Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.060740

ARTICLE

LogDA: Dual Attention-Based Log Anomaly Detection Addressing Data Imbalance

Chexiaole Zhang and Haiyan Fu*

School of Information Science and Technology, Hainan Normal University, Haikou, 571158, China

*Corresponding Author: Haiyan Fu. Email: 240058@hainnu.edu.cn

Received: 08 November 2024; Accepted: 29 January 2025; Published: 26 March 2025

ABSTRACT: As computer data grows exponentially, detecting anomalies within system logs has become increasingly important. Current research on log anomaly detection largely depends on log templates derived from log parsing. Word embedding is utilized to extract information from these templates. However, this method neglects a portion of the content within the logs and confronts the challenge of data imbalance among various log template types after parsing. Currently, specialized research on data imbalance across log template categories remains scarce. A dual-attention-based log anomaly detection model (LogDA), which leveraged data imbalance, was proposed to address these issues in the work. The LogDA model initially utilized a pre-trained model to extract semantic embedding from log templates. Besides, the similarity between embedding was calculated to discern the relationships among the various templates. Then, a Transformer model with a dual-attention mechanism was constructed to capture positional information and global dependencies. Compared to multiple baseline experiments across three public datasets, the proposed approach could improve precision, recall, and F1 scores.

KEYWORDS: Anomaly detection; system log; deep learning; transformer; neural networks

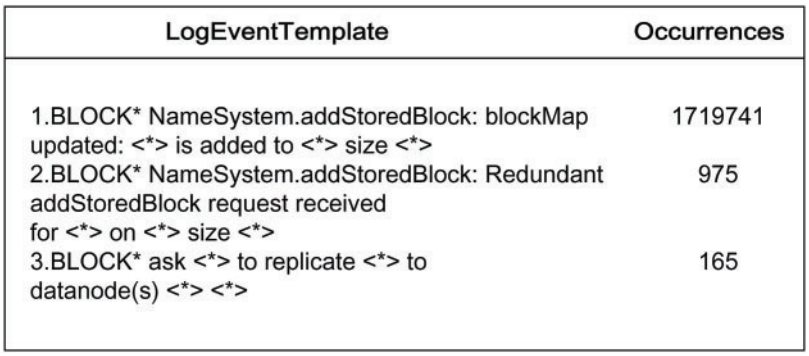
1 Introduction

Information technology has experienced significant growth with the arrival of the big data era. The exponential growth in data volume and the frequency of data exchanges highlight the need for a stable and secure network. Currently, most application systems are required to operate without interruption. Server overloads, application errors, or external data attacks affect millions of users, underscoring the critical importance of rapid error detection [1]. Given the importance of log anomaly detection, an increasing number of studies have been proposed. Initially, keyword matching or regular expressions [2,3] are used to manually detect log anomalies. Recently, deep learning models have been widely utilized for automatic anomaly detection [4,5]. Effective methodologies have been proposed, including the mining of log sequence patterns and word embeddings to extract semantic features. However, these methods fail to adequately address imbalanced log template data caused by log parsers. Some fundamental challenges remain as follows:

1. The impact of data noise. Currently, most log anomaly detection tasks rely on standard log parsing tools [6,7]. However, the use of these tools or the process of collecting and processing log data may encounter issues like data loss, duplication, and errors, which introduce a certain level of noise. This noise adversely affects the performance of log anomaly detection.
2. Imbalance in data across different types of log templates. Various types of log templates are generated after processing raw log data using log parsing tools. These log templates exhibit a significant imbalance

in data distribution (Fig. 1). However, most existing studies fail to address or adequately address the log-template class imbalance.

3. The absence of semantic vectors. Numerous approaches utilize word embeddings to obtain semantic vectors for log templates in many log anomaly detection tasks. However, this method discards some information from log templates, which results in incomplete semantic vectors.



LogEventTemplate	Occurrences
1.BLOCK* NameSystem.addStoredBlock: blockMap updated: <*> is added to <*> size <*>	1719741
2.BLOCK* NameSystem.addStoredBlock: Redundant addStoredBlock request received for <*> on <*> size <*>	975
3.BLOCK* ask <*> to replicate <*> to datanode(s) <*> <*>	165

Figure 1: Distribution of occurrences for some templates in the HDFS logs. The highest template occurrence reaches 1,719,741 times, while the lowest is only 165 times. '*' represents variables in the original logs

Based on dual attention and data imbalance (LogDA), a novel log anomaly detection model was proposed to address the aforementioned issues. A pre-trained model was used to obtain the semantic vectors, which captured the relationships between log templates. Drawing inspiration from research in text similarity, the work utilized cosine similarity to compute the relationships between vectors, constructing a scoring matrix in the process. These relationships were then integrated into the semantic vectors, which captured connections among various types of log templates. A dual-attention Transformer model equipped with a flexible and resource-efficient mechanism was developed to capture crucial log information. This design enhanced the extraction of both global and local details. The proposed approach was evaluated using real-world open-source datasets, including BGL, HDFS, and Thunderbird. The contributions of the work can be summarized as follows:

1. The work proposed the LogDA model, a dual-attention Transformer model based on data imbalance. The model could capture key information in log anomaly detection and reduce the impact of noise by calculating the similarity between templates and employing a dual-attention mechanism.
2. Most log anomaly detection approaches used word embedding to generate semantic vectors, which led to a loss of log information. The LogDA model leveraged a pre-trained model to extract sentence embeddings for log templates, which minimized semantic information loss. Additionally, data imbalance among log template categories was addressed by constructing a scoring matrix using the similarity of sentence embeddings.
3. A new perspective on log anomaly detection was presented, focusing on the highly imbalanced types of log templates generated by log parsers. The similarity between log template types was calculated to construct a scoring matrix capturing their interrelations. A Transformer model was designed with a dual-attention mechanism to better capture these relationships, which mitigated log-template data imbalance.
4. The LogDA model achieved state-of-the-art performance on three public benchmark datasets.

2 Related Work

Currently, most system logs are automatically generated and consist of multiple specific events that are represented as unstructured text. The task of log parsing is to extract these important events from the unstructured text and reconstruct them into structured text for subsequent log anomaly detection analysis (Fig. 2). In recent years, Drain [6] and Spell [7] are the most commonly used log parsing tools in various log anomaly detection tasks. Drain is a log parsing method based on deep parsing trees that uses regular expressions to separate constants from variables and extract parameters [8]. Spell is a log parsing method based on the longest common subsequence of log sequences. Since log parsing significantly impacts subsequent anomaly detection, and Drain shows high accuracy [9], Spell will be used in our future work.

