Hindawi
Security and Communication Networks
Volume 2023, Article ID 2803139, 12 pages
https://doi.org/10.1155/2023/2803139

Research Article

LogPal: A Generic Anomaly Detection Scheme of Heterogeneous Logs for Network Systems

Lei Sun [1] **and Xiaolong Xu** [1,2]

[1]*Jiangsu Key Laboratory of Big Data Security & Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*
[2]*School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China*

Correspondence should be addressed to Xiaolong Xu; xuxl@njupt.edu.cn

Received 5 February 2022; Revised 26 September 2022; Accepted 12 October 2022; Published 11 April 2023

Academic Editor: Lalit Garg

As a key resource for diagnosing and identifying problems, network syslog contains vast quantities of information. And it is the main source of data for anomaly detection of systems. Syslog presents the characteristics of large scale, diverse types and sources, data noise, and quick evolvement, which makes the detection methods not generic enough. To effectively address problem of log anomaly labelling caused by massive heterogeneous logs, we propose LogPal, a generic anomaly detection scheme of heterogeneous logs for network systems, which innovatively combines template sequences and raw log sequences to construct and generate log pattern events. By improving the self-attention mechanism of transformer, LogPal proactively synthesizes self-attention and handles log pattern events in a unique way. The model can make full use of log template and sequence semantic information, by automatically becoming aware of the pattern of logs. We implemented experiments to evaluate the performance of LogPal on publicly available datasets, and the outcome of the experiments shows that LogPal automatically adapts to log type changes and improves precision, recall, and $F1$ score to 99% on publicly available datasets.

1. Introduction

When the system is running, syslog is used to record the runtime state and events of the system, including the anomalies of the system. As the most reliable source of information for monitoring the health of a system, syslog contains massive amounts of information and is the main source of data for anomaly detection in the system [1]. For traditional standalone systems, developers write specific rules based on domain knowledge or manually check logs to detect system anomalies.

However, modern information systems usually adopt a distributed architecture. Syslog is multisourced and heterogeneous. Syslog usually originates from multiple subsystems with various types, structures, implementations, versions, and deployment environments [2, 3]. The approach to anomaly detection, which relies heavily on manual check of logs, is almost unworkable for large-scale system. Moreover, developers usually use free text to record system time for convenience and flexibility. Examples of heterogeneous logs are shown in Table 1.

More importantly, just like any other software maintenance, syslog is constantly evolving. Developers may frequently modify the source code, including logging statements. So, this can create a new log pattern that has not appeared and affected the results of anomaly detection. As Kabinna et al. [4] observed, in their research project, about 20%~45% of the logging statements changed during their lifecycle. Many new log events and log sequences are generated by dynamic logging statements.

Therefore, many automated anomaly detection methods based on logs have been proposed in recent years, and these methods are mainly classified into unsupervised learning and supervised learning. Unsupervised learning methods usually use machine learning techniques such as clustering and PCA [5–8], but unsupervised learning tends to be less

Table 1: Examples of heterogeneous logs.

Log type	Detailed message
Hadoop	2015-10-17 15 : 38 : 05,258 INFO [main] org.Apache.hadoop.metrics2.impl.MetricsSystemImpl: MapTask metrics system started
Thunderbird	2005.11.09 #8# Nov 9 12 : 20 : 55 #8#/#8# sshd[16228]: password authentication for user #41# accepted
Blue gene (L)	2005-06-03-15.42.50.363779 R02-M1-N0-C : J12-U11 RAS KERNEL INFO instruction cache parity error corrected

accurate compared to supervised learning methods. Supervised learning methods generally learn the anomaly patterns of logs based on anomaly labelling to achieve the purpose of anomaly detection. And supervised learning methods usually use deep learning methods such as LSTM and CNN [9–12]. Although some of the above methods can effectively detect anomalies, log sequence anomaly detection problems face the following challenges:

(1) It is rather difficult to achieve a balance between learning log templates and raw log semantic information. Thanks to the rapid development of natural language processing and deep learning, some methods build log anomaly detection methods based on raw log sequences when solving the heterogeneous log anomaly labelling problems, and there is hardly any parsing of the raw log sequences, making it difficult for models to utterly learn log word vector semantics or patterns. There are also approaches that parse the raw logs by extracting log sequence templates and use the log templates as input to build template-based anomaly detection network models. However, these approaches simply using log template sequences as training data to obtain word vectors for the templates, ignoring the key textual information specific to the raw logs, which can lead to more serious results. For example, two or more normal log sequences and anomalous log sequences are considered the same template by removing the critical variable part, and the model "considers" log sequences with different labels as the same input, which is quite fatal for anomaly detection. Therefore, how to make the model understand the log patterns more easily while retaining all the information of log semantics becomes one of the key issues for log-based anomaly detection.

(2) There is a large amount of noise in log data. A certain level of noise is inevitably interspersed in the collection and preprocessing of log data [13]. Log data are derived from various events that occur on distributed hardware and software systems. These events include both events that characterize the system as anomalous, such as being subject to DDoS attacks, storage failures, anomalous system behavior, and network jitter, and events that characterize the system as normal, such as successful ping sessions, successful subsystem startups, and file reads and writes. Since logs are usually generated by multiple processes or threads of the system, a log sequence often contains multiple normal/anomalous. This results in an anomalous log sequence often interspersed with one or more normal logs, presenting a significant challenge for log sequence anomaly detection. In addition, in large-scale systems, many logs are generated individually by geographically distributed components and then uploaded to a centralized location for further analysis. This collection process can lead to missing, duplicated, or disordered log sequences (e.g., due to network errors, limited system throughput, storage issues, etc.) [14]. A McKinsey network survey [15] found that 80% to 98% of logs are just noise, which makes processing and analyzing log data tricky. Noise in log data hinders the effectiveness of existing log-based anomaly detection methods.

(3) Accuracy and recall are still difficult to balance. In anomaly detection based on heterogeneous logs, the precision rate refers to the proportion of true anomalous logs among those predicted to be anomalous; the recall rate refers to the proportion of logs that are predicted to be anomalous among all true anomalous logs. As we all know, there exists a relation of "as one falls, another rises." It is an uphill battle to have both accuracy and recall. The system can generate hundreds of millions of system logs in just a few months, among which the anomalous logs can reach hundreds of thousands; even if there is a 1% error in the precision rate, there may be thousands of false positives, which is a great vexation for operations staff. Likewise, if the recall rate has 1% error, this means that there will be thousands of anomalous logs ignored, and some of them may be caused by fatal failures, which will cause serious losses. How to balance and improve the two is one of the most important challenges for researchers to overcome today.

