Hindawi
International Journal of Intelligent Systems
Volume 2023, Article ID 3777826, 18 pages
https://doi.org/10.1155/2023/3777826

WILEY | Hindawi

Research Article

LogBASA: Log Anomaly Detection Based on System Behavior Analysis and Global Semantic Awareness

Liping Liao,[1,2] **Ke Zhu**,[1] **Jianzhen Luo**,[1,2] **and Jun Cai**[1,2]

[1]*School of Cyber Security, Guangdong Polytechnic Normal University, Guangzhou, China*
[2]*Guangdong Provincial Key Laboratory of Intellectual Property and Big Data, Guangdong Polytechnic Normal University, Guangzhou, China*

Correspondence should be addressed to Jianzhen Luo; luojz@gpnu.edu.cn

Received 13 March 2023; Revised 22 August 2023; Accepted 30 August 2023; Published 20 September 2023

Academic Editor: Gianni Costa

System log anomaly detection is important for ensuring stable system operation and achieving rapid fault diagnosis. System log sequences include data on the execution paths and time stamps of system tasks in addition to a large amount of semantic information, which enhances the reliability and effectiveness of anomaly detection. At the same time, considering the correlation between system log sequences can effectively improve fault diagnosis efficiency. However, the existing system log anomaly detection methods mostly consider only the sequence patterns or semantic information on the logs, so their anomaly detection results show a high rate of missed and false alarms. To solve these problems, this paper proposed an unsupervised log anomaly detection model (LogBASA) based on the system behavior analysis and global semantic awareness, aiming to decrease the leakage rate and increase the log sequence anomaly detection accuracy. First, a system log knowledge graph was constructed based on massive, unstructured, and multilevel system log data to represent log sequence patterns, which facilitates subsequent anomaly detection and localization. Then, a self-attention encoder-decoder transformer model was developed for log spatiotemporal association analysis. This model combines semantic mapping and spatiotemporal features of log sequences to analyze system behavior and log semantics in multiple dimensions. Furthermore, a system log anomaly detection method that combines adaptive spatial boundary delineation and sequence reconstruction objective functions was proposed. This method uses special words to characterize the log sequence states, delineates anomaly boundaries automatically, and reconstructs log sequences through unsupervised training for anomaly detection. Finally, the proposed method was verified by numerous experiments on three real datasets. The results indicate that the proposed method can achieve an accuracy rate of 99.3%, 95.1%, and 97.2% on HDFS, BGL, and Thunderbird datasets, which proves the effectiveness and superiority of the LogBASA model.

1. Introduction

In recent years, computer systems have become larger and more functionally complex due to the rapid development of communication technology and artificial intelligence, as well as the explosive expansion in the number of linked smart devices. Anomaly detection has become an important task in the design of reliable computer systems. Once an anomaly occurs in a large-scale system, it can affect system availability and reliability or even directly lead to system disruption and paralysis [1]. Moreover, malicious system attacks may cause incalculable harm to society [2–5]. Therefore, to ensure the safe and reliable operation of large-scale systems, system administrators continuously collect real-time monitoring data, monitor the operational status of a system, and use different anomaly detection techniques to detect abnormal system behavior in a timely manner.

Currently, log analysis is one of the main techniques for system anomaly detection. System logs contain semantically rich event content and system runtime state data, which represent external manifestations of system behavior and denote one of the most important data types for anomaly detection and system monitoring tasks [6]. That is, if a system failure occurs, engineers can efficiently trouble-shoot and locate it by viewing the system log data. However, the amount of log data increases with the size and

complexity of a system, and a massive amount of log data is generated during the system runtime; typically, approximately 1 GB of log data is generated per hour for a commercial system [7]. The massive amount of log data makes manual detection of abnormal events infeasible. In addition, log data are inherently unstructured and contextually interdependent, and a set of log data can be concatenated by the same ID and generated into log sequences to track their status and critical events. Recently, correlating log data by IDs has been a typical approach for separating interleaved logs, and mining out the dependencies between log data can significantly improve the accuracy of anomaly localization. However, log forms change throughout the software system lifecycle [8–10], which can make anomalous behavior data mining challenging. In conclusion, the detection of anomalous logs represents a highly difficult task since log data are massive, unstructured, context-dependent, and diverse in form. Hence, it is essential to use precise and effective log anomaly detection methods.

The existing log-based anomaly detection methods can be roughly divided into three categories: pattern recognition methods based on log template counting vectors, pattern learning methods based on log event indexes, and log semantic-based representation learning methods. Pattern recognition methods based on log template counting vectors [11–15] use log event count vectors as input, and each event is individually treated as a dimension and mapped to the vector space using traditional machine learning methods; deviations from the direction of normal log sequences are classified as anomalies. However, these methods do not consider the sequential pattern and content of a log sequence and determine exceptions only based on the co-occurrence pattern between log templates. In the pattern learning methods based on log event indexes [16, 17], a log event sequence is analyzed using a sliding window, and the next event index is predicted based on the observation window data. However, these methods do not consider log content information and cannot learn semantic relationships between events, and a fixed sliding window size cannot capture sequential patterns outside the sliding window range. Also, these methods do not consider the correlation of the system behavior from a global perspective. In the log semantic-based representation learning methods [18, 19], event templates are extracted from log data, and natural language processing (NLP) techniques are used to generate log semantic representations for each template. However, these methods do not consider the correlation between the events in a log sequence and instead only use the semantic representation of each log text. The occurrence of anomalous events not only affects the system behavior but may also affect the subsequent execution path of the system. Therefore, analyzing only the log semantics cannot provide information on the contextual relationships between system behaviors. In addition, when anomalies occur in real log data, not only the sequence of events within a log sequence changes but also the time difference between adjacent events, so the information on time difference is also crucial for log analysis. Recent related research [6] has combined temporal embedding with log semantics to detect log sequence anomalies, aiming to uncover potential performance problems and detect anomalies from the perspective of system performance fluctuations that do not reflect sequential patterns of system behaviors.

To address the aforementioned challenges, this paper introduces an innovative approach named LogBASA for detecting log data anomalies. The LogBASA addressed the current challenges by combining system behavior analysis and global semantic awareness. First, a system log knowledge graph (SLKG) rooted in logs is designed. This graph integrates a substantial volume of log data and crafts a log relationship model, facilitating the identification of anomalous logs and the encapsulation of semantic nuances and sequential patterns within log event sequences. The transformation of log sequence patterns into a system functional path graph is initiated in the context of system behavior analysis. In this process, log event semantics are regarded as node features, which are amalgamated with the spatial configuration of the graph, and a graph convolution network (GCN) is used to amplify spatial attributes within a log sequence. Subsequently, a multilayer perceptron (MLP) is used to capture temporal features by representing the time differences between adjacent log events. Regarding the global semantic perception, the pretrained BERT model [20] is used to encode the log template semantics. The global semantics of the log sequences are subsequently mapped using the self-attention encoder-decoder transformer model. This mapping integrates spatiotemporal correlation features, providing a multidimensional system anomaly detection capability. By combining semantic insights and integrating temporal information with log sequence patterns, the proposed approach synergistically enhances the efficiency of anomaly detection within log event sequences.

The main contributions of this paper are as follows:

(1) An innovative anomaly detection model named LogBASA, which leverages global semantic awareness in combination with the system behavior analysis, is proposed. The LogBASA model demonstrates remarkable proficiency in merging semantic information from log sequences, spatial characteristics of log graphs, and temporal features of adjacent logs.

(2) An advanced model training methodology that harmonizes adaptive spatial boundary delineation with the objective function for sequence reconstruction is introduced. This approach employs distinct terminology to delineate log sequences and adopts unsupervised learning techniques to demarcate boundaries for normal logs dynamically. Furthermore, sequence representation reconstruction is performed to elucidate potential relationships between log events through semantic mapping and sequence patterns.

(3) A SLKG is designed to depict the interconnections within system logs. This innovative approach adeptly assimilates unstructured log data, providing

a significant improvement in both exception detection efficiency and fault localization accuracy.

(4) The proposed method is compared with eight typical methods on HDFS, BGL, and Thunderbird datasets, and the comparison results validate the effectiveness of the proposed method.

The rest of this paper is organized as follows. In Section 2, an overview of related work is presented. The proposed system model, including the SLKG, log graph feature representation, and log timing feature representation, is introduced in Section 3. The LogBASA model is described in detail in Section 4. In Section 5, the performance of the proposed model is verified on three real log datasets. Finally, the main conclusions are drawn in Section 6.

2. Related Work

Recently proposed methods for log anomaly detection can be broadly classified into two categories: data mining-based methods and deep learning-based methods.

2.1. Data Mining-Based Log Anomaly Detection. The existing data mining-based methods can be mainly classified into supervised learning-based methods and unsupervised learning-based methods. Supervised learning-based methods learn the fixed patterns of different labeled logs through training on labeled log data and then use support vector machine (SVM) [21], finite state automata (FSA) [22, 23], decision tree (DT) [24], logistic regression (LR) [25, 26], and other methods for anomaly detection. Their detection performance is usually higher than that of unsupervised learning-based methods [27]. However, the cost of labeling a large amount of log data is high. Unsupervised learning-based methods mainly include principal component analysis (PCA) [28, 29], frequent pattern mining [30], clustering [12, 31], and isolation forest (IF) [11]. Unsupervised learning-based methods have the advantages of low cost and interpretability, but despite these advantages, data mining-based methods tend to perform inconsistently in certain situations because they cannot capture sequential patterns and semantic information between log events.

