applied sciences

MDPI

Article

LTAnomaly: A Transformer Variant for Syslog Anomaly Detection Based on Multi-Scale Representation and Long Sequence Capture

Delong Han [1,2,†], Mengjie Sun [1,2,†], Min Li [1,2,*] and Qinghui Chen [1,2]

[1] Key Laboratory of Computing Power Network and Information Security, Ministry of Education, Shandong Computer Science Center (National Supercomputer Center in Jinan), Qilu University of Technology (Shandong Academy of Sciences), Jinan 250014, China; handl@sdas.org (D.H.); 10431210690@stu.qlu.edu.cn (M.S.); 10431210599@stu.qlu.edu.cn (Q.C.)
[2] Shandong Provincial Key Laboratory of Computer Networks, Shandong Fundamental Research Center for Computer Science, Jinan 250353, China
* Correspondence: limin@qlu.edu.cn
† These authors contributed equally to this work.

Abstract: Detailed information on system operation is recorded by system logs, from which fast and accurate detection of anomalies is conducive to service management and system maintenance. Log anomaly detection methods often only handle a single type of anomaly, and the utilization of log messages could be higher, which makes it challenging to improve the performance of log anomaly detection models. This article presents the LTAnomaly model to accomplish log anomaly detection using semantic information, sequence relationships, and component values to make a vector representation of logs, and we add Transformer with long short-term memory (LSTM) as our final classification model. When sequences are processed sequentially, the model is also influenced by the information from the global information, thus increasing the dependence on feature information. This improves the utilization of log messages with a flexible, simple, and robust model. To evaluate the effectiveness of our method, experiments are performed on the HDFS and BGL datasets, with the F1-measures reaching 0.985 and 0.975, respectively, showing that the proposed method enjoys higher accuracy and a more comprehensive application range than existing models.

Keywords: anomaly detection; deep learning; log analysis

Citation: Han, D.; Sun, M.; Li, M.; Chen, Q. LTAnomaly: A Transformer Variant for Syslog Anomaly Detection Based on Multi-Scale Representation and Long Sequence Capture. *Appl. Sci.* **2023**, *13*, 7668. https://doi.org/10.3390/app13137668

Academic Editor: Yajian Zhou

Received: 22 May 2023
Revised: 17 June 2023
Accepted: 27 June 2023
Published: 28 June 2023

1. Introduction

Currently, large-scale systems have been applied everywhere and provide diverse services in many situations. The quality of services greatly affects the user experience, but unstable software and hardware can cause many kinds of errors. The current network environment is complex, and novel attacks on computer systems often emerge, exposing service providers to security threats and economic losses. Therefore, the detection of large-scale system anomalies is necessary.

Large-scale systems generate many log files to record runtime states, including messages from users, applications, or the system itself [1]. Logs can be used for the timely detection of system anomalies, and output logs offer a possible means to detect abnormal system states [2–4]. However, logs are usually recorded in natural textual language, which is typically unstructured and requires manual analysis. Traditional log anomaly detection methods are based on such factors as the developer's domain knowledge, manual checks, and writing rules. The above methods have many drawbacks, such as experts with limited knowledge, a large number of logs, high costs to select features, and poor adaptability. The growth of deep learning has provided novel ideas to solve these problems. Researchers have proposed a series of initially effective schemes for log exception

detection. Log anomaly detection can represent logs by counting log events [5–8] or capturing log event semantic information [2,9] using support vector machine (SVM) [7,10], principal component analysis (PCA) [5], invariant mining (IM) [6], long short-term memory (LSTM) [8,11], and Transformer [12] for anomaly detection. However, these solutions also have the following problems. For example, SVM [7,10] requires log mass tagging. PCA [5] is prone to false positives. Only log sequence exceptions can be detected by IM [6]. DeepLog [8] only takes into account one type of exception. LogRobust [9] and HitAnomaly [12] have high computing costs. DeepSyslog [13] does not consider the potential dangers caused by abnormal component values.

To solve the above problems, we designed a log anomaly detection model, LTAnomaly, which uses LSTM and Transformer structures to model log semantic information, sequence relationships, and component information. In order to analyze logs more accurately, Word2Vec generates word vectors, and term frequency–inverse document frequency (TF-IDF) supplies weighted feature vectors of log sequences and component values. This method fully uses the potential relationships between semantic information, temporal information, and other features and solves the problem of high abnormal false alarm rates caused by potential information omissions. Both features are combined and fed to LTAnomaly for further detection. The Transformer embedded with LSTM has strong generalization and low time consumption. The primary objective of this article is to enhance the efficiency of system anomaly detection by improving the quality of automated services. In order to achieve this, innovative deep learning techniques are employed to improve the ability to detect anomalies in system logs, enabling accurate and swift prediction of abnormalities and defects in advance. This approach satisfies the demands for timely, efficient, and precise security detection in real-world scenarios. The main contributions of this article are as follows:

1. This paper presents a novel log representation considering component values. This approach precisely collects the component value information of the log and the semantic information captured on the log sequence. The potential relationships of multiple features are fully considered, improving the utilization of log information and anomaly detection accuracy.
2. This paper proposes LTAnomaly, a model for log anomaly detection based on LSTM and Transformer. LTAnomaly models the temporal dimension through LSTM and captures global contextual information using the attention mechanism in Transformer. When the model processes the sequence sequentially, it receives the influence from the global information, which improves the feature information dependency. This method is progressive and effective in log anomaly detection.
3. We conduct comparative experiments with LTAnomaly and six baseline models on the public log data sets HDFS and BGL and conduct two groups of ablation experiments. The experimental results show that our model is superior to other log anomaly detection methods in terms of detection accuracy and time performance.