To solve the above key challenges, in this study, we propose a generic anomaly detection mechanism for heterogeneous logs, called LogPal, which filters the raw system logs, then uses the FT-tree method to parse the log templates, and next splices the templates with the raw logs to generate log pattern events, thus realizing the automatic parsing of heterogeneous logs. Moreover, based on the semantic similarity of the anomalous sequences of heterogeneous logs, we combine natural language processing methods and deep learning methods to improve the transformer model to learn log patterns more adaptively and effectively to achieve anomaly detection of heterogeneous logs. The contributions of this study can be summarized in the following points:

(1) To address the difficult problem of balancing log templates and all semantic information of the raw

logs, a new log pattern event generation method for heterogeneous logs is proposed, which first filters the log sequences for noise reduction, then uses the FT-tree for template extraction, and then innovatively combines the filtered log sequences to build log pattern events, and the combined pattern events will consist of two parts (template number and filtered real logs).

(2) For the two parts that are different from each other by log pattern events, after embedding the pattern events into log pattern vectors, the synthetic attention approach is prospectively used to improve the transformer model to process log pattern events differently, so as to build a pattern-aware learning model for heterogeneous logs.

(3) To address the large amount of noise present in log sequences, in the synthetic attention part, the model's capability and computational complexity are balanced by the relative deviations of different tokens. The input tokens focus on each token, thinning out Tokens with different deviations away from it in a fine-to-coarse fashion, as a way to reduce or even ignore noise in the log sequence.

The rest of the study is organized as follows. Section 2 analyzes the work related to log-based anomaly detection. In Section 3, we introduce the framework of LogPal and the workflow of log parsing and anomaly detection in detail. Section 4 describes the experimental environment and datasets, evaluation indicators, experimental results, and the corresponding analysis. Section 5 concludes the study and looks forward to future work.

2. Related Work

The traditional machine learning approaches are playing an increasingly influential role in log anomaly detection. For example, Bodik et al. [16] use regression-based analysis techniques to automatically classify and identify performance crises by constructing a new representation of data center state, called a fingerprint, which is constructed by statistical selection and summarization of hundreds of performance metrics typically collected on such systems. It can be used to detect specific performance crises that have been seen before, but has limited effects on new unseen performance crises.

Chen et al. [17] proposed a decision tree learning method to diagnose failures in large Internet sites, which is the first application of decision trees to anomaly detection. The method records the runtime attributes of each request and applies automated machine learning and data mining techniques to determine the cause of failure. The algorithm was able to successfully identify 13 of the 14 true causes of failure, achieving a 93% identification rate.

Although effective, traditional machine learning methods often require manual extraction of features from the raw logs, and the results of the model output depend heavily on the extraction of features. In addition, traditional machine learning methods cannot effectively address the heterogeneity and evolution of logs, making the accuracy of anomaly detection based on traditional machine learning methods not very high. With the rapid development of deep learning and natural language processing, research has focused on the application of sequence-based [9–12, 18–21] models. Du et al. [9] designed the DeepLog framework using LSTM neural networks to realize online anomaly detection on system logs. DeepLog uses not only log keys, but also metric values in log entries to detect anomalies, and it relies only on a small training dataset consisting of "normal log entries." The LogMerge anomaly detection method proposed by Zhang et al. [13] combines LSTM and CNN methods to effectively extract the backward and forward dependencies of log sequences, yet significantly reduces the impact brought by noise in log sequences. LogMerge learns the semantic similarity of multisyntax logs, which enables the migration of log anomaly patterns across log types and greatly reduces the anomaly annotation overhead. LSTM with attention mechanism has also been used to improve the performance of complex sequence modeling tasks, such as those for which Zhang et al. [14] proposed the anomaly detection method LogRobust. LogRobust extracts semantic information of log events and represents them as semantic vectors. Then, it detects anomaly using an attention-based bi-LSTM model that captures contextual information in log sequences and automatically learns the importance of different log events. In this way, LogRobust can identify and handle unstable log events and sequences, is robust to unstable log data, and solves the problems of unstable log data in anomaly detection, but when the log sequences span is large and the network is deep, it can greatly increase the calculation. These are some explorations of log sequence anomaly detection with LSTM, but further improvements are needed in detecting accuracy and reducing computational overhead.

Transformer [22] is a state-of-the-art NLP architecture based on self-attention, it breaks the limitation that LSTM models cannot be computed in parallel, and the self-attention mechanism is a more interpretable model that has achieved many impressive results on natural language processing tasks, and in recent years, gradually more and more researchers have been applying this model to the field of log anomaly detection. For example, Nedelkoski et al. [18] proposed Logsy, a classification-based method to learn log representations that allow to distinguish between normal system log data and anomaly samples from auxiliary log datasets, easily accessible via the Internet. The idea behind Logsy is that the auxiliary dataset is sufficiently informative to enhance the representation of the normal data, yet diverse enough to regularize against overfitting and improve generalization. Steverson et al. [19] detect attacks on an enterprise network by applying mining NLP techniques to Windows Event Logs (WELs), using transformer models and self-supervised training methods. A self-supervised anomaly detection model was constructed by combining deep learning methods, traditional machine learning, and natural language processing. The model filters log into a series of words with a few simple steps. The model does not perceive template for input and has poor generalization

ability to logs of the same template that have not appeared, in addition to the simple filtering of logs makes it difficult to eliminate the effect of log noise and may even make log data noisier. Le and Zhang [23] proposed NeuralLog, a novel log-based anomaly detection approach that does not require log parsing. NeuralLog extracts the semantics from raw log sequences and represents them as semantic vectors. These representation vectors are then used to detect anomalies using a transformer-based classification model.

There are other deep learning methods for log anomaly detection. Qi et al. [24] proposed a novel log-based anomaly detection method called Adanomaly, which uses the BiGAN model for feature extraction and an ensemble approach for anomaly detection. Han et al. [25] proposed a data augmentation strategy that generates a set of anomalous sequences by negative sampling so that practitioners can use the observed normal sequences and the generated anomalous sequences to train a binary classification model.