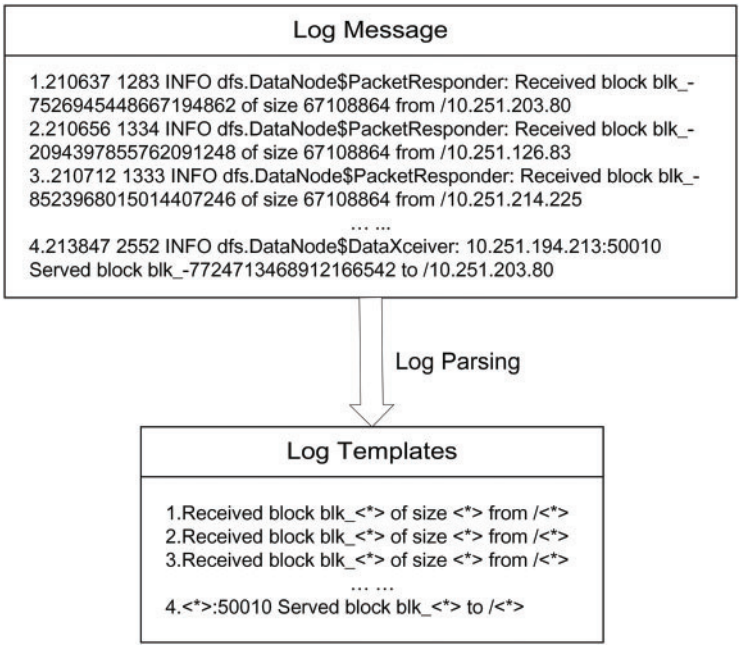


Figure 2: Log parsing process. The original logs are shown at the top, and the log templates extracted using Drain are displayed below. '*' represents variables in the original logs

These templates need to be converted into feature vectors for log anomaly detection after transforming the original logs into structured log templates through log parsing. The methods for log anomaly detection have gradually shifted from manual identification to machine learning and deep learning techniques. Some prominent machine learning methods in log anomaly detection include support vector machine (SVM) [10], decision trees [11], and principal component analysis (PCA) [12]. However, traditional machine learning methods fail to capture semantic information in log texts, which limits their effectiveness with increasing log volumes. Consequently, researchers have proposed numerous deep learning-based methods for log anomaly detection. Recurrent neural networks (RNNs) have become particularly prominent, especially long short-term memory networks (LSTM) [4,5], bidirectional long short-term memory (Bi-LSTM) [13–15], and gated recurrent units (GRUs). For instance, DeepLog [4] uses LSTM to predict whether the next log sequence is anomalous; if fails, the sequence is considered anomalous. Logs are characterized as being unstable and in continuous change [13]. A Bi-LSTM model is employed to capture contextual information within log sequences.

In addition to recurrent neural networks (RNNs) for log anomaly detection, Refs. [16–18] employ bidirectional encoder representations from transformers (BERT). Guo et al. introduced LogBert [15], a method that establishes two self-supervised training tasks specifically for log anomaly detection. Addressing the challenge of imbalanced learning in log anomaly detection, the most significant contributions include the MLog method proposed by Fu et al. [19], as well as research conducted by Yan et al. [20]. Recently, the advancement of large-scale models has significantly influenced the landscape of log anomaly detection research. He et al. introduced a method involving the fine-tuning of these large models, aiming to reduce the parameter count in log anomaly detection models [21,22]. Table 1 compares the work with diverse deep-learning approaches, focusing on their ability to handle the discrete attributes of logs, effectiveness in mitigating the loss of semantic information, and strategy for addressing the imbalance present among various log template types. However, most models for log anomaly detection rely on word embeddings to extract semantic vectors. This approach does not address data imbalance among log templates post-parsing. Therefore, the similarity between template types was calculated to enrich the semantic vectors with semantic similarity in the work. Besides, a dual-attention Transformer was used to capture semantic information for anomaly detection.

Table 1: Comparison with previous work

Methods	Discrete	Semantic loss	Imbalance templates
DeepLog	×	×	×
LogAnomaly	√	×	×
LogRobust	√	×	×
LogBert	√	×	×
LogDA (Ours)	√	√	√

3 Approach

This section presents the overall framework of LogDA and explains its workflow from log parsing, semantic vectorization, and classification based on dual attention.

3.1 Overview

As an innovative log anomaly detection method, LogDA was proposed to address the earlier issues. Fig. 3 shows the overall structure of LogDA. The procedure began with log parsing, aiming at deriving structured templates. Subsequently, a pre-trained model was utilized to extract the semantic vectors of these templates. LogDA was used to construct a scoring matrix, which captured relationships between various log template categories and handled data imbalance among different types of log templates post-parsing. Then LogDA employed a dual-attention Transformer for anomaly detection, which captured both the global semantic information of sequences and the relationships between templates. This method could address data imbalance among log templates.

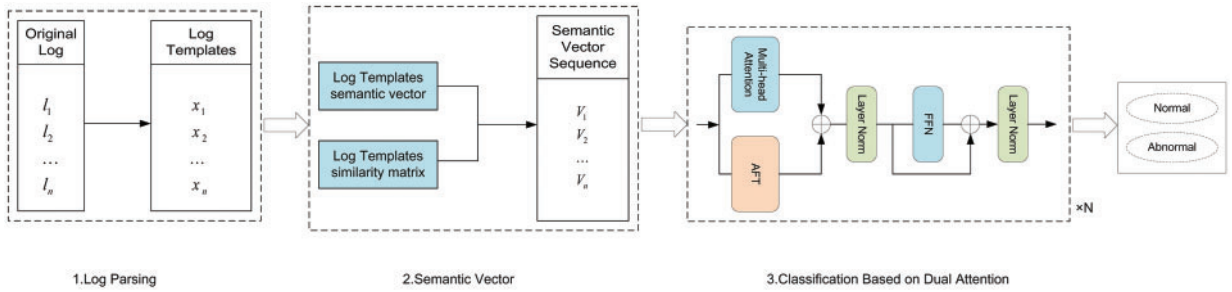


Figure 3: Overall structure of the LogDA model. Unprocessed logs are transformed into templates through the Drain log parser. These templates are then vectorized utilizing a pre-trained model, which facilitates the calculation of a similarity matrix. The fused semantic vectors are then input to a dual-attention classifier for anomaly detection

3.2 Log Parsing

Raw logs, typically unstructured, consist of a log header and log content [23]. In Fig. 2, the first log entry, "210637 1283 INFO dfs.DataNode$PacketResponder: Received block blk_-7526945448667194862 of size 67108864 from/10.251.203.80," has log header "210637 1283 INFO dfs.DataNode$PacketResponder:" and log content "Received block blk_-7526945448667194862 of size 67108864 from/10.251.203.80." Both the log header and content consist of constant and variable words. Constant words are fixed in the log sequence, while variable words record information like runtime details. In log parsing, variable words in the raw logs are removed, only with constant words retained. For instance, the parsed log template in the figure is "Received block blk_<*> of size <*> from/<*>". Here, the variable numeric details are substituted with wildcards, which constitute the final output.

