basis. The fundamental characteristics of cloud computing include resource pooling, which facilitates efficient sharing and utilization of resources, elastic scalability that allows dynamic adjustments based on business needs, and robust security that ensures reliable data protection and business continuity. With these advantages, many enterprises prefer leasing infrastructure from Cloud Service Providers (CSPs) rather than establishing and maintaining their own local server infrastructure, thereby mitigating operational overhead and complexity.

As depicted in Fig. 1, cloud computing encompasses several critical stages. Users submit tasks to the cloud platform, which provisions and allocates the required resources based on the specific requirements of the tasks, ensuring efficient deployment and execution. During the execution phase, the cloud system continuously monitors performance metrics to guarantee successful task completion. Upon task completion, the results are aggregated and returned to the user, and the allocated resources are swiftly released to optimize overall resource utilization.

Event log

Definition 1 (Event). Event is an execution step of activity in the workflow system, which can be expressed as a tuple $e = \{caseid, a, time, p_{\{1,2,\ldots,n\}}\}$. Among them, *caseid* marks a process instance, *a* is the activity of the event, *time* is the execution time of the event by

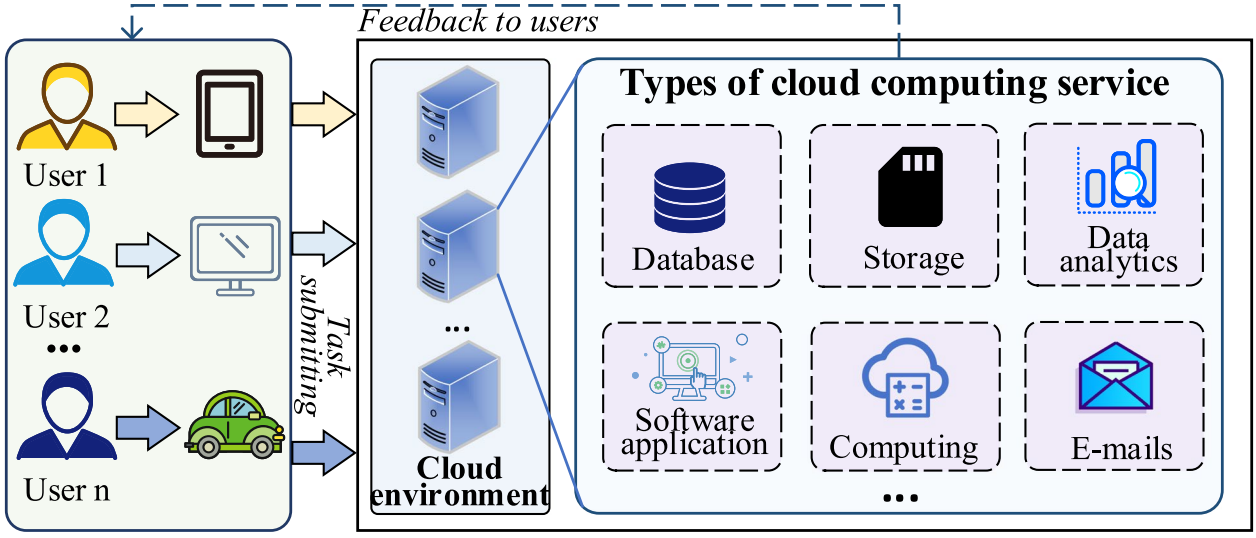

Fig. 1 The process of cloud computing

subtracting the end time of the event (*endtime*) and the start time of the event (*starttime*), $p_{\{1,2,\ldots,n\}}$ represent other attributes.

Definition 2 (Trace). Trace is a finite sequence of events, which can be expressed as $\sigma = \{e_1, e_2, \cdots, e_{|\sigma|}\}$. The execution time of trace is $\sigma_{time} = \sigma_{endtime} - \sigma_{starttime}$.

Definition 3 (Trace Prefix). Trace prefix refers to the first k events in the trace σ, expressed as $\sigma^{(k)} \subseteq \{e_1, e_2, \cdots, e_k\}$. Trace prefix remaining time $RT\left(\sigma^{(k)}\right) = \sigma_{endtime} - e_{k,endtime}$.

Definition 4 (Process Instance). Process instance refers to the complete execution of the business from the beginning to the end, which can be expressed as a triplet $z = \{Cid, \sigma, P_{\{1,2,\ldots,m\}}\}$. Among them, $P_{\{1,2,\ldots,m\}}$ represents other attributes in the process instance. In general, $\sigma(1)$ is the start event in the process instance, and $\sigma(|\sigma|)$ is the end event in the process instance.

Definition 5 (Event Log). Event Log is a set of process instances, which records the workflow system execution and is denoted as $L = \{\sigma_1, \sigma_2, \ldots, \sigma_{|u|}\}$.

Workflow process remaining time prediction

Figure 2 provides an overview of workflow process remaining time prediction in cloud computing, including event encoding, remaining time prediction, and comparative evaluation three pivotal components. Specifically, the business processes in the cloud computing environment are recorded in detail by the system to form event logs. Subsequently, these event logs are transformed into event vectors by diverse encoding techniques. These event vectors are used as inputs of the prediction model to predict the remaining time of the business process. Finally, the effect of diverse encoding techniques on the business process remaining time prediction is evaluated. This section introduces event encoding, dataset construction, model training, and prediction in detail.