The remainder of this article is structured as follows: Section 2 presents work related to log anomaly detection. The LTAnomaly model is described in Section 3. Section 4 discusses our experiments and their results. Section 5 discusses our conclusions and suggests future work.

2. Related Work

2.1. Log Parsing and Preprocessing

Since log messages are unstructured texts, log parsing aims to extract a set of event templates for logs and perform a structured analysis. Current methods of log parsing can be divided into offline and online methods, where offline methods are further split into rule- or clustering-based. Shiwen Cheng et al. [14] presented a method to identify log templates using regular expressions. However, it becomes more difficult to create and update regular expressions as the log volume increases. The first clustering-based log parsing algorithm, the Simple Logfile Clustering Tool (SLCT) [15], extracts log templates by clustering the

word sets occurring more frequently than a threshold in logs. Makanju et al. [16] proposed the IPLoM algorithm based on hierarchical clustering, which clusters logs by iterating their message length, token location, and mapping relationships layer by layer. Unlike SLCT, no threshold setting is required. A limitation of offline methods is that offline parsing cannot perform real-time anomaly detection and must be retrained for newly emerging log types. Online methods that can be parsed in real-time for timely subsequent anomaly detection have been proposed to meet industrial needs. Du et al. [17,18] proposed Spell, an algorithm based on the longest common sub-sequence method to cluster logs online, dynamically extracting log templates from incoming logs. The Drain algorithm [19] uses a deep fixed tree-based idea for clustering logs and can parse logs with stream processing and a just-in-time approach, which greatly reduces running time. Zhang et al. [20] established a frequent template tree (FT-Tree) to obtain log templates by extracting the longest combination of words with high frequency. Studiawan et al. [21,22] modeled event logs as graphs and utilized a graph clustering method to group log entries. Zhu et al. [23] evaluated multiple log parsing algorithms and found that the Drain method performed best. We choose Drain for log parsing due to its characteristics of high resolution and efficiency.

Feature extraction [24] is a crucial step in anomaly detection. Identifying and extracting the most useful features from data enables models to develop a better understanding of the information and generate more effective input for subsequent tasks such as classification and clustering. Extracting these key features allows models to process more accurate data, leading to improved overall performance. Feature extraction methods for detecting log anomalies can be based either on log event counts [5–8] or log event semantic information [2,9]. The first type extracts log events from log messages and converts them to a message count feature space, leaving out the order of log events and the semantic information in the log messages, and therefore it cannot accurately perform log parsing. The latter type models log streams as natural language sequences, using word embedding to convert log events to vectors, based on which the authors train the model. However, existing methods of word-vector-weighted aggregation do not focus on word order relationships, and there is no guarantee that logs will have a unique representation. Moreover, the method is confined to the semantics of log events and can easily overlook other key values in the log for exception detection. Some exceptions do not necessarily manifest as deviations from the normal sequence of log events but rather as potential threats, such as component values that behave as exceptions. Therefore, the values of some specific parameters can be considered important factors in the log-based exception detection mode.

2.2. Anomaly Detection

Anomaly detection [25] is widely employed in scenarios such as financial fraud [26], post-disaster situation analysis [27], social media event monitoring [28], and network traffic [29]. It can detect system anomalies in a timely and effective manner, thereby ensuring network security. Various machine learning [30] and deep learning methods are currently used in anomaly detection research.

Log anomaly detection was first proposed using machine learning, using either supervised or unsupervised learning. Many supervised learning methods have been applied to log anomaly detection. Liang et al. [10] used an SVM classifier to detect log event anomalies. Chen et al. [31] presented a detection method based on a decision tree. Farshchi et al. [32] introduced an approach for regression-based analysis that found a correlation between activity logs of operations and the impact of operational activities on cloud resources, enabling the discovery of anomalies in operational cloud application logs. Zhang et al. [33] proposed the PreFix framework, which uses a random forest algorithm to model template sequences that can intervene and "fix" potential failures. Bertero et al. [34] used Word2Vec for vector representation of logs and applied a classifier to detect whether they were anomalous. However, this approach is limited by its reliance on the quality of the dataset labels.

Subsequently, more unsupervised learning methods have been applied to anomaly detection in logs. In practical studies, the quantity of abnormal data is much less than that of normal data, and anomaly logs are generally unlabeled. Thus, the advantages of unsupervised learning can be better utilized. Lou et al. [6] proposed IM, which can detect more anomalous messages in distributed systems, and then, execution anomalies can be tested effectively. Xu et al. [5] utilized PCA to identify abnormal events and visualize the final results. Yang et al. [35] proposed LogOHC, which has high extraction efficiency in multi-source log datasets. Unsupervised learning with high automation saves labor costs, but most detection uses clustering and correlation analysis, and the degree of automated correlation between these methods is not high. Therefore, deep learning methods have been applied by more studies to jointly analyze events and automate the discovery of potential security threats.