3.3 Semantic Vectorization

Although structured log templates can be obtained through log parsing, they cannot be directly used for training in log anomaly detection tasks. Log templates must be vectorized into semantic vectors for computer learning. Moreover, the semantic vectorization procedure is segmented into two distinct phases (Fig. 4). This approach is designed to mitigate data imbalance across various log template types. Thus, semantic vectors encapsulate both the original meanings and the interrelationships between different templates. These include log template vectorization and log scoring matrix construction.

3.3.1 Log Template Vectorization

The log templates are converted into fixed-dimensional vectors of dimension d in the process of log template vectorization. Pre-trained model Sentence-BERT [24] is used to vectorize log templates in the LogDA task. The model employs vectorization for log templates at the sentence level, ensuring the retention of their parameters. This method preserves the entire semantic content of the templates. The embedding dimension of the pre-trained model is set to 768 by default. X represents the list of log templates obtained after parsing from different datasets, containing n log template entries. The resulting vectors can be represented as follows after vectorization.

$$E = VE(X) \tag{1}$$

where VE represents vectorization performed by the pre-trained model. The semantic vectors of the log templates are obtained after vectorization, denoted as E, $E \in R^{n \times d}$.

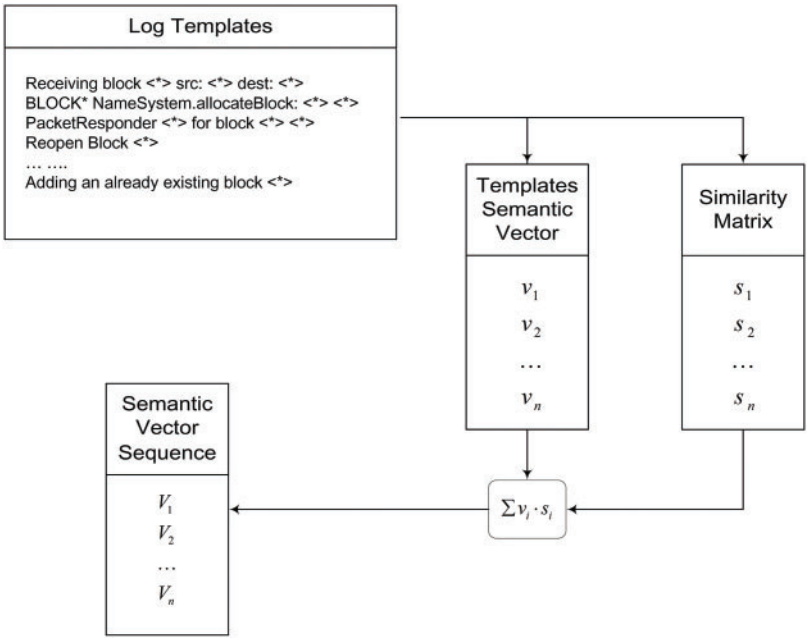


Figure 4: Semantic vectorization flowchart. Log templates are vectorized, and a similarity matrix between the log templates is computed. Weighted fusion is applied to generate new semantic vectors that retain the original semantics and relationships among different log templates. '*' represents variables in the original log

3.3.2 Construction of the Log Scoring Matrix

Once the log templates are transformed into semantic vectors, LogDA maps the interconnections among them by creating a similarity matrix. This matrix is found in the likeness between various types of log templates. The log templates with the highest and lowest similarity are selected to the template "instruction cache parity error corrected" in the BGL dataset (Fig. 5). The high similarity with the target log template is attributed to both involving parity error issues in the cache, which causes the system to perceive them as similar. Semantic vector $E = [v_1, v_2, \ldots, v_N]$, where $v_i \in R^d$; $i \in [1, N]$ represents the semantic vectors of different log templates; N represents the total number of log template types. The similarity between different log templates is calculated using a semantic vector E.

$$Log - similarity\left(v_i, v_j\right) = \frac{v_i \cdot v_j}{\|v_i\| \left\|v_j\right\|} \tag{2}$$

where v_i, v_j represent the semantic vectors of different log templates within the set of E. Similarity matrix $S, S \in R^{n \times n}$ is obtained by calculating the similarity between various templates, capturing the relationships among different types of log templates. The similarity matrix requires computing the cosine distance between all log templates. This approach results in the computational complexity of $O\left(N^2 \cdot d\right)$, where N is the number of templates; d is the embedding dimension (768). In the case of exceptionally large datasets, particularly where the number of log templates spans tens or even hundreds of thousands, the computational load becomes substantial. However, such large datasets are uncommon in real-world scenarios. Most log template datasets, such as all datasets used in this experiment, only contain a few thousand templates. Then the impact of computational complexity is negligible.

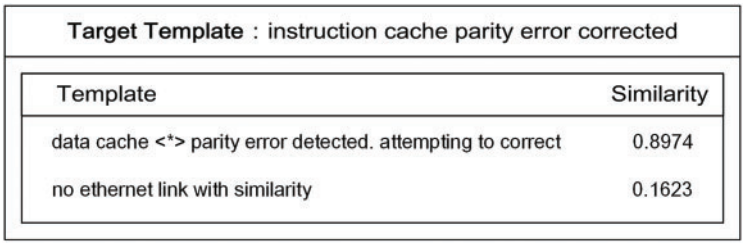


Figure 5: Log templates most similar and least similar to the target log template, along with their similarities. The log templates are selected from the BGL dataset. Log templates that are highly similar share the same type of issues as the target template, which results in their high similarity. '*' represents variables in the original log

