Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.051620

ARTICLE

# Enhancing Log Anomaly Detection with Semantic Embedding and Integrated Neural Network Innovations

**Zhanyang Xu*, Zhe Wang, Jian Xu, Hongyan Shi and Hong Zhao**

School of Computer and Software, Nanjing University of Information Science and Technology, Nanjing, 210044, China

*Corresponding Author: Zhanyang Xu. Email: zhanyang_xu@nuist.edu.cn

Received: 11 March 2024 Accepted: 14 July 2024 Published: 12 September 2024

## ABSTRACT

System logs, serving as a pivotal data source for performance monitoring and anomaly detection, play an indispensable role in assuring service stability and reliability. Despite this, the majority of existing log-based anomaly detection methodologies predominantly depend on the sequence or quantity attributes of logs, utilizing solely a single Recurrent Neural Network (RNN) and its variant sequence models for detection. These approaches have not thoroughly exploited the semantic information embedded in logs, exhibit limited adaptability to novel logs, and a single model struggles to fully unearth the potential features within the log sequence. Addressing these challenges, this article proposes a hybrid architecture based on a multiscale convolutional neural network, efficient channel attention and mogrifier gated recurrent unit networks (LogCEM), which amalgamates multiple neural network technologies. Capitalizing on the superior performance of robustly optimized BERT approach (RoBERTa) in the realm of natural language processing, we employ RoBERTa to extract the original word vectors from each word in the log template. In conjunction with the enhanced Smooth Inverse Frequency (SIF) algorithm, we generate more precise log sentence vectors, thereby achieving an in-depth representation of log semantics. Subsequently, these log vector sequences are fed into a hybrid neural network, which fuses 1D Multi-Scale Convolutional Neural Network (MSCNN), Efficient Channel Attention Mechanism (ECA), and Mogrifier Gated Recurrent Unit (GRU). This amalgamation enables the model to concurrently capture the local and global dependencies of the log sequence and autonomously learn the significance of different log sequences, thereby markedly enhancing the efficacy of log anomaly detection. To validate the effectiveness of the LogCEM model, we conducted evaluations on two authoritative open-source datasets. The experimental results demonstrate that LogCEM not only exhibits excellent accuracy and robustness, but also outperforms the current mainstream log anomaly detection methods.

## KEYWORDS

Deep learning; log analysis; anomaly detection; natural language processing

## 1 Introduction

As cloud computing and big data technologies advance, traditional applications migrate to cloud platforms, offering users convenience and scalability [1]. This migration means that these services and systems can now be extensively accessed through the Internet, thereby providing users with unprecedented convenience and scalability. However, this transition brings challenges, including

security threats. Global cloud service providers, such as Microsoft, Amazon, Google, Alibaba, and Tencent, face the task of providing uninterrupted services to millions worldwide, requiring high reliability and availability. Cloud service interruptions disrupt basic services, affecting all applications and potentially halting enterprises. For example, a network upgrade error caused a significant AWS outage in 2011, resulting in substantial losses. Logs, recording key events and information during runtime, are crucial for system management. They are often the sole data source for troubleshooting [2]. However, the diversity of log formats and the volume of logs generated daily make manual analysis cumbersome and error-prone. Thus, the demand for automated log analysis, particularly for prompt anomaly detection, is growing. This task aims to identify potential anomalies in real-time, aiding operators in issue resolution and improving system stability and reliability [3].

Automatic anomaly detection in system logs has become crucial due to growing data complexity. Various methods, including machine learning techniques like Invariant Mining (IM) [4], Support Vector Machines (SVM) [5], and Principal Component Analysis (PCA) [6], have been proposed. These methods extract key features from log sequences to train binary classifiers for anomaly detection. Despite their high accuracy, they rely on manual features, potentially leading to inaccurate feature selection and inability to capture deep hidden features. Furthermore, these methods have limited generalization ability, struggling with unknown logs.

The advent of deep learning has provided innovative solutions for log anomaly detection. Researchers have proposed a series of preliminarily successful strategies, such as DeepLog [7], LogAnomaly [8] and SwissLog [9] which primarily utilize Recurrent Neural Networks (RNN) to encapsulate comprehensive bidirectional context data, leveraging the inherent feedback mechanism to learn sequential event execution patterns in log data. However, these deep learning-based methods still face challenges in log data modeling and analysis. For instance, methods like DeepLog detect anomalies by mining quantitative features of log sequences, reflecting sequence patterns but overlooking semantic information of log text, leading to high false alarm rates for new log templates. While approaches like LogRobust extract semantic features based on log templates, improving accuracy to some extent, they simply map words to word vectors, disregarding the relationship between words and log statements [10].

Considering the reflections on the aforementioned issues, this paper presents a deep learning-based method for automatic log anomaly detection—LogCEM. The LogCEM approach represents log templates by employing weighted semantic vectors generated through Roberta combined with an improved Smoothed Inverse Frequency (SIF) algorithm, thereby thoroughly accounting for the impact of word meaning on log statements. This methodology effectively captures the semantic information latent within log templates. For the anomaly detection stage, a novel detection model is proposed, which integrates Multi-Scale Convolutional Neural Networks (MSECNN) and Mogrifier Gated Recurrent Units (GRU). The model comprises two components: MSECNN extracts crucial multi-scale local features via multi-scale one-dimensional convolutions and harnesses the Efficient Channel Attention (ECA) mechanism [11] to capture intrinsic interdependencies within the input. Complementarily, Mogrifier GRU leverages complex long- and short-term dependencies among log sequences for forecasting purposes. LogCEM was evaluated on public benchmark big data log datasets, with experimental outcomes demonstrating its proficiency in accurately identifying anomalous conditions. Comparative analyses with state-of-the-art methodologies further validated the superiority of our proposed approach. Our key contributions are summarized as follows: The contributions of this paper can be summarized as follows:

1. We propose a novel log semantic representation method based on RoBERTa and an enhanced SIF algorithm. This approach incorporates considerations of term frequency, semantics, and

part-of-speech, comprehensively addressing the impact of word importance on log statements while ensuring unique representations across different log templates. It is capable of precisely identifying semantically similar log events, discriminating between distinct log events, and characterizing evolving incidents. Consequently, it significantly contributes to enhancing the detection accuracy of various anomaly detection models.

2. We introduce a new end-to-end deep learning-based model for log sequence anomaly detection. The MSECNN module, by combining multi-scale features with the ECA mechanism, effectively captures both local and global characteristics of logs. Complementarily, the Mogrifier GRU component meticulously extracts informative content from both preceding and succeeding parts of the log sequence data, thereby augmenting the model's predictive capabilities. This integrated approach enhances overall performance in identifying anomalies within sequential log data.
3. Extensive system experiments were conducted on the HDFS and the BGL datasets to evaluate the LogCEM model. The results affirmatively demonstrate the efficacy of our proposed method in detecting a wide spectrum of anomalous log entries, exhibiting a substantial improvement in both accuracy and robustness compared to baseline models. These findings underscore LogCEM's capability to enhance anomaly detection in complex logging environments.

The remainder of this paper is organized as follows. Section 2 focuses on the background and related work. Section 3 describes the overview and details of LogCEM. In Section 4, we present our evaluation results. Finally, we summarize our work in Section 5.

## 2 Related Work

In the field of log anomaly detection, log parsing, feature extraction, and anomaly detection constitute three crucial stages. For each stage, we have conducted an exhaustive literature review and analysis of related work.

### *2.1 Log Parsing*

Log parsing is a key prerequisite step in log anomaly detection. Specifically, it is the task of extracting log templates and log parameters from raw log information. The automation of log parsing has gradually replaced the practice of manually reading logs and configuring regular expressions in the source code. Currently, the methods for log template extraction mainly include clustering, heuristic, and neural networks.

The operating assumption of cluster-based parsers is that logs sharing a common log template can be grouped based on specific features. LKE [12] employs a hierarchical clustering algorithm and a custom weighted edit distance measure, rather than directly clustering the logs. LogSig [13] transforms each log into a set of word pairs and clusters the logs based on the corresponding word pairs.

Heuristic-based log parsing methods find suitable heuristic algorithms for logs based on the format information or word information in the logs and extract log templates. Drain [14] is the most advanced parser in the public benchmark dataset, using a fixed-depth tree structure to help classify logs into groups. Spell [15] uses the longest common subsequence algorithm to extract log templates. Given that logs exhibit some of the same characteristics as natural language, some solutions have adopted neural network technology as a potential solution. PVE [16] employs a pre-trained bidirectional

transformer to extract relevant features from historical logs. It then utilizes a dual-path framework to extract word embeddings and labels, training a classifier for online log parsing.

### *2.2 Feature Extraction*

The second step of log processing is feature extraction, which includes log grouping and log representation, as illustrated in Fig. 1. Log grouping aims to divide logs into a finite number of sequences for feature extraction and input to the anomaly detection model. Three log grouping methods exist: fixed window, session window, and sliding window. A fixed window maintains an unchanged size; a session window groups logs by identifiers; a sliding window has two attributes: Window size, referring to the number of logs in a sequence, and stride, denoting the minimum distance between adjacent sequences.