2.2. Deep Learning-Based Log Anomaly Detection. Deep learning has received great attention in the past decade due to its superiority in model representation and modeling performance. Many studies have applied deep learning to log sequence anomaly detection under unsupervised learning-based schemes. There are three main types of neural networks: CNNs, RNNs, and transformers. Regarding the CNN-based approaches, Lu et al. [15] proposed a method to extract system log features using CNN models to detect anomalies in big data systems. Furthermore, to meet the demand for real-time detection, Wang et al. [32] proposed a lightweight log anomaly detection model named LightLog, which can increase the processing speed in anomaly detection tasks by creating a low-dimensional semantic vector space and using a lightweight temporal convolutional network (TCN). Regarding the RNN-based approaches, Brown et al. [33] proposed an RNN model that incorporates an attention mechanism to mine system logs for anomaly patterns. To model the sequence data, Zhang et al. [34] proposed an anomaly detection model for the automatic analysis of console logs. This model focuses on capturing the sequential features of log sequences using a long short-term memory (LSTM) network; this was the first time that the LSTM model was applied to log anomaly detection. Accordingly, a number of studies on log anomaly detection have been performed using the LSTM model and its variants. Based on [33], Du et al. [16] proposed the DeepLog model consisting of two main parts: log template anomaly detection and log variable anomaly detection. The first part extracts the sequential features of log sequences using the LSTM model, and the second part detects anomalies by modeling the log relationships. Compared to the model presented in [34], the DeepLog model can achieve more comprehensive log anomaly detection. Furthermore, Meng et al. [19] proposed the LogAnomaly model, which uses the template2vec log vectorization method. This model can effectively vectorize log templates and detect anomalies in log sequences by combining the LSTM with attention mechanisms. The LogAnomaly model's detection performance is better than that of the DeepLog model due to its improved log vectorization method. That is, it reuses the LSTM network for feature extraction and fine-tunes the classifier network parameters for anomaly detection. Wang et al. [35] proposed an anomaly detection model named OC4Seq, which used a multiscale RNN framework that considered the log data imbalance. In this way, different levels of sequence patterns can be captured by embedding discrete event sequences into the latent space, thus providing relatively easy anomaly detection. Regarding the anomaly detection methods based on the transformer model, Wibisono and Kistijantoro [36] used an adaptive general transformer-based model, following the work in [33], to learn long input sequences and improve the anomaly detection accuracy. Huang et al. [37] proposed the HitAnomaly model, which adopts hierarchical transformer structures to model log template sequences and parameter values and encode them using encoders. The anomaly detection mechanism used in the HitAnomaly model is based on attention. Guo et al. [17] developed the LogBERT model to learn the pattern of normal log sequences through mask log message prediction and hypersphere volume minimization.

Compared to the data mining-based log anomaly detection methods, the abovementioned models use neural networks to extract sequence features from log data and then capture the anomaly patterns of a system, showing relatively superior performance. However, most of the sequence pattern-based methods analyze only the sequence relationships between log events for anomaly detection, lacking consideration of spatial correlations between log events from the perspective of system behavior, which hinders further improvement of anomaly detection performance and can lead to unstable performance in specific cases. Recently, Wan et al. [38] proposed a method based on the graph neural networks (GNNs) named GLAD-PAW, which uses the graph attention networks (GAT) for log

anomaly detection, and verified the feasibility of the graph-based log anomaly detection methods. However, although this approach is graph-based, it still adopts the sequence model for spatial structure perception, namely, it uses only positional encoding while ignoring the interaction between node features and graph structure, which can lead to sub-optimal results. In contrast, the LogBASA model proposed in this study combines semantic mapping of log text, a node-centric spatial structure, and interevent temporal features for behavioral analysis and semantic perception of a given sequence of events. That is, in the model training phase, the semantic mapping is combined with the spatiotemporal association features by the self-attention encoder-decoder transformer model to improve the log anomaly detection performance and stability.

3. System Modeling

The LogBASA model proposed in this paper is a log anomaly detection method based on the self-attention encoder-decoder transformer model. The flowchart of the proposed method is shown in Figure 1, where it can be seen that it includes four main parts: log preprocessing, system behavior analysis, global semantic awareness, and anomaly detection. First, the LogBASA cleans the collected system log data, generates a multidimensional and fine-grained system description, and constructs a system log knowledge graph (SLKG) using the log data and system configuration files. Next, the log sequence is converted into a system functional path diagram to represent the system behaviors generated under the current log sequence. The node features provide semantic information on the log sequence, and the edges contain information on connectivity and weights between node pairs. Furthermore, the MLP network is used to extract the temporal features of log events, which are then fused with the spatial features of log sequences to generate the spatiotemporal association features of the log event sequences, and system behavior analysis is performed on them. The LogBASA method uses a pretrained BERT model to obtain semantic information on log event sequences and adds a special marker [SIGN] in front of sequences to characterize the semantic state of the sequence, which is then used as the encoder's input and encoded as a log semantic mapping. Furthermore, the system's behavioral features and global semantic sensing data are fused in the decoder for anomaly detection. Afterward, a training method that combines adaptive boundary partitioning based on unsupervised learning with the objective function of sequence reconstruction is employed to improve the performance of anomaly detection. Finally, the LogBASA provides the fault diagnosis result and feedback to operators for timely troubleshooting, which improves system maintenance efficiency.

3.1. System Log Knowledge Graph Construction. Previous studies have concluded that it is difficult to mine the correlation between system anomalies by analyzing only the log sequence data. To address this shortcoming, this study designs a system log knowledge graph (SLKG) to characterize the relationships between anomalies, which helps to enhance the anomaly detection effect. As shown in Figure 2, the SLKG consists of three layers: the system layer, the component layer, and the event layer, and their main functions are as follows:

(1) The system layer integrates the logs from various systems.

(2) The component layer describes the components of each system.

(3) The event layer represents log events.

The SLKG mainly contains three types of entities, the system, component, and event entities in the system, component, and event layers, respectively. Also, the SLKG mainly considers three types of relationships: system-component relationships, component-event relationships, and interevent relationships.

The SLKG construction process includes three main steps:

(1) Step 1: define the basic framework of entity, relationship, attributes, and SLKG based on the specific design principles.

(2) Step 2: automatically extract the corresponding entities and attributes from the original log data using log parsing techniques.

(3) Step 3: update the logs generated by the system to the SLKG in real time.

It is worth mentioning that the SLKG has a simple structure and is easy to design and apply. The hierarchical structure of the SLKG allows a clear analysis of relationships and a good understanding of the relationships between anomalies. This study uses the NLP techniques to extract the semantic features of logs, which can help to train the LogBASA model and effectively improve the performance of anomaly detection and fault diagnosis.

In this paper, the log content is parsed by the log parser Drain [39], and the component names and log templates in the logs are extracted as entities; the timestamp, pid, and level are used as attributes of the log template entities. Then, the relationships are set as hasComponent, hasTemplate, and hasAttribute using the predefined means. This paper stores the SLKG based on the Neo4j graphical database, whose data model is based on the graphical structures, making it well suited for representing complex relational data. In addition, for the SLKG queries, the Neo4j database uses indexing and query optimization techniques to support entity-based and relational queries effectively. In addition, the Neo4j effectively also supports horizontal and vertical expansion and thus can easily handle large-scale SLKG.

3.2. System Function Path Map. Each log in the system is generated by executing the log print statements in the source code. The order of logging events generated by a system task indicates the execution order of the function that completes the task. Therefore, the system execution process can be modeled as a system function path map, which is a directed

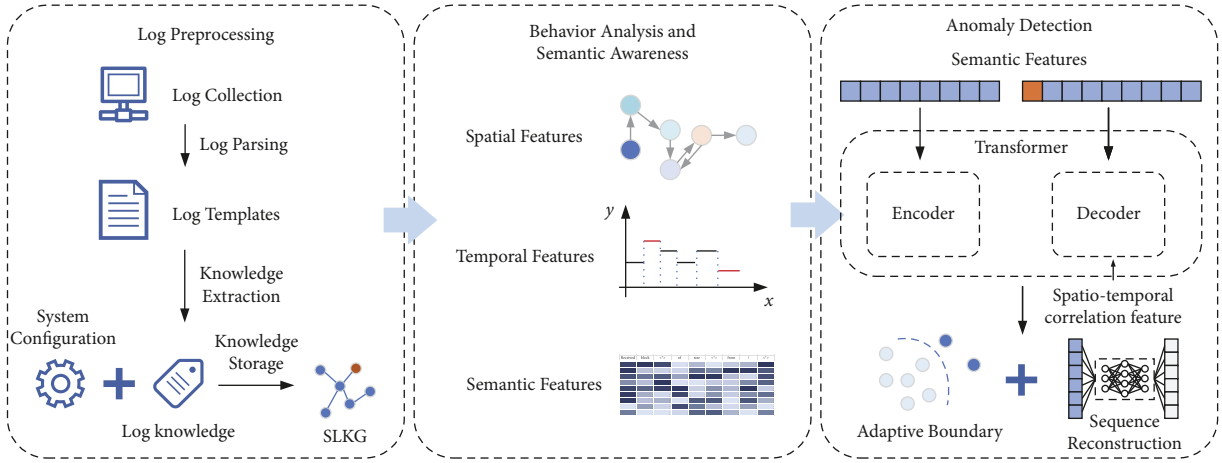

Figure 1: The overall architecture of the proposed method.

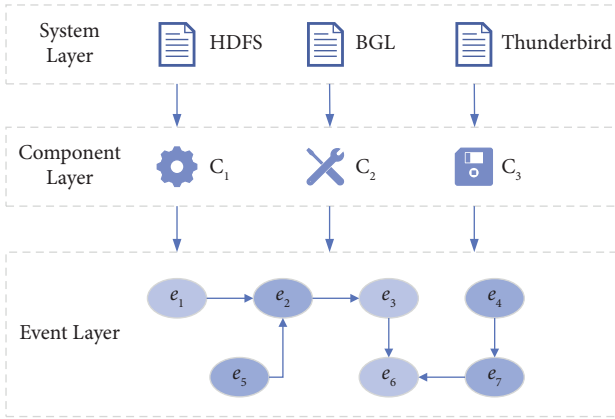

Figure 2: The SLKG structure.

graph, to capture the execution paths of tasks. When the anomaly detection model detects an anomaly, the system function path map can improve the efficiency of fault diagnosis. First, a log sequence is extracted from the SLKG, as explained in Section 3.1, and converted into a system functional path graph, denoted by $G = (V, E, M, N)$, where V represents the set of nodes v_i corresponding to the log events of the SLKG and E is the set of edges e_{v_i,v_j} (i.e., the sequence of events v_i linking events v_i to form a node pair). $M \in \mathbb{R}^{|V|\times d}$ denotes the node feature set corresponding to the semantic vector of log events generated by the NLP technique. $|V|$ is the number of nodes, and d is the feature dimension of the semantic vector. $N \in \mathbb{R}^{|E|\times 1}$ is the set of edge weights, which defines the frequency of occurrence of an edge e_{v_i,v_j} in the sequence. It is important to emphasize that self-looping edges are used for the initial event since there are no previous events before the initial event. In the previous steps, the system function path map is constructed from a given set of log sequences.