Deep learning methods have ushered in a boom in research and application in recent years, with good results in various fields. Ruff et al. [36] and Pang et al. [37] summarized the use of deep learning models in anomaly detection. Deep learning models provide new methods for log anomaly detection. Zhang et al. [11] applied LSTM for fault prediction in a log system, grouping logs with similar formats and content and processing rare marker data in training to capture long-term dependencies across log sequences. Du et al. [8] proposed DeepLog, which models logs as natural language sequences, enabling the automatic learning of log patterns from normal execution and discovering anomalies when log events deviate from the trained model under normal execution. However, the model can only be trained with large amounts of normal data. Attention-based RNN was proposed by Brown et al. [38] and focuses on log events and overlooks contextual relationships in the log sequence. Unstable log data can be dealt with by LogRobust [9]. An attention-based Bi-LSTM neural network grasps contextual information. This method can automatically analyze the significance of log events. However, large log data volumes result in slow speeds and high calculation costs. Marta Castillo [39] proposed AutoLog, which periodically samples logs, calculates numerical scores, trains a semi-supervised deep autoencoder, and uses the encoder for classification. Huang et al. [12] proposed HitAnomaly, which uses log template sequences and parameter values to represent logs and simultaneously applies a Transformer structure for anomaly detection. Recent studies have shown that LSTM outperforms Transformer in some tasks, but Transformer [40–42] replaces recurrent neural network models in some cases, such as LSTM. The performance of both LSTM and Transformer can be further improved by the semantic information of logs, so they can be integrated into a whole to deal with the instability of log anomaly detection with lower computational expenses, enhanced accuracy, and higher efficiency of anomaly detection.

3. Proposed Models

3.1. Overview

Figure 1 shows an overview of LTAnomaly. Log parsing is performed to convert semi-structured logs into log templates and log component parameters. Then, the logging template and logging component parameters are input to the LTAnomaly model. Log template content is put into Word2Vec to obtain the word vector. Combined with TF-IDF, the feature vector of the log event sequence can be obtained. Then, according to the TF-IDF of the component parameter sequence, the feature vector of the log component sequence is obtained. The feature vector matrices of the log event and log component sequences are spliced to realize the final feature matrix. With the final eigenvector matrix, the log anomaly model based on LSTM-Transformer can detect the logs.

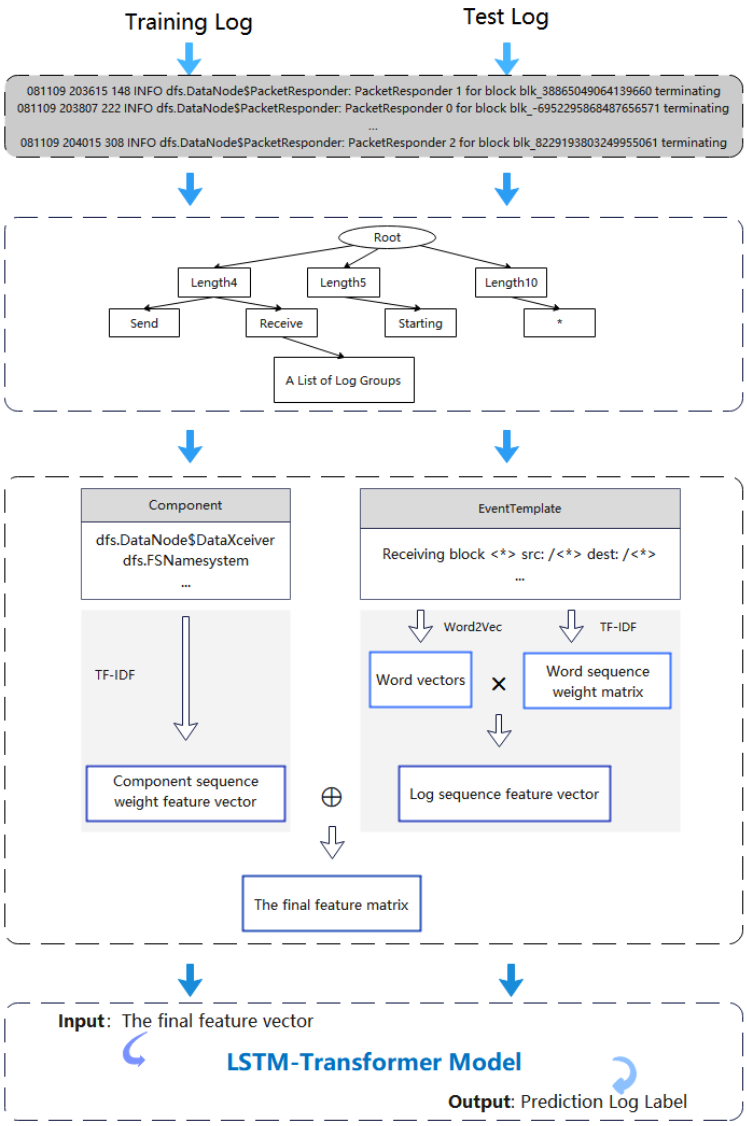

Figure 1. LTAnomaly framework. Drain is used for log parsing, and semantic relationships and log component values are captured from the log sequence to generate a log feature matrix. Transformer is combined with LSTM for anomaly detection. Note: ⊕ means a connection. * denotes a variable.