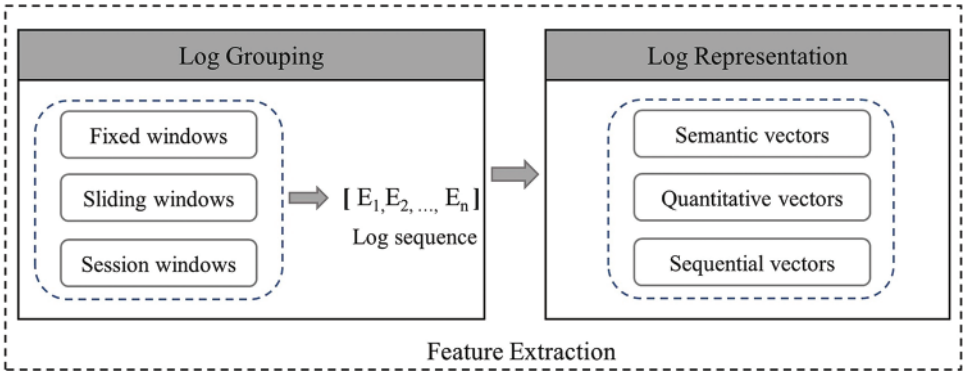


**Figure 1:** The workflow of feature extraction

After log grouping, logs need to be converted into a format recognizable by deep learning models. Existing anomaly detection models convert logs into three main types: Sequential vectors, quantity vectors, and semantic vectors. DeepLog assign indices to log events and generate fixed-dimension sequential vectors for each log sequence. LogAnomaly uses both sequential and quantity vectors. Although these methods model logs' sequential features, they ignore vital semantic features, rendering them unable to handle evolving logs. With NLP advancements, semantic vector-based methods are gaining attention, as log templates can be regarded as word sequences, and word vectorization can represent template semantics and extract temporal features. Currently, several techniques map log templates to semantic vectors. Bertero et al. [17] treat logs as regular text, applying Word2Vec-based word embedding to map log message words into high-dimensional vector space. However, Word2Vec cannot resolve synonymy issues. For instance, as shown in Fig. 2, "block" appears in both examples, but as a noun representing a data block in the former and a verb denoting the action of blocking in the latter. Word2Vec ignores sentence context, failing to fully comprehend log message semantics.

Log 1 : Receiving block blk_973160159069770286 src: /10.250.15.240:48815 dest: /10.250.15.240:50010

Log 2 : Firewall rule triggered: Blocking IP address 192.168.1.2 due to suspicious activity detected at 2024-03-05 12:56:18

**Figure 2:** Two log examples

LogRobust [18] and MLog [19] utilize the FastText model and Bert, respectively, to vectorize words. They then represent log templates by combining word vectors with the TF-IDF of a time window set as the corresponding weight. This paper uses a pre-trained model based on RoBERTa to obtain word embeddings. After word embedding, some weighted aggregation methods need to be applied to obtain sentence vectors, such as the average method and the TF-IDF weighted average method. Aggregating word vectors enables maintaining a fixed dimension for template vectors while mitigating the impact of varying template lengths. SwissLog uses an average mechanism to average all word vectors in the same log template to obtain the template vector. LayerLog [20] uses the TF-IDF mechanism to average all word vectors in the same template to obtain the log template vector. Unlike them, this study employs an unsupervised sentence vector synthesis method based on SIF [21]. This vector weighting calculation method has been proven to be superior to average weighting and TF-IDF weighting in many text tasks.

### *2.3 Anomaly Detection*

After the logs are grouped and log sequence representations are obtained, the log sequences are input into a deep learning model to perform anomaly detection tasks. At present, a variety of machine learning and deep learning techniques have been applied to log-based anomaly detection tasks, as summarized in Table 1. Liang et al. [5] trained a SVM classifier to detect anomalies in log events. Xu et al. [6] employed PCA with term weighting technology from information retrieval, achieving effective anomaly detection results without extensive parameter adjustment. While these methods partially capture pattern characteristics of log sequences, they overlook the sequential order among log sequences, potentially misclassifying anomalous data as normal and thereby resulting in a relatively high rate of false negatives.

**Table 1:** The summarization of log anomaly detection methods

| Method | Category | Log parser | Feature extraction | Anomaly detection model | Datasets |
|---|---|---|---|---|---|
| Liang et al. [5] | Supervised | – | Statistical feature count vector | SVM | BGL |
| Xu et al. [6] | Unsupervised | Source code-based | Event count vector | PCA | HDFS and private darkstar |
| Lu et al. [22] | Supervised | IPLoM | Event sequential vector | CNN | HDFS |
| Zhang et al. [23] | Supervised | Log clustering Tree | Pattern-based TF-IDF | LSTM | Enterprise system logs |
| DeepLog [7] | Unsupervised | Spell | Event count vector | LSTM | HDFS and OpenStack |

(Continued)

**Table 1 (continued)**

| Method | Category | Log parser | Feature extraction | Anomaly detection model | Datasets |
|---|---|---|---|---|---|
| LogCEM | Supervised | PVE | RoBERTa + SIF template semantic vector | MSECNN + Mogrifier GRU | HDFS and BGL |

With the development of deep learning technology, it is attracting more and more researchers, providing a new direction for log anomaly detection. Lu et al. [22] used CNN to automatically learn events in system logs. CNN can use multiple filters to mine relationships in the log context and capture the relevance in the semantic embedding of log templates. Since logs are generated over time, log sequences are composed of log messages in chronological order, essentially a type of time series data. Therefore, the common neural network in log sequence anomaly detection is RNN, because they can capture the temporal information in sequence data. Zhang et al. [23] first used LSTM for log system fault prediction. They collected logs with similar formats and contents, and processed the "rarity" of labeled data during the training process with LSTM to capture the remote dependencies between log sequences. DeepLog employs LSTM for modeling normal log sequences and frames anomaly detection as a multi-class classification task. However, they focus on log events and disregards context relationships in log sequences. Although RNNs and their variants have been widely employed in log anomaly detection, they still present certain inadequacies when modeling log sequences. This is due to the fact that conventional RNNs can only leverage information from preceding log messages, failing to capture the full context and thereby limiting their capacity to comprehend the comprehensive sequential dependencies inherent in log data. Consequently, this study leverages the Mogrifier mechanism to augment the contextual modeling capabilities of the GRU model, enabling a more sophisticated handling of sequential information. Additionally, the employment of MSCNN enhances the extraction of local features, capturing diverse granularity of patterns within the log sequences. Furthermore, the integration of an ECA module facilitates the automatic learning of the most salient features, thereby further boosting the detection efficacy and sensitivity to subtle anomalies.

## 3 Methodology

### *3.1 Overview*

Logs record operational state information of software systems, becoming crucial data sources to ensure system reliability and stability. However, detecting system anomalies from massive log data is challenging. To address this, LogCEM, a log anomaly detection method based on semantic embedding and hybrid neural networks, is proposed, as illustrated in Fig. 3.

The first step is log parsing, where log content is extracted from unstructured raw logs, and numbers and punctuation are removed through regular expression matching. After comparing log parsing methods, we employ PVE for efficient conversion of preprocessed log constants into log templates, effective for any log data. The second step is feature extraction. After grouping

the log templates, we input the log template sequence into the pre-trained language model RoBERTa, and then use the improved SIF algorithm to obtain the weighted average sentence vector for each log template. This sentence vector can effectively reflect the semantic information contained in the sentence. The final step is anomaly detection, where the log template sequence represented by numerical vectors is input into a mixed deep learning model for training. The model learns the hidden patterns of the log sequence and then performs binary classification to determine whether an anomaly is present.

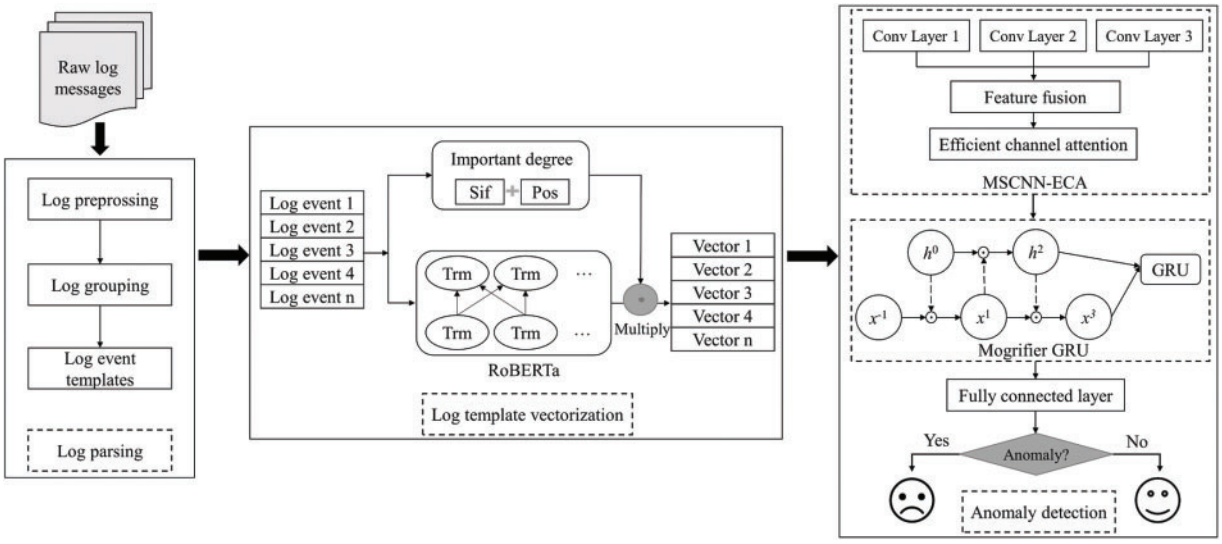


**Figure 3:** LogCEM framework

### *3.2 Log Processing*

Given the unstructured nature of raw logs and the structured data requirements of neural networks, log parsing aims to convert unstructured raw logs into structured log templates. Log parsing accuracy impacts the performance of downstream tasks, necessitating parameter tuning for specific log datasets. Nonetheless, the adaptability of some common log parsing methods remains relatively low. For instance, parse tree-based methods construct trees using token frequency-based specific rules, resulting in low parsing accuracy on certain log datasets due to the non-universal nature of these rules. Fig. 4 illustrates an example of Drain's incorrect parsing of a single BGL log. To enhance anomaly detection efficiency and accuracy, this paper compares several common log parsing methods and ultimately selects PVE as the log parser. PVE employs a variational autoencoder for log token classification. During log parsing, if a token is similar to the training data, PVE categorizes it as a template token. Consequently, PVE can effectively adapt to various log datasets, achieving high parsing accuracy without additional manual intervention.