According to Definition 1, the event is composed of the activity and other attributes. In this paper, the activity and the execution time are used to achieve the event encoding. The execution time needs to be discretized according to different activities to refine the time range. The execution time of event e after discretization is as below.

$$Execution\_Time(e) = \left\lfloor \frac{Time(e) - Time_{min}(e,a)}{Time_{max}(e,a) - Time_{min}(e,a)} \bullet N \right\rfloor \quad (1)$$

As shown in Eq. (1), *Time*(*e*) represents the execution time of event *e*, the activity performed by event *e* is recorded as *a*, $Time_{max}(a)$ and $Time_{min}(a)$ represent the longest and shortest execution time of the activity *a* in the event log. N represents the number of discrete classifications, and this paper sets N to 10 as default. Then, the activity in the event is spliced with $Execution\_Time(e)$ to obtain the event representation $\theta^{<a,Execution\_Time(e)>} \subset \mathbb{R}$.

The remaining time prediction task using deep learning models consists of two critical phases that are training and prediction. The training phase aims to fully use the historical process instances set in the event logs to construct the remaining time prediction model *f*, and the prediction phase aims to effectively predict the remaining time of the ongoing process instances (i.e., trace prefixes) using the prediction model *f*.

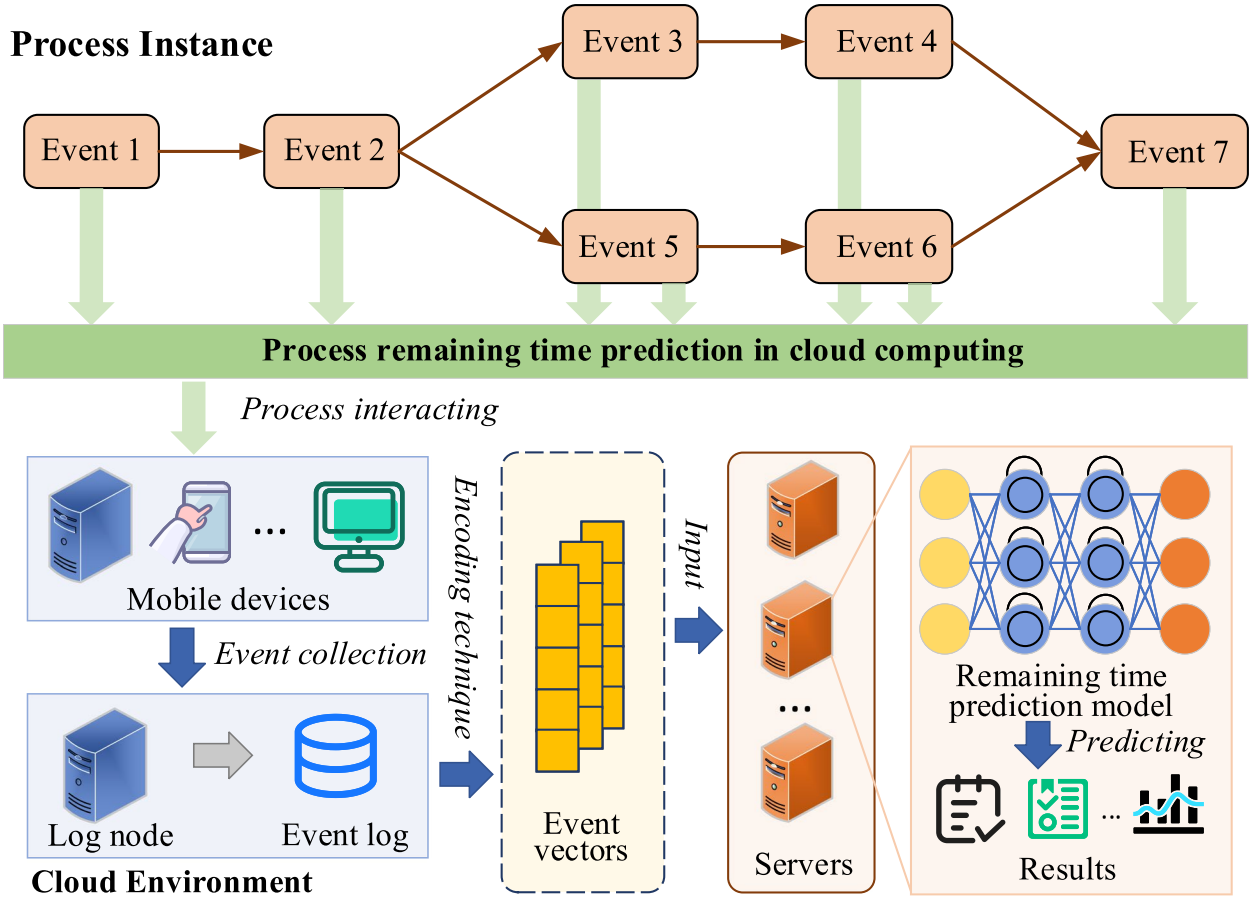

Fig. 2 Workflow process remaining time prediction in cloud computing

However, the event log should be converted into a dataset that the deep learning models can utilize, and the dataset is established according to different trace prefix lengths, as shown in Eqs. (2) and (3).

$$D_{\{1,2,\cdots,(len(\sigma)-k)\}} = \left\{\sigma^k, RT\left(\sigma^{(k)}\right) | k < len\left(\sigma\right), \sigma^{(k)} \in L\right\} \tag{2}$$

$$D_k = D_1 \bigcup D_2 \bigcup \cdots \bigcup D_{(len(\sigma)-k)} \tag{3}$$

Equation (2) is to traverse every trace in the event log, and intercept the trace within the length of $k \in [k_{min}, k_{max}]$. A trace prefix and its corresponding remaining time are obtained as a sample. Equation (3) assembles all samples to form the dataset D_k.