3.2. Log Parsing and Preprocessing

3.2.1. Log Parsing

Log parsing aims to fetch variables from unstructured logs and save invariant variables as log templates. Before log anomaly detection, it is necessary to parse the unstructured and free-text log so that the obtained log can be passed to the anomaly detection model. Drain is characterized by high parsing accuracy and efficiency. Therefore, logs parsed by Drain meet the requirements of the proposed model. This study uses the Drain method to parse all log data. As shown in Figure 2, Drain builds a fixed-depth parsing tree. The log is preprocessed by a simple regular expression according to domain knowledge. The log group is first distinguished by the length of the log message (the number of tokens), after which the tokens in front of the log are identified. Then, token similarity is calculated to find the best match, and finally, the parsing tree is updated.

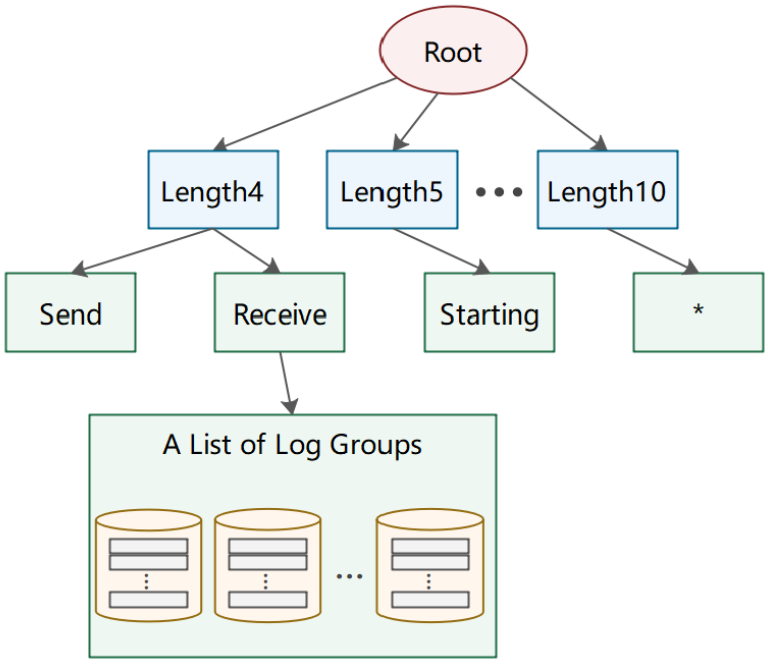

Figure 2. The log parsing process for Drain. Note: * denotes a variable.

Figure 3 shows that the log message is converted to a log template. For example, the log "081109 203615 148 INFO dfs. DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating" is parsed by Drain into the log template "PacketResponder <*> for block <*> terminating", where "<*>" is the wildcard character. The log template is composed of fixed text strings.

081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating

Log Messge

Drain

Structured Log

Date	81109
Time	203615
Level	INFO
Component	dfs.DataNode$PacketResponder
Content	PacketResponder 1 for block blk_38865049064139660 terminating
EventTemplate	PacketResponder <*> for block <*> terminating
ParameterList	['1', 'blk_38865049064139660']

Figure 3. Example of log parsing. Note: * denotes a variable.

3.2.2. Log Sequence Feature Matrix Generation

- **Log sequence vector:** Each dataset has a unique identifier. In the HDFS dataset, the block_id is set as the identifier for a specific sequence of operations. Log events are grouped on the strength of these identifiers, or log items generated by concurrent processes are broken into separate and single-threaded sequential sequences. Each row in Figure 4 refers to a log execution sequence, and each number (ID) in the execution sequence indicates a log. All logs in each log execution sequence have the same block_id. Logs with identical block_ids are arranged together in order to form a log sequence vector. All of these efforts help detect exceptions when the log execution sequence is incorrect.

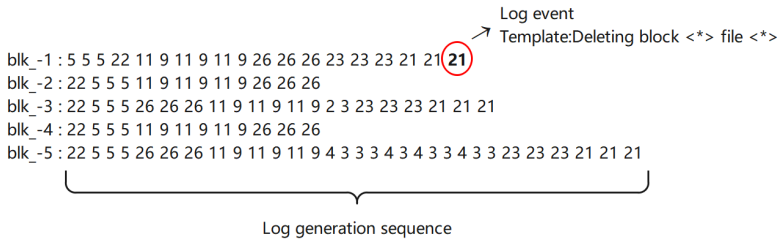

Figure 4. Log sequence vector example.