### *3.3 Feature Extraction*

A log sequence is a combination of a series of log events that record a specific execution flow. Errors in the execution order of log events or incomplete execution patterns can lead to system anomalies. Therefore, it is necessary to perform anomaly detection from the perspective of sequences. After log parsing, log events are grouped into log sequences according to sessions or sliding windows. If there is an abnormal log $L_i$ in the log sequence $Seq = [L_1, L_2, \ldots, L_n]$, then this sequence is an

abnormal sequence. After the log sequence is partitioned, semantic information is extracted from the log events using the pre-trained model RoBERTa. Subsequently, an improved SIF algorithm is applied to weight this information, thereby obtaining the sentence vector of the log events, as shown in Fig. 5.

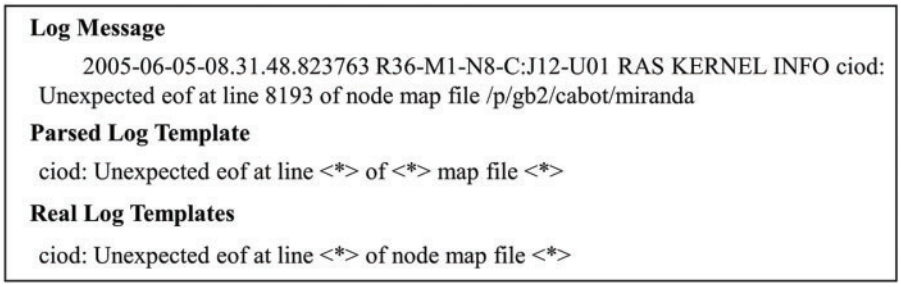


**Figure 4:** Examples of incorrect log parsing

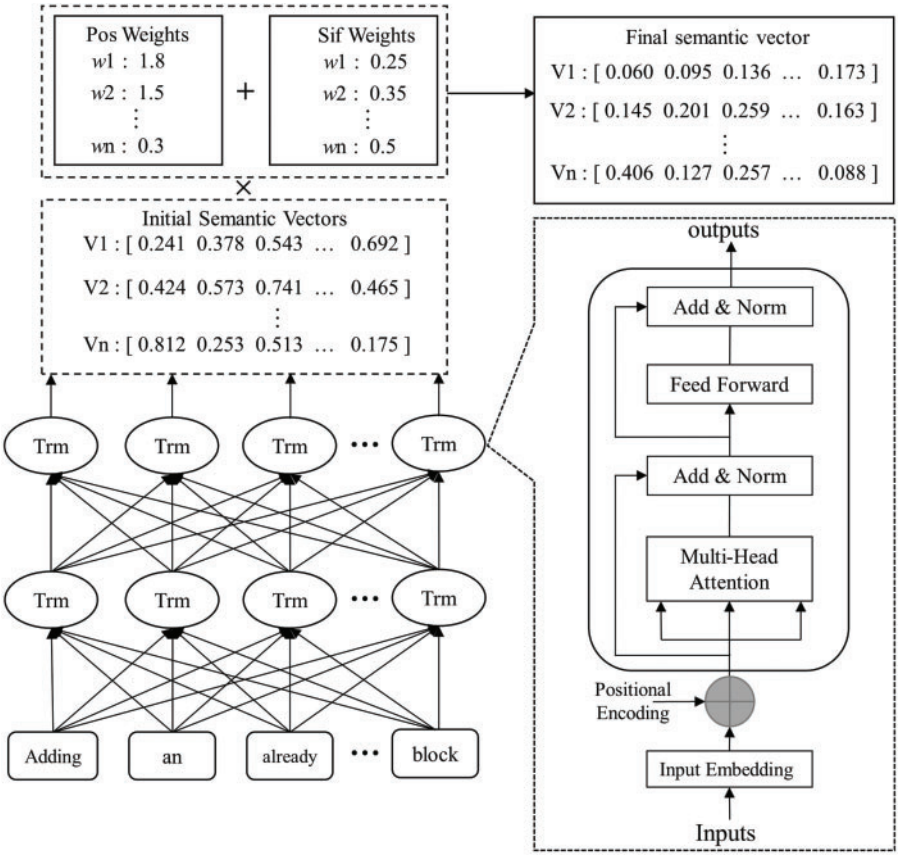


**Figure 5:** The workflow of semantic vectorization

### *3.3.1 Word Embedding*

After sequence partitioning, LogCEM employs the pre-trained RoBERTa model to generate word embeddings, providing rich and accurate data features for anomaly detection. This is because system iterations, upgrades, and improper log grouping can lead to changes in log templates, and relying

solely on statistical features such as order or quantity may affect detection performance. Considering that logs are natural language sentences written by engineers, with the main content being English words carrying semantic meanings. When logs are updated, although some words change, the main meaning remains unchanged. Therefore, LogCEM uses RoBERTa to generate semantic embeddings to represent log templates in the feature extraction stage. Previous studies have shown that a good log vector representation should meet two conditions: semantically similar words should have vectors with high cosine similarity in close proximity, and it should be able to represent the semantic differences between different log events. Although neural network methods such as Word2Vec and GloVe have addressed the word context relationship to some extent, they have not solved the issue of polysemy. This paper adopts RoBERTa for dynamic encoding of logs, which can obtain rich semantic features and address the polysemy problem.

The pre-trained RoBERTa model is first employed to generate the raw word embeddings for logs. RoBERTa is an optimized bidirectional encoder representation model pre-trained on a large 160 GB English corpus, with an order of magnitude more training data than BERT, and uses a byte-level BPE tokenizer instead of the character-level tokenizer in BERT, enabling more precise generalization and encoding of input log templates without introducing "unknown" tokens. Each log template is treated as a natural sentence, and the pre-trained RoBERTa tokenizer is used to tokenize it into n tokens $X = [t_1, t_2, \ldots, t_n]$, which helps retain semantics while mitigating the impact of out-of-vocabulary (OOV) words, i.e., words that are not present in the model's predefined vocabulary. After tokenization, each token is assigned a unique input ID representing its index in the RoBERTa vocabulary and an attention mask indicating contextual relevance. This mechanism enables the model to focus on salient tokens while disregarding less important ones, thereby enhancing performance on downstream tasks. Subsequently, input IDs and attention masks are fed into the RoBERTa model employed by LogCEM. This base model of RoBERTa comprises 12 Transformer encoders, each with 768 hidden units and 12 self-attention heads. More precisely, each token of the log template is sent into the Transformer encoder, which comprising a multi-head attention layer and a feed-forward layer. All sub-layers are followed by a residual connection and normalization layer to connect low-dimensional vectors and mitigate the adverse effects of data of different scales. Upon entering the encoder, the multi-head self-attention layer learns information from different positions in the log, and then adjusts weights and other parameters in the feed-forward layer. After passing through 12 such encoders, a state vector containing hidden layer information is output at the top. The multi-head attention mechanism is achieved by stacking multiple attention mechanisms to obtain the attention weight of each word in the log. The input to the attention mechanism consists of three matrices of the same dimension: $Q, K, V$. The attention value $\mathrm{A}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V}$ can be obtained through the query sequence $Q$, key sequence $K$, and value sequence $V$, where $\sqrt{d_k}$ represents the scaling factor, primarily intended to prevent gradient vanishing during the backpropagation process. The multi-head attention mechanism obtains semantic feature vectors of words from different representation spaces, then concatenates the results from multiple attention heads and applies a linear transformation:

$$\mathrm{h_i} = \mathrm{A}\left(\mathrm{W_i^Q Q}, \mathrm{W_i^K K}, \mathrm{W_i^V V}\right) \tag{1}$$

$$M = C\left(h_1, \ldots, h_h\right) W^O \tag{2}$$

where, $W_i^Q \in R^{dq \times d_h}$, $W_i^K \in R^{d_k \times d_h}$, $W_i^V \in R^{d_v \times d_h}$ and $W^O \in R^{d_o \times d_h}$ are weighting matrix. After each input passes through the multi-head attention layer, it undergoes residual network and layer normalization operations. Then, the feed-forward network transforms the hidden layer representation obtained by

the multi-head attention mechanism. For the log sequence $X$, its calculation is:

$$F(X) = max(0, XW_1 + b_1) W_2 + b_2 \tag{3}$$

where, $W_1$ and $W_2$ represent the weight matrices, and $b_1$ and $b_2$ represent the bias values. The hidden representation of $X$ is finally obtained through residual network and layer normalization calculations. Based on previous research, the output of the penultimate layer is chosen as the original word vector for each log word, as each hidden state layer can be used for sentence embedding. In summary, after the log template $X = [t_1, t_2, \ldots, t_n]$ is input into the RoBERTa model, the hidden vector is obtained as the feature representation of the log, denoted as $V = [v_1, v_2, \ldots, v_n]$, where $v_i$ represents the vector representation of the $i$-th token.

### *3.3.2 Important Degree Calculation*

After obtaining word embeddings, we need to aggregate them into sentence embeddings. Previous studies have shown that logs as natural language have two features: 1) Part-of-speech differences—Content words (e.g., nouns, verbs) are more important than function words (e.g., pronouns, prepositions) because they carry core semantics; 2) Context dependency—The importance of a word depends on its context, and the same word may have different importance in different log sequences. Traditional methods like average weighting and TF-IDF weighting cannot effectively represent the semantic features of logs. The SIF algorithm can convert word embeddings into sentence embeddings through weighting and denoising, and outperforms simple weighting methods in computing sentence similarity. However, SIF mainly performs weighting based on semantics and word frequency, ignoring part-of-speech factors. Therefore, Part-of-Speech (POS) Tagging is incorporated into the SIF algorithm to calculate the importance of words.

Specifically, we use the Natural Language Toolkit (NLTK), designed by the University of Pennsylvania, Philadelpha, United State to tag the part-of-speech weight for each word. Based on the idea that the contribution of nouns is greater than verbs, verbs greater than adjectives, and adjectives greater than other parts of speech, we set the corresponding weights for POS, as shown in Table 2. We denote the weight corresponding to POS as $W_p$. The higher the $W_p$, the more important the word is.