As shown in Figure 3, a log sequence with a set of semantic features of an event $l_i = [e_1, e_2, e_3, e_4, e_3, e_5, e_5], i \in |S|$, where $|S|$ is the total number of log sequences, is converted into a system functional path diagram consisting of node features and graph structure features. As can be seen from the graph in Figure 3, the graph provides a more intuitive spatial structure property than the log event sequence. The spatial structure of the graph is regarded as a node-centered global structure represented by a degree matrix. The accuracy and stability of anomaly detection can be improved by generating a more expressive spatial feature representation of the log sequence based on the graph convolution model. However, it should be noted that the system functional path graph does not contain duplicate nodes, which is different from log sequences. For instance, in the log sequence in Figure 3, events e_3 and e_5 appear twice, but nodes e_3 and e_5 denote single nodes in the transformed graph. The information on the frequency of occurrence of edges in each node pair is characterized by in and out degree to ensure that the log sequence information is represented completely in the graph.

3.3. Log Temporal Feature Representation. Most of the existing approaches construct anomaly detection models using the log template count vectors, template semantics, or their combination. However, the timestamp represents an important piece of information on a system log, which describes the moment when the system performs a particular task. As shown in Figure 4, the time interval of log events can reflect the current operating status of the system. Therefore, the timestamp can be used to calculate the running time between two log messages. In a stable system, the path traveled and the time consumed to execute a system task are relatively stable, so this study introduces the time difference between log events into anomaly detection and combines it with the system execution path to analyze the system logs from multiple dimensions.

Define the time difference sequence of a log sequence as $\Delta T = \{\Delta t_1, \Delta t_2, \ldots, \Delta t_{|l|}\}$, where Δt is the time difference between two adjacent log events E and $|l|$ is the log sequence length. As shown in Figure 4, Δt in the log sequence is closely related to the previous event e. For instance, Δt is smaller for IO tasks but larger for scheduling tasks [6]. Even in normal

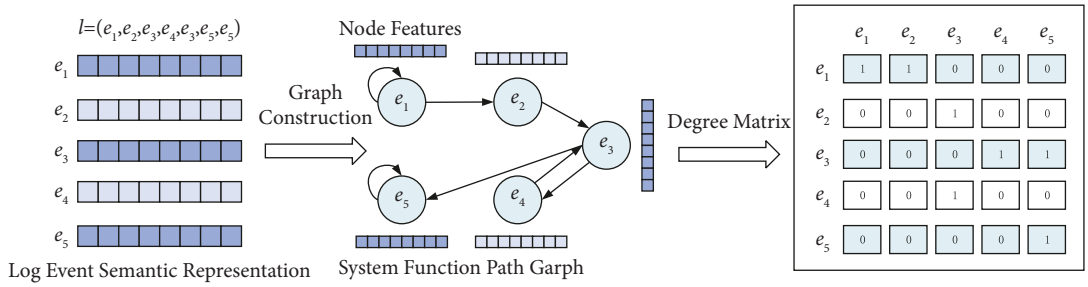

Figure 3: The system function path map.

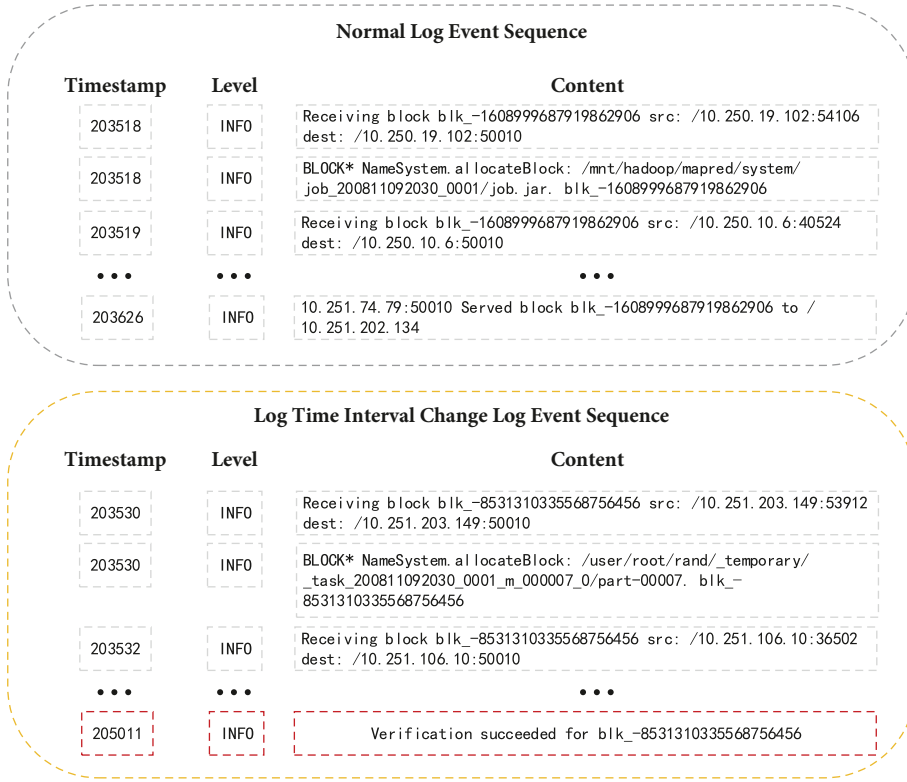

Figure 4: Illustration of normal and time-difference-fluctuating log event sequences.

system operation, the time interval oscillates within the time range associated with a task.

Since the original time difference series data denote one-dimensional temporal data, this study extends the one-dimensional temporal data using high-dimensional embedding to improve its interpretability. Li et al. [40] proposed a time-dependent event representation method, considering that one-hot coding could generate a large sparse matrix when processing multicategorical data, which could further cause large resource consumption. However, if there is a large Δt in the log sequence, it may lead to a dimensionality explosion and even affect the model training effect. In addition, the features of time difference sequences have nonlinear relationships, where the time difference between events in a normal log event sequence is uniform and shows certain continuity, whereas the occurrence of anomalies breaks the original continuity. Previous research [41] has shown that the MLP algorithm is effective in the feature extraction task for high-dimensional continuous data. The MLP algorithm can perform nonlinear modeling and feature extraction for continuous data, learn nonlinear features of data through hidden layers, and, due to its adaptive nature, can adaptively adjust the model parameters according to the data features to achieve feature extraction for time-difference sequences. Therefore, this study uses the MLP to perform feature extraction of time-difference sequences.

3.4. Definition of Specific Terms. In this section, formal and important terms used in this work are defined. Table 1 summarizes the mathematical notations used in this paper.

Definition 1. Log event: A log event consists of a set of words and numbers that are output from the system source code to reflect the textual content of the currently executed functions of

Table 1: Summary of symbols.

Symbol	Descriptions
token	Tokenized words, symbols, or numbers
e	Log event consisting of a sequence of tokens
$e^{[\mathrm{CLS}]}$	Add [CLS] and [SEP] tagged log events
l	A chronological sequence of log events
S	A training set consisting of sequences of log events
G	A directed graph that represents the system functional path
V	A set of all nodes in the system functional path map
E	A set of all edges in the system functional path map
M	A collection of node semantic features
N	Node degree matrix
v_i	The ith node in the node set
e_{v_i,v_j}	Edge between nodes v_i and v_j
Δt	Time difference between adjacent events in the event sequence
ΔT	Time-difference series representation
sem_i	The semantic representation of the ith token in the sequence
$\mathrm{sem}^{[\mathrm{CLS}]}$	Semantic representation of the additional word [CLS]
$\mathrm{sem}^{[\mathrm{SIGN}]}$	Semantic representation of the special addition word [SIGN]
Q, K, V	Matrix of queries, keys, and values in the attention mechanism
f_s	Spatial feature representation of the system functional path map
f_t	Log temporal features representation
X_e	Semantic representation of event sequences
X_d	Complete decoder input
Y_e	Semantic feature mapping of event sequences
Y_d	Event sequence decoder output representation
$Y^{[\mathrm{SIGN}]}$	The decoder output representation of the special word [SIGN]
Y	Full encoder output containing the special word [SIGN]
F	Feature representation, incorporating encoder output, spatial features, and temporal features
H_g^i	The hidden representation of the ith layer graph convolution
H_e^i	The hidden representation of the layer i encoder
H_d^i	The hidden representation of the layer i decoder
L	Loss functions, including L_{boundary} and L_{re}
C	Center of all decoder outputs for the special word [SIGN] in the training set S
y	Log event sequence label
R	Abnormal decision boundary

the system. In this study, the log parser Drain [39] is used to generate the event template after parsing, and all the log events in this paper denote event templates. Formally, a log event is expressed as $e = \{\mathrm{token}_1, \mathrm{token}_2, \ldots, \mathrm{token}_{|e|}\}$, where token_i denotes the ith word, number, or symbol and $|e|$ is the number of words in event e. For example, $e = \{$Receiving, block, <∗>, *src*, :, /, <∗>, *dest*, :, /, <∗>$\}$.

Definition 2. Log event sequence: A log event sequence can be described as a sequence of consecutive events output by the system in chronological order during the execution of a completed task. Formally, a log event sequence is defined as $l = \{e_1, e_2, \ldots, e_{|l|}\}$, where e_i denotes the ith event and $|l|$ is the number of events in a fixed time period. For example,

$$l = \left\{ \begin{array}{c} \text{Receiving block } \langle * \rangle \text{src: /> dest: />,} \\ \text{BLOCK} * \text{NameSystem.allocateBlock: } \langle * \rangle, \\ \text{PacketResponder } \langle * \rangle \text{ for block } \langle * \rangle \text{ terminating,} \\ \ldots, \\ \text{BLOCK} * \text{NameSystem.addStoredBlock: blockMap updated: } \langle * \rangle \\ \text{is added to } \langle * \rangle \text{ size } \langle * \rangle \end{array} \right\}. \quad (1)$$

Definition 3. Log event sequence anomaly detection: All collected log event sequences are combined into a training set denoted by $S = \{l_1, l_2, \ldots, l_{|S|}\}$, where $|S|$ is the total number of log event sequences in the training set. In the unsupervised training mode, the training set contains only normal log event sequences, and an anomaly detection model learns the feature patterns of the normal log time sequences; when the system generates a new event sequence, the model can determine whether it is a normal or anomaly sequence.

As mentioned before, semantic information cannot express event dependencies, and sequential patterns cannot reflect semantic features. Accordingly, this study aims to address these limitations from the perspective of system behavior analysis and global semantic perception. Therefore, this study jointly optimizes the log event semantic perception and event sequence pattern learning and combines them to achieve high-precision system log anomaly detection.