The training phase is to learn the remaining time prediction model f based on the generated dataset D_k. Moreover, the regular term is used to overcome over-fitting, as shown in Eq. (4).

$$f^* = \underset{f \in F}{argmin} \sum_{\sigma^{(k)} \in D_k} \left(f\left(\sigma\right) - t\right)^2 + \Omega\left(f\right) \tag{4}$$

Workflow process remaining time prediction model

Quasi recurrent neural network

Due to the advantages of Recurrent Neural Network (RNN) in processing sequential data, many researchers applied its common variants, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), to the remaining time prediction task, and achieved better prediction accuracy than traditional approaches, [22].

LSTM can solve the problems of RNN gradient disappearance and gradient explosion, [14]. LSTM neuron is calculated as follows.

$$(h_t, C_t) = LSTM(x_t, h_{t-1}, C_{t-1}) \tag{5}$$

As shown in Eq. (5), x_t is the input at time t, C_{t-1}, C_t, h_{t-1}, h_t represents the cell state and output at time t–1 and t.

GRU is simplified based on the LSTM. GRU changes the input gate, forget gate, and output gate of LSTM to reset gate and update gate, which not only preserves the LSTM characteristics, but also further improves the speed of network training, [7]. The GRU neuron is calculated as follows.

$$h_t = GRU(x_t, h_{t-1}) \tag{6}$$

However, both LSTM and GRU rely on the output of the previous time step when calculating the current time step, resulting in a longer computation time and difficulties in processing long sequence data. The Quasi-Recurrent Neural Network (QRNN) addresses this issue by using a neural sequence modeling method with alternating convolutional layers to achieve parallel processing of sequence data, [1]. QRNN includes two essential modules, i.e., convolution and pooling layers. When sequence data $X = (x_1, x_2, \ldots, x_T)$ is used as the input to the convolution layer, it is convolved in the time dimension using filters of number m and width d, producing a new sequence $Z = (z_1, z_2, \ldots, z_T)$. If the filter width d in the convolutional layer is changed to the variable parameter h, the width of z_t obtained after feeding the sequence data into the convolutional layer is $x_{t-h+1} \sim x_t$. As the filter width increases, more n-gram features can be computed. After obtaining the output of the convolutional layer, the pooling layer is used to extract feature information from the output, thereby reducing the number of features. QRNN neuron is calculated as follows.

$$(h_t, C_t) = QRNN(X, C_{t-1}) \tag{7}$$

Remaining time prediction models based on attention and bidirectional mechanisms

To make better use of the relationship between events in the trace prefix, the bidirectional mechanism and attention mechanism are integrated into the LSTM, GRU, and QRNN (denoted as Bi-LSTM, Att-Bi-LSTM, Bi-GRU, Att-Bi-GRU, Bi-QRNN, Att-Bi-QRNN), to further improve the remaining time prediction accuracy. The application details of the two mechanisms in the remaining time prediction model are as follows.

Context encoding based on bidirectional mechanism

Bi-LSTM, Bi-GRU, and Bi-QRNN consist of forward and reverse LSTM, GRU, and QRNN layers, respectively. The hidden vectors output by the forward LSTM, GRU, and QRNN are $\overrightarrow{h_{t\_LSTM}}$, $\overrightarrow{h_{t\_GRU}}$, $\overrightarrow{h_{t\_QRNN}}$, and the hidden vectors output by the reverse LSTM, GRU, and QRNN are $\overleftarrow{h_{t\_LSTM}}$, $\overleftarrow{h_{t\_GRU}}$, $\overleftarrow{h_{t\_QRNN}}$. The context encoding obtained by separately splicing the two modules is as follows.

$$h_{t\_LSTM} = (\overrightarrow{h_{t\_LSTM}}, \overleftarrow{h_{t\_LSTM}}) \tag{8}$$

$$h_{t\_GRU} = (\overrightarrow{h_{t\_GRU}}, \overleftarrow{h_{t\_GRU}}) \tag{9}$$

$$h_{t\_QRNN} = (\overrightarrow{h_{t\_QRNN}}, \overleftarrow{h_{t\_QRNN}}) \tag{10}$$

Trace encoding based on attention mechanism

After obtaining the context encoding for each time step of trace prefixes, the attention mechanism is used to learn the weight of each event, resulting in the final output. The calculation formula is as follows.

$$v = \sum_{t=1} a_t \cdot h_{t\_LSTM},\ h_{t\_GRU},\ h_{t\_QRNN} \tag{11}$$

As shown in Eq. (11), a_t represents the weight of the context encoding at time t, reflecting the importance of the event at time t to the remaining time prediction.

Event encoding techniques

In natural language processing, high-quality encoding techniques can significantly impact the training effectiveness of deep learning models. Therefore, the concept of pre-training is proposed to provide pre-trained word vectors for these models. Similarly, every event involved in a workflow process can be regarded as a word in natural language, implying that the quality of event encoding may affect the accuracy of remaining time prediction. To explore the influence of encoding techniques on remaining time prediction, this section introduces five different encoding techniques for encoding events.

One-Hot

One-Hot encoding is a common technique for encoding discrete data to adapt to the input of deep learning model. However, the shortcoming is that dimensional explosion can easily occur if the feature space further increases due to many categories. In addition, the semantic gap exists in One-Hot encoding, which means the event encoding cannot reflect the correlation between events.