3. Classification-Based Log Anomaly Detection

3.1. Framework. To address the challenges brought by the heterogeneity, evolution, and data noise of logs, we propose LogPal for generic anomaly detection for heterogeneous logs under massive noise. LogPal can automatically parse heterogeneous logs and improve the accuracy of syslog anomaly detection by combining the raw logs to obtain the final log pattern events, and LogPal can sense the log patterns through an improved transformer model to achieve anomaly detection. This section describes the overall framework of LogPal and the details of each part.

Figure 1 shows the overall framework of LogPal, which is divided into two modules: the offline training module and the online detection module. In the offline training module, LogPal first uses the FT-tree method to extract templates from the raw logs, and the templates are combined with the raw logs to parse them into new log pattern events, and construct pattern vectors based on the log pattern events. LogPal inputs the pattern vectors into the transformer deep neural network model of synthetic attention and trains a general anomaly detection model for heterogeneous logs. In the online detection module, LogPal maps online log sequences to pattern vectors based on the above method, judges whether an online log sequence is anomalous according to the trained anomaly detection model, and generates an alarm if it is an anomalous log sequence.

3.2. Pattern Vector Construction. Syslog is usually an unstructured natural language text written by different developers and often needs to be parsed by log parsers before it can be effectively applied for anomaly detection based on machine learning, deep learning, and other methods. Currently, it is a common practice to parse syslog by extracting templates from the syslog. A template is usually an invariant part of the syslog that represents the general type and meaning of the event expressed by the log sequence, and similar log sequences can be represented by the same templates, e.g., "** startup succeeded" is "syslog: klogd startup succeeded" which is a template for "syslog: klogd startup succeeded." Compared with the raw log, the template removes the variable part "syslog: klogd" and keeps the main part of the event, i.e., "A process or port started successfully." This template can represent not only the log sequence "syslog: klogd startup succeeded," but also other log sequences that describe the same event as this log sequence, such as "syslog: syslogd startup succeeded."

We use the FT-tree template parser [26] for template extraction. FT-tree is an extended prefix tree structure with the basic idea that a fixed part of a log sequence is usually the longest combination of frequently occurring words. Therefore, extracting templates is equivalent to identifying the longest combination of frequently occurring words from the logs. Numerous experiments based on production environment logs show that FT-tree supports incremental learning with high accuracy and high template matching efficiency. However, simply taking log template sequences as training data and constructing template vectors based on them, although effective, ignores key textual information peculiar to the raw logs, which results in two or more normal and exception log sequences, removing the critical variable parts, and generating the same template. This makes the model "think" of log sequences with different labels as the same input, which is fatal for the log anomaly detection model.

In the end, we adopt the frequently used textual preprocessing library torchtext, which filters abundant numbers and special character noise in the raw log sequences and applies character case conversion, then uses FT-tree for template extraction. The extracted log template sequences are encoded as natural number sequences from 1 to n, and each number represents the type of each template. So far, the raw log sequences have been transformed into template tag sequence, and finally new textual token sequences are generated and combined with the raw syslog. A combined pattern event will be composed of two parts (template number and filtered syslog). The new textual tokens sequences not only abstract the main part of each log sequence but also fully retains all the key information of the variable part. In addition, to preserve the semantics of the two parts of log pattern events and reduce or even eliminate the impact of heterogeneous log anomaly detection, LogPal uses all log pattern event tokens (template numbers arranged before syslog sequences) as training data to obtain word vectors of template words and raw syslog sequences and constructs pattern vectors based on them. GloVe [27] integrates latent semantic analysis based on singular value decomposition and the word2vec algorithm by introducing co-occurrence probabilities matrix, which uses both global statistical features of the corpus and local context features. GloVe uses the lexical co-occurrence statistics to change their weights in the objective function J, which is specified as follows:

$$J = \sum_{i,j}^{N} f\left(X_{i,j}\right)\left(v_i^T v_j + b_i + b_j - log\left(X_{i,j}\right)\right)^2, \quad (1)$$

where v_i and v_j are the word vectors of words i and j, b_i and b_j are two deviation terms, f is the weight function, and N is

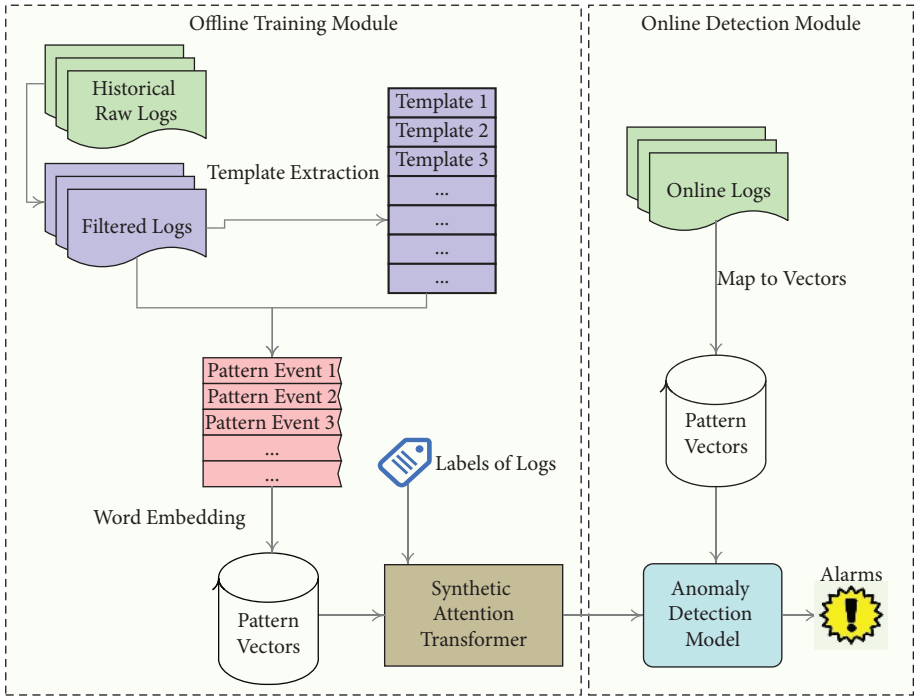

Figure 1: The framework of LogPal.

the size of the vocabulary table (co-occurrence matrix dimension is $N \times N$). The pattern vectors of log pattern events can be obtained by using GloVe. To facilitate the reader's understanding, Figure 2 shows the process of transforming the raw logs into new pattern vectors.

