


Article

Predictive Business Process Monitoring Approach Based on Hierarchical Transformer

Weijian Ni , Gang Zhao , Tong Liu *, Qingtian Zeng and Xingzong Xu

College of Computer Science and Engineering, Shandong University of Science and Technology,
Qingdao 266000, China

* Correspondence: liu_tongtong@foxmail.com

Abstract: Predictive business process monitoring is predicting the next stage of the business process based on the sequence of events that have occurred in the business process instance, which is positive for promoting the rational allocation of resources and the improvement of execution efficiency. There are drawbacks in modeling business process instances, such as conceptual drift phenomenon and long event sequences. Therefore, we propose a hierarchical Transformer-based business process prediction model to improve the performance of the Transformer-based predictive business process monitoring model. We encode the event features using two different encoding methods to obtain the relationship between activities and attributes. A drift detection algorithm is proposed to segment the business process and calculate the correlation between activities and segments by using cross-attention. Furthermore, learnable position encoding is designed to capture the relative position information of subsequences. Finally, the information of different granularity, such as event attributes, event subsequences, and complete instances, is fused by different weights. Experiments were run on seven real event logs for the next activity prediction and remaining time prediction, and the next activity prediction accuracy improved by 6.32% on average, and the mean absolute error of remaining time prediction reduces by 21% on average.

Keywords: business process; predictive business process monitoring; Transformer; concept drift detection



Citation: Ni, W.; Zhao, G.; Liu, T.; Zeng, Q.; Xu, X. Predictive Business Process Monitoring Approach Based on Hierarchical Transformer. *Electronics* **2023**, *12*, 1273. <https://doi.org/10.3390/electronics12061273>

Academic Editor: Domenico Ursino

Received: 13 February 2023

Revised: 20 February 2023

Accepted: 2 March 2023

Published: 7 March 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Process mining, as a cross-field of research in business process management fields, data mining, and formal modeling and analysis, aims to discover process knowledge from the massive event logs of modern information systems, ensuring monitoring and improvement of real business systems [1]. Process mining has recently received much attention from academia and industry and is recognized as one of the key technologies for achieving new business intelligence. Predictive business process monitoring is an emerging research direction in process mining, where the goal is to predict the possible next state of an executing business process instance. The common predictive goals are the next executed activity of the business process, the remaining execution time, the execution result, etc. Accurate business process prediction results enable effective predictive monitoring of business processes, which is valuable in tasks such as optimizing business process resource allocation, execution bottleneck analysis, efficiency optimization, and avoiding unintended execution outcomes [2].

As organizations such as enterprises become more complex, systems, applications, and devices in the organization generate a large number of logs and sensor data, so process mining requires the automation of analysis and the extraction of valid information from the massive amount of data. Moreover, the interaction between different business processes is very complex, so it is important to discover a comprehensive view of the entire business process by virtue of its comprehensive ability to analyze multiple business processes simultaneously. Finally, as processes change with technological advances or resource

scheduling, previous process models no longer match current processes, and process mining requires analyzing them in real-time, making process models more sensitive to data changes.

Early research in predictive business process monitoring focused on constructing formal models of business processes such as Transition Systems [1], Finite State Machine [3], and Stochastic Petri Nets [4] to predict the subsequent state of business process instances. With the growing development of information technology in various enterprises and institutions in recent years, more and more business process execution data have been recorded as event logs, greatly facilitating data-driven business process prediction research. Researchers have attempted to apply machine learning methods such as Hidden Markov Models [5], Support Vector Machines [6], and Cluster Analysis [7] to the task of business process prediction. For example, automobile insurance claims contain parallel execution of tasks, mapping them to an extended space Markov chain can make predictions. Support Vector Machines classify input object features relevant for the classification on the basis of its position in hyperspace and then check possible anomalies in its behavior. Cluster Analysis is applied to previous traces of the patient, and then a classifier is built for each cluster to discriminate whether or not a cancer patient. Machine learning methods can overcome the shortcomings of earlier formal modeling methods that could not adequately reflect the complex business process execution logic in real business systems and achieve better prediction results. However, the application of machine learning methods for business process prediction requires tedious feature engineering, and different business processes are often variable. It leads to no fixed pattern to follow in extracting business process features. In addition, traditional machine learning models have a more limited parameter space, making it difficult to effectively model large volumes of event logs. To address this problem, researchers start applying deep learning methods to business process prediction tasks in recent years, and the prediction results have further improved. As event logs are essentially sequential data, the Recurrent Neural Network (RNN) [8], which is used to process sequential data, is an early deep-learning model for business process prediction. RNN can consider the previous output and apply it to the current output. In addition, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), solving the problem of gradient disappearance and explosion in RNN, can better capture longer information dependencies in sequential data and have shown to perform well in business process prediction [9–11].

Transformer is a new type of model that has emerged in deep learning in recent years. It introduces multi-head attention, positional encoding, residual connection, layer normalization, and other mechanisms based on the self-attention model [12]. It solves the problems of recurrent neural networks that are difficult to run in parallel and convolutional neural networks with a restricted field of perception, and it has been used in natural language processing [13], recommender systems [14], computer vision [15], and many other tasks to achieve state-of-the-art results. The success of the Transformer in several fields has also attracted the attention of researchers in the field of business process prediction. Bukhsh et al. [16] were the first to apply the Transformer to business process prediction, and the basic idea is to compare a business process instance to a sentence in natural language processing, using it as input to the Transformer and the subsequent execution state of the instance as the model output. Overall, research related to the Transformer in business process prediction is relatively rare, and most of the existing work is limited to the direct application of the standard Transformer model architecture. We can argue that the potential of Transformer in business process prediction has not been well explored.

To this end, this paper will focus on Transformer-based business process prediction and aims to improve the performance of various business process prediction tasks. Firstly, the business process instance data recorded in the event log not only consists of sequences of activities, but each activity also has different attribute information when it is executed, which is a feature important for accurate business process prediction [17,18]. Transformer-based business process prediction models do not yet deal well with the relationships

between activities and their various types of attribute information. Secondly, realistic business processes often have phenomena such as conceptual drift during execution, and the backend business process models supporting business systems are unstable, making business process instances reflect different process model characteristics [19]. Concept drift can lead to different business process instance distributions. Various machine learning methods, including deep learning, are often based on the assumption of independent and homogeneous data distribution, not taking into account the effect of concept drift on the final result, so concept drift can reduce the accuracy of business process prediction [20]. The existing data-driven business process models do not match the processes where conceptual drift occurs, and outdated models lead to inaccurate predictions of future results. Furthermore, Transformer requires more time and computational resources for model training for its higher time complexity, which to some extent limits the effective application of Transformer-based business process prediction models, although it has a greater advantage over various RNN models in terms of accuracy for various tasks.

Based on the above considerations, a new business process prediction model, the Hierarchical Process Transformer (HiP-Transformer for short), is proposed. The basic idea of this model is to design a hierarchical Transformer model structure to better solve the problems mentioned above in predictive business process monitoring. Specifically, the HiP-Transformer model consists of three main layers. The first (bottom) layer is the event modeling layer, which uses the multi-head self-attention mechanism in the Transformer to model the activity of each event and its attribute information in a process instance. The second (middle) layer is the event sequence fragment modeling layer, of which the goal is to explore intra-fragment correlations and inter-fragment dependencies. The whole event sequence corresponding to the process instance is segmented based on the concept drift detection method to obtain a relatively stable event subsequence of process features. The process features within each event subsequence is then captured using cross-attention. The third (top) layer is the process instance modeling layer, where the goal is to further model the entire event sequence of a process instance using Transformer on top of the event subsequence modeling to obtain the representation of the process instance. In addition, for two representative business process prediction tasks, namely next activity prediction and remaining time prediction, the corresponding decoding modules are designed in the HiP-Transformer model to predict the subsequent states of the business process instances, respectively.

In this paper, experiments are conducted on several real business process datasets publicly available at the 4TU Center for Research Data. The results show that the proposed HiP-Transformer model achieves further improvement over traditional Transformer-based models for the two business process prediction tasks. We refer to the benchmark [2] of the deep learning model to validate the validity of the model. Our method is relatively more accurate than most deep learning-based business process prediction models, showing a significant advantage. The main contributions of this paper can be summarized as follows:

- We propose a novel model called HiP-Transformer, which infers accurate business process representations with three perspectives on the associated features. The internal associations between activities, events, and sub-sequences in the business process make up the comprehensive representation.
- We solve the problem of conceptual drift in long business processes. We capture the dependencies of preceding and following subsequences with conceptual drift behavior, which has a higher prediction accuracy and lower time complexity than traditional segmentation strategies.
- Comprehensive experiments on seven public datasets demonstrate the effectiveness of our HiP-Transformer model. Compared with deep learning methods, including LSTM [10], GRU [11], CNN [21], DNC [22], and Process Transformer [16], the next activity prediction accuracy improves by 6.32% compared to the average accuracy of the above models, and the average absolute error of remaining time prediction reduces by 21% compared to the average accuracy of the above models.