Most log templates exhibit low similarity with one another in the similarity matrix. Lacking adequate precautions, computing the similarity across all log templates results in a substantial expansion of data volume. This, in turn, will intensively deplete resources and severely diminish computational efficiency. For instance, when incorporating the scoring matrix, the duration required for semantic vectorization within the BGL dataset is 433.48 s, whereas it is 276.96 s when excluding the scoring matrix. A threshold is set in the experiment to retain only high similarity, improve computation, and filter out noise. Extensive experiments indicate that the threshold range is set between 0.90 and 0.95 for the HDFS and Thunderbird datasets with log entries exceeding 10 M. Setting the threshold too low can result in excessive use of resources and decreased efficiency in computations. The threshold range is between 0.75 and 0.80 for the BGL dataset. Given that the BGL dataset has a limited number of log entries (just 5 M), setting a threshold too high will result in the loss of most relationships between log templates, which reduces performance. Additionally, data are normalized to ensure numerical stability and facilitate subsequent processing, which enhances the accuracy of later analysis and processing. Thus, a similarity matrix S' is obtained. Finally, based on Eq. (3), the log semantic vector matrix V is obtained by weighting sentence vectors with the similarity matrix. S' represents the relationships among different types of log templates.

$$V = S' \cdot E \tag{3}$$

The final semantic vectors contain the similarity between different types of log templates and the original semantic information of log templates. In this way, LogDA can capture the relationships between different types of log templates, preserving the original semantic information of log templates.

3.4 Classification Based on Dual Attention

Log templates are transformed into semantic vectors after semantic vectorization., Semantic vector matrix V forms, and each row represents the semantic vector of a distinct log template (e.g., $[V_1, V_2, \ldots, V_n]$). Using these semantic vectors as input, LogDA employs a Transformer model with a dual attention mechanism to capture information within semantic vectors. The Transformer model significantly affects artificial intelligence development due to its self-attention mechanism [25]. It can process log information across the input sequence and capture relationships between logs. Nevertheless, the multi-head self-attention mechanism possesses certain limitations; it emphasizes global information at the expense of local detail capture. Attention-free transformer (AFT) [26], an efficient, low-complexity mechanism for capturing local information, is used to address these shortcomings. Similar to the multi-head attention mechanism, AFT

applies different linear layers to the input features, which obtains Q, K, V ($Q, K, V \in R^{n \times d}$). Learnable positional encoding $w_{t',t}$ is added to $K_{t'}$ in AFT, and $1 \leq t' \leq n$. Moreover, the traditional matrix multiplication in the Transformer model is adapted to an element-wise multiplication, which yields the final output.

$$\alpha_1 = softmax\left(\frac{QK^T}{\sqrt{d/h}}\right) \tag{4}$$

$$\alpha_2 = sigmoid\left(Q_t\right) \odot \frac{\sum_{t'=1}^{n} \exp\left(K_{t'} + w_{t,t'}\right) \odot V_{t'}}{\sum_{t'=1}^{n} \exp\left(K_{t'} + w_{t,t'}\right)} \tag{5}$$

$$\alpha = \alpha_1 + \alpha_2 \tag{6}$$

Attention values α_1 and α_2 are combined through residual connection to obtain the final attention value α. α_1 and α_2 represent the outputs from the multi-head attention mechanism and the AFT attention mechanism, respectively. The deep learning model dynamically fine-tunes the output weight of the two attention mechanisms by summing their attention values via a residual connection while training. This process, driven by gradient updates, shapes the final output of the model.

The probability of log anomalies (y) is obtained through the feedforward sublayer and subsequent normalization layer. The binary cross-entropy (BCE) loss function is employed, enabling the direct computation of logits without the necessity of separately applying the sigmoid function. This approach is more efficient and stable.

$$BCEWithLogitsLoss\left(y, \hat{y}\right) = \frac{1}{N}\sum_{i=1}^{N}\left(y_i \cdot \log\left(\sigma\left(\hat{y}_i\right)\right) + \left(1 - y_i\right) \cdot \log\left(1 - \sigma\left(\hat{y}_i\right)\right)\right) \tag{7}$$

where y_i is the true label (0 or 1); $\hat{y}_i$ represents the model's output logit; Eq. (8) is the sigmoid function that converts logits to 0–1.

$$\sigma\left(\hat{y}_i\right) = \frac{1}{1 + e^{-y_i}} \tag{8}$$

4 Experiments

This section compares LogDA on multiple datasets against existing methods. Then, ablation experiments and parameter analysis are performed on LogDA.

4.1 Experimental Design

4.1.1 Datasets

Three public datasets from LogHub [27] (e.g., HDFS [28], BGL, and Thunderbird [29]) were used for evaluation. The Hadoop distributed file system (HDFS) datasets consisted of 11,175,629 logs generated by Hadoop. Logs were manually annotated using tailored rules to differentiate between normal and abnormal events. The BGL dataset, collected from the BlueGene/L supercomputer system at Lawrence Livermore National Laboratory (LLNL), comprised 4,747,963 log messages, each categorized for analysis. Thunderbird, gathered from the Thunderbird supercomputer system at Sandia National Laboratories (SNL) in Albuquerque, also included similar labels to the BGL dataset. In this experiment, 10,000,000 log messages from Thunderbird were used as the research dataset.

Table 2 provides statistical data for these three log datasets. Log sequences for the HDFS dataset were extracted using block_IDs while sliding window methods were employed for the BGL and Thunderbird datasets. Log parsing was conducted with the Drain and custom regular expressions.

Table 2: Log dataset configurations

Datasets	Duration	Messages	Templates
HDFS	38.7 h	11 M	48
BGL	7 months	5 M	462
Thunderbird	244 days	10 M	4992

4.1.2 Evaluation Metrics

Precision, Recall, and F1 score are used to evaluate the log anomaly detection experiments (Eqs. (9)–(11)).

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

$$F1 = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} * \text{Recall}} \tag{11}$$

where TP represents the number of abnormal logs correctly detected as abnormal; TN represents the number of normal logs correctly detected as normal; FP represents the number of abnormal logs incorrectly detected as normal; FN represents the number of normal logs incorrectly detected as abnormal.

4.1.3 Environment Setup

The experimental configuration comprised a Windows 10 64-bit operating system, equipped with 32 GB of RAM, an 11th Gen Intel® Core™ i7-11700 CPU running at 2.50 GHz, and an Nvidia RTX3090Ti GPU. In the course of the experiments, I explored different configurations of the Transformer encoder, ranging from 1 to 4 layers. The batch size was consistently set at 64, with the model utilizing 8 attention heads. For optimization, the Adam algorithm was employed.