**Table 2:** The corresponding weights of POS

| | POS | Meaning | Examples | $W_p$ |
|---|---|---|---|---|
| Content word | NN | Noun, singular or mass | Error | 1.8 |
| | VB | Verb, base form | Connect | 1.5 |
| | JJ | Adjective | Local | 1.2 |
| | RB | Adverb | Next | 1 |
| Function word | CC | Coordination | and, either, or | 0.5 |
| | IN | Preposition or conjunction | on, in, at | 0.5 |
| | DT | Determiner | the | 0.3 |
| | TO | "to" as preposition or infinitive marker | to | 0.1 |

The SIF algorithm enhances the TF-IDF weighted average word vector method by using SIF as the weight for averaging all word vectors and subtracting the projection of the first principal component to get the sentence vector. This allows for a finer representation of a sentence's semantic

content. The SIF algorithm is based on the latent variable generative model, assuming word vectors in the corpus are dynamically generated by latent variables. The discourse vector $c_t$, representing the main meaning of the sentence, is an important latent variable that is updated over time. The SIF model views sentence generation as a dynamic random walk process. Each step generates a word and is determined by a topic vector $c_t$. For a given sentence $s$, its sentence vector is the maximum a posteriori estimate of the topic vector $c_t$. The smoothing process in the SIF model is based on two Assumptions: 1) Some words do not appear based on the context. 2) The occurrence of some high-frequency words is unrelated to the sentence's topic.

The probability of the occurrence of the word '$w$' in a sentence that is about the topic '$c_s$' is represented as:

$$Pr\left[w_s|c_s\right] = \alpha p\left(w\right) + \left(1-\alpha\right)\frac{\exp\left(< \tilde{c}_s, v_w >\right)}{{}^{Z}\tilde{c}_s} \tag{4}$$

where, $\tilde{c}_s = \beta c_0 + (1-\beta)c_s$, $c_0$ is orthogonal to $c_s$. The term $\alpha p(w)$ corresponds to Assumption 1), where $p(w)$ denotes the frequency of a word appearing in the entire corpus. $\alpha$ is a constant that allows for the probability of a word to be extremely small, yet it still appears with a probability of $\alpha p(w)$. The latter term corresponds to Assumption 2), which assumes that there is a common topic vector $c_0$ for all sentences. When the word $w$ is a high-frequency word, i.e., it is related to the common topic $c_0$, it can appear with a certain probability. $\beta$ is a constant, and $Z_{\tilde{c}_s}$ normalizes this term.

Based on the above assumptions, the generation probability of sentence $s$ with $c_s$ as the topic is:

$$p\left[s|c_s\right] = \prod_{w\in s} p\left(w|c_s\right) = \prod_{w\in s}\left[\alpha p\left(w\right) + \left(1-a\right)\frac{\exp < \boldsymbol{v}_w, \tilde{\boldsymbol{c}}_s >}{Z}\right] \tag{5}$$

The maximum a posteriori estimate for $\tilde{\boldsymbol{c}}_s$ is:

$$p\left[s|\boldsymbol{c}_s\right] = \sum_{w\in s} f_w\left(\tilde{\boldsymbol{c}}_s\right) \propto \sum_{w\in s}\frac{a}{p\left(w\right)+a}\boldsymbol{v}_w \tag{6}$$

where, $a = \dfrac{1-\alpha}{\alpha Z}$. Define the corresponding weight of the word $w$ in the sentence $s$ as:

$$w(x_i) = \frac{aWp}{a + p\left(w\right)} \tag{7}$$

where, $p\left(w\right)$ represents the probability of each word appearing in the corpus, which is calculated by dividing the word frequency by the total number of words in the corpus. α is a constant, with a value of 0.0001, and $W_p$ stands for the POS weight factor.

### *3.3.3 Template Vector Generation*

Finally, the log template vectors are constructed by combining the initial semantic vectors of each word with SIF weights that incorporate part-of-speech factors. This is done using a weighted summation approach that removes non-informative noise (i.e., the maximum principal component), as shown in Fig. 6. The specific calculation formula is presented as follows:

$$V_S^{'} = \frac{1}{n}\sum_{x\in X} w\left(x_i\right) V_{xi} \tag{8}$$

$$V_{\mathrm{S}} = v_{\mathrm{S}}^{'} - \mu\mu^{\mathrm{T}} V_{\mathrm{S}}^{'} \tag{9}$$

where, $V_s^{'}$ represents the sentence vector before removing the maximum principal component, $\mu$ is the feature matrix obtained through singular value decomposition of the matrix composed of all $V_s^{'}$; $\mu^T$ is the transpose of $\mu$; and $\mu\mu^T V_s^{'}$ is the maximum principal component vector of $V_s^{'}$.

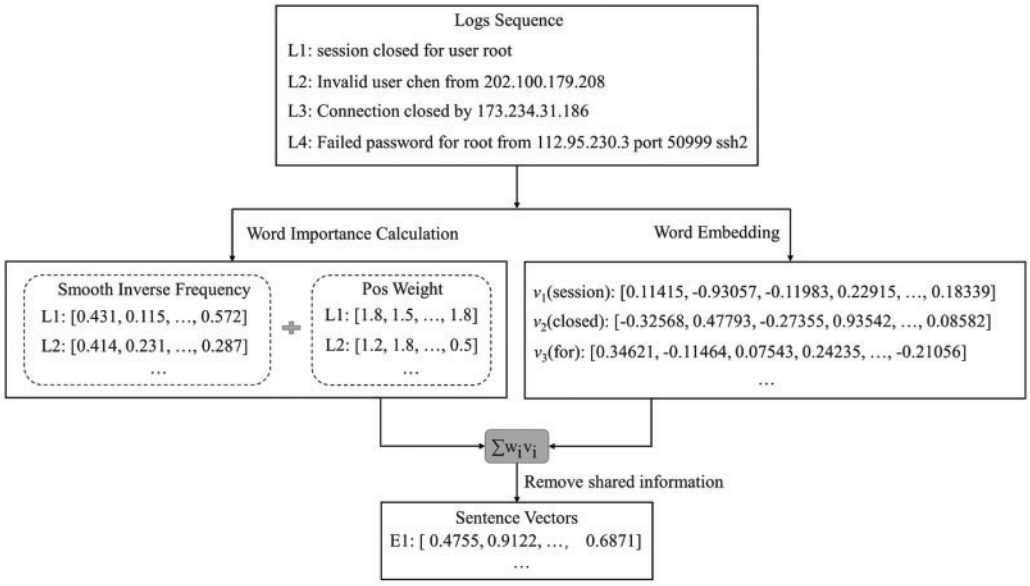


**Figure 6:** The specific implementation process of template vector generation

### *3.4 Anomaly Detection*

During the log data representation phase, each log event template is transformed into an embedding that reflects its semantic features. Consequently, the entire log sequence is constructed as a list of these embeddings, denoted as $[E_1, E_2, \ldots, E_n]$, forming the log embedding sequence. To discern anomalies, this sequence is input into an anomaly detection model for classification analysis. Our study introduces a hybrid neural network architecture that integrates MSECNN and Mogrifier GRU. This model adeptly captures both the local and global dependencies within log data and autonomously identifies salient log sequence features, thereby effectively detecting diverse anomaly patterns within system log sequences.

#### *3.4.1 Multiscale Efficient Channel Attention Convolutional Neural Network*

In binary and multiclass classification tasks, CNN with filters is effective for extracting local features and identifying local dependencies in multivariate time series data like log sequences where adjacent entries exhibit local correlations. One-dimensional convolution is particularly suited for processing temporal textual data like logs, as it enables CNN to comprehend the entire log sequence, not just isolated portions. However, a single convolutional layer with a fixed kernel size has limitations in capturing complex local patterns. To extensively mine local correlations within log sequences, this study employs multiple 1D convolutional layers with various kernel sizes. Within each kernel size, multiple filters are used to capture diverse features. This multi-scale 1D convolutional layer strategy allows capturing richer local dependencies in sequence data by combining information from different receptive fields and kernel sizes.

In our multi-scale convolutional layer design, we use kernels of sizes 2, 3, and 4 to extract features of varying granularities, as shown in Fig. 7. The size 2 kernel captures local features, the size 3 kernel broadens the receptive field for more extensive features, and the size 4 kernel further expands the receptive field for comprehensive understanding of the input data structure and context. These layers work in parallel on the same input. The output feature maps from all convolutions are concatenated along the channel dimension, forming a comprehensive feature representation. This structure allows thorough feature extraction from the input data by integrating different kernel sizes.

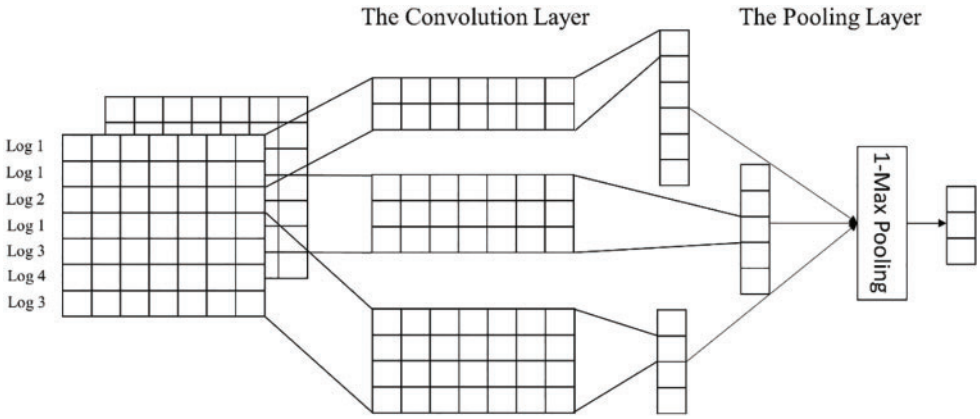


**Figure 7:** The structure of MSCNN

We utilize the Scaled Exponential Linear Unit (SELU) activation function in our CNN model, an improvement over the commonly used Rectified Linear Unit (ReLU). SELU mitigates the issues of vanishing and exploding gradients. As shown in Fig. 8, ReLU's derivative is always 1 for $x > 0$, preventing gradient decay, but is 0 for $x < 0$, leading to 'dead neurons'. SELU, with faster convergence speed, normalizes the sample distribution to zero mean and unit variance, ensuring stable gradients during training.