The rest of this paper is organized as follows: Section 2 reviews the related work on business process prediction research. Section 3 describes the problem to be solved formally and proposes the HiP-Transformer method. Section 4 shows the experimental results and analysis. Section 5 summarizes the work of this paper and describes future work.

2. Literature Review

Predictive Process Monitoring is different from traditional process mining tasks such as process discovery and consistency testing. It focuses not only on the existing execution of the business process but also on the future execution trends of the business process, contributing to the early determination of potential risks and bottlenecks in the business process and providing a basis for immediate optimization of the business process. Predictive business monitoring research can be broadly divided into formal modeling-based and data-driven approaches. The data-driven approach can better utilize the complex business process execution information contained in the massive event logs, overcoming the lack of flexibility of various formal approaches and gradually becoming one of the main streams of predictive business monitoring research. In recent years, deep learning techniques have shown increasing success in various real-world tasks such as natural language processing, recommender systems, and computer vision. The predictive business monitoring tasks are closer to the common applications of deep learning, so many researchers have started conducting relevant research using deep learning techniques and achieve better results.

The earliest research on predictive business monitoring based on deep learning focuses on applying RNNs. Evermann et al. [8] used RNNs to construct a basic framework for business process prediction, consisting of a two-layer unidirectional LSTM. Tax et al. [10] used the idea of parameter sharing to design a multi-task business process prediction method based on a multi-layer LSTM. Hinkka et al. [11] used another simplified variant of RNN, GRU, to design a business process classification method. To further improve the accuracy of RNN-based predictive business monitoring, researchers have proposed a variety of proven methods. For example, Camargo et al. [23] conducted feature engineering for subtypes, continuous-type activity attributes, and time series, respectively, and used the extracted features as input to the LSTM; Ni et al. [9] introduced attention, multi-layer stacking, and bidirectional connection and other mechanisms to effectively improve the accuracy of remaining time prediction; Liu et al. [24] designed a transfer learning based model training framework targeting the length characteristics of business process instances.

In addition to RNNs, researchers use other types of deep learning models for predictive business monitoring. For example, Pasquadibisceglie et al. [21] designed a method for predicting the next activity in a business process using convolutional neural networks with the help of modeling ideas from the image domain, so it can capture the spatial structure of the business process logs and enable parallelization training; Mehdiyev et al. [25] proposed a multi-stage deep learning approach, which extracts features using feature hashing and deep stacked autoencoders and then uses multi-layer feed-forward neural networks to construct a next activity prediction model; Ni et al. [26] used deep autoencoders to extract event sequence features and variational system structure features to achieve a more comprehensive feature representation, followed by residual time prediction using a multi-layer perceptron.

There are relatively few studies on predictive business monitoring based on the Transformer. Philipp et al. [27] used the self-attention mechanism in the Transformer to construct a business process next activity prediction model and achieved better prediction results than the traditional LSTM. Bukhsh et al. [16] used the encoder module of the Transformer to encode the activity sequence of a business process instance and predict the next activity or the remaining execution time of the business process instance based on the resulting vector representation. However, the above studies simply apply some of the modules in the Transformer directly (e.g., self-attention, encoder, etc.) and do not address the real-world challenges of business process prediction, leaving shortcomings such as high training overhead and difficulty in effectively modeling various types of attributes.

Therefore, there is still much room for improvement in the research of predictive business monitoring based on the Transformer.

3. Preliminaries and Methods

3.1. Definitions

3.1.1. Event

For a business process, let A denote the set of all activities; let T denote the set of occurrence times corresponding to all activities; and let M denote the set of attributes corresponding to all activities. An event can be denoted as $e = (a, t, m^{(1)}, \dots, m^{(k)})$, where $a \in A$ denotes the activity executed in the event; $t \in T$ denotes the time at which the event occurred; and $m^{(j)} \in M$ ($1 \leq j \leq k$) denotes the j -th attribute of the event, and usually there are multiple attributes of an event such as executor, execution cost, etc. Event attributes can be further classified into discrete and continuous attributes according to the range of values they take; for example, the executor attribute of an event is discrete, and the execution cost attribute is continuous.

3.1.2. Business Process Case and Prefix

A business process case is a sequence of events occurring in time order, which can be denoted as $\sigma = \langle e_1, e_2, \dots, e_n \rangle$, where e_i ($1 \leq i \leq n$) denotes the i -th executed event in the instance, and n denotes the number of events contained in the instance, satisfying $e_i.t < e_{j+1}.t$ for $1 \leq i < j \leq n$. A complete business process instance contains a start event e^{start} and an end event e^{end} , i.e., $e_1 = e^{start}$, $e_n = e^{end}$.

A business process prefix is a case that has not yet ended and is usually a subsequence of a full business process instance. Given a business process case $\sigma = \langle e_1, e_2, \dots, e_n \rangle$, its prefix can be given as $prefix(\sigma) = \langle e_1, e_2, \dots, e_{n'} \rangle$, where $1 \leq n' \leq n$.

3.1.3. Business Process Event Log

A business process event log is a set of numerous business process cases, denoted as $L = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$, where m denotes the number of business process cases contained in the event log. Business process event logs are usually collected during the running of a business system and reflect the actual running behavioral characteristics of the business system.

3.2. Task Description

In the predictive business monitoring task, given an event log $L = \{\sigma_1, \sigma_2, \dots, \sigma_m\}$, the goal is to build a predictive model to predict the likely subsequent execution state of a business process prefix (i.e., a business process case that has not yet completed execution). Specifically, given a business process case $\sigma = \langle e_1, e_2, \dots, e_n \rangle$ and its one prefix $prefix(\sigma) = \langle e_1, e_2, \dots, e_{n'} \rangle$, the prediction goals for the next activity prediction task and the remaining execution time prediction task are as follows, respectively,

$$y^{(act)}(prefix(\sigma)) = e_{n'+1}, \quad (1)$$

$$y^{(time)}(prefix(\sigma)) = e_n.time - e_{n'}.time, \quad (2)$$

The predictive business monitoring task is carried out in two main segments: firstly, the business process prefix set $prefix(L)$ is generated based on the given event log $prefix(L)$ in the following Equation (3).

$$prefix(L) = \{prefix(\sigma) | \sigma \in L, len(prefix(\sigma)) > n_0\}, \quad (3)$$

where $len(\cdot)$ denotes the number of events contained in an input business process case or prefix, and n_0 is a pre-assigned length threshold. In short, for each business process case in the event log L , its prefix set is generated, and the prefix set of each case is combined. Since prefixes containing a small number of events have less information and are of less value for subsequent model training, a threshold n_0 is specified to filter the shorter prefixes.

Then, according to the prediction target of the predictive business monitoring task (such as the next activity or the remaining execution time), the prediction target value is calculated for each business process prefix in $prefix(L)$ according to Equation (1) or Equation (2), respectively, and the training dataset is then built as follows:

$$D(L) = \left(\left(prefix(\sigma), y^{(act/time)}(prefix(\sigma)) \right) \mid prefix(\sigma) \in prefix(L) \right) \quad (4)$$

3.3. Transformer

Transformer was originally a Seq2Seq model for machine translation tasks, which uses a self-attention mechanism to compute the correlation between words within an input sentence and extract the most important parts of the input sentence for the prediction target [12]. In recent years, Transformer has been widely used for a variety of tasks beyond machine translation and has achieved good results, especially in tasks related to time series analysis.

Typically, the input to the Transformer is a sequence $X = \langle x_1, x_2, \dots, x_n \rangle$, where x_i ($1 \leq i \leq n$) denotes the i -th element in the sequence, and n denotes the length of the sequence. For example, in a natural language processing task such as machine translation, X is a sentence composed of a series of words; in a time series analysis task, X is a time series composed of a series of elements. In deep neural network models such as Transformer, each input element is represented by using a numerical vector. Formally, the input sequence can be represented as $X = (x_1, x_2, \dots, x_n)^T \in \mathbb{R}^{n \times d}$, where x_i is the vector representation of element x_i , and d is the vector dimension. The Transformer's self-attention mechanism is shown in the following Equations (5) and (6).

$$SelfAttn(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d^{(K)}}}\right)V, \quad (5)$$

$$Q = XW^{(Q)}, K = XW^{(K)}, V = XW^{(V)}, \quad (6)$$

where $Q \in \mathbb{R}^{n \times d^{(Q)}}$, $K \in \mathbb{R}^{n \times d^{(K)}}$, and $V \in \mathbb{R}^{n \times d^{(V)}}$ are the original input after multiplying with matrices $W^{(Q)} \in \mathbb{R}^{d \times d^{(Q)}}$, $W^{(K)} \in \mathbb{R}^{d \times d^{(K)}}$ and $W^{(V)} \in \mathbb{R}^{d \times d^{(V)}}$ and are also known as the query matrix, the key matrix, and the value matrix, respectively. $\sqrt{d^{(K)}}$ acts as a scaling factor.