3.3. Synthetic Attention Transformer. LogPal is modeled by an encoder with a multihead attention transformer and takes the constructed log pattern vectors as input, which differs from the input of a traditional transformer in that each pattern vector contains two parts of tokens (the template number and the filtered real log). Therefore, an improved transformer for synthesizing attention is designed to learn the constructed log pattern vectors more efficiently.

Synthetic attention is represented by a synthetic attention matrix, which is divided into global attention and sparse attention. Global attention is applied to the log template, and sparse attention is applied to the log sequence. The log template pays attention to every token of log pattern, including the log template itself, because it can even directly determine the anomaly itself.

However, not every token needs to deal with contextual representation. In the typical self-attention mechanism, every token needs to attend all other tokens; however, for a trained transformer, the learned attention matrix K is usually very sparse at most data points. Therefore, the computational complexity can be reduced by combining structural biases to limit the number of keyword key pairs per query. For a given input token, we can group its contexts into nonoverlapping spans of different sizes, and the size of the spans increases with their relative distance. That is, the input token attends each token, processing the different spans away from it in a fine-to-coarse fashion. To obtain the synthetic attention keyword matrix, the template token attention and the sparse log token attention are constructed successively.

3.3.1. Template Global Attention. Global attention is used for the template token of the constructed log vector, "global" means that the template token can both attend all other tokens and let all other tokens pay attention to it. The attention formula is as

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{2}$$

where $\text{Attention}(Q, K, V)$ is the value of attention and Q, K, and V are the query vector matrix, key vector matrix, and value vector matrix, respectively. Every row of these three matrices represents a vector corresponding to a token, and we need to calculate a Score Matrix for the template vector before calculates the template attention:

$$\text{Score} = \text{QK}^{\text{T}}. \tag{3}$$

Templates are very important for anomaly detection, so global attention is applied to templates. That is, only the attention between the template token and other tokens, and the attention of other tokens with the template token are calculated. Figure 3 illustrates this process.

3.3.2. Log Sparse Attention. Unlike templates, each token of log sequences is processed from center to both ends. Every token pays more attention to the log sequence token that is closer to itself, and the further distant token is not as

Raw Log Sequences:
Log1: syslog: klogd startup succeeded.
Log2: syslog: syslogd startup succeeded.
Log3: MapTask metrics system stopped.
Log4: MapTask metrics system shutdown complete.

Templates → S/N:
Template1 →1 : * * startup succeeded
Template2 →2 : maptask metrics system * *

Pattern Tokens:
Pattern1: 1 syslog klogd startup succeeded
[35, 1140, 805, 832, 3577, 1]
Pattern2: 1 syslog: syslogd startup succeeded
[35, 1140, 706, 832, 3577, 1]
Pattern3: 2 maptask metrics system stopped
[543, 1856, 653, 4551, 56, 1]
Pattern4: 2 maptask metrics system shutdown complete
[543, 1856, 653, 4551, 56, 1860]

Pattern Vectors:
Pattern Vector1: [[-0.0411, -0.0023,…], [-0.0310, 0.0423,…], […]…[…]]
Pattern Vector2: [[-0.0411, -0.0023,…], [-0.0310, 0.0423,…], […]…[…]]
Pattern Vector3: [[0.1334, -0.6031,…], [-0.0654, -0.0630,…], […]…[…]]
Pattern Vector4: [[0.1334, -0.6031,…], [-0.0654, -0.0630,…], […]…[…]]

Figure 2: Examples of mapping raw log to pattern vector.

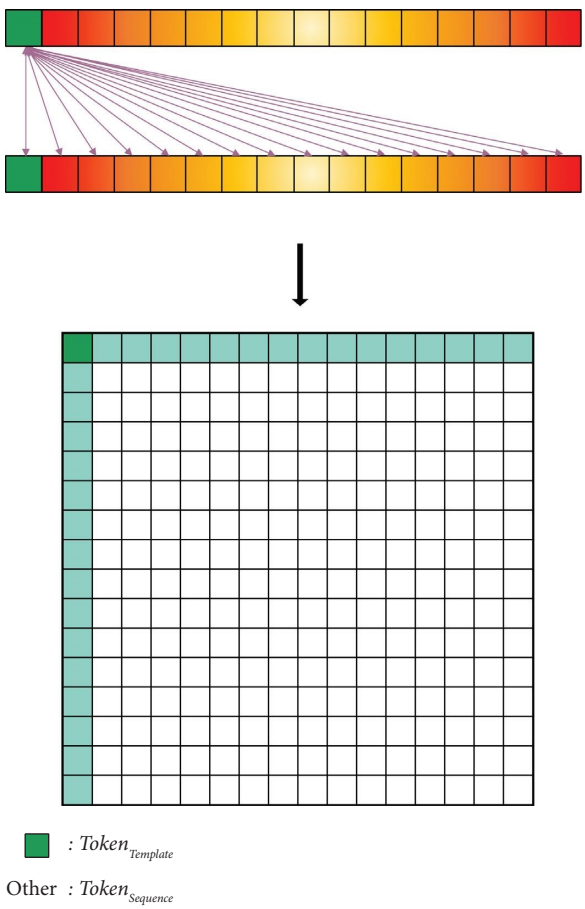

Figure 3: Global attention process of template token vector.

concerned, which can significantly decrease the subsequent parameters in quantity by sparsity. The following Q matrix, K matrix, and Score matrix all only describe the raw log sequence tokens, without the template tokens.

In the K matrix, we take a heterogeneous log sequence with m tokens as an example. For a certain token_i, the distance deviations of every raw log token_j (without considering the template token) from token_i is calculated as follows:

$$\text{Deviation}_j = |i - j|, \tag{4}$$

and then, we input m deviations to the minimum heap

$$\text{MinHeap}_{\text{Deviation}} = \{\text{Deviation}_0, \text{Deviation}_1, \ldots, \text{Deviation}_{m-1}\}. \tag{5}$$

By inputting the deviation into the minimum heap, we can ensure that the next selected token has the minimum deviation from token_i. And then, LogPal selects several groups of tokens from the minimum heap, and the number of tokens in each group is 2^0, 2^1, 2^2, ..., total N tokens. The vector in each cluster takes the maximum value. Next, the maximum vector is used to calculate the Score vector. Then, process each token_i in sequence to get the sparse matrix W^{Score}. The ith row and jth column of the raw log vectors (excluding the template vector) are the sparse attention values of token_i with its own and other token, and finally further calculated by equation (2) to obtain the sparse attention matrix from fine to coarse. This is based on the assumption that for any token, the nearest token requires more attention, while the distant token has little impact on it. This can reduce the effect of noise away from the distant token. At the same time, it also decreases the parameter quantity of subsequent calculations. Finally, the sparse attention matrix obtained in part 2 is spliced to the lower right of the template global attention matrix.