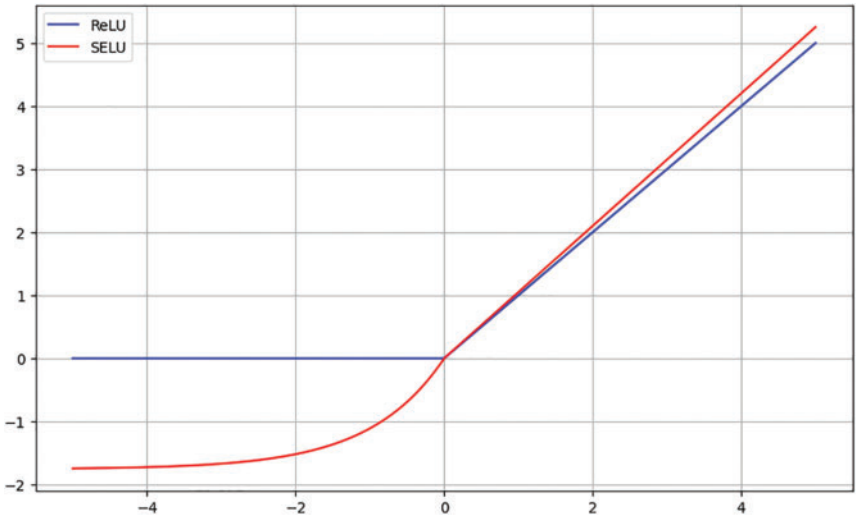


**Figure 8:** The difference between RELU and SELU activation functions

Through the multi-scale convolution module, we obtain an aggregation of features at different scales. We then adopt the ECA mechanism to derive channel-wise feature weights, allowing the neural network to attend to the importance of each channel, reducing redundant features, and improving feature utilization efficiency. Unlike traditional stacked convolutional layers, ECA enhances CNN's core attributes by employing a local cross-channel interaction strategy and adaptively selecting the one-dimensional convolution kernel size. This alleviates the adverse effects of dimension reduction encountered in mechanisms like SENet. As illustrated in Fig. 9, the ECA module applies global average pooling, then uses a sigmoid function to derive channel weights via a fast one-dimensional convolution of size k. ECA determines each channel's attention weight by incorporating a one-dimensional convolution operation along the channel dimension. This convolution operation acts as a local receptive field, considering only the correlation of each channel within a local scope, rather than globally. Consequently, the ECA mechanism reduces computational complexity while enhancing efficiency.

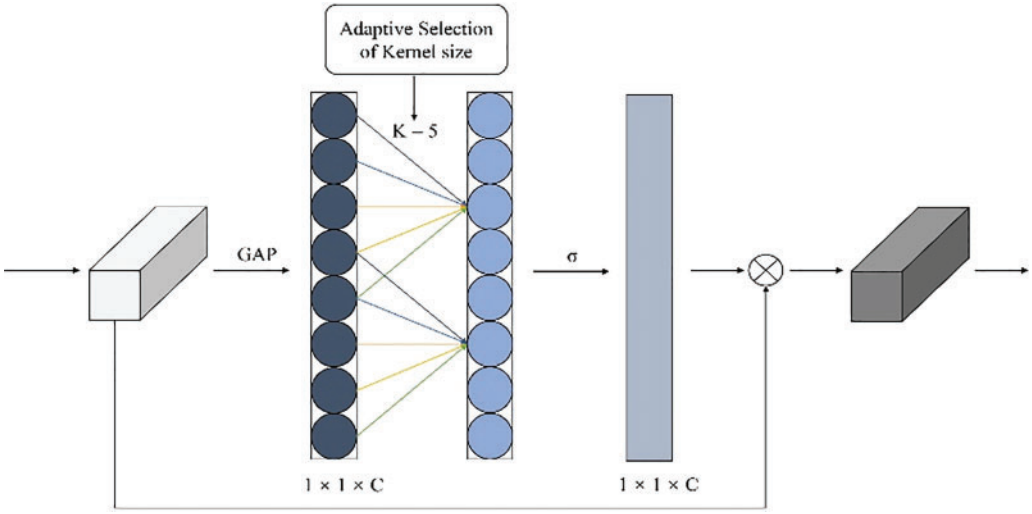


**Figure 9:** The structure of ECA

While CNN can capture local correlation features of log data, they struggle to mine the dependency relationships between data. However, the RNN framework can capture long-distance dependency information. Both GRU and LSTM belong to the RNN network architecture and introduce a gating mechanism to control the flow of information. The difference lies in that LSTM has three gates, including a single input gate, a forget gate, and an output gate, while GRU is a simplified version of LSTM, containing only a reset gate and an update gate. Compared with LSTM, GRU has a simpler model structure and fewer parameters. Furthermore, in order to enhance classification capabilities, the model needs to thoroughly learn contextual information and extract deeper hidden feature. However, in the GRU network, the input data $x_t$ and the hidden state $h_t$ from the previous moment are independent of each other. They only interact within the gates, and there is no correlation before this, which may lead to the loss of log context information. To overcome this limitation and better enhance the information interaction between log data, we introduce the Mogrifier data coupling module.

### 3.4.2 Mogrifier Gated Recurrent Unit

Specifically, the input is processed through the ECA module, resulting in a fused feature matrix. Assuming the length of the log sequence is n and the number of filters in the convolutional layer is m, we obtain a matrix of size n × m. Before feeding it into the model, we first perform an interaction operation to enhance the context modeling capability, and then input it into the original GRU for traditional operations. Fig. 10 shows the structure of a Mogrifier GRU model with three rounds of interaction updates. A one-way Mogrifier GRU unit is shown in Fig. 10.

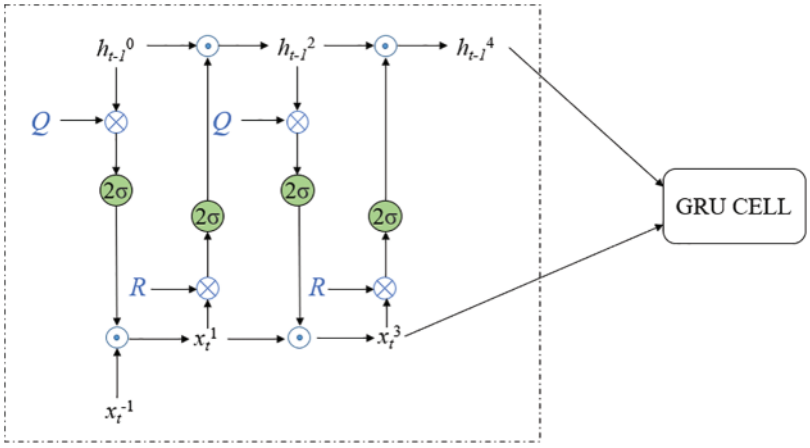


**Figure 10:** The structure of Mogrifier GRU

As depicted in Fig. 10, the intricate information interplay between the current input $x_t$ and the preceding hidden state $h_{t-1}$ is orchestrated through Eqs. (5)–(8).

$$x_t^1 = 2\sigma\left(Qh_{t-1}^0\right) \odot x_t^{-1} \tag{10}$$

$$h_{t-1}^2 = 2\sigma\left(Rx_t^1\right) \odot h_{t-1}^0 \tag{11}$$

$$x_t^3 = 2\sigma\left(Qh_{t-1}^2\right) \odot x_t^1 \tag{12}$$

$$h_{t-1}^4 = 2\sigma\left(Rx_t^3\right) \odot h_{t-1}^2 \tag{13}$$

where, $Q$ and $R$ are learnable parameter matrices; the symbol $\odot$ represents the Hadamard product. Once $x_t^3$ and $h_{t-1}^4$ are acquired, the corresponding hidden state $h_t$ at the present time instance can be derived as outlined below:

$$z_t = \sigma\left(W_z x_t^3 + U_z h_{t-1}^4 + b_z\right) \tag{14}$$

$$r_t = \sigma\left(W_r x_t^3 + U_r h_{t-1}^4 + b_r\right) \tag{15}$$

$$\widetilde{h}_t = \tan\mathrm{h}\left(W_h x_t^3 + U_h\left(r_t \odot h_{t-1}^4\right) + b_h\right) \tag{16}$$

$$h_t = z_t \odot h_{t-1}^4 + (1 - z_t) \odot \tilde{h}_t \tag{17}$$

where $W_z$, $W_r$, $W_h$, and $U_z$, $U_r$, $U_h$ are the wegiht matrix, and $b_z$, $b_r$, $b_h$ are the bias; $z_t$ and $r_t$ represent the output values of the update gate and reset gate at time step $t$, respectively; $\widetilde{h}_t$ represents the summary of the input and past hidden layer states and $h_t$ is the output of the hidden layer state.

Finally, the output features are used as input to the fully connected layer and the Softmax classifier. After supervised learning on the training dataset, this binary classifier can evaluate whether a given log sequence is abnormal.

## 4 Experiments

### *4.1 Experimental Setup*

To validate the effectiveness of LogCEM, this study utilizes two authoritative datasets, HDFS and BGL, available in LogHub [24]. These two log datasets are widely used in the field of log anomaly detection. For each dataset, 60% of the data was used as the training set, 20% as the validation set, and the remaining 20% as the test set.

The Hadoop Distributed File System (HDFS) dataset is a manually categorized collection of normal and abnormal logs generated using benchmark workloads within the Hadoop ecosystem [8]. Each log entry is associated with a unique block ID, allowing for natural session window organization based on these IDs. The dataset comprises 11,175,629 logs, segmentable into 575,061 sessions, with 16,838 identified as anomalous.