Due to the limited capability of modeling a single self-attention, multiple self-attention is usually calculated independently in Transformer, and the results are concatenated, which in turn forms a multi-head self-attention, calculated as shown in the following Equations (7) and (8).

$$MultiHead(X) = (head_1 \oplus head_2 \oplus \dots \oplus head_k)W^{(O)} \quad (7)$$

$$head_i = SelfAttn(XW_i^{(Q)}, XW_i^{(K)}, XW_i^{(V)}) \quad (8)$$

After that, the Transformer integrates the multi-head self-attention encoding with the original encoding of the input sequence using residual concatenation. Then, we obtain the final encoding of the input sequence X' using Layer Normalization and a position-wise Feed-Forward Network, calculated as shown in the following Equations (9)–(11).

$$X' = FFN(H') = ReLU(H'W^{(1)} + b^{(1)})W^{(2)} + b^{(2)} \quad (9)$$

$$H' = LayerNorm(FFN(H) + H) \quad (10)$$

$$H = LayerNorm(MultiHead(X) + X) \quad (11)$$

The X' in Equation (9) can be used as a feature representation of the input sequence, which can subsequently be fed into the task-related network structure to complete the prediction target. The overall calculation cost of the Transformer model consists mostly

of computing the attention matrix QK^T in Equation (5), and its time complexity is $O(n^2)$, which makes the Transformer model less efficient when dealing with long sequences.

3.4. Hierarchical ProcessTransformer

In this paper, a Hierarchical Process Transformer (HiP-Transformer) model is proposed for predictive business process monitoring tasks. In contrast to existing approaches that apply the Transformer model directly to business process instance data, this approach decomposes the predictive business process monitoring task at three levels and designs a specific Transformer model at each level. Firstly, to better model the activity, time, and various attribute information contained in events, the Transformer model at the first (bottom) level is designed to obtain a fine-grained event encoding representation using an event-level self-attention mechanism. The second (middle) layer of the Transformer model is designed to encode the event subsequence. The third (top) layer of the Transformer model is designed to globally encode the whole business process instance based on the multiple event subsequences. The three levels of the Transformer model can effectively fuse multiple granularities of information such as event attributes, event sequences, and complete cases to achieve a combined encoded representation of the business process case. It finally achieves accurate prediction of the next activity, remaining time, and other key states of the business process case by designing a task-related prediction network.

The framework of the HiP-Transformer model is shown in Figure 1. The three layers of the HiP-Transformer model are described in detail at what follows.

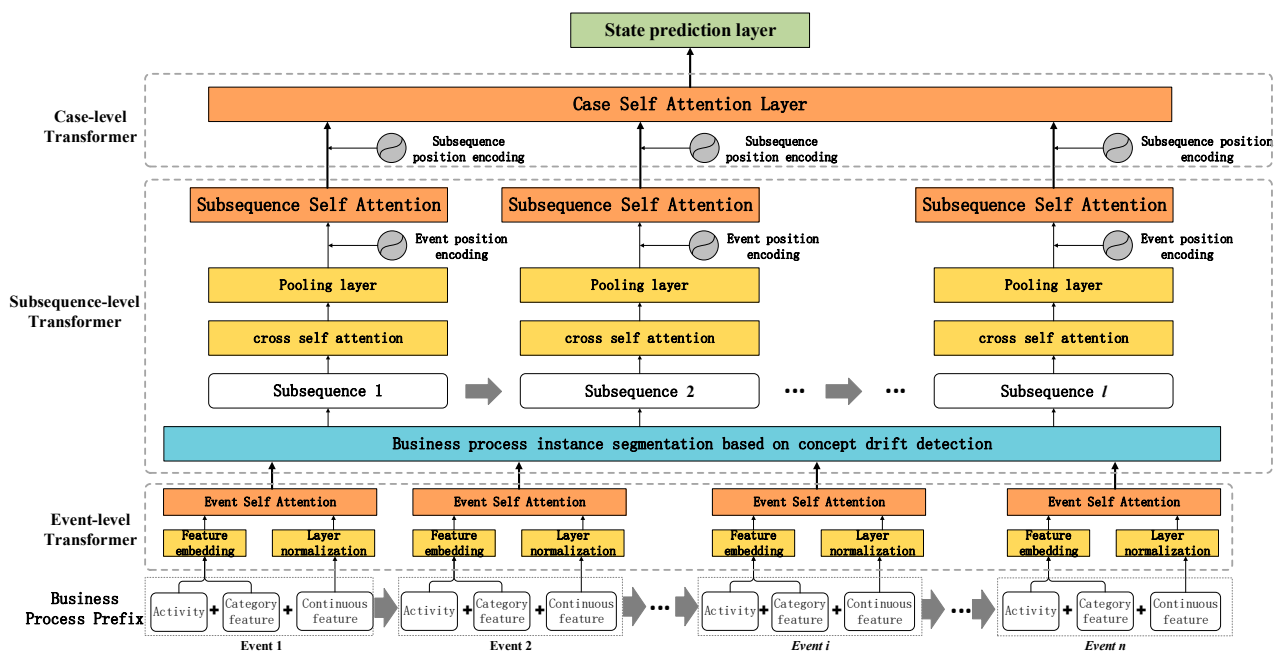


Figure 1. HiP-Transformer Model Framework.

3.4.1. First Layer: Event-Level Transformer

Events are the fundamental building blocks of business process cases, and valid event coding is one of the keys to the task of predictive business process monitoring. According to the definition of an event in Section 3.1, an event is a composition of multiple attributes, so the encoding of events requires effective integration of event-related attribute information. To make the HiP-Transformer model proposed in this paper more adaptable, all attributes are divided into discrete and continuous attributes. Different encoding methods are designed for these two types of attributes.

In general, assuming that an event contains a discrete and a continuous attribute, i.e., $e = (a, t, m_1, m_2) \in A \times T \times M_1 \times M_2$, where m_1 and m_2 denote discrete and continuous attributes, respectively. The subsequent way of encoding the attributes can be easily

extended to zero or more attributes. The discrete attribute m_1 is first represented as a one-hot coding vector $\tilde{m}_1 \in \{0, 1\}^{|M_1|}$ according to its value domain M_1 . A learnable coding matrix is constructed for the attribute $M_1 \in \mathbb{R}^{|M_1| \times d}$, and the encoding vector for this attribute can be obtained as $m_1 = \tilde{m}_1 M_1$. For the continuous attribute m_2 , attribute learnable encoding vector $\tilde{m}_{type(m_2)}$ is constructed, and then the specific values of the attribute are normalized, which in turn obtains its encoding vector as $m_2 = m_2 \odot \tilde{m}_{type(m_2)}$, where \odot denotes the operation of the product of a scalar and a vector.

To easily understand, further examples of encoding discrete and continuous attributes are given. Suppose that the discrete attribute of an event is “activity executor is Bob” and the continuous attribute is “execution cost is 0.5 (after normalization)”, i.e., $m_1 = \text{Bob}$, $m_2 = 0.5$. The activity executor takes the value space as $M_1 = \{\text{Bob}, \text{Alice}, \text{Tom}\}$. Let the encoding dimension of each attribute be 4. The encoding process for this discrete attribute is as follows.

$$\begin{aligned}\tilde{m}_1 &= (1, 0, 0) \\ M_1 &= \begin{pmatrix} e_{1,1}, e_{1,2}, e_{1,3}, e_{1,4} \\ e_{2,1}, e_{2,2}, e_{2,3}, e_{2,4} \\ e_{3,1}, e_{3,2}, e_{3,3}, e_{3,4} \end{pmatrix} \\ m_1 &= \tilde{m}_1 M_1 = (e_{1,1}, e_{1,2}, e_{1,3}, e_{1,4})\end{aligned}$$

The encoding process for this continuous attribute is as follows.

$$\begin{aligned}\tilde{m}_{cost} &= (h_1, h_2, h_3, h_4) \\ m_2 &= m_2 \odot \tilde{m}_{cost} = (0.5 \times h_1, 0.5 \times h_2, 0.5 \times h_3, 0.5 \times h_4)\end{aligned}$$