Figure 4 illustrates the mapping process from the Q matrix and K matrix to the *Score* matrix with the raw log sequence whose length of token is 15 as an instance. token_7 and token_8 are shown in Figure 4.

The pseudocode of the sparsity algorithm is shown in Algorithm 1.

3.4. Parameter Setting. In the online detection module, every pattern tokens in the log pattern events is mapped to a 300-dimensional vector in the same way, 4 heads are used for multihead attention, a cross-entropy function is used as the loss function to train the LogPal neural network, and a Dropout layer is used to prevent overfitting, a sigmoid layer is employed to output the classification, and we use a weight decay factor 0.001, the initial learning rate is set to 0.001 for the Adam optimizer, and the final training epoch is set to 10. In addition, the random seed can be initialized to a fixed value to ensure that the experimental results can be reproduced. Our model is implemented using PyTorch and trained on an NVIDIA GeForce RTX 3090 GPU.

4. Experiments

To quantify the performance of LogPal, we conducted various experiments. We compare this method with four exposed baselines on two real-world syslog datasets. We

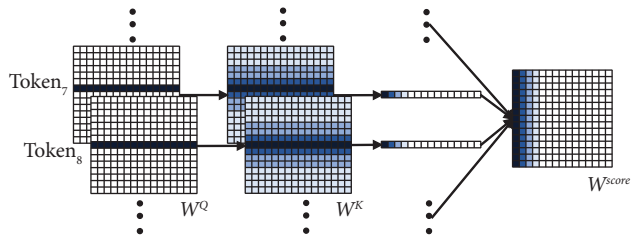

Figure 4: Mapping process from Q matrix and K matrix to sparse score matrix.

Input: W^Q, W^K
Output: W^{score}
(1) Array = $N_1, N_2, N_3, \ldots$ and sum equals Token's numbers;
(2) Min − Heap for offering and polling Deviation;
(3) **for** row_i in W^Q **do**
(4) **for** row_j in W^K **do**
(5) Min − Heap add $|i - j|$;
(6) **end for**;
(7) **for** N_k in Array **do**
(8) **while** not end of N_k **do**
(9) ListKey_k add (Min − Heap poll);
(10) $\text{Key}_k \longleftarrow \text{Max}(\text{ListKey}_k)$;
(11) $W^{\text{score}}_{i,k} \longleftarrow Q_k \cdot K_k$;
(12) **end for**;
(13) **end for**;
(14) returnW^{score};

Algorithm 1: Sparsity of score matrix.

describe the main information in the datasets, discuss the experimental settings and evaluation indicators, and give the results.

4.1. Datasets. We evaluate the proposed method on the following three open log datasets: BGL dataset [28], HDFS dataset [7], and Thunderbird [28]. A brief summary is shown in Table 2, and the details are as follows:

The BGL dataset is an open dataset of logs collected from a BlueGene/L supercomputer system at Lawrence Livermore National Labs (LLNL) in Livermore, California, with 131,072 processors and 32,768 GB memory. The log contains alert and nonalert sequences identified by alert category tags. In the first column of the log, "–" indicates nonalert sequences while others are alert sequences. The label information is amenable to alert detection and prediction research. It has been used in several studies on log parsing, anomaly detection, and failure prediction.

The HDFS dataset is generated in a private cloud environment using benchmark workloads and manually labeled through handcrafted rules to identify the anomaly. The logs are sliced into traces according to block IDs. The HDFS dataset marks each block sequence as normal or anomalous. The HDFS dataset consists of 11,175,629 logs collected in 38.7 hours on more than 200 Amazon EC2 nodes. There are 575,061 log blocks in the dataset, of which 16,838 are marked as "exception" by Hadoop experts.

Table 2: A brief summary of datasets.

Dataset	Time span	Size	No. of messages	Anomalies
BGL	214.7 days	708 MB	4,747,963	348,460
HDFS	38.7 hours	1.47 GB	11,175,629	288,250
Thunderbird	244 days	29.60 GB	211,212,192	43,087,287

Thunderbird dataset is an open dataset of logs collected from a Thunderbird supercomputer system at Sandia National Labs (SNL) in Albuquerque, with 9,024 processors and 27,072 GB memory. The log contains alert and nonalert sequences identified by alert category tags. In the first column of the log, "–" indicates nonalert sequences, while others are alert sequences.

4.2. Experimental Setup. The experimental setup for this study is explained as follows.

4.2.1. Comparisons

We compared LogPal with five published baseline methods, namely, PCA [7], DeepLog [9], Swisslog [29], HitAnomaly [30], and InterpretableSAD [26]. The parameters of these methods have been optimized to

produce their best evaluation scores. A brief description of these methods is as follows:

PCA: in this model, Logs are converted to count vectors and divided into normal and anomaly spaces using a principal component analysis (PCA) algorithm.

DeepLog: a deep neural network using LSTM models the system logs as natural language sequences.

SwissLog: SwissLog combines semantic embedding and time embedding methods to train a bi-LSTM model based on unified attention for anomaly detection

HitAnomaly: a log sequence encoder and a parameter value encoder are designed to obtain their corresponding representations. The hierarchical transformer structure is used to model the log template sequence and parameter values.

Interpretable SAD: the authors propose a data augmentation strategy that generates a set of anomalous sequences with negative sampling so that a binary classification model can be trained based on the observed normal sequences and the generated anomalous sequences.

4.2.2. Evaluation Indicators. To measure the effectiveness of LogPal in anomaly detection, we use precision, recall, and $F1$ score as indicators.