4.2 Experimental Evaluation

This section validates the performance of LogDA on datasets HDFS, BGL, and Thunderbird. LogDA was compared with various machine learning methods in the experiments, including invariant mining (IM) [30], principal component analysis (PCA), and deep learning methods (e.g., DeepLog, LogAnomaly, LogRobust, and LogBert). For the machine learning methods, the work utilized the loglizer package [31] for evaluation, while the deep learning methods were implemented using the open-source toolkits LogADEmpirical [32] and log deep. Table 3 displays the comparative outcomes of various methodologies, including LogDA, on the HDFS, BGL, and Thunderbird datasets.

Table 3: Performance of different methods on various datasets

Datasets	HDFS			BGL			Thunderbird		
	Precision	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**	**Precision**	**Recall**	**F1 score**
IM	0.77	**1.00**	0.87	0.44	0.25	0.32	0.50	0.05	0.09
PCA	0.99	0.26	0.41	**1.00**	0.23	0.38	1.00	0.19	0.32
DeepLog	0.96	0.25	0.40	0.92	0.78	0.84	0.88	0.99	0.93
LogAnomaly	0.97	0.40	0.57	0.89	0.79	0.84	0.88	0.99	0.93
LogRobust	0.95	0.95	0.95	0.96	0.93	0.94	0.66	**1.00**	0.80
LogBert	0.71	0.78	0.74	0.86	0.92	0.89	0.98	0.95	0.96
LogDA	**0.99**	0.99	**0.99**	0.99	**0.97**	**0.98**	**1.00**	0.97	**0.98**

Note: Bold numbers represent the best performance.

Experimental data in Table 3 show that traditional machine learning methods (e.g., PCA and IM) exhibit poor performance in log anomaly detection, especially on the BGL and Thunderbird datasets. Although these methods may yield high accuracy or recall rates, a common challenge in machine learning approaches is the trade-off between accuracy and recall, which leads to suboptimal F1 scores. This issue likely occurs because event-counting methods fail to capture relationships between log events in complex datasets. In contrast, deep learning approaches exhibit significantly superior performance compared to traditional machine learning techniques, with particularly impressive results observed on the BGL and Thunderbird datasets. Deep learning approaches can capture the relationships between logs in more complex datasets. Notably, LogDA attains exceptional performance across all three datasets, with metrics surpassing 0.95 and achieving 0.99 on HDFS, which outperforms the second-best baseline by a 4% margin. Table 3 illustrates that LogDA outperforms other log anomaly detection methods on all three datasets. LogDA is superior in capturing the interrelations between various log templates by handling diverse log templates and employing a dual attention mechanism, which enhances the precision of anomaly detection in logs. Specifically, LogDA's dual-attention mechanism in its Transformer encoder captures both local and global log information more effectively.

Fig. 6a–c presents the ROC curves and AUC values constructed for the BGL, Thunderbird, and HDFS datasets, which can better evaluate the LogDA model. The LogDA model achieves an AUC of 1.00 on both the Thunderbird and HDFS datasets and 0.99 on the BGL dataset. LogDA encompasses a more thorough understanding of log template semantics by leveraging pre-trained models and preprocessing diverse log template types. Furthermore, a dual-attention mechanism enhances its capability to discern the relationships between various log template types, which yields high performance in log anomaly detection.

The confusion matrix is an important evaluation method. Fig. 7a–c illustrates the confusion matrices for the LogDA model on the BGL (a), Thunderbird (b), and HDFS (c) datasets, based on the previously mentioned true positive (TP), true negative (TN), false positive (FP), and false negative (FN). The proportion of detected positive cases by LogDA in log anomaly detection is significantly higher than that of negative cases, indicating the effectiveness of the LogDA model.

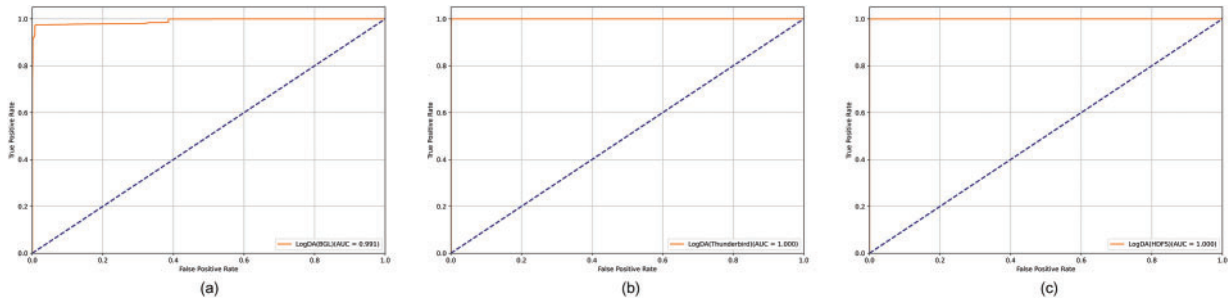


Figure 6: ROC curves and AUC values of the LogDA model on different datasets. (a) BGL dataset (AUC = 0.991); (b) Thunderbird dataset (AUC = 1.000); (c) HDFS dataset (AUC = 1.000). AUC represents the model's classification performance, with values closer to 1 indicating better classification effectiveness

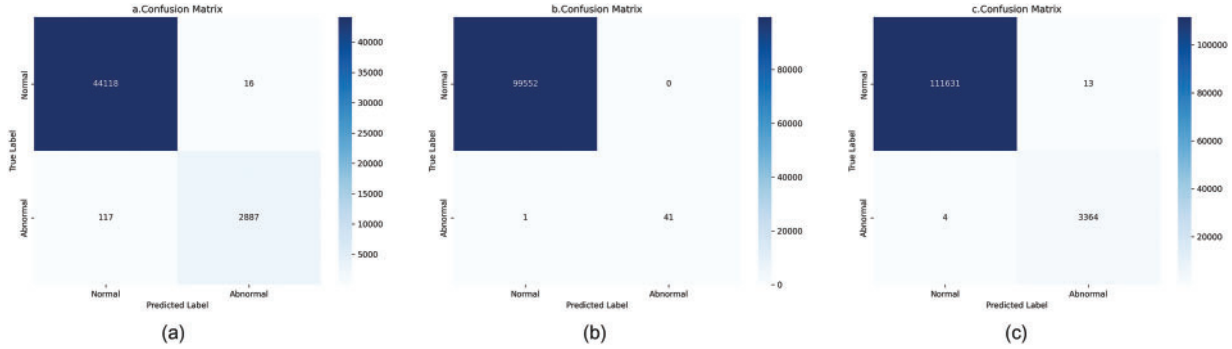


Figure 7: Confusion matrices of the LogDA model on different datasets. (a) BGL dataset; (b) Thunderbird dataset; (c) HDFS dataset. "True" and "False" represent correct and incorrect predictions, respectively. The high True counts across all datasets indicate the model's high accuracy

4.3 Ablation Study

The experiment compared the performance of the traditional Transformer encoder and the dual attention mechanism encoder on the BGL and Thunderbird datasets to validate the effectiveness of the proposed dual attention mechanism. As depicted in Fig. 8a, the dual attention mechanism exhibits slightly lower recall on the BGL dataset compared to the conventional Transformer encoder; however, it attains superior accuracy and F1 scores. Fig. 8b presents the performance of the traditional encoder and the dual attention encoder on the Thunderbird dataset. The dual attention mechanism encoder outperforms the traditional Transformer encoder, indicating that the dual attention mechanism can better capture the relationships between different types of log templates.