Word2Vector

Word2Vector, [21], proposed by Mikolov, is an approach that can quickly train word vectors. It can predict words with strong correlations to given words, effectively overcoming the problems of dimensional explosion and semantic gap in One-Hot encoding. Word2Vector can be classified into two types, i.e., CBOW (Continuous Bag-of-Word) model and Skip-Gram (Continuous Skip-gram) model.

CBOW

The idea of CBOW model-based events encoding is to use contextual events to predict the current event, thus showing the similarity of adjacent events in process instances. The input layer of the CBOW model is the binary vector θ_{con} of contextual events, the interception

window size of contextual events is set to 2 in this paper. The contextual event vector is multiplied with the corresponding weight matrix $W_{V\times N}$ between the input and the hidden layer, and then the hidden layer vector h_{out} is obtained by weighted average. Then, the hidden layer vector h_{out} will be multiplied with the weight matrix $W^{'}_{V\times N}$ between the hidden layer and the output layer, and the predicted event probability distribution p is obtained by *Softmax* function (a normalization operation).

$$h_{out} = \frac{1}{c} W_{V\times N} \cdot \left(\sum_{j=1}^{c} \theta_j \right) \tag{12}$$

$$p = Softmax\left(h_{out} \cdot W^{'}_{V\times N} \right) \tag{13}$$

Skip-Gram

The basic idea of Skip-Gram model-based event encoding is quite similar to CBOW model-based event encoding. However, the difference is that the Skip-Gram model uses the current event to predict contextual events. The input layer of the Skip-Gram model is the One-Hot encoding vector θ_{cur} of the current event. When the input weight matrix is $W^{'}_{V\times N}$, the current encoding vector is multiplied by the weight $W^{'}_{V\times N}$ and weighted average to obtain the hidden vector h_{out}. Then, the hidden vector h_{out} is subsequently multiplied with the weight matrix $W_{V\times N}$ between the output layers, and the contextual event probability distribution p' to be predicted is obtained by *Softmax* function.

$$h_{out} = sum\left(\theta \cdot W^{'}_{V\times N} \right) \tag{14}$$

$$p' = Softmax(h_{out} \cdot W_{V\times N}) \tag{15}$$

FastText

FastText, [16], proposed by Mikolov, is a new text classification algorithm that followed Word2Vector. Its structure is similar to CBOW, but the key difference is that the FastText model inputs all events and their n-gram features to predict their corresponding labels. More specifically, the central role of the n-gram feature is to optimize and refine the relationship between event sequences, [20]. The n-gram feature is mapped to the hidden layer after linear transformation, where it is then weighted and averaged. The category label of each event is obtained in the output layer using the *Softmax* function.

In addition, FastText uses a hierarchical Softmax method based on Huffman trees to reduce computational complexity. This method mainly uses the frequency of different categories as weights to build the Huffman tree, with each leaf node representing a label. The more frequently a label appears, the shorter its path in the Huffman tree. If the depth of a node is d+1, its parent nodes are $S1, S2, \ldots, S_d$, calculated as shown in Eq. (16).

$$P\left(S_{d+1}\right) = \sum_{j=1}^{d} P\left(S_j\right) \tag{16}$$

GloVe

GloVe, [15], is proposed based on Word2Vector. Its basic idea is to decompose the word co-occurrence matrix and use the local context window to train the word vector model. Compared with Word2Vector, GloVe uses global word frequency statistics to obtain dense vectors of words, which can accurately reflect semantic information. Additionally, when measuring word-to-word similarity, GloVe calculates either cosine similarity or Euclidean distance. The implementation of event encoding using GloVe includes three steps as follows.

Step 1: The co-occurrence matrix X is constructed based on the events extracted from the event logs. In the co-occurrence matrix X, θ_{ij} represents the number of co-occurrence of event θ_i and its contextual event θ_j under the fixed context window size. According to the distance between two events in the context window, the corresponding weight is obtained by calculating the decay rate, that is, the correlation degree of events.

Step 2: The approximate relationship between the event vector and the co-occurrence matrix X is established, and the calculation process is as follows.

$$\theta_i^T \tilde{\theta_j} + b_i + \tilde{b_j} = log\theta_i\theta_j \tag{17}$$

As shown in Eq. (17), θ_i^T represents the transpose vector of the contextual event θ_i. $\tilde{\theta_j}$ represents the vector of the current event θ_j, b_i and $\tilde{b_j}$ represents the event vector bias.

Step 3: The loss function is built according to Eq. (17). The basic idea is to construct residual least square method, in which the difference is calculated by co-occurrence matrix and event vector, and different co-occurrence matrix event vectors have different weights.

$$J = \sum_{i,j}^{R} f\left(\theta_{ij}\right)\left(\theta_i^T \tilde{\theta_j} + b_i + \tilde{b_j} - log\left(\theta_{ij}\right)\right) \tag{18}$$

As shown in Eq. (18), R represents the number of events in the trace. f represents the weighting function. When the co-occurrence matrix X reaches the minimum value after being computed by the loss function J, the final event vector is obtained.

Table 1 Event log basic information statistics

Datasets	Traces	Variants	Events	Activities	Traces Length	
					Max traces length	Min traces length
BPIC_2012_A	13087	4457	73022	10	10	3
BPIC_2012_O	5015	1819	41728	7	39	4
BPIC_2012_W	9658	2465	147450	6	153	1
Helpdesk	3804	1393	13710	9	14	1
Hospital_Billing	100000	1020	451359	18	217	1
Sepsis_Cases	1050	846	15214	16	185	3
Prepaid_Travel_Costs	2099	202	18246	29	21	1
Order	1000	9	6358	8	8	6

Experimental evaluation

This section shows the experimental setup and results analysis using eight real-life event logs. Our approach is implemented in Pytorch framework. The source code of our implementation and all experimental results are available[1]. The computer configuration used is: PC Intel Core i5-10400F 2.90 GHz, NVIDIA GeForce RTX 2070 SUPER environment.