Precision: it is the percentage of true anomalies among all anomalies detected by the approach:

$$PR = \frac{TP}{TP + FP}. \tag{6}$$

Recall: it is the percentage of anomalies among the dataset being detected:

$$RC = \frac{TP}{TP + FN}. \tag{7}$$

$F1$ score: it is the harmonic mean of precision and recall:

$$F1 = \frac{2 \times PR \times RC}{PR + RC}. \tag{8}$$

TP is the number of anomalous log sequences correctly detected by the model, FP is the number of normal log sequences incorrectly identified as anomalies by the approach, FN is the number of anomalous log sequences that are not detected by the approach, and F1 score is used as a metric that considers both precision and recall, which does not favor one metric over another and does not lose scientific validity due to the imbalance problem of the dataset.

4.3. Experimental Results. Firstly, we compare LogPal with transformer on BGL dataset and HDFS dataset. We convert the log sequences of the datasets into log pattern vectors in the same way, and then, these pattern vectors are input into the transformer model, and relevant parameter settings are consistent with LogPal. We conducted the comparison experiment by controlling the ratios of the training set and test set. Figures 5 and 6 show the comparative results of the experiment.

The horizontal axis of Figures 5 and 6 represents the ratios of the training set and test set, and the vertical axis represents the $F1$ score of different anomaly detection models. It can be seen that when the training set ratio of LogPal is large, the optimal $F1$ score of the LogPal method for anomaly detection is 99% and that of transformer model is 98%, which has a weak advantage over transformer; when the ratio of training sets is small, it can better reflect the performance advantages of LogPal. It shows that even if a small amount of training data is obtained from the target syslog, LogPal can extract the key information leading to the normal or anomalous log sequences and can produce accurate prediction even in invisible samples. When the ratio of training sets is 2 : 1, LogPal's anomaly detection rate is 90%, while transformer is 86%, LogPal increased $F1$ score by 4.7%. Even with a large training set, the $F1$ score improves by more than 1%. LogPal uses synthetic attention to perceive the relationship between the template vector and the raw vector differently and obtains a better $F1$ score than the transformer model. The transformer model does not consider this special feature of the raw log template but only considers the self-attention relationship matrix of the raw log sequence itself making a lot of important information ignored, which may lead to false alarm. LogPal can quickly perceive and learn the semantic information of the pattern vectors to improve the accuracy of anomaly detection. In addition, LogPal adopts a sparse attention method for the raw log sequence token, which can reduce the noise impact even in the long text log sequence and adopting a general and unified pattern event extraction method to embed the pattern vectors, as we expect, enable robust representation and accurate anomaly detection even for heterogeneous logs or invisible new logs.

To further evaluate the performance of LogPal, we also evaluated LogPal and baselines on BGL dataset, HDFS dataset, and Thunderbird dataset. We show the overall performance of LogPal compared with baselines in Table 3 and Figures 7–9. Based on the three datasets, generally speaking, LogPal has the best performance, and all evaluation indicators are close to 99%. LogPal can generally filter massive heterogeneous logs to generate pattern vectors and perceive log templates and log sequences, respectively, by synthetic attention. PCA and DeepLog use the index of log template to learn anomalous and normal patterns. It ignores the meaning of log sequences and words, and the actual performance is not high. Although InterpretableSAD performs well on the Thunderbird dataset, where all indicators are balanced and the indicator value is not low, the method does not perform so well on the BGL dataset and the HDFS dataset, which may be related to the fact that the method is not universal and may be more suitable for a certain dataset.

It is worth noting that comparing SwissLog and HitAnomaly, SwissLog has a precision of 97% and a recall of 100% in the experiments on the BGL dataset and the HDFS dataset, while the two indicator values of HitAnomaly are exactly the opposite, it has a precision of 100% and a recall of

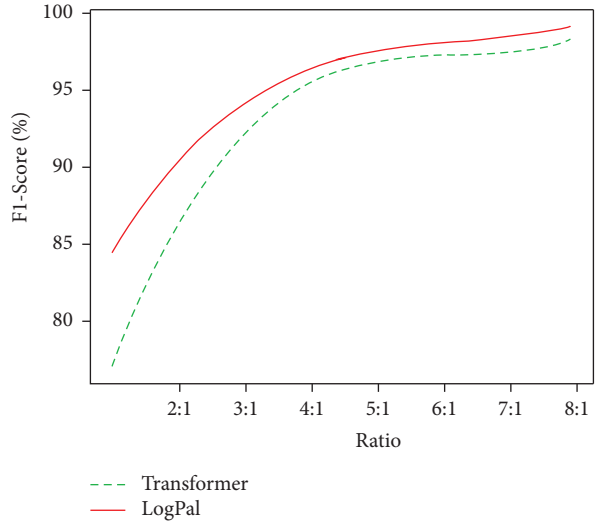

Figure 5: Comparisons of different ratios on the BGL dataset.

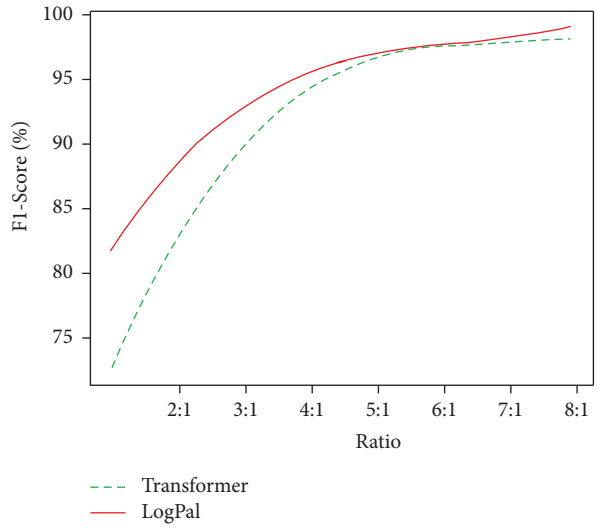

Figure 6: Comparisons of different ratios on the HDFS dataset.

Table 3: Comparison of methods on three datasets.

Methods	Datasets								
	BGL			HDFS			Thunderbird		
	PR (%)	RC (%)	*F*1 score (%)	PR (%)	RC (%)	*F*1 score (%)	PR (%)	RC (%)	*F*1 score (%)
PCA	98	67	80	50	61	55	87	90	89
DeepLog	95	96	93	92	92	91	95	92	93
HitAnomaly	**100**	97	98	**100**	97	98	97	96	97
SwissLog	97	**100**	**99**	97	**100**	98	95	**97**	96
InterpretableSAD	94	88	91	92	87	89	97	96	96
LogPal	99	99	**99**	98	99	**99**	**98**	**97**	**98**

The bold values show the best performances of method based on the experimental results with the specific indicator and dataset.