- **Word vector:** Word2Vec is an effective method of embedding words that takes the context of the words into account during training to establish a language model f(x) = y between the goal word (x) and its context (y). In this way, the targeted word is obtained according to the context word or vice versa. Word2Vec is performed in LTAnomaly to effectively process semantic information in log messages.
  Word2Vec requires words as input, so we need to first split the log template content. The log can be separated by spaces or other methods. For instance, "PacketResponder <*> for block <*> terminating" is divided into four words: "PacketResponder", "for", "block", and "terminating." Then, we use Word2Vec to predict targeted words from context words. As shown in Figure 5, one-hot encoding was performed on the top n words and the bottom n words of the target word. Next, the hidden and softmax layers are used to predict the targeted word, obtaining a word vector for each word. The word vectors of all words form a word vector matrix $V \in \mathbb{R}^{a \times b}$ according to their order of appearance in the log sequence vector.
- **Word sequence weight matrix**: TF-IDF is a popular weighting technique implemented in both information search and data mining. The importance of words in a sentence can be measurable by TF-IDF, which meets the demand for high resolution. A word's weight tends to grow according to its quantity in the log sequence. However, if a word occurs frequently, the weight of the word decreases. For instance, if the word "Block" occurs regularly in a log event, then the word may better describe the log event. In contrast, if "Block" occurs in whole log events, then it is not possible to distinguish log events based on this word, so its weight should be reduced. Therefore, we denote significance with the term frequency (TF) and measure the occurrence of log events by the inverse document frequency (IDF). These are calculated as

$$TF(word_i) = \frac{n_i}{\sum_k n_k}, \tag{1}$$

and

$$IDF(word_i) = log\left(\frac{s_i}{s+1}\right), \tag{2}$$

where n_i is the number of occurrences of $word_i$, $\sum_k n_k$ is summed over words in whole log sequences, and s and s_i, respectively, denote the sum of log sequences and the sum of log sequences with $word_i$. The TF-IDF weight W of each word is computed as

TF×IDF. The TF-IDF weights of the log sequences constitute a word sequence vector weight matrix $W \in \mathbb{R}^{c \times a}$.

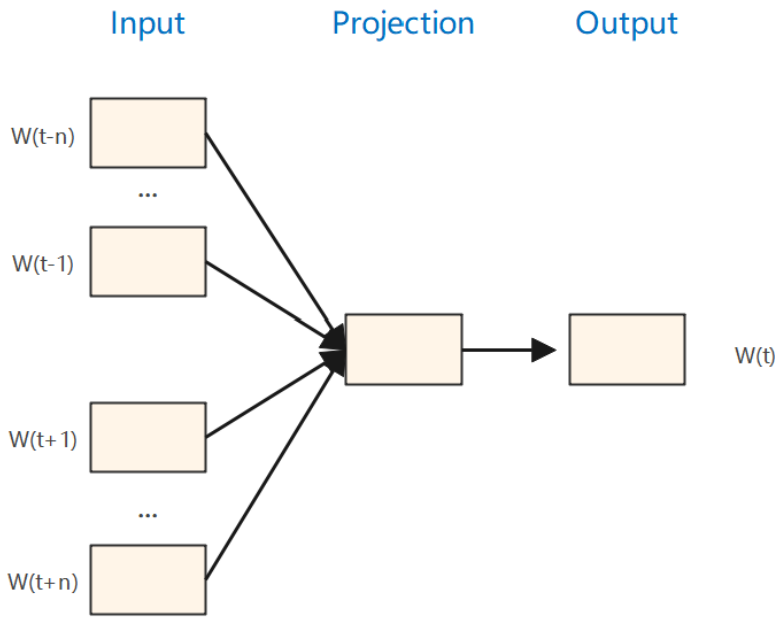

Figure 5. Word2Vec model structure.

- **Log sequence feature matrix**: To produce the log sequence feature matrix, we multiply the word vector matrix V by the word sequence weight matrix W to obtain the log sequence feature vector $Ew \in \mathbb{R}^{c \times b}$, i.e.,

$$Ew = W \times V. \tag{3}$$

Therefore, the log sequence characteristic matrix can pay attention not only to exceptions that occur in the log execution order but also to the semantic information of the log. This method represents log messages more accurately than existing log representations.

3.2.3. Component Sequence Matrix Generation

We discuss how to obtain the representation of component values in LTAnomaly. Log component values with the same block_id form a vector of component value sequences in order of execution. Then, TF-IDF is used for weight analysis. The TF-IDF algorithm here differs slightly from that of word sequence weight calculation. TF is still employed to describe its importance, and IDF measures the occurrence of log events. These are calculated as

$$TF(component_i) = \frac{m_i}{\sum_k m_k}, \tag{4}$$

and

$$IDF(component_i) = log\left(\frac{c_i}{c+1}\right), \tag{5}$$

where m_i is the number of appearances of $component_i$ in whole log sequences, $\sum_k m_k$ is the sum of the component values in whole log sequences, and c and c_i are, respectively, the sum of log sequences and of those with $component_i$. The TF-IDF weight W for each component value is computed as TF×IDF. The TF-IDF weights of all component sequences constitute a component sequence weight matrix $Wc \in \mathbb{R}^{c \times d}$. Finally, the log component value sequence matrix is equal to the component sequence weight matrix, $Ec \in \mathbb{R}^{c \times d}$, i.e.,