Datasets

Eight real-life event logs from 4TU Center for Research are used in our experiments, i.e., BPIC_2012_A, BPIC_2012_O, BPIC_2012_W[2], Helpdesk[3], Hospital_Billing[4], Sepsis_Cases[5], Prepaid_Travel_Costs[6] and Order[7]. These event logs come from different fields, such as financial institutions, software companies and medical systems. The detail information is shown in Table 1. It can be seen that these event logs cover business processes with different data scales and complexity, which can comprehensively evaluate the effectiveness of event encoding techniques applied in this paper.

Experimental setting

In this experiment, Mean Absolute Error (MAE) is used as the evaluation measure for the workflow process remaining time prediction model, which calculates the absolute value of the difference between the predicted value $f\left(\sigma^{(k)}\right)$ and the actual value $RT\left(\sigma^{(k)}\right)$ of trace prefixes $\sigma^{(k)}$. The lower the MAE value, the higher the prediction accuracy. The calculation process is shown as below.

$$MAE\left(f\right)=\sum_{\left(\left(\sigma^{(k)},RT\left(\sigma^{(k)}\right)\right)\in D_k\right)}\left|f\left(\sigma^{(k)}\right)-RT\left(\sigma^{(k)}\right)\right| \quad (19)$$

To verify the impact of encoding techniques on the process remaining time prediction task, the event vectors generated by five encoding techniques (i.e., One-Hot, Skip-Gram, CBOW, FastText and GloVe) are used as the input of nine prediction models (i.e., LSTM, Bi-LSTM, Att-Bi-LSTM, GRU, Bi- GRU, Att-Bi-GRU, QRNN, Bi-QRNN, and Att-Bi-QRNN). The experimental results of remaining time prediction models are obtained by 5-fold cross-validation. The basic parameters of encoding techniques and prediction models are also adjusted for obtaining high-quality event vectors: (1) The dimensions of the input event vector: {3, 5, 300}; (2) Learning rate: {0.0025, 0.005, 0.001}; (3) Epoch: 200; (4) Batch size: {8, 16, 32, 512}; and (5) Optimization algorithm: Adam.

Results of encoding techniques on the remaining time prediction

To verify the influence of event encoding techniques on the remaining time prediction accuracy, the test is carried out on eight event logs. The experimental results are shown in Fig. 3, where $Encoding\_Avg$ represents the average MAE value measured by all prediction models using a specific event encoding technique. The One-Hot encoding is used as the baseline technique because it cannot reflect the relationship between events.

As shown in Fig. 3, compared with One-Hot, the other four event encoding techniques all improve remaining time prediction accuracy in eight event logs with different data scales. This indicates that they can represent the relationships between events more effectively and provide the prediction model with more abundant input information by mapping adjacent events to similar positions in the vector space. Meanwhile, it also shows that these encoding techniques are scalable in large data scales. Furthermore, the $Encoding\_Avg$ of FastText event encoding technique is lower than that of Skip-Gram and CBOW, demonstrating that FastText fully considers the characteristics of n-grams and uses hierarchical Softmax to find the maximum event probability label, thereby obtaining a better event representation vector. Compared with Skip-Gram, CBOW, and FastText based