The activity a performed in the event is, in essence, a discrete attribute of the event, and the activity vector \mathbf{a} can be obtained by the encoding method of the discrete attribute described above. The encoding vector \mathbf{e} of the entire event e is obtained in the event-level Transformer using the self-attention mechanism. Considering that the activity is the main part of an event, the activity vector is first used as the query vector in the self-attention mechanism to calculate the weight vector of each attribute concerning the activity, and the vectors of the attributes are fused in a weighted manner. Let $\mathbf{X} = (m_1; m_2)$, the process is calculated as follows:

$$\mathbf{m} = \text{softmax}\left(\frac{\mathbf{q}\mathbf{K}^T}{\sqrt{d^{(K)}}}\right)\mathbf{V} \quad (12)$$

$$\mathbf{q} = \mathbf{a}\mathbf{W}^{(Q)}, \mathbf{K} = \mathbf{X}\mathbf{W}^{(K)}, \mathbf{V} = \mathbf{X}\mathbf{W}^{(V)} \quad (13)$$

where $\mathbf{W}^{(Q)}$, $\mathbf{W}^{(K)}$, and $\mathbf{W}^{(V)}$ are the linear transition matrices shown in Equation (6). The initial event vector is formed by concatenating the activity vector with the fused attribute vector, i.e., $\tilde{\mathbf{e}} = \mathbf{a} \oplus \mathbf{m}$. After that, the final event vector is obtained using techniques such as multi-head attention, layer normalization, and location feed-forward networks in the Transformer model. Specifically, multiple initial event vectors are obtained using Equations (7) and (8), which are concatenated into the layer normalization and position feedforward networks shown in Equations (9)–(11). We obtain the final event encoding vector \mathbf{e} as the output of the first layer event-level Transformer.

3.4.2. Second Layer: Subsequence-Level Transformer

As described in Section 3.3, the time complexity of the Transformer model is the square level of the length of the input sequence. To improve the efficiency of applying the Transformer model in predictive business process monitoring tasks, the HiP-Transformer model splits the complete sequence of events of a business process case to obtain a subsequence of events. Traditional sequence splitting methods allow for uniform splitting using a fixed-length window. However, this approach ignores the dependency characteristics of the internal events. It leads that subsequences lack semantic integrity and distributional consistency. It is especially true for business process cases, where the business process

is unstable and often evolves. In this paper, we use the concept drift feature of business process cases as a basis for sequence splitting so that the resulting subsequence has a relatively stable internal distribution, thus helping to improve the performance of Transformer modeling. The proposed method for detecting conceptual drift in business process cases will be presented first, followed by the Transformer model for event subsequences.

Concept Drift Detection

The first layer Transformer model obtains the encoding vector of each event that provides a basis for concept drift detection. Therefore, in contrast to the direct application of existing concept drift detection methods, this paper proposes a concept drift detection method based on the encoding representation. The basic idea of this method is splitting the time series of business process instances to obtain different subsequences. Then, we conduct significance tests on the data distribution of each subsequence to determine whether concept drift has occurred between the subsequences. Through this method, many candidate concept drift points may be obtained and may thus apply the Optics algorithm to cluster on the candidate concept drift points. The most representative point in the cluster of the candidate concept drift points will be retained as the segmentation point, and the sequence segmentation is completed.

In the candidate concept drift point search stage, supposing the encoding representations of two subsequences are $\mathbf{W}_1 = (e_{1,1}, e_{1,2}, \dots, e_{1,n_1})$ and $\mathbf{W}_2 = (e_{2,1}, e_{2,2}, \dots, e_{2,n_2})$. Let μ_1 and μ_2 be the mean of the encoding vectors of the subsequence \mathbf{W}_1 and \mathbf{W}_2 , and let λ_1 and λ_2 be the trace of the covariance matrix of the subsequence \mathbf{W}_1 and \mathbf{W}_2 . From the perspective of the differences test, the conceptual drift detection task is to test whether the subsequence \mathbf{W}_2 has changed significantly in distribution compared to the subsequence \mathbf{W}_1 . The following statistic testing is first constructed to quantitatively assess the difference between the two subsequences.

$$s = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}}} \quad (14)$$

After that, let the original hypothesis be that the difference between the two subsequences is not significant, i.e., no conceptual drift has occurred. Given a confidence threshold δ , the acceptance region of the original hypothesis can be expressed formally as follows:

$$\Pr\left(\frac{|\mu_2 - \mu_1|}{\sqrt{\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}}} \geq \varepsilon\right) \leq \delta \quad (15)$$

Equation (15) can be equivalently written as:

$$\Pr\left(|\mu_2 - \mu_1| \geq \varepsilon \sqrt{\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}}\right) \leq \delta \quad (16)$$

According to Hoeffding's Inequality [28] and its corollaries [29], that the following holds:

$$\Pr(|\mu_2 - \mu_1| - |E(\mu_2) - E(\mu_1)| \geq \varepsilon) \leq \exp\left(\frac{-2\varepsilon^2}{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)|\mathbf{b} - \mathbf{a}|^2}\right) \quad (17)$$

where $E(\mu_2)$ and $E(\mu_1)$ denote the expectation of the subsequence encoding representation, respectively. \mathbf{b} and \mathbf{a} denote the upper and lower deterministic bounds of the event encoding vector, respectively. These values are usually not directly computable. However, considering that the event encoding is learned through the previously described event-level Transformer by random initialization, it can be assumed that: $E(\mu_2) = E(\mu_1) = 0$. $|\mathbf{b} - \mathbf{a}|^2$ can be written as a fixed constant Δ^2 , so we can obtain Equation (18).

$$\Pr(|\mu_2 - \mu_1| \geq \varepsilon) \leq \exp\left(\frac{-2\varepsilon^2}{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\Delta^2}\right) \quad (18)$$

Substituting the denominator term in Equation (15) into Equation (17), we obtain Equation (19).

$$\exp\left(\frac{-2\varepsilon^2\left(\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}\right)}{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\Delta^2}\right) \leq \delta \quad (19)$$

Through solving Equation (19), we obtain Equation (20).

$$\varepsilon \geq \sqrt{\frac{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\Delta^2 \ln \frac{1}{\delta}}{2\left(\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}\right)}} \quad (20)$$

According to Equation (20), given a confidence threshold δ , if the mean values of the encoding vectors of the two subsequences satisfy Equations (21) and (22).

$$s = \frac{|\mu_2 - \mu_1|}{\sqrt{\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}}} \geq \sqrt{\frac{\left(\frac{1}{n_1} + \frac{1}{n_2}\right)\Delta^2 \ln \frac{1}{\delta}}{2\left(\frac{\lambda_1}{n_1} + \frac{\lambda_2}{n_2}\right)}} \quad (21)$$

$$|\mu_2 - \mu_1| \geq \Delta \sqrt{\frac{1}{2} \left(\frac{1}{n_1} + \frac{1}{n_2}\right) \ln \frac{1}{\delta}} \quad (22)$$

Then, it can be assumed that a conceptual drift has occurred between these two subsequences.

Based on the above theoretical analysis, the specific implementation of the candidate concept drift point search is shown in Algorithm 1. The algorithm uses a recursive approach to search for candidate concept drift points. Firstly, the input business process case is continuously segmented (lines 4~5). Secondly, the statistics of each subsequence are calculated (lines 6~7) and then judged whether there is a concept drift between the adjacent two subsequences according to Equation (22). If there is a concept drift, the current segment position is marked as a candidate concept drift point (lines 8~9). Thirdly, the above candidate concept drift point search process is iteratively executed for the current two subsequences (lines 10~13) until all subsequences no longer meet the minimum threshold. Finally, all candidate concept drift points are combined and returned (line 14).

Algorithm 1 needs several parameters such as the smallest sequence length threshold and the lowest confidence threshold. Different parameter settings will greatly impact the search results. Therefore, this paper does not directly take the results of Algorithm 1 as the final concept drift points but adjusts different threshold parameters by running Algorithm 1 to obtain a series of candidate concept drift points, then clustering them to obtain the concept drift points for sequence segmentation. In this paper, the density-based OPTICS clustering algorithm is used in particular. Its main advantage is that it can adaptively discover successive high-density regions in the sample space, treating each high-density region as a cluster and overcoming the shortcomings of clustering algorithms such as K-Means that can only discover spherical clusters. Using the OPTICS clustering algorithm, we can obtain a high-density set of all candidate concept drift points, after which a representative concept drift point from each set is selected for sequence segmentation. The OPTICS-based clustering algorithm for candidate concept drift points can be found in the reference [30] and will not be described further in this paper.