For ease of reading, all symbols used in this paper are listed in Table 1.

4. Anomaly Detection

The proposed LogBASA model has the self-attention encoder-decoder transformer architecture. To obtain the global semantic information on log sequences, this study employs the log parser Drain [39] to parse log events to obtain log templates and uses a pretrained BERT model to extract semantic information on events. In the encoder, a sequence of log event templates is encoded and used as input for semantic feature mapping. In the decoder, a special sequence token [SIGN] is added to the beginning of the input event template sequence to characterize the sequence state. Furthermore, through a combination of adaptive spatial boundary partitioning and reconstruction errors, the feature representations of [SIGN] and log event sequences are learned separately, and sequence prediction is realized by model training using unsupervised learning. In addition, by combining the global semantics of event template sequences with the system behavior analysis, the spatial distribution of normal log event sequences is obtained in multiple dimensions, which can improve the efficiency of anomaly detection. The LogBASA is described in detail in the following.

4.1. Semantic Awareness. Previous studies [41, 42] have shown that semantic information on log event sequences has a significant impact on the log anomaly detection effect. To prevent semantic confusion introduced by polysemous words and event changes from affecting the anomaly detection effect, this study extracts semantic information from log events using the superior-performance BERT model. This model is adopted to capture and learn similarities and differences between log events in a more efficient way. As described in [20], the original BERT model has two main functions in a specific downstream task: model fine-tuning and feature extraction. To enable an anomaly detection model to capture the textual semantic information on different log events accurately, this study performs feature extraction to generate event-corresponding contextual semantic representations using the BERT model as a substrate for the anomaly detection model.

The block diagram of the BERT model is presented in Figure 5. As shown in Figure 5, first, the log event template is divided into multiple tokens. The [CLS] and [SEP] tokens are added to the beginning and end of each sentence to mark the start and end positions of a sentence, respectively, which can be expressed as $e^{[\mathrm{CLS}]} = [\mathrm{CLS}]\mathrm{token}_1, \mathrm{token}_2, \ldots, \mathrm{token}_{|e|}[\mathrm{SEP}]$. In terms of semantic information extraction, semantic information on the special word [CLS] in the log event is directly extracted as an event representation and denoted by $\mathrm{sem}^{[\mathrm{CLS}]}$.

For a log event sequence $l = \{e_1, e_2, \ldots, e_{|l|}\}$, the corresponding event semantic sequence $X_e = \{\mathrm{sem}_1^{[\mathrm{CLS}]}, \mathrm{sem}_2^{[\mathrm{CLS}]}, \ldots, \mathrm{sem}_{|l|}^{[\mathrm{CLS}]}\}$ is generated by the BERT model, where $\mathrm{sem}_i^{[\mathrm{CLS}]}$ represents semantic information on event e_i.

4.2. System Behavior Analysis. To obtain a multidimensional, fine-grained system description of log data, this study characterizes each system function in a sequence at a fixed time as a system behavior. In the spatial dimension, a log event sequence is converted into a system functional path graph, and the GCN is used to extract the spatial structure features of the graph, which facilitates the graph representation learning of the subsequent anomaly detection model. In the temporal dimension, the timestamps of the log events are used to generate the time-difference sequences corresponding to the log event sequences to characterize the temporal correlation between events. Finally, the MLP model is employed for temporal feature extraction to facilitate the subsequent anomaly detection model to learn the temporal dependencies between logs.

4.2.1. Spatial Structural Features. For log sequences, a sequence pattern between log events represents an important discriminator of normal or abnormal behavior. That is, the sequential pattern between log events reflects the information of the positions of log events in a sequence. The sequence-based approaches exploit the positional structure of log events implicitly by sequentially processing a sequence (e.g., LogRobust [9]) or performing explicit positional encoding to augment the sequence representation (e.g., NeuralLog [43]). Unlike sequential data, there is no node order pattern in the directed graph because the nodes' positions are fixed. In this study, the degree matrix of the system function path map is used to realize structure-aware encoding of the graph using the GCN to enhance the interpretability of the graph representation and perform spatial feature extraction of the graph. In addition, the node's entry and exit degrees can intuitively reflect the local topology of graph nodes, which not only represent the importance of nodes in the graph but also reflect the similarity between the nodes and thus can be used to complement the semantic similarity between the nodes. The spatial feature extraction is defined as follows [44]:

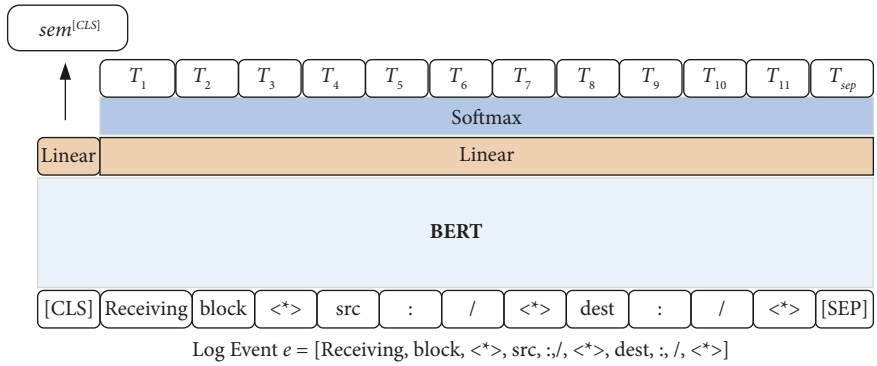

Figure 5: The block diagram of the BERT model.

$$H_g^{i+1} = \mathrm{ReLU}\left(\mathrm{GCNConv}(m,n)\right),$$
$$f_s = \mathrm{Pool}\left(H_g^i\left(H_g^{i-1}\right)\right), \quad i = 1,2, \tag{2}$$

where m and n denote the node semantic features and the node degree matrix of the graph, respectively, $\mathrm{GCNConv}(\cdot)$ represents the graph convolution network, $\mathrm{Pool}(\cdot)$ is the mean pooling operation, and the spatial structure features of the graph are expressed as $f_s = \left\{{fs_1}^d, {fs_2}^d, \ldots, {fs_{|l|}}^d\right\}$, where fs_i^d denotes the feature representation of node v_i and d is the feature dimension.

4.2.2. Temporal Relationship Features. The timestamps in log events denote a potential indicator for abnormal judgment. As shown in Figure 4, the analysis of timestamps in event sequences shows that the time difference of abnormal event sequences fluctuates significantly. These fluctuations can be caused by system performance problems, such as network congestion or system abnormalities. If there is a performance problem, the log events usually maintain the same sequence order as a normal sequence, so system anomalies commonly show a change in sequence order or a large fluctuation in time difference. For such performance problems, static feature extraction methods [45] have been typically used to discover performance defects, or intrusion detection has been performed to detect them [46]. However, complete extraction of sequence features is challenging and can increase the system load and affect the system's operation efficiency. Therefore, this study identifies possible anomalous problems in both spatial and temporal dimensions by extracting the temporal relationship features of log events and combining them with the graph space structural features, as explained in Section 4.2.1. The MLP model is used to extract the temporal relationship features of event sequences as follows:

$$\mathrm{MLP} = \sum_{i=1}^{|\Delta T|} \Delta t_i w_{ij} + b_j,$$
$$f_t = \mathrm{Sigmod}\left(\mathrm{MLP}(\Delta T)\right), \tag{3}$$

where ΔT is the time-difference sequence of a log event sequence, Δt_i is the ith difference in the time-difference sequence, w_{ij} and b_j are the weight and bias of the j th hidden layer, respectively, and they are used to obtain the temporal relationship feature $f_t = \left\{{ft_1}^d, {ft_2}^d, \ldots, {ft_{|l|}}^d\right\}$ of the log event sequence, ${ft_i}^d$ denotes the feature representation of the ith time difference Δt_i, and d denotes the feature dimension.

4.3. Self-Attention Encoder-Decoder Transformer Structure. In recent years, the self-attention encoder-decoder transformer structure has been proven to be very effective in various NLP tasks [20, 47]. Compared to recurrent neural network models, such as LSTM, transformer structures can efficiently handle long sequences of input and output data, which represents an advantage when encoding complex log sequence patterns and dealing with high-dimensional and multilevel log event semantics. In addition, to improve a model's semantic-aware performance, a multiheaded attention mechanism has often been used. In this paper, the LogBASA model adopts the encoder with the self-attention encoder-decoder transformer structure to encode event semantic sequences into semantic mappings and fuse the spatiotemporal correlation features of log event sequences, as presented in Section 4.2, which are further used as key and value inputs. In addition, anomaly detection is performed by outputting the semantic representation vector and sequence pattern mapping of the special word [SIGN] from the decoder.

4.3.1. Semantic Awareness Encoder. An encoder with a multilayer encoder structure based on self-attention is adopted to sense the semantic relevance of log sequences from a global perspective, as shown in Figure 6. For a log event sequence $l = \left\{e_1, e_2, \ldots, e_{|l|}\right\}$, the event semantic representation sequence $X_e = \left\{\mathrm{sem_1}^{[\mathrm{CLS}]}, \mathrm{sem_2}^{[\mathrm{CLS}]}, \ldots, \mathrm{sem_{|l|}}^{[\mathrm{CLS}]}\right\}$, where $\mathrm{sem_i}^{[\mathrm{CLS}]}$ is the semantic representation of the start word [CLS] corresponding to event e_i, is obtained by the pretrained BERT model. The self-attention mechanism [47] is employed to capture the semantic features in the sequence of semantic representations of events as follows:

$$\mathrm{Attention}(Q,K,V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{4}$$

where Q, K, and V denote the query, key, and value matrices, respectively, and d_k is the dimensionality coefficient.

Each encoder self-attentive module is connected to a fully connected feedforward neural network (FFN), which consists of two layers with linear transformation functions with a ReLU-activation-function layer between them, which can be expressed as follows:

$$\mathrm{FFN}(x) = \mathrm{ReLU}\left(xW_1 + b_1\right)W_2 + b_2, \tag{5}$$

where x denotes the semantic representation of the attention module output and W_1, W_2, b_1,and b_2 are the weight parameters and biases of the first and second FFN layers, respectively.

The final encoder output provides the semantic feature mapping Y_e of an event sequence, which is then fused with the spatiotemporal correlation features of the log event sequence in the system behavior analysis and used as the key and value inputs of the encoder-decoder attention module in the decoder.