[1] https://github.com/gn874682003/Event-Enoding-Evaluation

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

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

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

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

[6] https://doi.org/10.4121/uuid:5d2fe5e1-f91f-4a3b-ad9b-9e4126870165

[7] https://doi.org/10.4121/uuid:0f09a7ff-3ba8-4acd-9451-1e3f32ee31f3

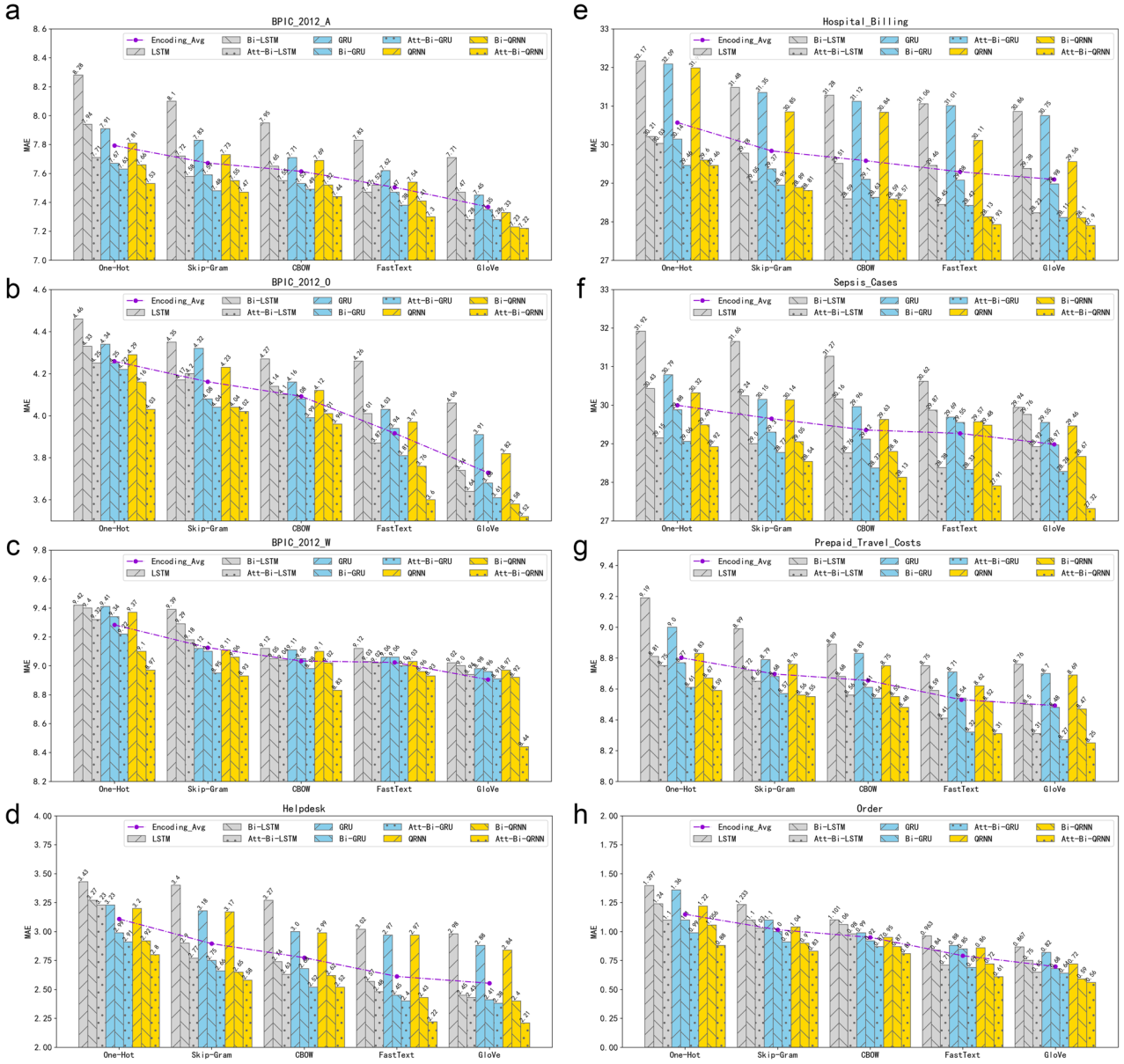

Fig. 3 Results of different encoding techniques in remaining time prediction

on local corpus, the GloVe event encoding technique based on global corpus, achieves the best results. This is because GloVe builds a co-occurrence matrix by a sliding window, which fully captures global process information. This global context helps GloVe to better represent the dependencies between events in workflow processes, which is crucial for accurate remaining time prediction. However, there is an exceptions, i.e., the Att-Bi-LSTM model of the Sepsis_Cases log. One possible explanation is that the Sepsis_Cases log contains many variants and has a relatively small data scale, which may impact the stability of model training. To sum up, GloVe is more suitable for predicting the process remaining time among the five event encoding techniques evaluated.

For prediction models, the $Encoding\_Avg$ of QRNN is lower than that of LSTM and GRU across all event logs, indicating that QRNN is more suitable for the remaining time prediction task. After integrating the bidirectional and attention mechanisms, the MAE value of the prediction model decreases to varying degrees. This is because the bidirectional and attention mechanisms enhance the context and focus on important information, thereby improving prediction accuracy. Among all prediction models, the Att-Bi-QRNN prediction model achieves the lowest MAE value, verifying its superiority.

Based on the experimental results and the above analysis, the exploration conclusions are as follows.

- Compared with other event encoding techniques, GloVe event encoding has the greatest improvement on the prediction accuracy, so it is considered to be more suitable for the remaining time prediction task.
- For different prediction models, the combination of GloVe event encoding technique and Att-Bi-QRNN prediction model realizes the minimum MAE. Therefore, this combination is more suitable for the remaining time prediction task.

To further illustrate the influence of event encoding on the semantic correlation between events, the event vectors obtained by FastText and GloVe are visualized using the BPIC_2012_O event log as an example. The results are shown in Fig. 3. In this experiment, the event vector dimension is set to 5, and the window size is set to 2. Additionally, principal component analysis is performed to ensure the presentation on a 2-dimensional plane.

In Fig. 4, events are represented using the activity name and the discrete value of execution time, and events of the same activity are marked with the same color. The distance between every two events indicates their correlation. As can be seen from Fig. 4, after being encoded by event encoding techniques, some events appear to be clustered. For instance, there is a trace of the event log ⟨ SELECTED_7, CREATED_8, SENT_8, SELECTED_5, CANCELLED_7, CREATED_5, SENT_5, SENT_BACK_6, ACCEPTED_0⟩. Where events SENT_8, SELECTED_5, CANCELLED_7, SENT_BACK_6, ACCEPTED_0 are clustered in brown boxes, events SELECTED_7, CREATED_8 are clustered in red boxes, and events CREATED_5 and SENT_5 are clustered in purple boxes. Accordingly, the clustering relationship basically reflects the existence of a