Fig. 9a shows the performance of the dual attention mechanism model with different neural network layer counts on the BGL dataset. Fig. 9b depicts the performance of the traditional Transformer encoder with varying neural network layer counts on the BGL dataset. F1 scores of the model employing the dual attention mechanism consistently surpass those of the traditional Transformer encoder across diverse layer configurations. Furthermore, as the layer count increases, the stacking of the traditional Transformer encoder adversely affects its performance, which leads to a decline. In contrast, the dual attention mechanism model maintains stable performance, which makes it more resilient to various uncertainties encountered in log anomaly detection.

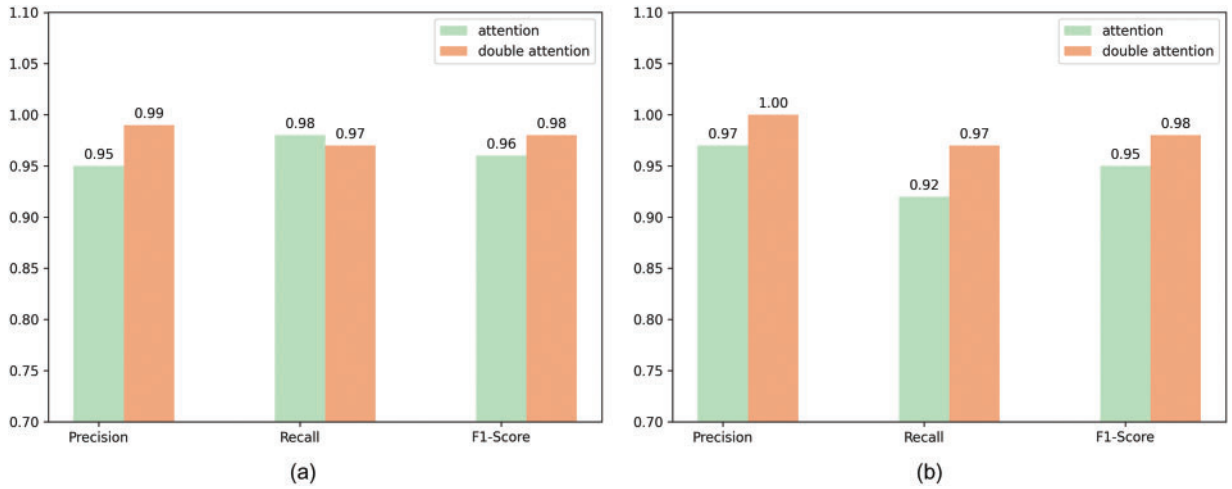


Figure 8: Performance of different attention mechanisms on various datasets. (a) BGL dataset; (b) Thunderbird dataset. The double attention mechanism achieves superior performance, demonstrating its enhanced ability to capture relationships between templates

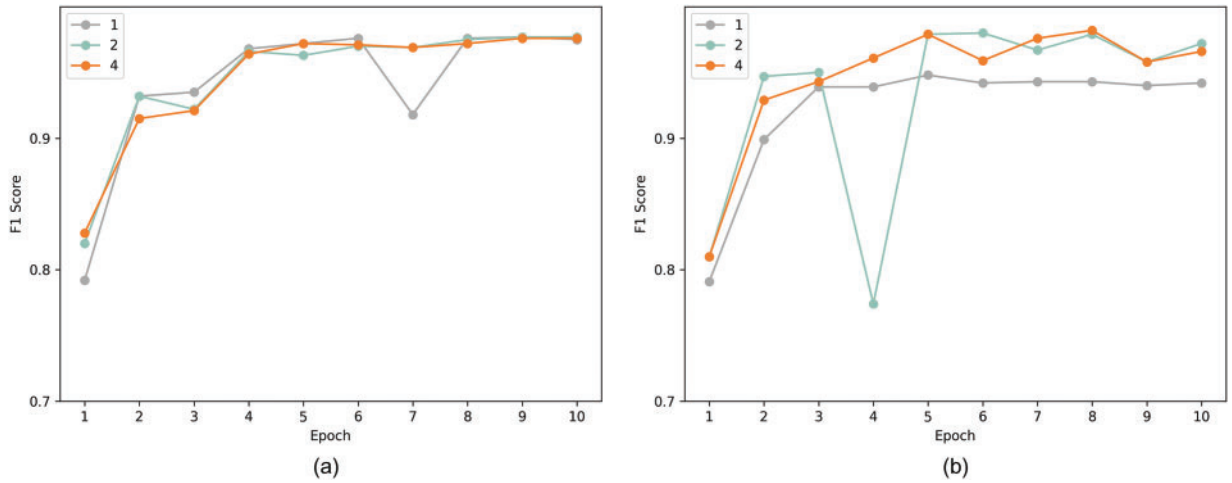


Figure 9: Comparison of F1 scores of different models with varying layers on the BGL dataset. (a) Dual-attention Transformer model; (b) Traditional Transformer model. The curve in (a) is smoother than in (b). The dual-attention mechanism model outperforms the traditional model with greater stability

The work selected thresholds of 0.75, 0.80, and 0.85 on the BGL dataset and conducted experiments utilizing both the traditional Transformer encoder and the proposed dual-attention mechanism, which investigated the effect of similarity threshold values on performance between different log template types. The increased threshold improved the F1 scores achieved by the dual-attention model (Fig. 10). Additionally, higher thresholds could reduce data resource consumption and unnecessary noise. However, the threshold should not be excessively high, as this might exclude important relationships between different types of log templates in semantic information. The adjusted threshold affected the generation of the similarity matrix. When the scoring matrix was within a reasonable range, the relationships between different types of log templates were captured without compromising the original semantics of the log templates.