4.3.2. Anomaly Detection Decoder. In this study, a log event sequence anomaly prediction decoder is designed based on global semantic awareness and system behavior analysis. Adaptive anomaly boundaries with sequence reconstruction are used as training objectives. Unlike the encoder, the decoder uses the multihead attention module to improve semantic perception performance, which can be expressed as

$$\begin{aligned}\mathrm{MultiHead}(Q,K,V) &= \mathrm{Concat}\left(\mathrm{head}_1, \mathrm{head}_2, \ldots, \mathrm{head}_h\right)W^O,\\ \text{where head}_i &= \mathrm{Attention}\left(QW_i^Q, KW_i^K, VW_i^V\right),\end{aligned} \tag{6}$$

where $W_i{}^Q$, $W_i{}^K$, and $W_i{}^V$ are the linear projection weights of an attention head_i. W^O is the projection matrix.

As shown in Figure 6, each decoder layer includes the encoder-decoder attention module and FFN module. The decoder input is similar to that of the encoder, and a special word [SIGN] is added to the beginning of a sequence to capture the semantic features of the whole time sequence based on the semantic representation sequence of events X_e to obtain the complete decoder input X_d, which is defined by

$$X_d = \left[\mathrm{sem}^{[\mathrm{SIGN}]}\right] \| \left[X_e\right]_{\mathrm{dim}=1}, \tag{7}$$

where $[\bullet] \| [\bullet]_{\mathrm{dim}=1}$ denotes the splicing of two matrices in the column dimension.

Next, the spatiotemporal correlation features of the event sequences are fused, as explained in Section 4.2, with the semantic feature mappings generated by the semantic-aware encoder and used as key and value inputs to the encoder-decoder attention module. The LogBASA model can capture the patterns and global semantic correlations of event sequences represented by the node topology through the attention mechanism as follows:

$$F_i = \sum_{i=1}^{|I|} y_e^i + f_s^i + f_t^i, \tag{8}$$

where $y_e{}^i \in Y_e$, $f_s{}^i \in f_s$, $f_t{}^i \in f_t$, and $F = \left\{F_1, F_2, \ldots, F_{|I|}\right\}$ denote the new event sequence feature representation.

$$H_d^{i+1} = \mathrm{FFN}\left(\mathrm{Attention}\left(\mathrm{MultiHead}\left(H_d^i\right), F, F\right)\right). \tag{9}$$

The encoder-decoder attention module of the last decoder layer is followed by a fully connected FFN, the same as the encoder, and the final output of the decoder is finally obtained as follows:

$$Y = \left[Y^{[\mathrm{SIGN}]}\right] \| \left[Y_d\right]_{\mathrm{dim}=1}. \tag{10}$$

To ensure that the model can learn inherent differences between normal and abnormal log samples, this paper defines an adaptive boundary with a sequence reconstruction loss function. As a result, the model can map normal log events to adjacent positions in the implicit space, where normal samples are close to each other, thus differentiating normal from abnormal samples. The semantic mappings of log event sequences are obtained from the decoder output of the added [SIGN] tokens. It is assumed that the spatial mappings of all normal log event sequences are closely distributed, forming a hypersphere. The goal of introducing the adaptive boundary is to form a spatially compact hypersphere that can contain semantic mappings of all normal event sequences. In the testing phase, any sequence that is mapped by the model outside the hypersphere boundary is denoted as anomalous. The proposed adaptive boundary loss function L_{boundary} is defined as follows:

$$\begin{aligned}C &= \mathrm{Mean}\left(\sum_{i=1}^{|S|} Y_i^{[\mathrm{SIGN}]}\right),\\ L_{\mathrm{boundary}} &= \underset{\Phi}{\mathrm{argmin}} \sum_{i}^{N} \left\| Y_i^{[\mathrm{SIGN}]} - C \right\|^2,\end{aligned} \tag{11}$$

where $Y_i^{[\mathrm{SIGN}]}$ is the decoder output representation of the feature character [SIGN] for the ith log event sequence, C is the center of all decoder outputs for the special word [SEQ] in the training set S, and $\mathrm{Mean}(\cdot)$ represents the mean value.

The goal of sequence reconstruction is to learn the sequence pattern of normal log events, where the encoder first encodes the input data in a low-dimensional space and then tries to reconstruct them to their original form by means of a decoder. Therefore, if the input samples are difficult to reconstruct, namely, if they generate large reconstruction

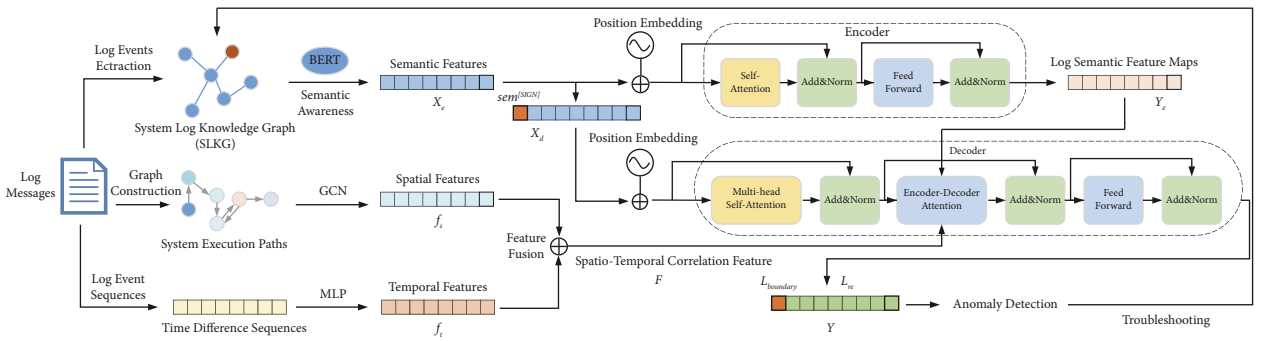

Figure 6: The LogBASA architecture.

errors, they are considered anomalous. The sequence reconstruction loss function L_{re} is defined as follows:

$$L_{re} = \underset{\Phi}{\operatorname{argmin}} \frac{1}{|Y_d|} \sum_{i=1}^{|Y_d|} \left\| y_d^i - y_e^i \right\|^2, \tag{12}$$

where $y_d{}^i$ and $y_e{}^i$ are the ith events of the encoder output Y_e and the decoder output Y_d, respectively.

The proposed adaptive boundary and sequence reconstruction loss function L is expressed by

$$L = \lambda L_{\text{boundary}} + (1 - \lambda) L_{re}, \tag{13}$$

where λ is the weight parameter that expresses the balance of the adaptive boundary error L_{boundary} and the reconstruction error L_{re}.

In the training process of the anomaly detection decoder, the anomaly decision boundary R is continuously updated by calculating the difference between $Y^{[\text{SIGN}]}$ and C. The trained R is used to determine whether the new log event sequence is anomalous. If the decoder of the special character [SIGN] of the log event sequence indicates that the difference between $Y^{[\text{SIGN}]}$ and C is less than R, the new log event sequence is determined to be normal, and the label y is represented by the value zero. Otherwise, the new log event sequence is considered abnormal, and the label y is represented by the value of one. The determination method is defined by

$$y = \begin{cases} 0, \text{if } Y^{[\text{SIGN}]} - C < R, \\ 1, \text{if } Y^{[\text{SIGN}]} - C \geq R. \end{cases} \tag{14}$$

5. Evaluation Results

5.1. Experimental Datasets. In this paper, three popular and publicly available datasets, the HDFS, BGL, and Thunderbird datasets, were used to evaluate the proposed approach and related baseline methods. These datasets have been widely used in log analyses [9, 14, 16, 17, 48], and they were constructed from real-world data and tagged manually by system administrators or using system-generated alert tags. The three datasets were obtained from publicly available websites. The statistical information on the three datasets is given in Table 2.

(1) HDFS [29]: The HDFS dataset was generated in the Hadoop-based MapReduce cloud environment using a benchmark workload and manually flagged based on the hand-developed rules to identify exceptions. In this dataset, logs are sliced into tracks based on the block ID, and logs with the same block ID identifier are arranged in order of execution time to form a log sequence. This dataset includes 575,061 log sequences synthesized from 11,175,629 logs. Hadoop domain experts flagged the log sequences as normal or abnormal, and 558,223 log sequences (97.1%) were regarded as normal, while the remaining 16,838 log sequences (2.9%) were denoted as abnormal.

(2) BlueGene/L [49]: The BGL is an open-log dataset collected from the BlueGene/L supercomputer system at Lawrence Livermore National Laboratory (LLNL) in Livermore, California, with 131,072 processes and 32,768 GB of memory. The logs in this dataset contain both alarm and nonalarm messages identified by alarm category tags. This dataset includes 4,747,963 logs, of which 348,460 logs denote exception logs. Unlike the HDFS dataset, the BGL logs do not contain specific identifiers, so in this paper, a fixed window was used to divide the BGL dataset into log sequences of different lengths for subsequent experiments. In the divided log sequences, if there was an abnormal log entry, the log sequence was considered abnormal.

(3) Thunderbird [49]: The Thunderbird is an open-log dataset collected from the Thunderbird supercomputer system at Sandia National Laboratories (SBNL) in Albuquerque, with 9,024 processors and 27,072 GB of memory. Tagging information was obtained from alert and nonalert messages identified by alert category tags. This study sampled 20,000,000 log messages from the original Thunderbird dataset to compose the dataset for experimental verification, of which 758,562 log messages denote anomalies. Similar to the BGL dataset, a fixed window was used to divide the log data into sequences with different lengths for training.

Table 2: Statistical information on the three datasets used in this study.

	HDFS	BGL	Thunderbird
Duration	38.7 hours	214.7 days	244 days
Total log number	11175629	4747963	20000000
Template number/node number	48	127	4992
Total sequence number/map number	575061	18002	99593
Abnormal sequence number/map number	16838	1205	843
Window type	Session window	Fixed window	Fixed window

5.2. Experimental Setup. The LogBASA model training was performed on a PC with 12 GB RAM and an i5-9500 processor with six cores and six threads, using Python 3.8.12 and the PyTorch 1.13.0 development environment for script development and programming. The number of encoder and decoder layers in the LogBASA model was set to six, the number of attention heads was eight, the size of the feed-forward neural network was 2,048, and the hidden-layer dimensions of the GCN and MLP models were 32 and 64, respectively. The LogBASA was trained using the Adam optimizer, after repeated tests, the learning rates of the graph structure coding training and anomaly detection training were set to 0.01 and 0.0001, and the cross entropy and mean square error were selected as loss functions, respectively.