The BlueGene/L (BGL) dataset consists of logs from a supercomputer system at Lawrence Livermore National Labs, bifurcated into alert and non-alert messages based on alert category tags. Encompassing 4,747,963 logs, with 348,460 identified as anomalous, BGL logs lack unique identifiers like HDFS's "block ID" for job sessions. Consequently, we employ sliding windows to segment logs into sequences, considering a sequence anomalous if it contains at least one anomalous log. For training purposes, 40,000 log sequences were randomly selected from each of the normal and anomalous categories within the BGL dataset. Table 3 provides detailed descriptions.

**Table 3:** Detailed information on two public datasets

| Dataset | Span of time | File size | Logs | Anomalies |
|---|---|---|---|---|
| HDFS | 38.7 h | 743,185,031 bytes | 11,175,629 logs | 16,838 blocks |
| BGL | 214.7 days | 1,577,982,906 bytes | 4,747,963 logs | 348,469 logs |

We implement our proposed model on a Windows server with 12th Gen Intel(R) Core(TM) i7-12650H CPU @ 2.30 GHz, 16 G memory, NVIDIA GeForce RTX 2080Ti GPU and Pytorch 1.7.1. The parameters of our algorithm are described in Table 4. We used the control variable method to adjust the hyperparameters size and step to improve model performance.

**Table 4:** Initial hyperparameter configuration table of the model

| Parameters | Value |
|---|---|
| Dimension of template vector | 768 |
| Learning rate | 5e–5 |
| HDFS batch size | 24 |
| BGL batch size | 32 |
| Log sequence | 18 |
| Number of filters | 48 |

(Continued)

**Table 4 (continued)**

| Parameters | Value |
|---|---|
| Filter sizes | [3, 4, 5] |
| Activation function | SELU |
| Optimizer | AdamW |
| Loss function | CrossEntropyLoss |
| Epoch | 180 |

### *4.2 Evaluation Metrics*

To evaluate the effectiveness of LogCEM in anomaly detection, this paper employs Precision, Recall and F1-score as evaluation metrics.

Precision, denoting the proportion of correctly identified malicious log entries among all positive predictions:

$$Precision = \frac{TP}{TP + FP} \tag{18}$$

Recall, representing the ratio of correctly detected malicious log entries to all such entries:

$$Recall = \frac{TP}{TP + FN} \tag{19}$$

F1-score, the harmonic mean of precision and recall, taking both into account:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{20}$$

This paper also uses the Area Under the ROC Curve (AUC) to evaluate the classification effect of the model. AUC signifies the region enclosed between the ROC curve and the $X$-axis. The ROC is a curve plotted on a two-dimensional plane, where the False Positive Rate (FPR) constitutes the $X$-axis and the True Positive Rate (TPR) forms the $Y$-axis, the calculation is shown in Eqs. (18) and (19).

$$FPR = \frac{FP}{FP + TN} \tag{21}$$

$$TPR = \frac{FP}{TP + FN} \tag{22}$$

A larger AUC value corresponds to the curve being nearer to the upper left corner, indicative of a superior classification outcome. It is primarily employed in the evaluation of a binary classification model's performance. The calculation formula for AUC is shown as follows:

$$AUC = \frac{\sum_{i \in positive} rank_i - \frac{p*(n+1)}{2}}{p \times n} \tag{23}$$

where, ranki refers to the position of the $i$-th sample when arranged in a certain order. $p$ and $n$ denote the total count of positive and negative samples, respectively. $\sum_{i \in positive}$ represents the summation of the ranks of all positive samples.

### *4.3 Baselines*

To verify the effectiveness and advancement of LogCEM, this paper selected various popular supervised and unsupervised methods as baselines for validation on various datasets. The baseline methods are briefly introduced as follows:

Principal Component Analysis is a methodology grounded in Principal Component Analysis. It isolates the main constituents of input sequence attributes and designates test sequences surpassing a specific limit as anomalies.

Invariant Mining prioritizes the utilization of indices from log templates as input over semantics. It extracts invariants from log events based on vectors of log template counts. Log sequences that do not adhere to these invariants are flagged as anomalies.

DeepLog is a conventional approach to log anomaly detection, utilizing the LSTM neural network. It adopts one-hot encoding for the vectorization of templates and has demonstrated effective results in tasks related to anomaly detection.

LogAnomaly harnesses a unique template2Vec approach to distill semantic data from log templates. Additionally, it relies on the LSTM neural network for identifying anomalies.

LogBERT is a model that mirrors the structure of BERT. It's trained on sequences of log template indices, utilizing both the loss from predicting masked template indices and the hypersphere objective loss.

### *4.4 The Impact of Template Extraction*

Previous research has demonstrated a significant impact of log parsing on the effectiveness of log anomaly detection. Our method also requires initial log parsing to obtain structured log messages. Log parsing enhances the quality of log representation, thereby improving the performance of the anomaly detection model. It reduces the noise in the representation and lightens the learning burden of the model by removing dynamic fields in the logs. Proper preprocessing and modeling of these dynamic fields may be crucial for optimal log representation. Therefore, it is essential to evaluate these different template extraction components for their impact on anomaly detection performance. This will help us understand the strengths and weaknesses of various methods more comprehensively and provide direction for future improvements.

In these experiments, we tested some of the current representative and novel log parsing methods, including Drain, Logram, IPLoM, Spell and PVE. As shown in Table 5, the experimental results show that there are subtle differences in performance when different parsers are used to parse the dataset. Since PVE performs best in anomaly detection and has good adaptability to different log datasets, we choose PVE as the default log parser in the following experiments.

**Table 5:** The impact of different log parsers on detection performance

| Dataset | | Drain | Spell | IPLoM | Logram | PVE |
|---|---|---|---|---|---|---|
| HDFS | Precision | 0.97 | 0.96 | 0.98 | 0.98 | **0.99** |
| | Recall | 0.98 | 0.99 | 0.97 | 0.97 | 0.99 |
| | F1-score | 0.98 | 0.98 | 0.98 | 0.97 | **0.99** |

(Continued)

**Table 5 (continued)**

| Dataset | | Drain | Spell | IPLoM | Logram | PVE |
|---|---|---|---|---|---|---|
| BGL | Precision | 0.98 | 0.98 | 0.97 | 0.97 | **0.99** |
| | Recall | 0.97 | 0.98 | 0.99 | 0.98 | 0.99 |
| | F1-score | 0.98 | 0.98 | 0.98 | 0.97 | **0.99** |

### *4.5 The Impact of Log Representation*

In order to evaluate the contribution of different log representation methods to log anomaly detection, we conducted a set of additional experiments on the original HDFS and BGL datasets. In these experiments, we compared the log template semantic vectorization method based on RoBERTa and improved SIF proposed by us with other commonly used vectorization methods. Specifically, while keeping the rest of the model the same, we changed the log template semantic embedding module to methods such as Word2vec, GloVe, Bert, and TF-IDF. The experimental results are shown in Table 6. We can see that the template vectorization based on RoBERTa performs better in terms of F1-score than the methods based on Word2vec and GroVe. This is because the use of Word2vec and Glove methods will have many OOV problems, which cannot accurately represent the semantic information of the log sequence. Compared with the method combined with BERT and TF-IDF, the method we proposed has improved in both Precision and F1-score. This shows that RoBERTa automatically captures the potential semantic relationship between important words and sentences more effectively than other models, which is mainly due to the dynamic masking strategy of the RoBERTa model that allows the model to rely more on the context to predict the masked vocabulary, thus fully obtaining semantic information. The improved SIF algorithm removes the common noise and considers the word property factors, so our method takes into account semantics, word frequency, and word properties, and more effectively represents the feature information of the logs.

**Table 6:** The impact of different feature vectors on detection performance

| Method | HDFS (%) | | | | BGL (%) | | | |
|---|---|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Specificity | Precision | Recall | F1-score | Specificity |
| Word2vec | 95.42 | 97.88 | 96.63 | 99.10 | 96.22 | 96.65 | 96.43 | 99.20 |
| GroVe | 96.33 | 96.96 | 96.69 | 99.18 | 97.45 | 97.67 | 97.56 | 99.45 |
| BERT | 97.39 | 97.54 | 97.46 | 99.42 | 98.12 | 97.54 | 97.83 | 99.61 |
| BERT + TF-IDF | 98.15 | 98.66 | 98.40 | 99.57 | 97.96 | **98.72** | 98.34 | 99.56 |
| RoBERTa | 97.76 | 96.57 | 97.66 | 99.62 | 98.23 | 97.75 | 97.99 | 99.63 |
| RoBERTa + Improved SIF | **98.85** | **98.73** | **98.79** | **99.74** | **99.11** | 98.57 | **98.84** | **99.81** |

### *4.6 The Impact of Anomaly Detection Module*

To verify the impact of our proposed model, we compared it with different network model structures. We assess the influence of the anomaly detection module on the overall experimental framework. Within our proposed solution, we employ deep neural network methodologies to devise

an anomaly prediction model, which integrates 1D-MSCNN, ECA, and Mogrifier GRU. We aim to enhance the model's overall performance for detecting anomalies.

As shown in the Table 7, the F1-score of MSCNN is 1.5% higher than that of a single CNN, indicating that MSCNN can more effectively extract local features of the sequence by using parallel multi-scale convolution modules to generate multiple receptive fields. After adding the ECA module, all indicators of MSCNN have increased, which shows that introducing the ECA module into CNN can adaptively extract important information in the log sequence, enhance feature expression ability, suppress invalid features, and capture the dependence between channels through cross-channel interaction strategies. Finally, although adding sequence models such as LSTM and GRU can improve the effect of anomaly detection, the experimental results have declined compared with Mogrifier GRU. This is because the hidden state and current input value of neural networks such as LSTM and GRU are independently input, which may lead to the loss of some important information due to the lack of context. In contrast, Mogrifier GRU enhances the modeling ability of the context by first allowing the current input and the previous hidden state to interact, thereby improving the detection performance of the neural network.