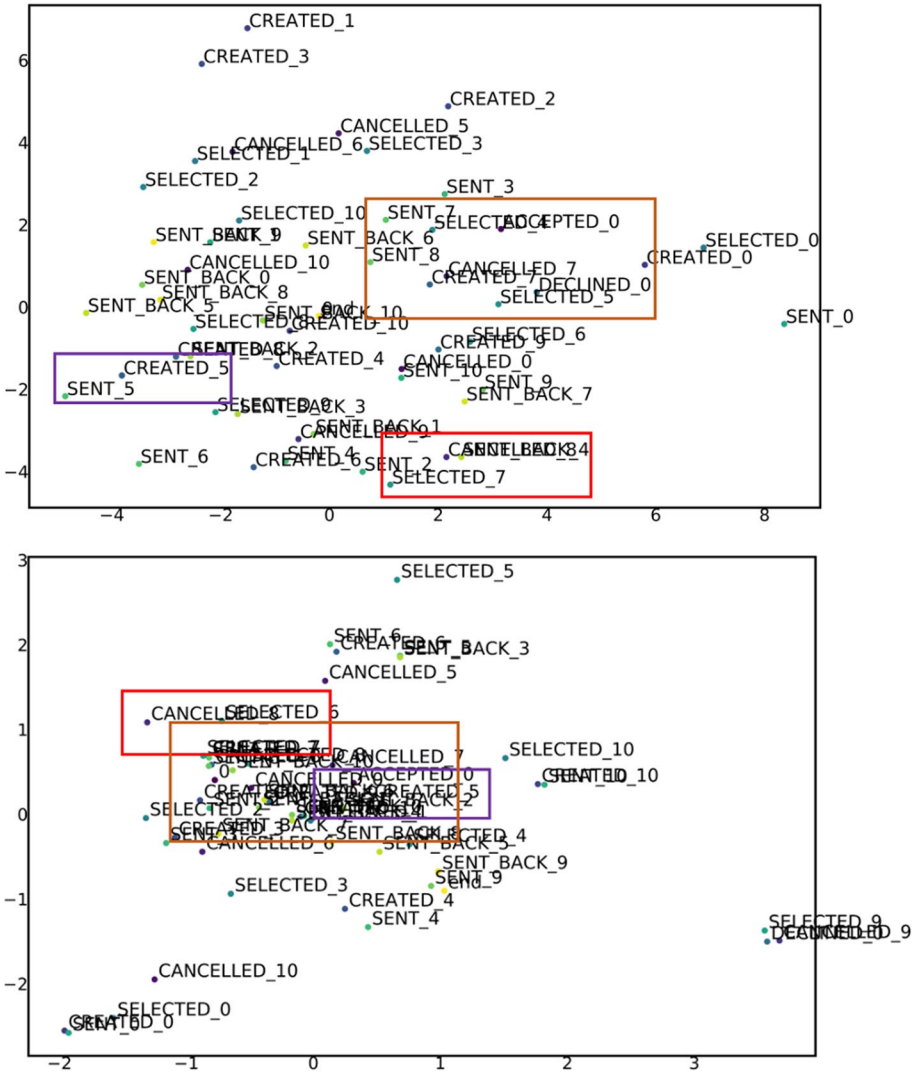

Fig. 4 The diagram of event vector visualization

Table 2 Training time comparison in different event encoding (Unit: Second)

	BPIC_2012_A	BPIC_2012_O	BPIC_2012_W	Helpdesk	Hospital_Billing	Sepsis_Cases	Prepaid_Travel_Costs	Order
Skip_Gram	718.999	489.147	1965.043	104.536	5393.180	487.129	600.952	158.294
CBOW	600.036	354.369	1324.373	113.576	4481.048	362.263	442.105	149.503
FastText	490.300	302.685	1222.422	101.388	4005.434	348.511	371.159	104.509
GloVe	400.849	212.630	621.260	92.420	12013.450	274.313	1353.296	71.234

correlation between events which is also consistent with the semantic relationship of events.

Results of encoding techniques training time

This section tests the training time of different event encoding techniques on eight event logs, and the results are shown in Table 2. In the experiments, the same parameter settings are used for each event encoding technique: (1) Number of iterations: 200; (2) Learning rate: 0.1; (3) Batch size: 32; (4) Event vector dimension: 300.

The selection of event encoding techniques not only considers the influence on prediction accuracy but also takes into account the time performance. Since One-Hot encoding is simple and does not require training, Table 2 only shows the training time of the other four event encoding techniques. It can be seen that GloVe has the shortest training time across all event logs, demonstrating its superiority in time performance. However, there are two exceptions, i.e., Hospital_Billing and Prepaid_Travel_Costs event logs. The possible explanation is that these two event logs have a large number of activities and large data scale, which leads to a long calculation time of the co-occurrence matrix. Overall, the GloVe event encoding technique has advantages in both impact on prediction accuracy and training time, making it effectively suitable for workflow process remaining time prediction tasks.

Conclusion

This paper mainly combines different remaining time prediction models and event encoding techniques to evaluate the impact of different encoding techniques on the remaining time of workflow processes, so as to better apply them to cloud workflow processes. Using the event vectors obtained by different event encoding techniques as the input of the prediction models, the remaining time prediction effect obtains different degrees of improvement, which indicates that high-quality event encoding helps to capture the correlation between process events.

The application of event encoding techniques can improve the prediction accuracy of remaining time, but there is still a lack of effective approaches to determine the encoding dimension for different workflow processes. Additionally, more event encoding techniques and event attribute information should be considered. Therefore, our future work focus on expanding the application of advanced event encoding techniques in process prediction and exploring methods to provide appropriate encoding dimensions for different workflow processes.

Authors' contributions

Cong Liu conceived the project and directed the research. Wenjuan Liu contributed to the development and fine-tuning of the algorithm. Guo Na performed the data analysis and wrote the main manuscript text. Yan Gu prepared figures 1-2. Rongjia Song, Long Cheng, and Qingtian Zeng provided essential theoretical insights and critically revised the manuscript for important intellectual content. All authors contributed to the writing and editing of the manuscript.