5.2.1. Baseline Methods. To evaluate the performance of the proposed LogBASA model, it was compared with several advanced baseline algorithms. It should be noted that only normal log sequences were used in all experiments during the training phase.

The comparison models were as follows:

(1) Methods based on pattern recognition and similarity comparison:

   (a) PCA [29]: this method is based on the principal component analysis (PCA) that extracts the major components of the input sequence and discriminates test sequences that are above a predefined threshold as anomalous sequences.
   (b) Isolation forest [11]: this is a simple and effective unsupervised machine learning-based approach based on a tree-based model.
   (c) LogCluster [12]: this is a cluster-based approach that divides log sequences into normal and abnormal clusters based on the template count vector.

(2) Content-independent methods using only the log template index as input data:

   (a) DeepLog [16]: this is a stacked LSTM-based model that predicts the next log template index based on the sequence features within the observation window.
   (b) LogBERT [17]: this is an anomaly detection method based on the BERT model that trains the log template index sequences using the masked language model loss and single-class objectives.
   (c) OC4Seq [35]: this is a multiscale anomaly detection model based on the RNN model, which integrates anomaly detection targets with the RNN output to detect anomalies in the potential space.

(3) Semantic awareness methods:

   (a) LogAnomaly [19]: this is a model based on the Bi-LSTM and attention mechanisms that adopts an improved template embedding method to predict log templates.
   (b) AutoEncoder [50]: this is a self-encoder-based approach that embeds log sequences in the latent space and uses isolated forests (iForest) for anomaly detection.

5.2.2. Evaluation Metrics. Anomaly detection is a binary classification task, and its performance is typically evaluated using precision, recall, and F1-Score evaluation metrics. These metrics are calculated using the following variables: true positive (TP), false positive (FP), false negative (FN), and true negative (TN), which denote the number of abnormal sequences detected by a model, the number of normal sequences misclassified as abnormal by the model, the number of abnormal sequences misclassified as normal by the model, and the number of normal sequences detected by the model, respectively.

The specific calculation expressions of the three evaluations metrics are as follows:

(1) Precision: this metric denotes the percentage of correctly detected anomaly log sequences by a model among all detected anomaly log sequences, and it is calculated by

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{15}$$

(2) Recall: this metric represents the percentage of log sequences that are correctly identified as exceptions among all true exceptions, and it is calculated as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{16}$$

(3) $F1 - \text{Score}(F_1)$: this metric denotes the summed average of precision and recall, and its value is obtained by

$$F_1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{17}$$

5.3. Experimental Results

5.3.1. Training Efficiency Evaluation. To verify the superior performance of the proposed LogBASA, this section presents the comparison results of training states of the DeepLog, LogAnomaly (both based on the LSTM model), LogBERT (similar to the BERT model), AutoEncoder, and the proposed LogBASA model. The number of epochs required by different models to reach the convergence state during the training process is presented in Figure 7. In this experiment, all models were trained on the HDFS dataset. The results indicated that the proposed LogBASA model was the fastest method among all models in reaching the convergence state, but it had a higher loss than the other models due to the higher model complexity since it integrated the GCN, MLP, and transformer models and a large number of parameters. The results showed that the proposed LogBASA model reached convergence after 13 epochs, while the second-fastest model regarding reaching convergence was the LogAnomaly model, which reached convergence after 30 epochs. The results demonstrated that the proposed model had the highest training efficiency among all models, despite its more complex structure.

5.3.2. Effects of Weight and Sequence Length on Model Performance. The experiments were conducted for different weights λ and sequence lengths to test the robustness of the proposed model and evaluate its performance in different situations.

(1) Objective Function. The performance comparison results for different λ values are shown in Figure 8. In the experiments on the HDFS dataset, due to the short length of log event sequences in this dataset, the importance of the adaptive boundary objective function was higher, and the model performance was optimal at $\lambda = 0.8$. In the experiments on the BGL dataset, the performance of the proposed model was more stable, which was mainly because the anomalous event sequences were mostly long and uniformly distributed, and the sequence reconstruction objective function dominated the training process. In this experiment, the proposed model's performance was optimal for $\lambda = 0.3$. In the experiments on the Thunderbird dataset, since the anomaly rate of log data in this dataset was low (0.85%) and most of the log sequences were long, like those in the BGL dataset, the adaptive boundary and the sequence reconstruction objective function reached the equilibrium state at $\lambda = 0.5$, and the model performance was optimal at $\lambda = 0.5$. The results indicated that the proposed objective function could effectively improve the model's performance.

(2) Sequence Length. Additional experiments were conducted to investigate whether the proposed LogBASA model can work stably under different window settings. Considering the specificity of the HDFS datasets, the experiments were conducted only on the BGL and Thunderbird datasets. The experiments were performed for nine window sizes, ranging from 20 to 100. The maximum sequence length was set to 100 because it was considered that setting a larger window could affect the system response efficiency and lead to delayed detection of anomalies, and a larger window is unlikely to be adopted in practical application scenarios. The experimental results are shown in Figure 9.

As shown in Figure 9, the performance of the proposed LogBASA model was stable for all sequence lengths on the BGL dataset. However, on the Thunderbird dataset, the model performance tended to increase significantly as the window size decreased. This could be due to the increase in the number of anomalies caused by the decrease in the window size, which allowed the model to distinguish between normal and abnormal log sequences more efficiently, thus improving the anomaly detection performance.

5.3.3. Overall Performance. To evaluate the overall performance of the proposed LogBASA method based on the system behavior analysis and global semantic awareness, first, the LogBASA method was compared with the common methods for log sequence anomaly detection, including the template count vector-based methods (the PCA, isolation forest, and LogCluster methods), template-based prediction methods (the DeepLog, LogBERT, and OC4Seq methods), and semantic awareness-based methods (the LogAnomaly and AutoEncoder methods). The experimental results are shown in Figures 10–12.

The experimental results indicated that the overall performance of the proposed LogBASA method outperformed that of the comparison methods. The template count vector-based methods had neither high accuracy nor high recall. These methods completely ignored the sequential relationships between events despite a certain reflection on the pattern characteristics of log event sequences, which resulted in a relatively high false-negative rate and a low recall. Only the isolation forest method had a recall of 0.727. The template-based prediction methods performed obviously better than the template count vector-based methods. The DeepLog, LogAnomaly, and OC4Seq methods all used the LSTM or RNN models to construct the anomaly detection network, but this sequential network could not effectively mine the dependency relationship between longer event sequences because each forward transmission was performed based only on the processing result of the previous time step. Although the LogBERT method based on the transformer model did not perform the best regarding all indexes, it had stable performance.

Furthermore, the LogAnomaly and AutoEncoder methods consistently outperformed the DeepLog method, which demonstrated the benefits of considering semantic content information in event sequence anomaly detection, as well as the excellent performance of the transformer model in sequence pattern learning, to a certain extent. However, these methods still had certain limitations in the log event sequence anomaly detection task.

Finally, the proposed LogBASA method outperformed the baseline methods in terms of precision, recall, and F1-Score metrics on both HDFS and BGL datasets. The LogBASA performed poorly on the Thunderbird dataset

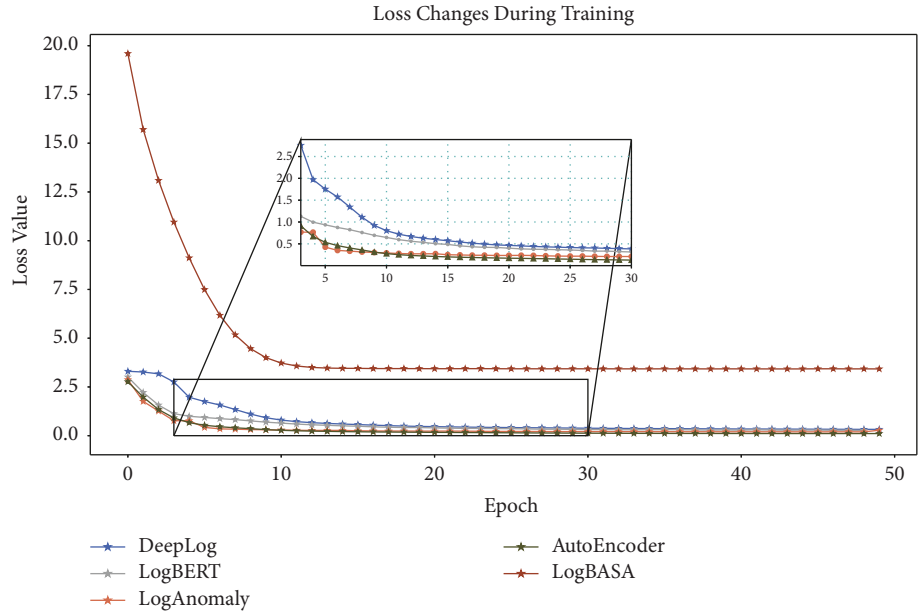

Figure 7: The changes in the loss function value during the model training process.

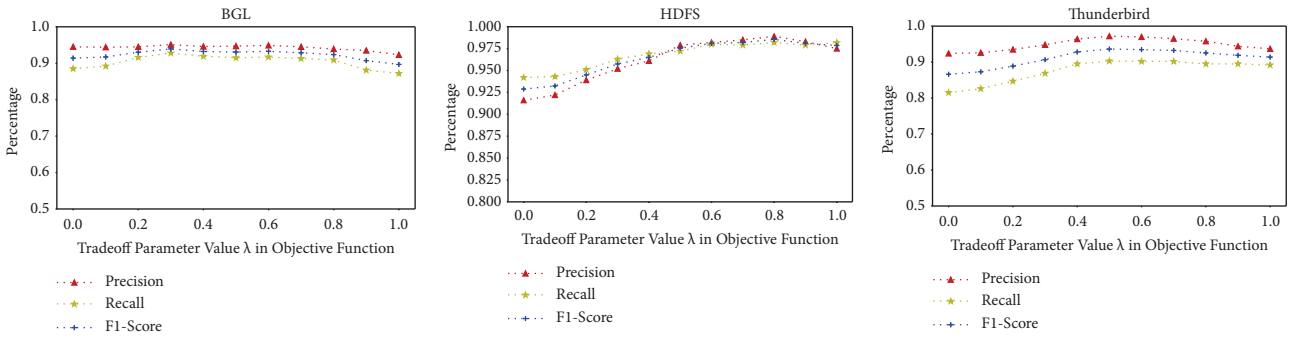

Figure 8: Performance comparison results for different λ values. (a) HDFS. (b) BGL. (c) Thunderbird.