$$Ec = Wc. \tag{6}$$

3.2.4. Final Feature Matrix Generation

The final feature matrix is the pivotal point in log anomaly detection. The log exception detection model takes the final feature matrix as its input. The $concatenate(\cdot)$ means to stitch two matrices together. The final feature matrix, $E \in \mathbb{R}^{c\times(b+d)}$, is composed of a log sequence feature matrix and component value sequence feature matrix, which are connected as

$$E = concatenate(Ew, Ec). \tag{7}$$

3.3. Anomaly Detection

LTAnomaly uses LSTM and Transformer to model the feature matrix of the input and detect anomaly logs, taking advantage of LSTM modeling of the time dimension and the self-attention mechanism in Transformer to capture global contextual information. In doing so, we add LSTM to the Transformer subset (Figure 6). For Transformer with LSTM, the input eigenmatrix is processed by LSTM and transmitted to the Transformer block.

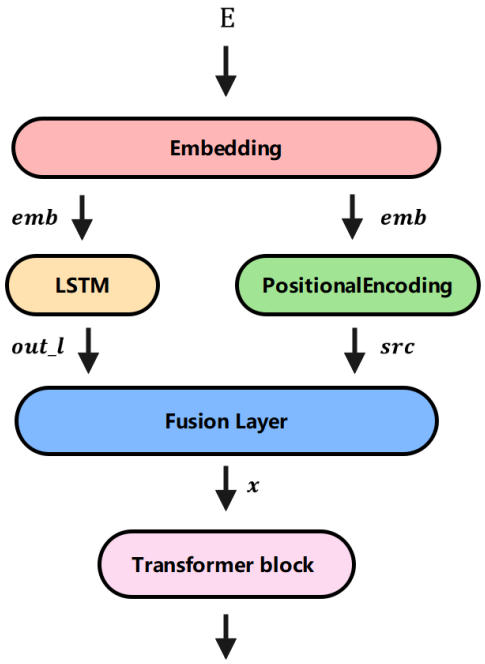

Figure 6. LT model: Transformer with LSTM.

The final feature matrix E, including the log sequence feature matrix and component sequence feature matrix, is processed by Word2Vec and TF-IDF. The feature matrix E in the proposal is input to the anomaly detection model based on LSTM-Transformer, which is processed by the Embedding layer and transmitted to the LSTM layer and the PositionalEncoding layer. The output matrix of the LSTM layer and the PositionalEncoding layer are combined and sent to the Transformer block. The calculation process can be summarized as follows:

$$\boldsymbol{emb} = Encoder(E), \tag{8}$$

$$\boldsymbol{out\_l}, h = LSTM(\boldsymbol{emb}), \tag{9}$$

$$\boldsymbol{src} = Pos\_encoder(\boldsymbol{emb} \times \sqrt{ninp}), \tag{10}$$

$$\boldsymbol{x} = Fusion(\boldsymbol{src}, \boldsymbol{out\_l}). \tag{11}$$

where ***emb*** denotes the output of Embedding, ***out_l*** and ***src*** are the outputs of LSTM and PositionalEncoding, respectively, and *ninp* is the embedded dimension in the Embedding layer. $Fusion(\cdot)$ is a combined operation for the Fusion layer, and $\boldsymbol{x}$ is the output of the Fusion layer, which is input to the next module.

We next discuss the decoder parts of the Transformer model, involving the MultiheadAttention, normalization, and full connection layers, as shown in Figure 7. The output $\boldsymbol{x}$ of the Fusion layer is regarded as the input of the module, and the calculation process of the module is as follows:

$$Q = \boldsymbol{x}W^Q, \tag{12}$$

$$K = \boldsymbol{x}W^K, \tag{13}$$

$$V = \boldsymbol{x}W^V, \tag{14}$$

$$H_i = Softmax\left(\frac{Q_i K_i^T}{\sqrt{d_k}}\right)V_i \tag{15}$$

$$\boldsymbol{O} = Concat(H_1, H_2, \ldots, H_{n-1}, H_n)W^O. \tag{16}$$

where W^Q, W^K, and W^V are learnable parameter matrices, and the input data are transferred to query, key, and value. The dimension of the key vector is d_k. The $Concat(\cdot)$ function concatenates the output of each header with the help of the parameter matrix W^O.

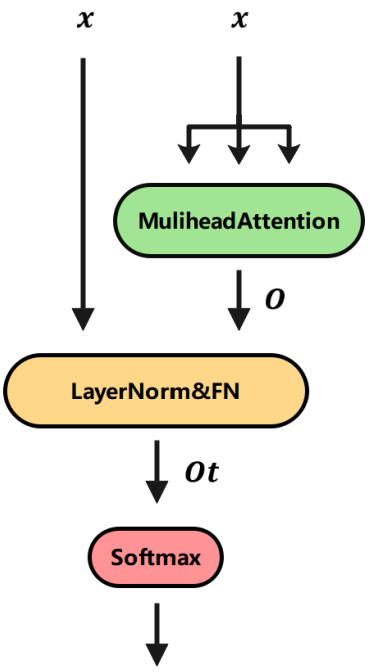

Figure 7. Transformer block.