97% in the experiments. This means that SwissLog is more inclined to detect log sequences as anomalous, in other words, it is more capable of uncovering anomalous logs in the logs. Although the recall rate is satisfactory, it is clear from the precision rate that SwissLog mistakes some log sequences that are really normal as anomalous. Thus, generating a large number of false positive predictions, and if a log anomaly detection method generates too many false alarms, which will consume energies of O&M staff to verify the system condition and add a lot of unnecessary work; in

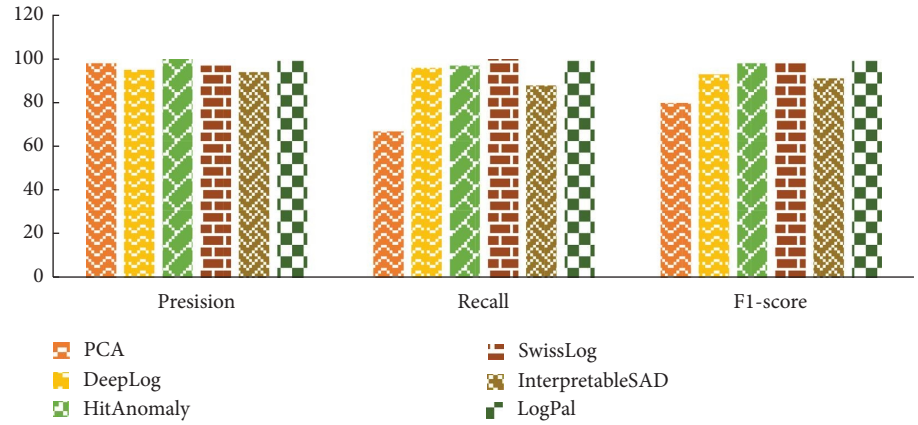

Figure 7: Comparison of different methods on the BGL dataset.

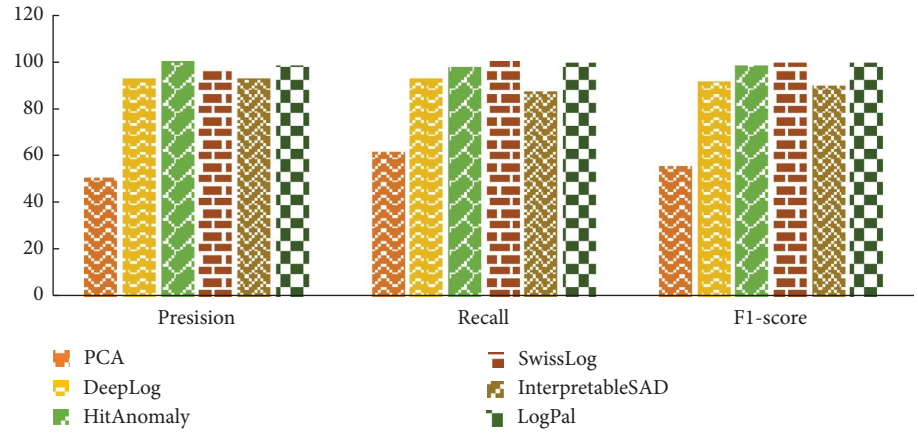

Figure 8: Comparison of different methods on the HDFS dataset.

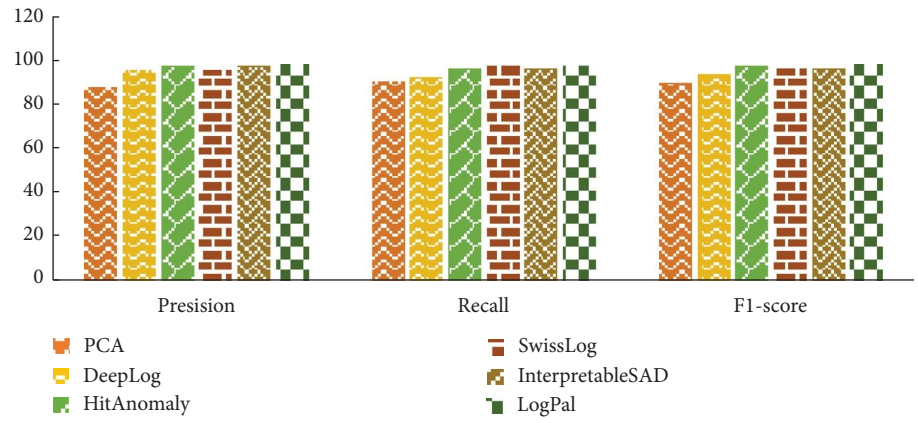

Figure 9: Comparison of different methods on the thunderbird dataset.

contrast, HitAnomaly can be very precise in logs identified as anomalous, without so much false positive predictions. But it misses some anomalous logs, thus generating a large number of false negative predictions, which may be a more serious problem if it fails to detect system anomaly. System failures may not be resolved in a timely manner for a long time, which will cause serious losses. Generally speaking, LogPal is able to balance precision and recall and has an improved overall performance. Overall, compares favorably to baselines, LogPal not only learns the semantic information of log word vectors but also focuses differently on the attention relation between template tokens and log sequence tokens. Finally, LogPal achieves an excellent performance on the BGL dataset, the HDFS dataset, and the Thunderbird

dataset. Compared with existing methods, LogPal improves the $F1$ score by 1% on the HDFS dataset. Besides, LogPal improves the precision and $F1$ score by 1% on the Thunderbird dataset.

5. Conclusion

As a kind of data reflecting system status and events, syslog provides an important support for detecting various software and hardware system anomalies. Many log-based methods have been proposed to detect anomaly in large-scale software and hardware systems. However, the existing methods make it difficult to effectively deal with the labelling problems in heterogeneous logs. To overcome these problems, this study proposes a generic anomaly detection mechanism for heterogeneous logs, called LogPal. The model innovatively utilizes the synthetic attention transformer encoder network, which prospectively thins out the semantics of log sequences and weakens the influence of noise. Compared with other methods, it achieves better generalization ability on multisource and heterogeneous samples. Experiments based on public datasets show that the overall performance of LogPal is better than the current machine learning and deep learning methods. In future work, we will further improve the accuracy of anomaly detection by introducing the weight coefficient to learn the contribution degree of the template and the raw log token. In addition, we will explore the synthesis strategy of synthetic attention to reduce the computational complexity and improve the early warning speed of anomaly detection.