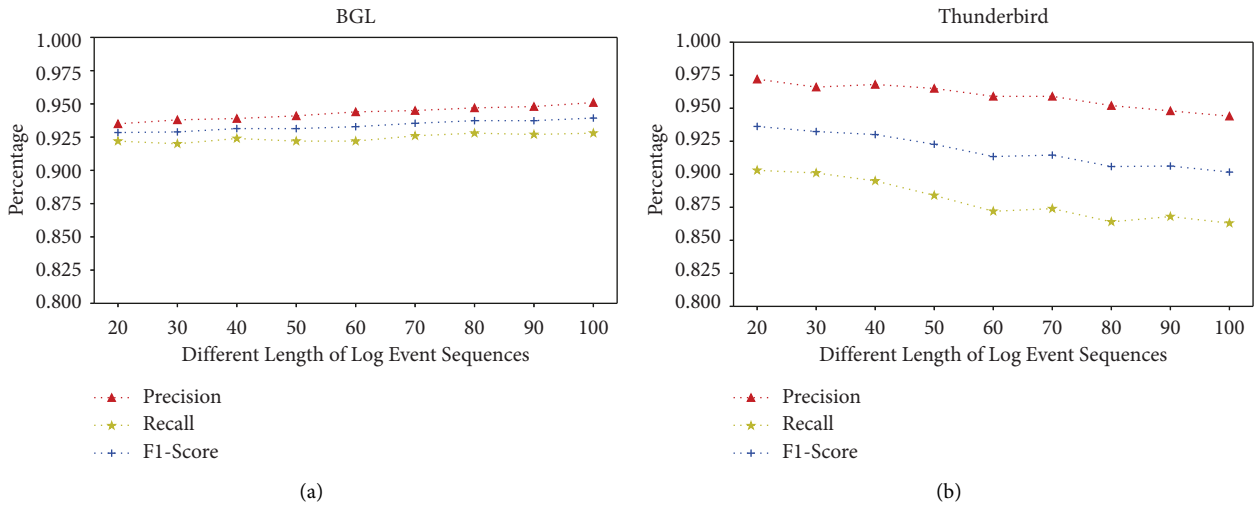

Figure 9: Performance comparison results for different sequence length values. (a) BGL. (b) Thunderbird.

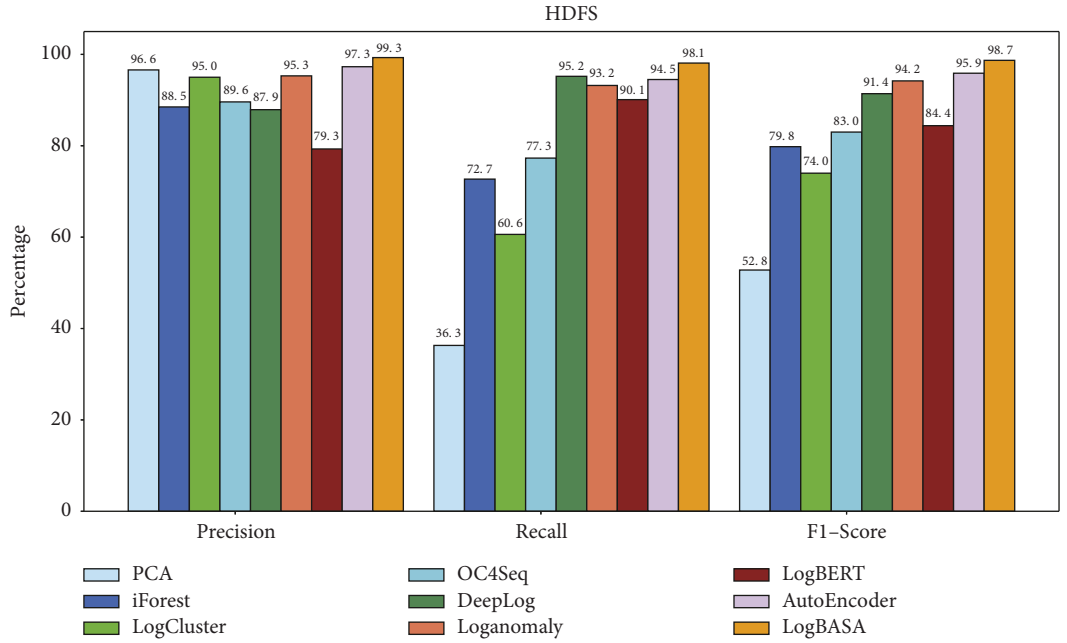

Figure 10: Performance comparison results on the HDFS dataset.

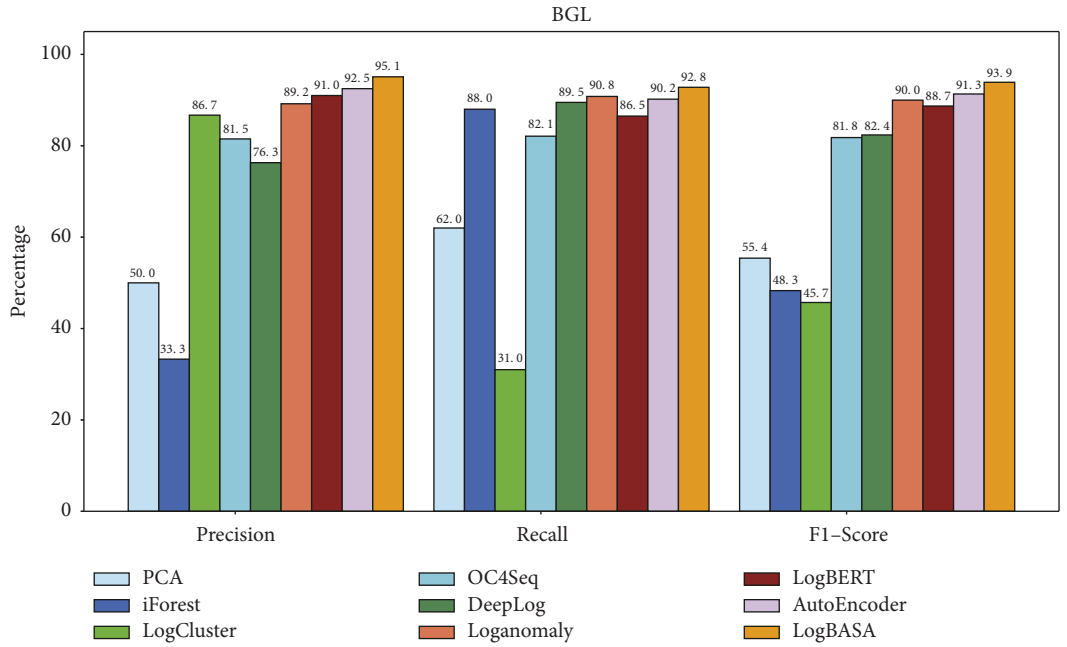

Figure 11: Performance comparison results on the BGL dataset.

only because the model complexity was too high for the Thunderbird dataset to easily learn noisy and irrelevant features, which resulted in the model's inaccurate judgment of normal data, which further led to the high rate of missed reports. In the LogBASA, the GCN and MLP models were used to capture local spatial feature information and global temporal feature information on the log event sequences, respectively, and the self-attention encoder-decoder transformer model was used to learn the sequence patterns and semantic correlations in the whole log sequence. By combining the abovementioned improvements and optimizations, the LogBASA achieved excellent performance in log event sequence anomaly detection tasks.

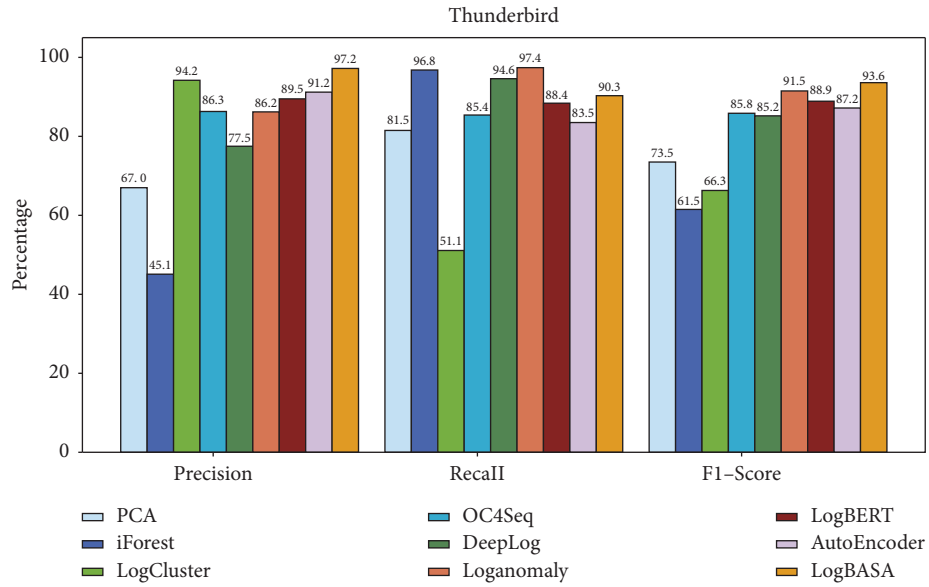

Figure 12: Performance comparison results on the Thunderbird dataset.

6. Conclusion

To address the problems of poor feature recognition and unstable performance in the existing sequence pattern recognition methods and content-aware methods, this study proposes a log anomaly detection method named LogBASA, which is based on system behavior analysis and global semantic awareness. First, log event sequences are modeled as system functional path diagrams to characterize system behavior, and the GCN and MLP models are combined to perform the spatiotemporal correlation analysis of the system behavior, obtaining comprehensive event sequence patterns and generating spatiotemporal feature representations of event sequences. Then, a self-attention encoder-decoder transformer-based architecture is designed. In the encoder module, event semantic sequences are used as input and encoded into semantic feature mappings. In the decoder module, the special word [SIGN] is added to the beginning of a sequence to characterize the log event sequence state, and adaptive boundaries and sequence reconstruction objective functions are used in the anomaly detection task. The proposed LogBASA model provides a high-performance solution for handling large-scale, unstructured, and multi-level log event data. Finally, the proposed method is verified by extensive experiments on HDFS, BGL, and Thunderbird datasets, and the experimental results demonstrate that the LogBASA model outperforms common methods in system log anomaly detection, achieving accuracy rates of 99.3%, 95.1%, and 97.2% on the three datasets.