Algorithm 1 Candidate Drift Point Search

Input: Business process case encoding $\sigma = (\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_n)$, subsequence length threshold l , event vector module Δ , confidence threshold δ

Output: Candidate Drift Point Set P

```

1:  $start \leftarrow 1, end \leftarrow n$  //Initialization
2:  $P \leftarrow \phi$  //Initialization
3: if  $end - start > 2l$  do //Concept drift detection when the sequence length is greater than the threshold value
4:    $tmp \leftarrow \lfloor (start + end)/2 \rfloor$  //Calculate the sequence center position
5:    $W_1 \leftarrow [start, tmp], W_2 \leftarrow [tmp + 1, end]$  //Split left and right subsequences
6:    $n_1 \leftarrow tmp - start + 1, n_2 \leftarrow end - tmp$  //Calculate the length of the left and right subsequences
7:    $\mu_1 \leftarrow \sum_{t=start}^{tmp} \mathbf{e}_t, \mu_2 \leftarrow \sum_{t=tmp+1}^{end} \mathbf{e}_t$  //Calculate the mean of the left and right subsequence encoding vectors
8:   if  $|\mu_2 - \mu_1| \geq \Delta \sqrt{\frac{1}{2} \left( \frac{1}{n_1} + \frac{1}{n_2} \right) \ln \frac{1}{\delta}}$  do //Judge whether there is conceptual drift in the left and right subsequences according to Equation (18)
9:      $P \leftarrow tmp$ 
10:  if  $n_1 \geq 2l$  do //Candidate concept drift point search for left subsequence iterations
11:     $P \leftarrow P \cup \text{ConceptDriftSearch}(\sigma_{start:tmp}, l, \Delta, \delta)$ 
12:  if  $n_2 \geq 2l$  do //Candidate concept drift point search for right subsequence iterations
13:     $P \leftarrow P \cup \text{ConceptDriftSearch}(\sigma_{tmp+1:end}, l, \Delta, \delta)$ 
14: return  $P$ 

```

Transformer for Subsequence

The second layer Transformer model takes the event sub-sequence obtained after concept drift detection as input, which can be formally represented as $\mathbf{M} = \langle \mathbf{e}_s, \mathbf{e}_{s+1}, \dots, \mathbf{e}_{s+l-1} \rangle \in \mathbb{R}^{l \times d}$ after encoding by the event-level Transformer model, where l denotes the length of the sub-sequence, and d denotes the event encoding vector dimension. In addition, the activity, as an important component of the event, has a more significant impact on subsequent tasks such as the next activity prediction, remaining execution time prediction, etc. Therefore, the activity encoding of the event will be explicitly used as another input in the second layer Transformer model, which can be expressed formally as $\mathbf{A} = \langle \mathbf{a}_s, \mathbf{a}_{s+1}, \dots, \mathbf{a}_{s+l-1} \rangle \in \mathbb{R}^{l \times d}$. To fully exploit the intrinsic relationship between event and activity, this paper proposes a cross-self-attention mechanism. The basic idea is to use event and activity encoding as queries, respectively, calculate the weights of each other's encoding and then fuse them in a weighted manner. First, to model the sequence relationship between the events in the sub-sequence, position encoding is added to the event and activity encodings, specifically using the traditional Sinusoidal absolute position encoding $\mathbf{P} \in \mathbb{R}^{l \times d}$ to obtain $\mathbf{A}' = \mathbf{A} + \mathbf{P}$ and $\mathbf{M}' = \mathbf{M} + \mathbf{P}$, where each element in \mathbf{P} is set to a fixed value [12] as follows.

$$\mathbf{P}_{t,i} = \begin{cases} \sin\left(10000^{-\frac{i}{d} \cdot t}\right) & \text{if } i \text{ is even number} \\ \cos\left(10000^{-\frac{i}{d} \cdot t}\right) & \text{if } i \text{ is odd number} \end{cases} \quad (23)$$

After that, the event encoding matrix and the activity encoding matrix are linearly converted to obtain:

$$\mathbf{Q}^{(A)} = \mathbf{A}'\mathbf{W}^{(Q-A)}, \mathbf{K}^{(M)} = \mathbf{M}'\mathbf{W}^{(K-M)}, \mathbf{V}^{(M)} = \mathbf{M}'\mathbf{W}^{(V-M)} \quad (24)$$

$$\mathbf{Q}^{(M)} = \mathbf{M}'\mathbf{W}^{(Q-M)}, \mathbf{K}^{(A)} = \mathbf{A}'\mathbf{W}^{(K-A)}, \mathbf{V}^{(A)} = \mathbf{A}'\mathbf{W}^{(V-A)} \quad (25)$$

Each of the \mathbf{W} matrices in the above equation represents a linear transformation weight matrix, taking $\mathbf{W}^{(Q-A)}$ as an example, with Q denoting the query transformation used for this matrix and A denoting the transformation applied to the active encoding. Based on the

self-attention mechanism shown in Equation (5), the following calculations are performed using each of the transformation matrices in the above equations, respectively.

$$S^{(A-M)} = \text{softmax}\left(\frac{Q^{(A)}K^{(M)T}}{\sqrt{d^{(K)}}}\right)V^{(M)} \quad (26)$$

$$S^{(M-A)} = \text{softmax}\left(\frac{Q^{(M)}K^{(A)T}}{\sqrt{d^{(K)}}}\right)V^{(A)} \quad (27)$$

$$S^{(A-A)} = \text{softmax}\left(\frac{Q^{(A)}K^{(A)T}}{\sqrt{d^{(K)}}}\right)V^{(A)} \quad (28)$$

$$S^{(M-M)} = \text{softmax}\left(\frac{Q^{(M)}K^{(M)T}}{\sqrt{d^{(K)}}}\right)V^{(M)} \quad (29)$$

Each matrix S in the above equations represents the result of fusing the event encoding with the activity encoding, with each row corresponding to the fusion encoding of an event in the subsequence. Taking $S^{(A-M)}$ as an example, A indicates that the fusion result uses the activity encoding as the query, and M indicates that the event encoding is fused. $S^{(A-M)}$, $S^{(M-A)}$ use different encoding as query and key value, so it can be considered as the fusion result of cross-self-attention, where $S^{(A-A)}$, $S^{(M-M)}$ are the fusion results of traditional self-attention. Next, these four fusion results will be further fused, specifically using a maximum pooling as follows:

$$\tilde{S} = \text{max_pool}\left(S^{(A-M)}, S^{(M-A)}, S^{(A-A)}, S^{(M-M)}\right) \quad (30)$$

where max_pool maximizes each row of the specified parameter matrix separately, such that $S^{(sub)} \in \mathbb{R}^{l \times d'}$, with d' denotes the dimension of the fused encoding vector. Finally, the final event subsequence coding matrix is obtained using techniques such as multi-head attention, layer normalization, and position feed-forward networks in the Transformer model, as shown in Equation (7) to Equation (11).

3.4.3. Third Layer: Case-Level Transformer

The third layer Transformer model takes the encoding matrix of each event subsequence obtained in the second layer as input, which can be denoted formally as $X^{(all)} = (S_1, S_2, \dots, S_k)$, where $S_i \in \mathbb{R}^{l_i \times d}$ denotes the encoding matrix of the i th subsequence; l_i is the length of the subsequence; and d is the dimension of each event encoding vector. Since the length of each subsequence is different, the encoding matrix of each subsequence is processed using maximum pooling to convert it into a vector format. Suppose $S_i = (s_{i,1}, s_{i,2}, \dots, s_{i,k})$, the maximum pooling vector can be denoted as follows.

$$s'_i = \text{max_pool}(s_{i,1}, s_{i,2}, \dots, s_{i,k}) \quad (31)$$

Then, position encoding is introduced for each event subsequence to model the positions of different event subsequences throughout the business process cases. In contrast to the traditional Sinusoidal absolute position encoding used in the second layer of the Transformer model, this layer uses learnable position encoding, i.e., learning a position vector p_i for each position $i \in [1, k]$. Compared to the Sinusoidal absolute position encoding with fixed values, learnable position encoding is more flexible in the third layer. The main reasons are that the number of event subsequences contained in the business process cases is relatively small. Because the value of k is small, so learning the location vector does not bring a high additional computational cost; Sinusoidal absolute location coding has a relatively small location difference in short sequences, and learnable position encoding can

better distinguish different location information according to the actual characteristics of the data.

The pooled event subsequence vector and the position vector can be combined to obtain $\Gamma' = (s'_1 + p_1, s'_2 + p_2, \dots, s'_k + p_k)^T$, which is used as the input of the Transformer model, and Γ is calculated according to Equation (5) to Equation (11) as the final encoded representation of the business process case.