Data Availability

Previously reported log data were used to support this study and are available at https://doi.org/10.48550/arXiv.2008.06448.

Conflicts of Interest

The authors declare that they have no known conflicts of financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China, under Grant 62072255.

References

[1] S. L. He, P. J. He, Z. B. Chen, T Yang, Y. Su, and M. R. Lyu, "A A Survey on Automated Log Analysis for Reliability Engineeringurvey on automated log analysis for reliability engineering," *ACM Computing Surveys*, vol. 54, no. 6, pp. 1–37, 2021.

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

[3] H. Li and Y. Li, "LogSpy: system log anomaly detection for network systems," in *Proceedings of the 2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE)*, pp. 347–352, Hangzhou, Zhejiang, China, October 2020.

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

[5] J. Chen, J. Y. Ouyang, and A. Q. Feng, "DoS anomaly detection based on isolation forest algorithm under edge computing framework," *Computer Science*, vol. 47, no. 2, pp. 293–299, 2020.

[6] Q. W. Lin, H. Y. Zhang, and J. G. Lou, "Log clustering based problem identification for online service systems," in *Proceedings of the 38th International Conference*, pp. 102–111, Austin, Texas, USA, May 2016.

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

[8] S. Ying, B. M. Wang, L. Wang et al., "An An Improved KNN-Based Efficient Log Anomaly Detection Method with Automatically Labeled Samplesmproved KNN-based efficient log anomaly detection method with automatically labeled samples," *ACM Transactions on Knowledge Discovery from Data*, vol. 15, no. 3, pp. 1–22, 2021.

[9] M. Du, F. Li, and G. Zheng, "Deeplog: anomaly detection and diagnosis from system logs through deep learning," in *Proceedings of the 2017 ACM SIGSAC Conference*, pp. 1285–1298, Dallas, Texas, USA, July 2017.

[10] B. Sharma, P. Pokharel, and B. Joshi, "User behavior analytics for anomaly detection using lstm autoencoder - insider threat detection," in *Proceedings of the IOE GC*, T. Bangkok, Ed., pp. 1–9, March 2020.

[11] X. Y. Duan, S. Ying, H. L. Cheng, W Yuan, and X Yin, "OILog: An online incremental log keyword extraction approach based on MDP-LSTM neural network," *Information Systems*, vol. 95, pp. 101618–11, 2021.

[12] S. Garg, K. Kaur, N. Kumar, G Kaddoum, AY Zomaya, and R Ranjan, "A A Hybrid Deep Learning-Based Model for Anomaly Detection in Cloud Datacenter Networksybrid deep learning-based model for anomaly detection in cloud datacenter networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 3, pp. 924–935, 2019.

[13] S. L. Zhang, W. D. Li, and Y. Q. Sun, "Unified anomaly detection for syntactically diverse logs in cloud," *Science CSRD*, vol. 57, no. 4, pp. 778–790, 2020.

[14] X. Zhang, Y. Xu, and Q. Lin, "Robust log-based anomaly detection on unstable log data," in *Proceedings of the 2019 27th ACM Joint Meeting*, pp. 807–817, New York, NY, USA, August 2019.

[15] I. Kornoukhov and H. Soller, *Mitigating Cyber Risks with Smart Log Management,* Mckinsey & Company, New York, NY, USA, 2020, https://www.mckinsey.com/business-functions/mckinsey-digital/our-insights/tech-forward/mitigating-cyberrisks-with-smart-log-management.

[16] P. Bodik, M. Goldszmidt, and A. Fox, "Fingerprinting the datacenter: automated classification of performance crises," in *Proceedings of the European Conference on Computer Systems, Proceedings of the 5th European conference on Computer systems*, pp. 111–124, New York, NY, USA, January 2010.

[17] M. Chen, A. X. Zheng, and J. Lloyd, "Failure diagnosis using decision trees," in *Proceedings of the International Conference*

on Autonomic Computing, 2004. Proceedings, pp. 36–43, New York, NY, USA, May 2004.

[18] S. Nedelkoski, J. Bogatinovski, and A. Acker, "Self-attentive classification-based anomaly detection in unstructured logs," in *Proceedings of the 2020 IEEE International Conference on Data Mining (ICDM)*, pp. 1196–1201, Sorrento, FL, USA, February 2020.

[19] K. Steverson, C. Carlin, and J. Mullin, "Cyber intrusion detection using natural language processing on windows event logs," in *Proceedings of the 2021 International Conference on Military Communication and Information Systems (ICMCIS)*, pp. 1–7, The Hague, Netherlands, May 2021.

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

[21] W. B. Meng, Y. Liu, and Y. C. Zhu, "Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proceedings of the International Joint Conference on Artificial Intelligence*, pp. 4739–4745, Macao, China, August 2020.

[22] A. Vaswani, N. Shazeer, and N. Parmar, "Attention is all you need," in *Proceedings of the Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pp. 5998–6008, Long Beach, CA, USA, July 2017.

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

[24] J. X. Qi, Z. Z. Luan, and S. H. Huang, "Adanomaly: adaptive anomaly detection for system logs with adversarial learning," in *Proceedings of the Network Operations and Management Symposium*, pp. 1–5, Budapest, Hungary, August 2022.

[25] X. Han, H. Cheng, and X. Depeng, "InterpretableSAD: interpretable anomaly detection in sequential log data," in *Proceedings of the 2021 IEEE International Conference on Big Data (Big Data)*, pp. 1183–1192, Orlando, FL, USA, August 2021.

[26] S. L. Zhang, W. B. Meng, and J. H. Bu, "Syslog processing for switch failure diagnosis and prediction in datacenter networks," in *Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–10, Barcelona, Spain, June 2017.

[27] J. Pennington, R. Socher, and C. Manning, "Glove: global vectors for word representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543, Doha, Qatar, January 2014.

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

[29] X. Y. Li, P. F. Chen, and L. X. Jing, "Swisslog: robust and unified deep learning based log anomaly detection for diverse faults," in *Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, pp. 92–103, Portugal, October 2020.

[30] S. Huang, Y. Liu, C. Fung et al., "HitAnomaly: HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Logierarchical transformers for anomaly detection in system log," *IEEE Transactions on Network and Service Management*, vol. 17, no. 4, pp. 2064–2076, 2020.