The $\boldsymbol{O}$ processed by multiple attention is added and connected with its input $\boldsymbol{x}$ to provide $\boldsymbol{Ot}$ through the full connection layer,

$$\boldsymbol{Ot} = FNN(\boldsymbol{O} + \boldsymbol{x}), \tag{17}$$

where $FNN(\cdot)$ is a feedforward network. An $\boldsymbol{Ot}$ with the same size and shape as $\boldsymbol{x}$ can be input to the next module.

Softmax calculates the prediction label y_pred for the log sequence, in which 0 indicates that the input data are normal and 1 indicates abnormality. Cross-entropy for

our loss function can be utilized in the training phase, making full use of the adaptive moment estimation optimization algorithm (Adam) when tuning model parameters. LSTM and Transformer are combined in a powerful and robust model through intersectional information representation.

4. Experiments

4.1. Datasets

Experiments were conducted on two public datasets: HDFS [5], a distributed system log; and BGL [19], a supercomputer log.

The HDFS log dataset is based on map-reduce jobs from running Hadoop with more than 200 AmazonEC2 nodes. It has 11,175,629 items of log information, including time (year, month, day, hour, minute, and second), source IP address, and data size. The block_id can be divided into 575,061 operation sequences, with 2.9% of them marked as exceptions by Hadoop domain-related experts.

BGL was formed by the BlueGene/L supercomputer system at Lawrence Livermore National Laboratory (LLNL) [18]. Its 4,747,963 log messages include 348,460 logs regarded as anomalous. Unlike HDFS, BGL logs do not capture the block_id generated by each log event. Consequently, we must split the log messages into log sequences with sliding windows. Once there is an error log in a log sequence, it is deemed an exception.

Log Anomaly Characteristics

As computer systems continue to increase in complexity and size, complex log data and various types of anomalies become more common. A large-scale system will inevitably encounter failures that lead to changes in log patterns. Anomalies in log data sets refer to events that do not conform to the normal behavior pattern recorded over a certain period of time, which may be due to software errors, hardware failures, malicious attacks, or other reasons. The anomaly types of log data include log sequence anomalies, log event anomalies, and component value anomalies, as shown in Figure 8. A log sequence anomaly mainly refers to an obvious deviation or inconsistency between the sequence of log occurrences and the usual sequence. Abnormal log events mainly refer to content in the log that is inconsistent with normal situations, such as the appearance of unknown IP addresses or software version information. Component value anomalies mainly refer to component value information generated by logs that does not match normal situations, which may be caused by illegal acquisition of permissions or malicious attacks.

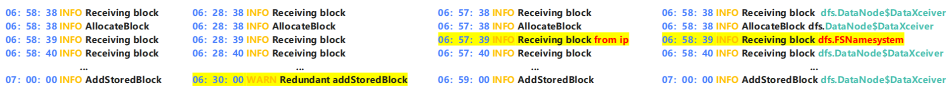

Figure 8. Log anomaly characteristics.

If log anomalies in a system are not identified in a timely manner, they may lead to potential security risks and threats to the system, which can result in data breaches, system attacks, or malware infections.

4.2. Evaluation Metrics

Log anomaly detection is typically viewed as a dichotomous issue, where Precision, Recall, and F1-measure are used to measure model efficiency. These indicators [23,43] have also been mentioned in previous studies. These indicators are calculated as

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

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

$$F1 - measure = 2 \times \left(\frac{Precision \times Recall}{Precision + Recall} \right). \quad (20)$$

where TP means that an exception log sequence is correctly predicted, FN represents an incorrect prediction of an abnormal log sequence, FP indicates the prediction of a normal log sequence as an exception, and TN indicates a correct normal log sequence prediction. Precision is the ratio of actual anomalies among all reported anomalies. Recall is the ratio of correctly recognized log sequences to all actual anomalies. $F1 - measure$ is the harmonic average of precision and recall.

4.3. Experimental Results

Experiments on LTAnomaly were implemented on a CPU-equipped Windows machine with PyTorch. Calculation speed and results become better on a GPU with specific hardware. The window size of the log sequence and the batch_size were set to 15 and 128, respectively. With two layers of LSTM and two layers of Transformer, we set the initial learning rate to 0.0001 and trained the LTAnomaly model for 20 epochs. LTAnomaly was compared with six baseline methods with unsupervised learning: LogCluster [44], DeepLog [8], LogRobust [9], HitAnomaly [12], CL2MLog [45], and LogLS [46].

4.3.1. HDFS Dataset Evaluation

Table 1 shows the performance of LTAnomaly and the six baseline methods on the HDFS dataset, with the highest score expressed in bold. LTAnomaly shows the highest recall rate, with an F1-measure of 0.985. High F1-measures of LogRobust, HitAnomaly, CL2MLog, LogLS, and LTAnomaly make it clear that the semantic information of the log template has an impact on exception detection. LogRobust produces more errors than LTAnomaly because it does not use component value information provided by the dataset. HitAnomaly is better than LTAnomaly when detecting normal logs. However, there are more false detections of anomalous logs, with serious ramifications for the system. Further observation shows that many anomaly detection methods work efficiently on HDFS datasets, i.e., the log parser does a good job extracting log templates from HDFS datasets. Therefore, the results show that the log parser plays an important role in the anomaly detection of logs.