Funding

This work was partly supported by the National Key R&D Program of China under Grant 2022ZD0119501, National Natural Science Foundation of China under Grant 62472264 and 52374221, the Taishan Scholars Program of Shandong Province under Grants ts20190936 and Grant tsqn201909109, the Natural Science Excellent Youth Foundation of Shandong Province under Grant ZR2021YQ45, and the Youth Innovation Science and Technology Team Foundation of Shandong Higher School under Grant 2021KJ031.

Data availability

Data is provided within the manuscript.

Declarations

Competing interests

The authors declare no competing interests.

Received: 14 October 2024 / Accepted: 1 June 2025
Published online: 03 July 2025

References

1. Bradbury J, Merity S, Xiong C et al (201 6) Quasi-recurrent neural networks. arxiv:1611.01576.
2. Bukhsh Z, Saeed A, Dijkman R (2021) Processtransformer: predictive business process monitoring with transformer network. arxiv:2104.00721.
3. Cao R, Ni W, Zeng Q et al (2021) Remaining time prediction for business processes with concurrency based on log representation. China Commun 18(11):76–91
4. Cao R, Zeng Q, Ni W et al (2023) Business process remaining time prediction using explainable reachability graph from gated rnns. Appl Intell 53(11):13178–13191
5. Cao R, Zeng Q, Ni W et al (2023) Explainable business process remaining time prediction using reachability graph. Chin J Electron 32(3):625–639
6. Cheng L, Du L, Liu C et al (2024) Multi-modal fusion for business process prediction in call center scenarios. Inf Fusion 108:102362
7. Cho K, Merrienboer B, Bahdanau D et al (2014) On the properties of neural machine translation: Encoder-decoder approaches. arxiv:1409.1259
8. Dongen V, Crooy W, Van Der Aalst (2008) Cycle time prediction: When will this case finally be finished. In: OTM Confederated International Conferences on the Move to Meaningful Internet Systems. Springer Berlin Heidelberg, p 319–336
9. El-Gharib NM, Amyot D (2019) Process mining for cloud-based applications: A systematic literature review. In: 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, United States, p 34–43

10. Folino F, Guarascio M, Pontieri L (2014) Mining predictive process models out of low-level multidimensional logs. In: Proceedings of the International Conference on Advanced Information Systems Engineering. Springer-Verlag, Berlin, p 533–547
11. Goyal P, Ferrara E (2018) Graph embedding techniques, applications, and performance: A survey. Knowl-Based Syst 151:78–94
12. Guo N, Liu C, Li C et al (2024) Explainable and effective process remaining time prediction using feature-informed cascade prediction model. IEEE Trans Serv Comput 17(3):949–962
13. Hake P, Zapp M, Fettke P (2017) Supporting business process modeling using rnns for label classification. In: International Conference on Applications of Natural Language to Information Systems. Springer-Verlag, Berlin, p 283–286
14. Hochreiter S, Schmidhuber J (1977) Long short-term memory. Neural Comput 9(8):1735–1780
15. Jeffrey P, Richard S, Christopher D (2014) Glove: Global vectors for word representation. In: In Proceedings of the 2014 conference on emporocal methods in natural language processing. Association for Computational Linguistics, USA, p 1532–1543
16. Joulin A, Edouard G, Piotr B et al (2016) Bag of tricks for efficient text classification. arxiv:1607.01759.
17. Koninck P, Broucke S, Weerdt J (2018) act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes. Lect Notes Comput Sci 2018:305–321
18. Leontjeva A, Conforti R, Francescomarino C (2016) Complex symbolic sequence encodings for predictive monitoring of business processes. In: International Conference on Business Process Management. Springer-Verlag, Berlin, p 297–313
19. Liu T, Ni W, Sun Y (2020) Predicting remaining business time with deep transfer learning. Data Anal Knowl Discov 4(Z1):134–142
20. Mascardi V, Locoro A, Rosso P (2009) Automatic ontology matching via upper ontologies: A systematic evaluation. IEEE Trans Knowl Data Eng 22(5):609–623
21. Mikolov T, Corrado G, Kai C et al (2013) Efficient estimation of word representations in vector space. arxiv:1301.3781
22. Ni W, Sun Y, Liu T (2020) Business process remaining time prediction using bidirectional recurrent neural networks with attention. Comput Integr Manuf Syst 26(6):1564–1572
23. Pham TP, Durillo JJ, Fahringer T (2017) Predicting workflow task execution time in the cloud using a two-stage machine learning approach. IEEE Trans Cloud Comput 8(1):256–268
24. Polato M, Sperduti A, Burattin A (2018) Time and activity sequence prediction of business process instances. Computing 100(9):1005–1031
25. 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, Berlin, p 389–403
26. Senderovich A, Weidlich M, Gal A (2015) Queue mining for delay prediction in multi-class service processes. Inf Syst 53:278–295
27. Tax N, Verenich I, La Rosa M (2017) Predictive business process monitoring with lstm neural networks. In: Proceedings of the International Conference on Advanced Information Systems Engineering. Springer-Verlag,Berlin, p 477–492
28. Teinemaa I, Dumas M, Rosa M et al (2019) Outcome-oriented predictive process monitoring: Review and benchmark. ACM Trans Knowl Discov Data 13(2):17
29. Van Der Aalst W, Schonenberg M, Song M (2011) Time prediction based on process mining. Inf Syst 36(2):450–475
30. Verenich I, Nguyen H, Rosa M (2017) White-box prediction of process performance indicators via flow analysis. In: Proceedings of the International Conference on Software and System Process. ACM, New York, p 85–94
31. Zhao H, Li S, Chen Q (2019) Method of time prediction for business process. J Chin Comput Syst 40(2):42–48

Publisher's Note