**Table 7:** Experimental results of different abnormaly detection models

| Method | HDFS (%) | | | BGL (%) | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F1-score | Precision | Recall | F1-score |
| CNN | 94.45 | 93.87 | 94.16 | 95.15 | 96.53 | 95.84 |
| MSCNN | 95.77 | 95.55 | 95.66 | 97.89 | 98.25 | 98.07 |
| MSCNN + ECA | 96.32 | 96.47 | 96.39 | 98.34 | 98.66 | 98.50 |
| MSCNN + ECA + Lstm | 97.51 | 97.83 | 97.67 | 99.07 | 97.75 | 98.41 |
| MSCNN + ECA + GRU | 97.77 | 97.54 | 97.65 | 98.78 | 97.94 | 98.36 |
| MSCNN + ECA+ BIGRU | 98.11 | 97.89 | 98.00 | 98.63 | 98.21 | 98.42 |
| MSCNN + ECA + Mogrifier GRU-1 | 97.70 | 98.63 | 98.16 | 98.64 | 98.76 | 98.70 |
| MSCNN + ECA + Mogrifier GRU-2 | 98.13 | **98.92** | 98.53 | 98.89 | **98.64** | 98.76 |
| MSCNN + ECA + Mogrifier GRU-3 | **98.85** | 98.73 | **98.79** | **99.11** | 98.57 | **98.84** |
| MSCNN + ECA + Mogrifier GRU-4 | 97.38 | 97.96 | 97.67 | 98.35 | 98.44 | 98.39 |

Further, we found that the number of interaction times used in Mogrifier GRU significantly affects the detection performance. We changed the number of interaction times in Mogrifier GRU from 1 to 4, where Mogrifier GRU-n represents the Mogrifier GRU model with n interaction times. The results show that the F1-score increases first and then decreases with the increase of the number of interaction times. When the number of interaction times is 3, our method performs best among all corresponding methods. Specifically, we achieved F1-score of 98.79% and 98.84% on HDFS and BGL, respectively. This indicates that the interaction times enhance the main features of the input $x_t$ and weakens its secondary features. However, the more interactions, the higher the homogeneity of $x_t$ and $h_{t-1}$, which negatively affects performance and also increases computational costs.

### *4.7 Overall Performance and Comparison*

To demonstrate the superiority of LogCEM, we conducted a performance evaluation and compared it with other advanced methods on the HDFS and BGL datasets.

As depicted in Fig. 11, we conducted multiple repeated experiments on log datasets for LogCEM and five baseline methods, with the results representing their averages. On the HDFS dataset, LogCEM performed exceptionally well, achieving precision and recall rates exceeding 98.70%. Although PCA exhibited excellent precision, its lower recall rate led to a decreased F1-score, indicating an inability to effectively identify numerous anomalies, potentially due to its focus on sequence pattern characteristics while disregarding sequential relationships and contextual information between logs. Furthermore, methods solely relying on template counting struggle to truly reflect data relationships due to log sequence data's strong dependence, resulting in larger false negatives and lower recall rates. Template counting vector-based methods ignore semantic features when processing log sequences, causing the model to ineffectively learn sequence patterns, thus reducing overall performance. In anomaly detection, recall rate is considered more critical than precision, as missed anomalies may lead to serious economic losses and potential risks. While DeepLog and LogAnomaly employ LSTM models for anomaly detection networks, their sequential nature limits effective mining of long-distance data dependencies and global log connection characteristics, as each forward propagation is based solely on the previous time step. LogAnomaly's slightly better performance than DeepLog indicates that introducing more templates can enhance anomaly information capture. However, LogAnomaly did not perform well on the BGL dataset, possibly due to the large number of log templates causing sparse template statistical vectors, unable to capture linear relationships between log templates. On the BGL dataset, LogCEM exhibited the best anomaly detection effect, with an F1-score of 98.79%.

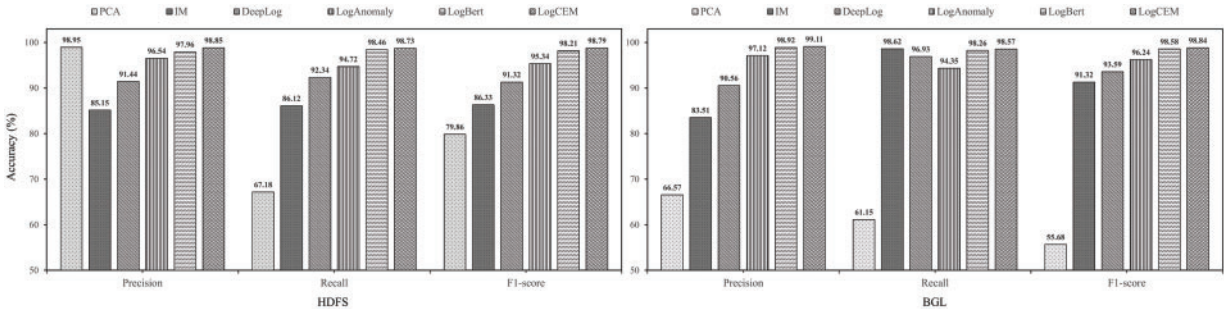


**Figure 11:** Experiment results on different datasets

Moreover, Fig. 12 illustrates the ROC curves and AUC values for various methods across two datasets. As evident from Fig. 12, our proposed method outperforms other baseline approaches with a higher average AUC, manifesting superior performance on both datasets. This is attributed to LogCEM's utilization of a combination of the RoBERTa pre-trained language model and an improved SIF algorithm with mean-weighted scheme, which affords a more advantageous semantic representation. Furthermore, the multi-scale local feature extraction prowess of MSECNN and the Mogrifier GRU's adeptness at integrating contextual semantics enable the LogCEM method to effectively address the instability of log sequence data and interdependencies among data, thereby demonstrating a high degree of reliability in log anomaly detection. The synergy of these components equips LogCEM with the capability to navigate the complexities inherent in log data, reinforcing its position as a robust solution for anomaly identification.

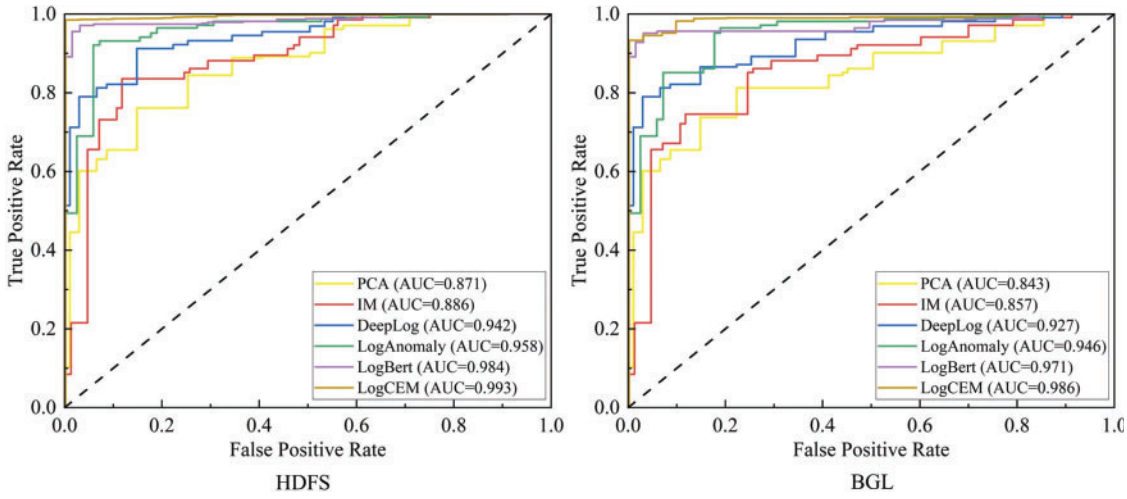


**Figure 12:** ROC curves of different methods on different datasets

### *4.8 Parametric Analysis*

Given the differing partitioning methods between BGL and HDFS datasets, BGL employs a sliding window approach to segment log sequences. The sliding window's size and stride directly determine the number of segmented log sequences. We evaluated the impact of window length on performance, gradually increasing it from 8 to 20 with a step size of 2. As shown in Fig. 13, LogCEM's performance on the BGL dataset improves as the window size increases, achieving optimal results at length 18. Smaller window lengths result in weaker learned semantic correlations, while larger sizes enhance the model's ability to capture abnormal information by increasing the number of log sequences. However, performance gradually declines when the window length exceeds 18, potentially due to excessive subsequences extracted from the enlarged sliding window, leading to overfitting and challenges in capturing log sequence correlations.

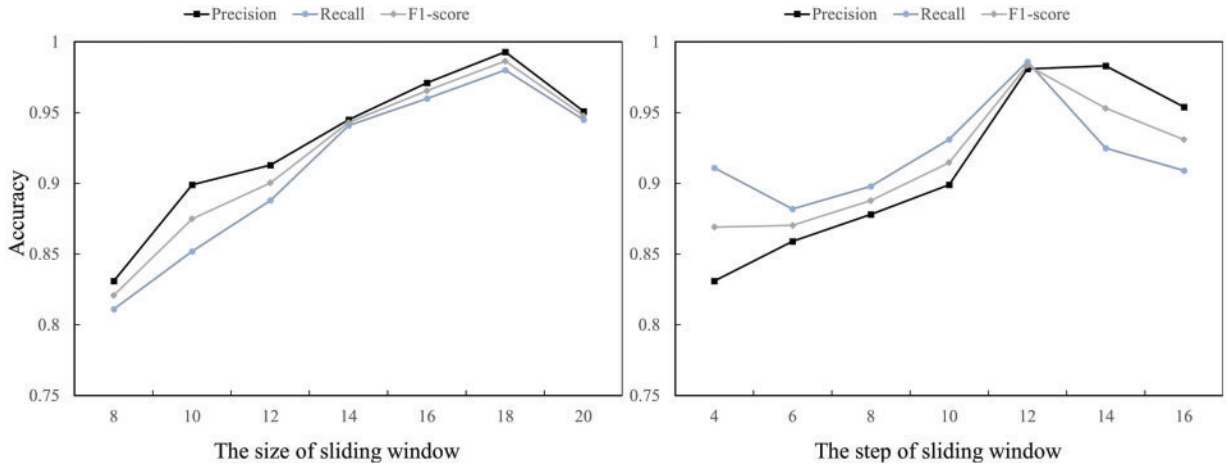


**Figure 13:** Effect of size and step on Precision, Recall and F1-score

The impact of the sliding window's size during model training on the model's feature learning capability was investigated in our study. We experimented with varying sliding step lengths to evaluate

the model's performance, as depicted in Fig. 13. Our observations indicate that the model's capacity to detect log anomalies initially improves with an increase in the number of sliding steps, but subsequently deteriorates. The optimal overall detection performance is achieved when the sliding step length is 12. This can be primarily attributed to the fact that an excessively long sliding step length results in a lax learning of the normal log sequence's features by the model, thereby hindering its ability to accurately discern the differences between abnormal and normal logs. This maintains the core meaning, reduces repetition, and aligns with the style of academic papers.