3.4.4. State Prediction Layer

The purpose of the state prediction layer is to predict the possible subsequent states of the business process case, including the next activity to be executed, the remaining execution time, etc., based on the obtained business process case encoding by the HiP-Transformer model. Given a business process case $\sigma = \langle e_1, e_2, \dots, e_n \rangle$, suppose that $\sigma = \langle \sigma'_1, \sigma'_2, \dots, \sigma'_k \rangle$ is obtained after concept drift detection segmentation, where σ'_i ($1 \leq i \leq k$) denotes the i -th event subsequence. As mentioned above, the encoding of each event, event subsequence, and the whole business process case can be obtained after the three-layer Transformer model. In general, the most recently executed information in the business process case is more important for the subsequent state prediction, so the following three types of information are selected as inputs in the state prediction layer.

- The encoding vector of the last event e_n is selected in the first layer Transformer, noted as e_n .
- The encoding matrix of the last event subsequence σ'_k is selected in the second layer Transformer and is denoted as $S_k^{(sub)}$. Since this encoding matrix is related to the subsequence length, the last r_1 column is selected as the input and is denoted as $S_{k, (end-r_1+1:end)}^{(sub)}$.
- The encoding matrix of the business process case σ is selected in the third layer Transformer and is denoted as Γ . Since this encoding matrix is related to the length of the sequence, the last r_2 columns are selected as input and are denoted as $\Gamma_{(end-r_2+1:end)}$.

The state prediction layer inputs the above three types of information into a Multi-layer Perceptron (MLP) network to obtain the prediction results, and the calculation process is as follows.

$$y = MLP\left(e_n \oplus \text{flatten}\left(S_{k, (end-r_1+1:end)}^{(sub)}\right) \oplus \text{flatten}\left(\Gamma_{(end-r_2+1:end)}\right)\right) \quad (32)$$

where \oplus denotes the vector join operation, and $\text{flatten}()$ denotes flattening the matrix into vectors by rows. The MLP used in this paper consists of a three-layer neural network.

Since different business process case states have different data types. For example, the activity prediction task is a multi-classification problem, and the remaining execution time prediction task is a regression problem. Therefore, the cross-entropy loss function and mean square error loss function are used for the next execution activity prediction and remaining execution time prediction, respectively, which enable back-propagation training of the whole HiP-Transformer model.

4. Results and Discussion

4.1. Experiment Data

The experimental data in this paper are obtained from the public event log dataset of the 4TU Center for Research Data accessed on 1 July 2020. <https://data.4tu.nl/portal>. Since the research in this paper focuses on predictive business process monitoring research for long event sequences, the event logs with long event sequences are mainly used in selecting the dataset, including BPI2012, BPI2012W, BPI2013i, BPI2015, BPI2017, Hospital, and Sepsis.

- BPI2012 is a Holland financial institution loan applying process during the period from October 2011 to March 2012; W indicates the status of work items related to the approval process.

- BPI2013i is the process related to the Volvo back office management system during the period from April 2010 to May 2012.
- BPI2015 is the Holland building permit applying process during the period from April 2014 to September 2014.
- BPI2017 is the Holland financial institution loan applying process during the period from January 2016 to December 2016.
- Hospital is the business process in a regional hospital patient billing information during the period from December 2012 to January 2016.
- Sepsis is a hospital sepsis case visit information related process during the period from October 2014 to December 2014.

Table 1 shows the statistical information of these seven datasets. These seven datasets from 4TU Center for Research Data have more complex business relationships and contain more additional features, and most of the log information has a large time span. The above characteristics meet the concerns of this paper. To verify the relevance of the method in this paper for long event sequences, only business process examples with more than the average sequence length in each dataset are used in the experiments.

Table 1. Descriptive statistics of event logs. Duration related measures are shown in days.

Event Log	Cases Number	Events Number	Activities Number	Average Length	Maximum Length	Average Duration	Maximum Duration
BPI2012	13,087	262,200	24	20	175	8.6	137.2
BPI2012W	9658	170,107	7	17.6	156	10.5	132
BPI2013i	568	6499	13	11.4	123	11.9	768
BPI2015	1199	27,409	38	22.8	61	95.9	1486
BPI2017	31,509	1,202,267	26	35	180	18	395
Hospital	3494	97,794	18	27.9	217	127.2	1034
Sepsis	1049	15,214	16	14.5	185	28.5	422.3

When generating the training sets consisting of business process prefixes from the original event log data according to Equations (3) and (4), the event sequences of each prefix have different lengths. To facilitate the neural network model to process the data, zero-padding is used at the front of the event sequences so that all business process prefixes have the same length.

4.2. Experiment Setting

Accuracy (Acc) and Mean Absolute Error (MAE) were used as the model evaluation metrics in this experiment for the next activity prediction task and the remaining time prediction task, respectively. The two metrics are calculated as follows:

$$Acc = \frac{1}{N} \sum_{i=1}^N \mathbb{I}[y_i = y_i^{(pred)}]$$

$$MAE = \frac{1}{N} \sum_{i=1}^N |y_i - y_i^{(pred)}|$$

where N is the number of business process cases in the test set. y_i and $y_i^{(pred)}$ are the true state (next activity, remaining time) and the corresponding prediction value of the i -th test case, respectively.

In the evaluation of the prediction model, the experiment divides the datasets into an 80% training set, 10% validation set, and 10% testing set, where the training set is used for training the model parameters; the validation set is used to select the optimal model based on the above metrics; and the test set is used to calculate the final prediction effect of the model.

To obtain better prediction results, most parameters of the model were adjusted in the experiments based on the effect of the validation set, and better parameter combinations were obtained as: batch size is 64; the learning rate is 0.01; activity and attribute encoding dimension is 64; self-attention layer encoding dimension is 36; attention head number is 3; attention layer number is 2; dropping rate is 0.5; and step length is 2.

4.3. Comparison Experiments

4.3.1. Next Activity Prediction

The following methods are selected as the comparison methods in the next execution activity prediction task.

- LSTM [10]: next activity prediction using Long Short-Term Memory (LSTM).
- GRU [11]: next activity prediction using Gated Recurrent Unit (GRU).
- CNN [21]: next activity prediction using Convolutional Neural Networks (CNN).
- DNC [22]: a memory unit is added to the LSTM model to constitute a differential neural computer (DNC) to implement the next activity prediction.
- Transformer [16]: the next activity prediction is predicted using the traditional Transformer directly.

Table 2 shows the accuracy of the above five comparison methods and the method in this paper (HiP-Transformer) on the next execution activity prediction task. Overall, the HiP-Transformer model achieves the best prediction results on five of the seven datasets, especially compared to the prediction methods using the traditional Transformer directly on all datasets. The HiP-Transformer model did not achieve the highest accuracy on the BPI2012 dataset, because the business process instances in the dataset have only activity information in each event. The lack of attribute information makes the HiP-Transformer model unable to fuse various attributes, which is not conducive to improving prediction results. The Sepsis dataset is a log of sepsis patients' visit information, and the patient visit process changes with its state. Different business process cases have large differences, so it is more difficult to predict the task of the next execution activity, and the accuracy of all methods is relatively low.

Table 2. Accuracy of next executing activity prediction (accuracy: %).

Datasets	LSTM [10]	GRU [11]	CNN [21]	DNC [22]	Transformer [16]	HiP-Transformer
BPI2012	85.46	86.65	83.25	77.70	85.20	86.30
BPI2012W	85.35	84.78	81.19	60.15	89.30	89.88
BPI2013i	70.09	74.69	46.03	51.91	62.11	79.33
BPI2015	68.58	71.02	58.43	70.56	71.98	73.02
BPI2017	83.15	84.25	78.45	84.75	81.88	85.88
Hospital	78.50	79.46	75.89	70.26	80.98	84.70
Sepsis	64.22	63.50	56.15	21.01	45.98	62.54
Avg.	76.48	77.76	68.48	62.33	73.92	80.24

4.3.2. Remaining Time Prediction

The following method are selected as the comparison method in the remaining execution time prediction task.

- LSTM [10]: remaining execution time prediction using LSTM.
- Accurate-LSTM [23]: using LSTM to model activity sequences, role sequences, and relative time sequences separately to enhance the effectiveness of LSTM models.
- Transformer [16]: remaining execution time prediction is directly implemented using the traditional Transformer.

Table 3 shows the MAE values of the above three comparison methods and the method in this paper (HiP-Transformer) on the remaining execution time prediction task. Overall, the HiP-Transformer model achieves the lowest error on six of the seven datasets.

Furthermore, comparing the two Transformer-based methods with the two LSTM-based methods, we can find that the Transformer model is significantly better than the LSTM model. Compared with the next execution activity prediction task, the prediction target of the remaining execution time prediction task is the state of the next stage of business process instance execution, which is more difficult to predict and puts forward higher requirements on the business process instance modeling method. The results of this experiment reflect the advantages of Transformer in modeling event sequences. In each dataset, the methods have poor prediction results on Hospital and BPI2015, and a common feature of these two datasets is the existence of a large number of events executing repeated activities consecutively, which indicates that deep learning models in modeling this phenomenon are still more room for improvement.