Tasks with fewer than 100,000 training samples were regarded as low-resource conditions to assess the performance of the LogDA model in low-resource scenarios. The experiment was conducted on the Thunderbird dataset, with 30 epochs of training. F1 scores for different amounts of training samples ranging from 50 to 100 k were compared (Fig. 11). The dual-attention model consistently outperformed the traditional Transformer encoder, particularly with limited training data. For instance, with only 50 k training samples, the F1 score of the traditional Transformer encoder was 0.68, while the dual-attention model achieved an F1 score of 0.78. Similarly, with 100 k training samples, the F1 scores were 0.76 and 0.83, with the dual-attention model outperforming the traditional Transformer encoder. Moreover, the F1 scores obtained from the dual attention model remained more stable without significant fluctuations. The dual attention mechanism was effective in capturing the relationships between different types of log templates, even when the training samples were limited. Overall, LogDA provided acceptable results under low-resource settings. The model demonstrated considerable reliability in scenarios where computational resources were scarce or when the volume of anomalous logs was minimal.

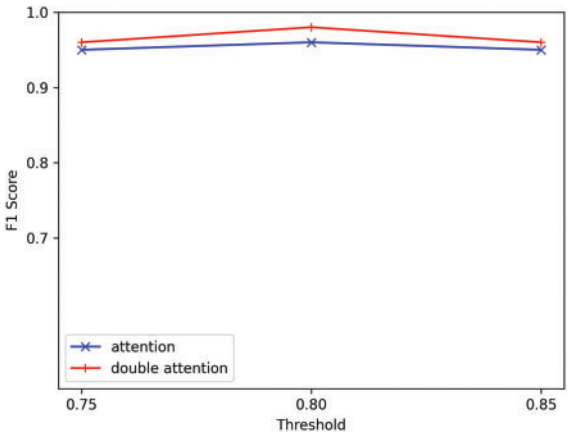


Figure 10: Impact of different thresholds on performance in the BGL dataset. Thresholds of 0.75, 0.80, and 0.85 are tested. The F1 score is highest at a threshold of 0.80, suggesting that both too-high and too-low thresholds hinder the model's ability to capture semantic information. A balanced threshold selection is essential

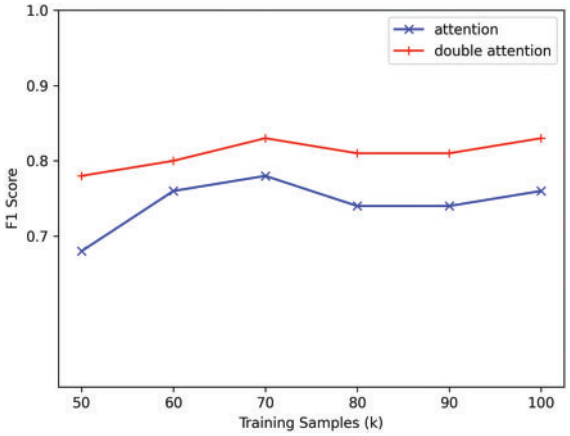


Figure 11: Comparison of F1 scores with different training samples on the Thunderbird dataset.50, 60, 70, 80, 90, and 100 k training samples are used. The dual-attention mechanism outperforms other mechanisms, particularly under resource-limited conditions

5 Conclusion

The work addressed the challenge of heterogeneity in various types of log template data by introducing LogDA, a log anomaly detection method leveraging the disparities within log template data. A scoring matrix was constructed based on different types of log templates and was integrated into the semantic vectors. These vectors were allowed to retain both their original semantic information and the relationships between log templates. Moreover, a dual-attention mechanism model, capturing the correlations among log templates, was presented to overcome the constraints of conventional Transformer encoders. The capture of relationships between different types of log templates was enhanced by constructing a scoring matrix and employing a dual-attention mechanism. Although this approach might entail increased computational complexity relative to other methods, its impact on large-scale log processing was minimized by employing varied threshold values and normalization techniques. The model maintained high performance even when handling large-scale log data. Comprehensive experiments have shown that LogDA excelled in capturing the intricate relationships between different types of log templates, thanks to its integration of pre-trained models, sophisticated preprocessing of template interrelations, and the innovative dual-attention mechanism. This enabled LogDA to achieve superior performance in log anomaly detection, enhancing the stability of the anomaly detection process.

However, the challenge of increased training parameters and computational load remains when dealing with large-scale datasets. Therefore, our future research will prioritize the reduction of parameter size in model training through the application of fine-tuning techniques derived from larger models. This approach aims to simultaneously increase the efficiency and accuracy of log anomaly detection methods.

Acknowledgement: The authors would like to express gratitude to all reviewers for their valuable feedback on the work.

Funding Statement: The work was funded by the Hainan Provincial Natural Science Foundation Project (Grant No. 622RC675) and the National Natural Science Foundation of China (Grant No. 62262019).

Author Contributions: The authors confirm contribution to the work as follows: Conception and design: Chexiaole Zhang; Data collection: Chexiaole Zhang; Analysis and interpretation of results: Chexiaole Zhang; Draft manuscript preparation: Chexiaole Zhang; Supervision: Haiyan Fu. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: This study utilized publicly available datasets. The research data can be accessed at https://github.com/logpai/loghub (accessed on 20 March 2024).

Ethics Approval: Not applicable.

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

References

1. Su J, Jiang C, Jin X, Qiao Y, Xiao T, Ma H, et al. Large language models for forecasting and anomaly detection: a systematic literature review. arXiv:2402.10350. 2024.
2. Cherkasova L, Ozonat K, Mi N, Symons J, Smirni E. Automated anomaly detection and performance modeling of enterprise applications. ACM Trans Comput Syst. 2009;27(3):1–32. doi:10.1145/1629087.1629089.
3. Yen TF, Oprea A, Onarlioglu K, Leetham T, Robertson W, Juels A, et al. Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Proceedings of the 29th Annual Computer Security Applications Conference; 2013 Dec 08–12; New Orleans, LA, USA. New York, NY, USA: ACM; 2013. p. 199–208. doi:10.1145/2523649.2523670.