## 5 Conclusion

Logs have always been one of the most valuable data sources for ensuring the cybersecurity of power systems. However, leveraging the semantics of logs to improve sequential anomaly detection poses particular challenges. Consequently, there is a need for anomaly detection schemes based on log semantics.

This paper introduces LogCEM, a robust log anomaly detection method based on deep learning. Unlike traditional methods that use statistical or sequential features for log template representation, we innovatively integrate the RoBERTa language model with the SIF algorithm. This approach takes into account the impact of part of speech on semantics, effectively extracting hidden semantic features in the log sequence. Furthermore, we employ a multi-scale MSCNN with the ECA attention mechanism to capture local features. Global features are then captured through an improved GRU model, effectively distinguishing between normal and abnormal logs. Experiments on large-scale system logs show that LogCEM outperforms most current methods. The focus of semantic representation has been primarily on a single type of log. However, in practical systems, security experts often analyze security status based on multiple types of logs collectively. Therefore, in our future work, we will investigate the semantic representation of multi-logs. Additionally, LogCEM is not designed for incremental updates, and we will consider online incremental training in subsequent work to further enhance the efficiency of analyzing large data in practical systems.

**Acknowledgement:** The authors wish to express their appreciation to the reviewers for their helpful suggestions which greatly improved the presentation of this paper.

**Funding Statement:** This research was supported by the Science and Technology Program State Grid Corporation of China, Grant SGSXDK00DJJS2250061.

**Author Contributions:** Conceptualization: Zhanyang Xu; investigation: Zhe Wang; analysis and interpretation of results: Jian Xu; draft manuscript preparation: Hongyan Shi; supervision: Hong Zhao. All authors reviewed the results and approved the final version of the manuscript.

**Availability of Data and Materials:** Publicly available dataset was analyzed in this study. The research data can be found on https://github.com/logpai/loghub (accessed on 20 August 2023).

**Ethics Approval:** Not applicable.

**Conflicts of Interest:** The authors declare that they have no conflicts of interest to report regarding the present study.

## References


[1] A. Hany Fawzy, K. Wassif, and H. Moussa, "Framework for automatic detection of anomalies in DevOps," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 3, pp. 8–19, Mar. 2023. doi: 10.1016/j.jksuci.2023.02.010.

[2] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *2016 IEEE 27th Int. Symp. Softw. Reliab. Eng. (ISSRE)*, Ottawa, ON, Canada, IEEE, Oct. 2016, pp. 207–218. doi: 10.1109/ISSRE.2016.21.

[3] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen and F. Pianese, "System log parsing: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 35, no. 8, pp. 8596–8614, Aug. 2023. doi: 10.1109/TKDE.2022.3222417.

[4] J. -G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in *Proc. 2010 USENIX Conf. USENIX Annu. Tech. Conf., USENIXATC'10*, USA, USENIX Association, Jun. 2010, pp. 24.

[5] Y. Liang, Y. Zhang, H. Xiong, and R. Sahoo, "Failure prediction in IBM BlueGene/L event logs," in *Seventh IEEE Int. Conf. Data Mining (ICDM 2007)*, Omaha, NE, USA, IEEE, Oct. 2007, pp. 583–588. doi: 10.1109/ICDM.2007.46.

[6] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM SIGOPS 22nd Symp. Oper. Syst. Princ.*, Big Sky, MT, USA, ACM, Oct. 2009, pp. 117–132. doi: 10.1145/1629575.1629587.

[7] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. 2017 ACM SIGSAC Conf. Comput. Commun. Secur.*, Dallas, TX, USA, ACM, Oct. 2017, pp. 1285–1298. doi: 10.1145/3133956.3134015.

[8] W. Meng *et al.*, "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. Twenty-Eighth Int. Joint Conf. Artif. Intell.*, Macao, China, International Joint Conferences on Artificial Intelligence Organization, Aug. 2019, pp. 4739–4745. doi: 10.24963/ijcai.2019/658.

[9] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "SwissLog: Robust anomaly detection and localization for interleaved unstructured logs," *IEEE Trans. Dependable Secur. Comput.*, vol. 20, no. 4, pp. 2762–2780, Jul. 2023. doi: 10.1109/TDSC.2022.3162857.

[10] S. He, T. Deng, B. Chen, R. Simon Sherratt, and J. Wang, "Unsupervised log anomaly detection method based on multi-feature," *Comput. Mater. Contin.*, vol. 76, no. 1, pp. 517–541, 2023. doi: 10.32604/cmc.2023.037392.

[11] Q. Wang, B. Wu, P. Zhu, P. Li, W. Zuo and Q. Hu, "ECA-Net: Efficient channel attention for deep convolutional neural networks," in *2020 IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Seattle, WA, USA, IEEE, Jun. 2020, pp. 11531–11539. doi: 10.1109/CVPR42600.2020.01155.

[12] Q. Fu, J. -G. Lou, Y. Wang, and J. Li, "Execution anomaly detection in distributed systems through unstructured log analysis," in *2009 Ninth IEEE Int. Conf. Data Min.*, Miami Beach, FL, USA, IEEE, Dec. 2009, pp. 149–158. doi: 10.1109/ICDM.2009.60.

[13] L. Tang, T. Li, and C. -S. Perng, "LogSig: Generating system events from raw textual logs," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag., CIKM'11*, New York, NY, USA, Association for Computing Machinery, Oct. 2011, pp. 785–794. doi: 10.1145/2063576.2063690.

[14] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *2017 IEEE Int. Conf. Web Serv. (ICWS)*, Honolulu, HI, USA, IEEE, Jun. 2017, pp. 33–40. doi: 10.1109/ICWS.2017.13.

[15] M. Du and F. Li, "Spell: Streaming parsing of system event logs," in *2016 IEEE 16th Int. Conf. Data Min. (ICDM)*, Barcelona, Spain, IEEE, Dec. 2016, pp. 859–864. doi: 10.1109/ICDM.2016.0103.

[16] W. Yuan, S. Ying, X. Duan, H. Cheng, Y. Zhao and J. Shang, "PVE: A log parsing method based on VAE using embedding vectors," *Inf. Process. Manag.*, vol. 60, no. 5, pp. 103476, Sep. 2023. doi: 10.1016/j.ipm.2023.103476.

[17] C. Bertero, M. Roy, C. Sauvanaud, and G. Tredan, "Experience report: Log mining using natural language processing and application to anomaly detection," in *2017 IEEE 28th Int. Symp. Softw. Reliab. Eng. (ISSRE)*, Toulouse, IEEE, Oct. 2017, pp. 351–360. doi: 10.1109/ISSRE.2017.43.

[18] X. Zhang *et al.*, "Robust log-based anomaly detection on unstable log data," in *Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Tallinn Estonia, ACM, Aug. 2019, pp. 807–817. doi: 10.1145/3338906.3338931.

[19] Y. Fu, K. Liang, and J. Xu, "MLog: Mogrifier LSTM-based log anomaly detection approach using semantic representation," *IEEE Trans. Serv. Comput.*, vol. 16, no. 5, pp. 3537–3549, Sep. 2023. doi: 10.1109/TSC.2023.3289488.

[20] C. Zhang *et al.*, "LayerLog: Log sequence anomaly detection based on hierarchical semantics," *Appl. Soft. Comput.*, vol. 132, no. 6, pp. 109860, Jan. 2023. doi: 10.1016/j.asoc.2022.109860.

[21] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," presented at the Int. Conf. Learn. Rep., Jul. 2022. Accessed: Jun. 27, 2024. [Online]. Available: https://openreview.net/forum?id=SyK00v5xx

[22] S. Lu, X. Wei, Y. Li, and L. Wang, "Detecting anomaly in big data system logs using convolutional neural network," in *2018 IEEE 16th Int. Conf. Dependable, Auton. Secure Comput., 16th Int. Con. Pervasive Intell. Comput., 4th Int. Conf. Big Data Intell. Comput. Cyber Sci. Technol. Congr. (DASC/PiCom/DataCom/CyberSciTech)*, Athens, IEEE, Aug. 2018, pp. 151–158. doi: 10.1109/DASC/PiCom/DataCom/CyberSciTec.2018.00037.

[23] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis and H. Zhang, "Automated IT system failure prediction: A deep learning approach," in *2016 IEEE Int. Conf. Big Data (Big Data)*, Washington, DC, USA, IEEE, Dec. 2016, pp. 1291–1300. doi: 10.1109/BigData.2016.7840733.

[24] J. Zhu, S. He, P. He, J. Liu, and M. R. Lyu, "Loghub: A large collection of system log datasets for AI-driven log analytics," in *2023 IEEE 34th Int. Symp. Softw. Reliab. Eng. (ISSRE)*, Florence, Italy, IEEE, Oct. 2023, pp. 355–366. doi: 10.1109/ISSRE59848.2023.00071.