Table 3. Remaining execution time prediction effect (MAE: Day).

Datasets	LSTM [10]	Accurate-LSTM [23]	Transformer [16]	HiP-Transformer
BPI2012	383.00	29.95	4.60	3.89
BPI2012W	157.05	30.484	4.87	5.56
BPI2013i	30.08	28.13	8.36	4.17
BPI2015	42.36	40.10	28.42	24.88
BPI2017	127.80	16.85	10.28	9.93
Hospital	28.30	38.77	24.87	14.28
Sepsis	421.60	17.82	19.47	17.26
Avg.	170.02	28.87	14.41	11.4

4.4. Experiment Analysis

4.4.1. Ablation Experiment

In contrast to existing predictive business monitoring methods based on deep learning, the advantage of the HiP-Transformer model proposed in this paper is that it introduces the idea of concept drift detection to split business process cases. Traditional sequence segmentation methods usually use Piecewise Aggregate Approximation (PAA) for segmentation, which cannot reflect the distribution changes within the sequence. To verify the positive effect of concept drift detection on business process instance prediction, this experiment replaces the concept drift point detection (DPD) in the HiP-Transformer model with equally spaced sequence segmentation. It keeps the structure of the three-layer Transformer model unchanged and then compares the prediction effects of the model before and after the replacement.

Figure 2 shows the prediction accuracy of the two segmentation strategies on the next activity prediction task. Since segmentation of business process cases only involves longer event sequences, only the prediction results of business process cases with event sequence lengths greater than 20 are considered in this experiment. It can be found that the concept drift detection method for event sequence segmentation can improve the prediction results on most datasets, especially as the number of training iterations increases. It is because the encoding of events and their subsequences can be continuously optimized during the training process, which in turn continuously improves the accuracy of concept drift detection and makes the segmented subsequences more consistent with the overall distribution characteristics of the business process cases.

4.4.2. Different Sequence Lengths on Prediction Results

To further understand the prediction performance of the HiP-Transformer model, this experiment groups the business process cases in the testing dataset according to the length of the event sequence and calculates the prediction performance of the HiP-Transformer model on different length case groups.

Figure 3 shows HiP-Transformer's prediction accuracy for the next activity and the Mean Absolute Error for the remaining time on the four datasets. By plotting the evaluation

results of different length business processes on the two prediction tasks, we found that as the sequence length increases, HiP-Transformer's prediction accuracy for the next activity tends to increase, and the prediction error for the remaining time tends to decrease. We can conclude that the model can solve the problem of decreasing prediction accuracy for long sequence business processes to some extent. The above comparison experiments reveal that the three-layer stacked self-attention model, by virtue of fully mining the internal knowledge of the business process, enables the model to fully model the historical information of the sequence when predicting the subsequent states of longer sequences. Thus, HiP-Transformer can achieve good results in most lengths, and the richer the historical information, the better the prediction results.

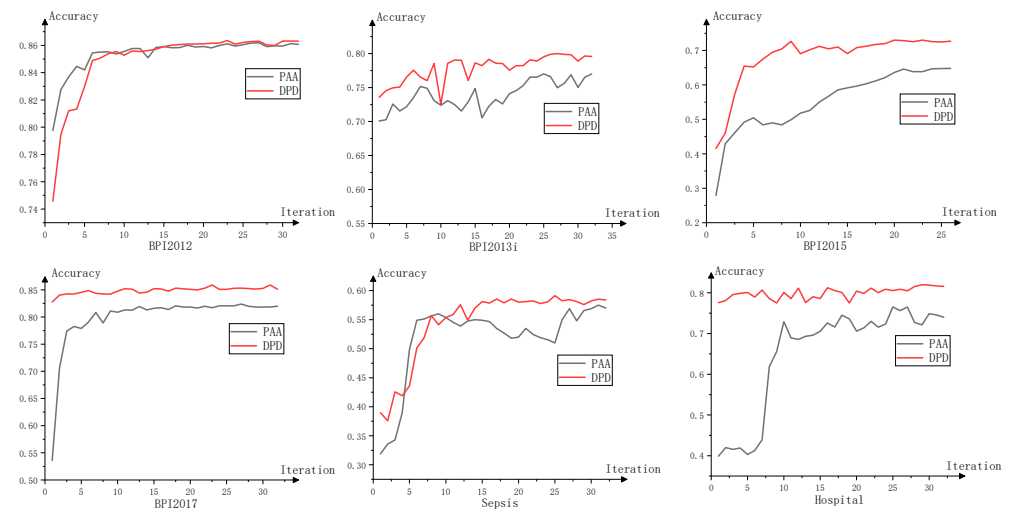


Figure 2. Comparison of the Prediction Accuracy of Business Processes under Different Segmentation Strategies.

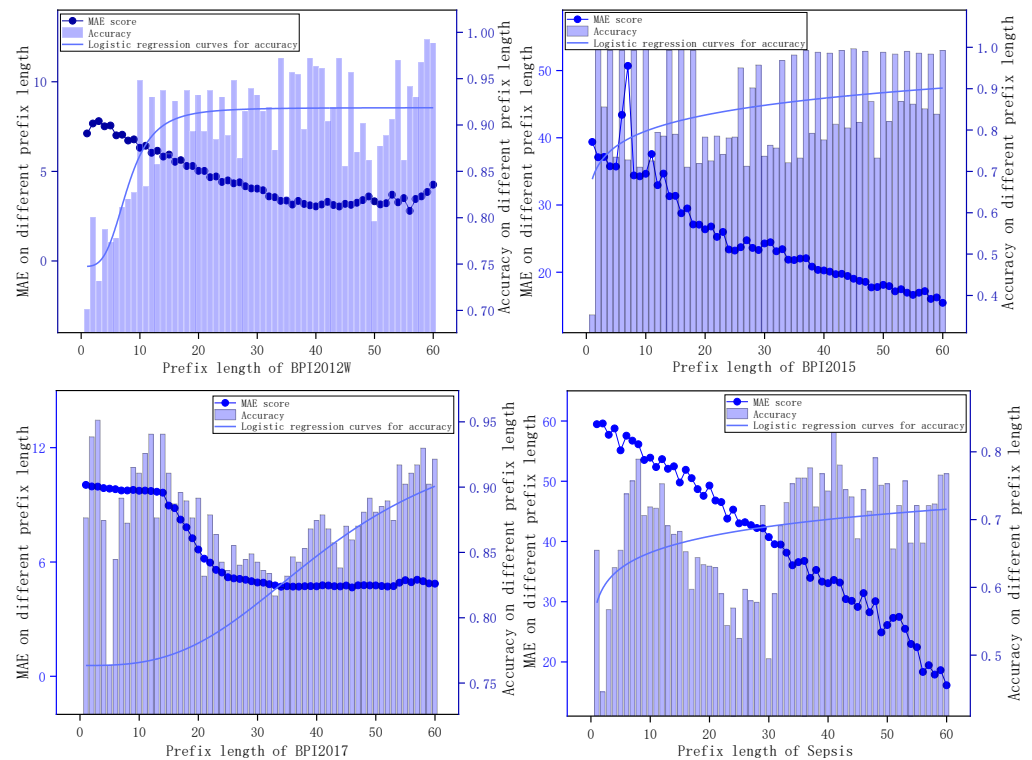


Figure 3. Influence of Prefix Length on Accuracy and MAE of HiP-Transformer.

5. Conclusions

This paper presents HiP-Transformer, a predictive business monitoring model containing a three-layer Transformer architecture, and focuses on the tasks of the next execution activity prediction and remaining execution time prediction. The model specifically contains an event-level Transformer, subsequence-level Transformer, and instance-level Transformer, which encodes the representation of business process cases at different levels. To better deal with the concept drift phenomenon existing during the execution of business processes, this paper also proposes a concept drift detection algorithm to achieve a valid segmentation of business process cases. It can capture different execution phases of business processes and their distribution changes and provide a basis for modeling the subsequence-level Transformer. This paper conducts an experimental study using several real event logs, and the results show that the HiP-Transformer model has a better performance. Compared with benchmark business process prediction models, the HiP-Transformer's accuracy improves by 6.32% in the next activity prediction, and the mean absolute error reduces by 21% in the remaining time prediction. In addition, we design a comparison experiment to compare the prediction accuracy under different segmentation strategies, and it is found that the drift detection-based segmentation strategy proposed in this paper can achieve higher accuracy in the end. Further experimental analysis on different prefix length reveals that the activity prediction accuracy tends to increase, and the remaining time prediction error tends to decrease with the increase in prefix length and its corresponding feature information, which verifies that the model can enhance the modeling of long business process cases by properly segmenting event subsequences.