Data Availability

The log data used to support the findings of this study have been deposited in the GitHub repository (https://github.com/logpai/loghub).

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This study was supported in part by the National Natural Science Foundation of China (nos. 61902080, 61972104, 62002072, and 61702120), National Key Research and Development Program of China (nos. 2019YFB1804403 and 2018YFB1802200), Special Projects in Key Areas of Guangdong Province (no. 2019B010118001), Science and Technology Project in Guangzhou (no. 201803010081), Foshan Science and Technology Innovation Project, China (no. 2018IT100283), Guangdong Provincial Key Laboratory Project of Intellectual Property and Big Data (no. 2018B030322016), Guangzhou Key Laboratory (no. 202102100006), Science and Technology Program of Guangzhou, China (nos. 202002020035 and 202102021078), Research Projects in Guangdong Province (no. 2021ZDJS026), and Guangdong Provincial University Key Field Special Project (no. 2021ZDZX1031).

References

[1] Y. Xie and K. Yang, "Domain adaptive log anomaly prediction for Hadoop system," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 20778–20787, 2022.

[2] S. M. Hosseini Bamakan, H. Wang, and Y. Shi, "Ramp loss K-Support Vector Classification-Regression; a robust and sparse multi-class approach to the intrusion detection problem," *Knowledge-Based Systems*, vol. 126, pp. 113–126, 2017.

[3] R. Gandhi, A. Sharma, W. Mahoney, W. Sousan, Q. Zhu, and P. Laplante, "Dimensions of cyber-attacks: cultural, social, economic, and political," *IEEE Technology and Society Magazine*, vol. 30, no. 1, pp. 28–38, 2011.

[4] S. Lu, X. Wang, and L. Mao, "Network security situation awareness based on network simulation," in *Proceedings of the 2014 IEEE Workshop on Electronics, Computer and Applications*, pp. 512–517, Ottawa, Canada, May 2014.

[5] B. Watkins, "The impact of cyber attacks on the private sector," *Briefing Paper, Association for International Affair*, vol. 12, pp. 1–11, 2014.

[6] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: robust anomaly detection and localization for interleaved unstructured logs," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 4, pp. 2762–2780, 2023.

[7] H. Mi, H. Wang, Y. Zhou, M. R. T. Lyu, and H. Cai, "Toward fine-grained, unsupervised, scalable performance diagnosis for production cloud computing systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 6, pp. 1245–1255, 2013.

[8] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–37, 2021.

[9] X. Zhang, Y. Xu, Q. Lin et al., "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pp. 807–817, Tallinn, Estonia, August 2019.

[10] S. Kabinna, C. P. Bezemer, W. Shang, M. D. Syer, and A. E. Hassan, "Examining the stability of logging statements," *Empirical Software Engineering*, vol. 23, no. 1, pp. 290–333, 2018.

[11] F. T. Liu, K. M. Ting, and Z. H. Zhou, "Isolation forest," in *Proceedings of the 2008 eighth ieee international conference on data mining*, pp. 413–422, Pisa, Italy, December 2008.

[12] Q. Lin, H. Zhang, J. G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference on Software Engineering Companion*, pp. 102–111, Austin, TX, USA, May 2016.

[13] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001.

[14] J. G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu, "Mining program workflow from interleaved traces," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 613–622, Washington, DC, USA, July 2010.

[15] S. Lu, X. Wei, Y. Li, and L. Wang, "Detecting anomaly in big data system logs using convolutional neural network," in *Proceedings of the 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pp. 151–158, Athens, Greece, August 2018.

[16] M. Du, F. Li, G. Zheng, and V. Srikumar, "Deeplog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1285–1298, Dallas, Texas, USA, October 2017.

[17] H. Guo, S. Yuan, and X. Wu, "Logbert: log anomaly detection via bert," in *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Shenzhen, China, July 2021.

[18] W. Meng, Y. Liu, Y. Huang et al., "A semantic-aware representation framework for online log analysis," in *Proceedings of the 2020 29th International Conference on Computer Communications and Networks (ICCCN)*, pp. 1–7, Honolulu, HI, USA, August 2020.

[19] W. Meng, Y. Liu, Y. Zhu et al., "LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs," *IJCAI*, vol. 19, pp. 4739–4745, 2019.

[20] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: pre-training of deep bidirectional transformers for language understanding," 2018, https://arxiv.org/abs/1810.04805.

[21] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in ibm bluegene/l event logs," in *Proceedings of the IEEE International Conference on Data Mining (ICDM 2007)*, pp. 583–588, Omaha, NE, USA, October 2007.

[22] B. Debnath, M. Solaimani, M. A. G. Gulzar et al., "LogLens: a real-time log analysis system," in *Proceedings of the 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1052–1062, Vienna, Austria, July 2018.

[23] I. Beschastnikh, Y. Brun, M. D. Ernst, and A. Krishnamurthy, "Inferring models of concurrent systems from logs of their behavior with CSight," in *Proceedings of the 36th International Conference on Software Engineering*, pp. 468–479, Hyderabad, India, May 2014.

[24] M. Chen, A. X. Zheng, J. Lloyd, M. I. Jordan, and E. Brewer, "Failure diagnosis using decision trees," in *Proceedings of the International Conference on Autonomic Computing, 2004. Proceedings*, pp. 36–43, New York, NY, USA, May 2004.

[25] T. Qin, Y. Gao, L. Wei, Z. Liu, and C. Wang, "Potential threats mining methods based on correlation analysis of multi-type logs," *IET Networks*, vol. 7, no. 5, pp. 299–305, 2018.

[26] P. Bodik, M. Goldszmidt, A. Fox, D. B. Woodard, and H. Andersen, "Fingerprinting the datacenter: automated classification of performance crises," in *Proceedings of the 5th European conference on Computer systems*, pp. 111–124, Paris, France, April 2010.

[27] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: system log analysis for anomaly detection," in *Proceedings of the 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)*, pp. 207–218, Ottawa, Canada, October 2016.

[28] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Online system problem detection by mining patterns of console logs," in *Proceedings of the 2009 Ninth IEEE International Conference on Data Mining*, pp. 588–597, Miami Beach, FL, USA, December 2009.

[29] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, pp. 117–132, Big Sky, MT, USA, October 2009.

[30] L. Feremans, V. Vercruyssen, B. Cule, W. Meert, and B. Goethals, "Pattern-based anomaly detection in mixed-type time series," in *Proceedings of the Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2019*, pp. 240–256, Springer, Würzburg, Germany, September 2020.

[31] R. Vaarandi, "A data clustering algorithm for mining patterns from event logs," in *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764)*, pp. 119–126, Kansas City, MO, USA, October 2003.

[32] Z. Wang, J. Tian, H. Fang, L. Chen, and J. Qin, "LightLog: a lightweight temporal convolutional network for log anomaly detection on the edge," *Computer Networks*, vol. 203, Article ID 108616, 2022.

[33] A. Brown, A. Tuor, B. Hutchinson, and N. Nichols, "Recurrent neural network attention mechanisms for interpretable system log anomaly detection," in *Proceedings of the First Workshop on Machine Learning for Computing Systems*, pp. 1–8, Tempe, AZ, USA, June 2018.

[34] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated IT system failure prediction: a deep learning approach," in *Proceedings of the 2016 IEEE International Conference on Big Data (Big Data)*, pp. 1291–1300, Washington, DC, USA, December 2016.

[35] Z. Wang, Z. Chen, J. Ni, H. Liu, H. Chen, and J. Tang, "Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection," in *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pp. 3726–3734, Singapore, August 2021.

[36] S. R. Wibisono and A. I. Kistijantoro, "Log anomaly detection using adaptive universal transformer," in *Proceedings of the 2019 International Conference of Advanced Informatics: Concepts, Theory and Applications (ICAICTA)*, pp. 1–6, Yogyakarta, Indonesia, September 2019.

[37] S. Huang, Y. Liu, C. Fung et al., "Hitanomaly: hierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.

[38] Y. Wan, Y. Liu, D. Wang, and Y. Wen, "Glad-paw: graph-based log anomaly detection by position aware weighted graph attention network," in *Proceedings of the Pacific-asia conference on knowledge discovery and data mining*, pp. 66–77, Springer, Berlin, Germany, May 2021.

[39] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: an online log parsing approach with fixed depth tree," in *Proceedings of the 2017 IEEE International Conference on Web Services (ICWS)*, pp. 33–40, Honolulu, HI, USA, June 2017.

[40] Y. Li, N. Du, and S. Bengio, "Time-dependent representation for neural event sequence prediction," 2017, https://arxiv.org/abs/1708.00065.

[41] P. Ryciak, K. Wasielewska, and A. Janicki, "Anomaly detection in log files using selected natural language processing methods," *Applied Sciences*, vol. 12, no. 10, p. 5089, 2022.

[42] V. H. Le and H. Zhang, "Log-based anomaly detection with deep learning: how far are we?" in *Proceedings of the 44th International Conference on Software Engineering*, pp. 1356–1367, Pittsburgh, PA, USA, May 2022.

[43] V. H. Le and H. Zhang, "Log-based anomaly detection without log parsing," in *Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pp. 492–504, Melbourne, Australia, November 2021.

[44] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, https://arxiv.org/abs/1609.02907.

[45] C. Lou, P. Huang, and S. Smith, "Understanding, detecting and localizing partial failures in large system software," in *Proceedings of the 17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pp. 559–574, Santa Clara, CA, USA, February 2020.

[46] D. Aksu and M. A. Aydin, "MGA-IDS: optimal feature subset selection for anomaly detection framework on in-vehicle networks-CAN bus based on genetic algorithm and intrusion detection approach," *Computers & Security*, vol. 118, Article ID 102717, 2022.

[47] A. Vaswani, N. Shazeer, and N. Parmar, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[48] Z. Chen, J. Liu, W. Gu, Y. Su, and M. R. Lyu, "Experience report: deep learning-based system log analysis for anomaly detection," 2021, https://arxiv.org/abs/2107.05908.

[49] A. Oliner and J. Stearley, "What supercomputers say: a study of five system logs," in *Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)*, pp. 575–584, Edinburgh, UK, June 2007.

[50] A. Farzad and T. A. Gulliver, "Unsupervised log message anomaly detection," *ICT Express*, vol. 6, no. 3, pp. 229–237, 2020.