4. Du M, Li F, Zheng G, Srikumar V. DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security; 2017 Oct 30–Nov 3; Dallas, TX, USA. New York, NY, USA: ACM; 2017. p. 1285–98. doi:10.1145/3133956.3134015.
5. Meng W, Liu Y, Zhu Y, Zhang S, Pei D, Liu Y, et al. LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence; 2019 Aug 10–16. Macao, China. p. 4739–45. doi:10.24963/ijcai.2019/658.
6. He P, Zhu J, Zheng Z, Lyu MR. Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS); 2017 Jun 25–30; Honolulu, HI, USA. p. 33–40. doi:10.1109/ICWS.2017.13.
7. Du M, Li F. Spell: online streaming parsing of large unstructured system logs. IEEE Trans Knowl Data Eng. 2019;31(11):2213–27. doi:10.1109/TKDE.2018.2875442.
8. Lee Y, Kim J, Kang P. LAnoBERT: system log anomaly detection based on BERT masked language model. Appl Soft Comput. 2023;146:110689. doi:10.1016/j.asoc.2023.110689.
9. Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, et al. Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP); 2019 May 25–31; Montreal, QC, Canada. p. 121–30. doi:10.1109/ICSE-SEIP.2019.00021.
10. Liang Y, Zhang Y, Xiong H, Sahoo R. Failure prediction in IBM BlueGene/L event logs. In: Seventh IEEE International Conference on Data Mining (ICDM 2007); 2007 Oct 28–31; Omaha, NE, USA. p. 583–8. doi:10.1109/ICDM.2007.46.
11. Chen M, Zheng AX, Lloyd J, Jordan MI, Brewer E. Failure diagnosis using decision trees. In: International Conference on Autonomic Computing; 2024 May 17–18; New York, NY, USA. p. 36–43. doi:10.1109/ICAC.2004.1301345.
12. Behera A, Panigrahi CR, Pati B. Unstructured log analysis for system anomaly detection—A study. In: Borah S, Mishra SK, Mishra BK, Balas VE, Polkowski Z, editors. Advances in data science and management. Singapore: Springer Nature Singapore; 2022. p. 497–509. doi:10.1007/978-981-16-5685-9_48.
13. Zhang X, Xu Y, Lin Q, Qiao B, Zhang H, Dang Y, 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; 2019 Aug 26–30; Tallinn, Estonia. New York, NY, USA: ACM; 2019. p. 807–17, doi:10.1145/3338906.3338931.
14. Zhang C, Wang X, Zhang H, Zhang J, Zhang H, Liu C, et al. LayerLog: log sequence anomaly detection based on hierarchical semantics. Appl Soft Comput. 2023;132:109860. doi:10.1016/j.asoc.2022.109860.
15. Yang L, Chen J, Wang Z, Wang W, Jiang J, Dong X, et al. PLELog: semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering: Companion Proceedings (ICSE-Companion); 2021 May 25–28; Madrid, Spain. p. 230–1. doi:10.1109/ICSE-Companion52605.2021.00106.
16. Qi J, Luan Z, Huang S, Fung C, Yang H, Li H, et al. LogEncoder: log-based contrastive representation learning for anomaly detection. IEEE Trans Netw Serv Manag. 2023;20(2):1378–91. doi:10.1109/TNSM.2023.3239522.
17. Guo H, Yuan S, Wu X. LogBERT: log anomaly detection via BERT. In: 2021 International Joint Conference on Neural Networks (IJCNN); 2021 Jul 18–22; Shenzhen, China. p. 1–8. doi:10.1109/IJCNN52387.2021.9534113.
18. Huang S, Liu Y, Fung C, Wang H, Yang H, Luan Z. Improving log-based anomaly detection by pre-training hierarchical transformers. IEEE Trans Comput. 2023;72(9):2656–67. doi:10.1109/TC.2023.3257518.
19. Fu Y, Liang K, Xu J. MLog: mogrifier LSTM-based log anomaly detection approach using semantic representation. IEEE Trans Serv Comput. 2023;16(5):3537–49. doi:10.1109/TSC.2023.3289488.
20. Yan L, Luo C, Shao R. Discrete log anomaly detection: a novel time-aware graph-based link prediction approach. Inf Sci. 2023;647:119576. doi:10.1016/j.ins.2023.119576.
21. He S, Lei Y, Zhang Y, Xie K, Sharma PK. Parameter-efficient log anomaly detection based on pre-training model and LORA. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE); 2023 Oct 9–12; Florence, Italy. p. 207–17. doi:10.1109/ISSRE59848.2023.00038.

22. Guo H, Yang J, Liu J, Bai J, Wang B, Li Z, et al. LogFormer: a pre-train and tuning pipeline for log anomaly detection. Proc AAAI Conf Artif Intell. 2024;38(1):135–43. doi:10.1609/aaai.v38i1.27764.
23. Yu S, He P, Chen N, Wu Y. Brain: log parsing with bidirectional parallel tree. IEEE Trans Serv Comput. 2023;16(5):3224–37. doi:10.1109/TSC.2023.3270566.
24. Reimers N, Gurevych I. Sentence-BERT: sentence embeddings using Siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP); 2019 Nov 3–7; Hong Kong, China. Stroudsburg, PA, USA: Association for Computational Linguistics; 2019. p. 3980–90.
25. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: 31st Conference on Neural Information Processing Systems (NIPS 2017); 2017 Dec 3–9; Long Beach, CA, USA.
26. Zhai S, Talbott W, Srivastava N, Huang C, Goh H, Zhang R, et al. An attention free transformer. arXiv:2105.14103. 2021.
27. Zhu J, He S, He P, Liu J, Lyu MR. Loghub: a large collection of system log datasets for AI-driven log analytics. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE); 2023 Oct 9–12; Florence, Italy. p. 355–66. doi:10.1109/ISSRE59848.2023.00071.
28. Xu W, Huang L, Fox A, Patterson D, Jordan MI, Xu W, et al. Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles; 2009 Oct 11–14; Big Sky, MT, USA. New York, NY, USA: ACM; 2009. p. 117–32. doi:10.1145/1629575.1629587.
29. Oliner A, Stearley J. What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07); 2007 Jun 25–28; Edinburgh, UK. p. 575–84. doi:10.1109/DSN.2007.103.
30. Lou JG, Fu Q, Yang S, Xu Y, Li J, Lou JG, et al. Mining invariants from console logs for system problem detection. In: 2010 USENIX Annual Technical Conference (USENIX ATC 10); 2010 Jun 23–25; Boston, MA, USA. New York, NY, USA: ACM; 2010.
31. He S, Zhu J, He P, Lyu MR. Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE); 2016 Oct 23–27; Ottawa, ON, Canada. p. 207–18. doi:10.1109/ISSRE.2016.21.
32. Le VH, Zhang H. Log-based anomaly detection with deep learning: how far are we? In: Proceedings of the 44th International Conference on Software Engineering; 2022 May 21–29; Pittsburgh, PA, USA. New York, NY, USA: ACM; 2022. p. 1356–67. doi:10.1145/3510003.3510155.