Future work can apply the HiP-Transformer model proposed in this paper to other predictive business monitoring tasks, such as business process execution result prediction, abnormal activity checking, etc., to better validate the value of the method in this paper. Large institutions, such as banks and credit companies, can apply our method on extracting consumer information and other factors to estimate remaining repayment time. The division of different consumer segments enables a more accurate prediction of their payment capacity, which can help to address potential financial risks in advance.

Author Contributions: Conceptualization, W.N. and G.Z.; methodology, W.N.; software, G.Z.; validation, W.N., G.Z. and T.L.; formal analysis, X.X.; investigation, W.N. and G.Z.; data curation, G.Z. and X.X.; writing—original draft preparation, W.N., G.Z., T.L. and Q.Z.; writing—review and editing, W.N., G.Z., T.L. and Q.Z.; funding acquisition, W.N. All authors have read and agreed to the published version of the manuscript.

Funding: The work is supported by Shandong Provincial Natural Science Foundation (No. ZR2022MF319), National Natural Science Foundation of China (No. U1931207), Taishan Scholars Program of Shandong Province (TS20190936), Talented Young Teachers Training Program of Shandong University of Science and Technology (BJ20211110).

Data Availability Statement: The data presented in this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical.

Acknowledgments: The authors wish to thank all who assisted in conducting this work.

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

References

1. Aalst, W.; Adriansyah, A.; Medeiros, A.; Arcieri, F.; Wynn, M.T. *Process Mining Manifesto*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 169–194.
2. Rama-Maneiro, E.; Vidal, J.; Lama, M. Deep learning for predictive business process monitoring: Review and benchmark. *IEEE Trans. Serv. Comput.* 2021; *early access*. [[CrossRef](#)]
3. Becker, J.; Breuker, D.; Delfmann, P.; Matzner, M. Designing and Implementing a Framework for Event-based Predictive Modelling of Business Processes. In *Proceedings of the International Workshop on Enterprise Modeling and Information Systems Architectures*, Luxembourg, 25–26 September 2014.

4. Rogge-Solti, A.; Weske, M. Prediction of business process durations using non-Markovian stochastic Petri nets. *Inf. Syst.* **2015**, *54*, 1–14. [[CrossRef](#)]
5. Lakshmanan, G.T.; Shamsi, D.; Doganata, Y.N.; Unuvar, M.; Khalaf, R. A markov prediction model for data-driven semi-structured business processes. *Knowl. Inf. Syst.* **2015**, *42*, 97–126. [[CrossRef](#)]
6. Cabanillas, C.; Ciccio, C.D.; Mendling, J.; Baumgrass, A. Predictive task monitoring for business processes. In *International Conference on Business Process Management*; Sadiq, S., Soffer, P., Völzer, H., Eds.; Springer: Cham, Switzerland, 2014; Volume 8659, pp. 424–432.
7. Francescomarino, C.D.; Dumas, M.; Maggi, F.M.; Teinemaa, I. Clustering-Based Predictive Process Monitoring. *IEEE Trans. Serv. Comput.* **2016**, *12*, 896–909. [[CrossRef](#)]
8. Evermann, J.; Rehse, J.R.; Fettke, P. A Deep Learning Approach for Predicting Process Behaviour at Runtime. In *International Conference on Business Process Management*; Dumas, M., Fantinato, M., Eds.; Springer: Cham, Switzerland, 2017; Volume 281, pp. 327–338.
9. Ni, W.J.; Sun, Y.J.; Liu, T. Business process remaining time prediction using bidirectional recurrent neural networks with attention. *Comput. Integr. Manuf. Syst.* **2020**, *26*, 1564–1572.
10. Tax, N.; Verenich, I.; Rosa, M.L.; Dumas, M. Predictive Business Process Monitoring with LSTM Neural Networks. In *International Conference on Advanced Information Systems Engineering*; Dubois, E., Pohl, K., Eds.; Springer: Cham, Switzerland, 2017; Volume 10253, pp. 477–492.
11. Hinkka, M.; Lehto, T.; Heljanko, K.; Jung, A. Classifying Process Instances Using Recurrent Neural Networks. In *International Conference on Business Process Management*; Daniel, F., Sheng, Q., Motahari, H., Eds.; Springer: Cham, Switzerland, 2018; Volume 342, pp. 313–324.
12. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5998–6008.
13. Ma, X.; Zhang, P.; Zhang, S.; Duan, N.; Zhou, M. A tensorized transformer for language modeling. In Proceedings of the 33rd International Conference on Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019.
14. Sun, F.; Liu, J.; Wu, J.; Pei, C.; Lin, X.; Ou, W.; Jiang, P. BERT4Rec: Sequential recommendation with bidirectional encoder representations from transformer. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1441–1450.
15. Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. A survey on vision transformer. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**; early access. [[CrossRef](#)]
16. Bukhsh, Z.A.; Saeed, A.; Dijkman, R.M. Process Transformer: Predictive Business Process Monitoring with Transformer Network. *arXiv* **2021**, arXiv:2104.00721.
17. Li, L.; Wen, L.; Wang, J. MM-Pred: A Deep Predictive Model for Multi-attribute Event Sequence. In Proceedings of the 2019 SIAM International Conference on Data Mining, Calgary, AB, Canada, 2–4 May 2019; pp. 118–126.
18. Schnig, S.; Jasinski, R.; Ackermann, L.; Jablonski, S. Deep Learning Process Prediction with Discrete and Continuous Data Features. In Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, Funchal, Portugal, 23–24 March 2018; pp. 314–319.
19. Sato, D.M.V.; Freitas, D.S.C.; Barddal, J.P.; Scalabrin, E.E. A survey on concept drift in process mining. *ACM Comput. Surv.* **2021**, *54*, 1–38. [[CrossRef](#)]
20. Teinemaa, I.; Dumas, M.; Rosa, M.L.; Maggi, F.M. Outcome-oriented predictive process monitoring: Review and benchmark. *ACM Trans. Knowl. Discov. Data* **2019**, *13*, 1–57. [[CrossRef](#)]
21. Pasquadibisceglie, V.; Appice, A.; Castellano, G.; Malerba, D. Using Convolutional Neural Networks for Predictive Process Analytics. In Proceedings of the IEEE International Conference on Process Mining, Aachen, Germany, 24–26 June 2019; pp. 129–136.
22. Khan, A.; Le, H.; Do, K.; Tran, T.; Ghose, A.; Dam, H.; Sindhgatta, R. Memory-Augmented Neural Networks for Predictive Process Analytics. *arXiv* **2018**, arXiv:1802.00938v1.
23. Camargo, M.; Dumas, M.; González-Rojas, O. Learning Accurate LSTM Models of Business Processes. In *International Conference on Business Process Management*; Hildebrandt, T., van Dongen, B., Röglinger, M., Mendling, J., Eds.; Springer: Cham, Switzerland, 2019; Volume 11675, pp. 286–302.
24. Liu, T.; Ni, W.; Sun, Y.; Zeng, Q. Predicting Remaining Business Time with Deep Transfer Learning. *Data Anal. Knowl. Discov.* **2020**, *4*, 134–142.
25. Mehdiyev, N.; Evermann, J.; Fettke, P. A Multi-stage Deep Learning Approach for Business Process Event Prediction. In Proceedings of the IEEE 19th Conference on Business Informatics, Thessaloniki, Greece, 24–27 July 2017; pp. 119–128.
26. Ni, W.; Yan, M.; Liu, T.; Zeng, Q. Predicting remaining execution time of business process instances via auto-encoded transition system. *Intell. Data Anal.* **2022**, *26*, 543–562. [[CrossRef](#)]
27. Philipp, P.; Jacob, R.; Robert, S.; Beyerer, J. Predictive Analysis of Business Processes Using Neural Networks with Attention Mechanism. In Proceedings of the IEEE International Conference on Artificial Intelligence in Information and Communication, Fukuoka, Japan, 19–21 February 2020; pp. 225–230.
28. Hoeffding, W. Probabilities inequalities for sums of bounded random variables. *J. Am. Stat. Assoc.* **1963**, *58*, 13–30. [[CrossRef](#)]

29. Frias-Blanco, I.; del Campo-Ávila, J.; Ramos-Jimenez, G.; Morales-Bueno, R.; Ortiz-Díaz, A.; Caballero-Mota, Y. Online and non-parametric drift detection methods based on Hoeffding's bounds. *IEEE Trans. Knowl. Data Eng.* **2014**, *27*, 810–823. [[CrossRef](#)]
30. Ankerst, M.; Breunig, M.M.; Kriegel, H.P.; Sander, J. OPTICS: Ordering points to identify the clustering structure. In Proceedings of the ACM SIGMOD International Conference on Management of Data, New York, NY, USA, 31 May–3 June 1999; Volume 28, pp. 49–60.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.