Table 1. HDFS dataset for evaluation.

Technique	Precision	Recall	F1-Measure
DeepLog	0.92	0.95	0.934
LogCluster	0.96	0.83	0.890
LogRobust	0.96	0.96	0.970
HitAnomaly	**0.99**	0.97	0.979
CL2MLog	0.96	0.98	0.970
LogLS	0.96	0.98	0.970
LTAnomaly	0.98	**0.99**	**0.985**

For large systems that are not easily manageable, small improvements can be crucial for efficiency. In some cases, making LTAnomaly 0.02 more accurate than LogLS may result in more precise identification of potential failures or anomalies, thus enabling businesses or organizations to react faster and avoid potential losses. Moreover, in critical domains such as finance and healthcare, even a relatively minor accuracy improvement can have a significant impact on system security and reliability.

The LTAnomaly model achieves high accuracy in log anomaly detection by combining semantic information, sequence relationships, and component values to represent logs as vectors. This is achieved by incorporating a Transformer with long short-term memory (LSTM) as the final classification model, which enables the model to consider both local and global information when processing sequences. The flexibility, simplicity, and robustness of

the LTAnomaly model make it more effective than existing log anomaly detection models, and it has been shown to perform well in detecting HDFS data anomalies. On the other hand, Transformer-based anomaly detection methods also use deep learning techniques, and the Transformer architecture has been adapted for time series analysis by using a self-attention mechanism to learn the relationship between different parts of time series data. The reason why LTAnomaly may outperform Transformer-based methods in HDFS anomaly detection is due to various factors, such as the size and complexity of the dataset, the quality of the data, and the hyperparameter tuning of each algorithm. The design of its model architecture increases the sensitivity to long sequence information, making the model more accurate in considering the relationship between local and global information. Moreover, the unique characteristics of the dataset, such as the size and complexity of the logs or the nature of the detected anomalies, may favor the LTAnomaly model over other methods, confirming that there are scenarios where our proposed model performs better at log anomaly detection.

Log anomaly detection requires not only high accuracy but also high efficiency. This paper compares the time-consuming performance of Deeplog, HitAnomaly, and LTAnomaly methods on HDFS datasets. To ensure the consistency of the experimental results, Deeplog, HitAnomaly, and LTAnomaly are conducted in the same experimental environment.

Table 2 shows the results of the experiment. DeepLog takes less time than LTAnomaly at the expense of low accuracy. HitAnomaly takes more time than LTAnomaly because the number of parameters in the Transformer-encoder is much larger than that in LSTM. Based on the experimental results, LTAnomaly dramatically reduces time consumption while ensuring high accuracy. LTAnomaly is an efficient log anomaly detection model.

Table 2. Time performance comparison of different anomaly detection models.

Model	Number of Logs	Time Consumption
Deeplog	787,095	2 h 17 m 29 s
HitAnomaly	787,095	4 h 29 m 56 s
LTAnomaly	787,095	3 h 22 m 6 s

4.3.2. BGL Dataset Evaluation

Table 3 compares the performance of LTAnomaly and the six baseline models on the BGL dataset, with the highest score expressed in bold. LTAnomaly has the best accuracy, with an F1-measure of 0.975, a 4% increase over CL2MLog. The detection accuracy of LogCluster is not high, due to the high-dimensional sparsity of the event count matrix. Hence, it is difficult to distinguish between anomalies and normal situations with log clusters, leading to many false predictions. DeepLog only considers the temporal characteristics of log sequences and ignores the semantic characteristics of log messages, resulting in worse performance than LTAnomaly. The LogLS model performs better on HDFS and shows a poor performance on BGL datasets, indicating that it has a small application range. As the experimental results show, LTAnomaly performs well on both HDFS and BGL, indicating a wide range of applications.

Table 3. BGL dataset for evaluation.

Techniques	Precision	Recall	F1-Measure
DeepLog	0.91	0.71	0.797
LogCluster	0.42	0.87	0.541
LogRobust	0.91	0.78	0.840
HitAnomaly	0.95	0.90	0.924
CL2MLog	0.91	0.97	0.939
LogLS	0.68	**0.99**	0.809
LTAnomaly	**0.97**	0.98	**0.975**

In the experiments, we examined the effects of window size and number of layers on the performance of our model. We tested the BGL dataset, changing one parameter value and setting the others to default values.

The partition of the log sequence is determined by the window size. To investigate the influence of window size on the model, we set it to 5, 10, 15, 20, and 25. As we can see from Figure 9, LTAnomaly has relatively stable accuracy for different window sizes, with a certain influence on recall. With an increased window size, the efficiency of LTAnomaly anomaly detection does not show significant degradation, and it remains high. However, too small a window results in low recall, so the model cannot detect some anomalies. The model performs best with a window size of 15. The number of LSTM layers is also important for LTAnomaly. To observe its influence on LTAnomaly, we set the number of layers as 1, 2, 3, 4, and 6. Figure 10 shows its influence on LTAnomaly, from which we can find that the anomaly detection of the model is basically the same when the number of layers is greater than or equal to 2. However, the more parameters there are, the longer the model training and detection times. Hence, we prefer two LSTM layers.

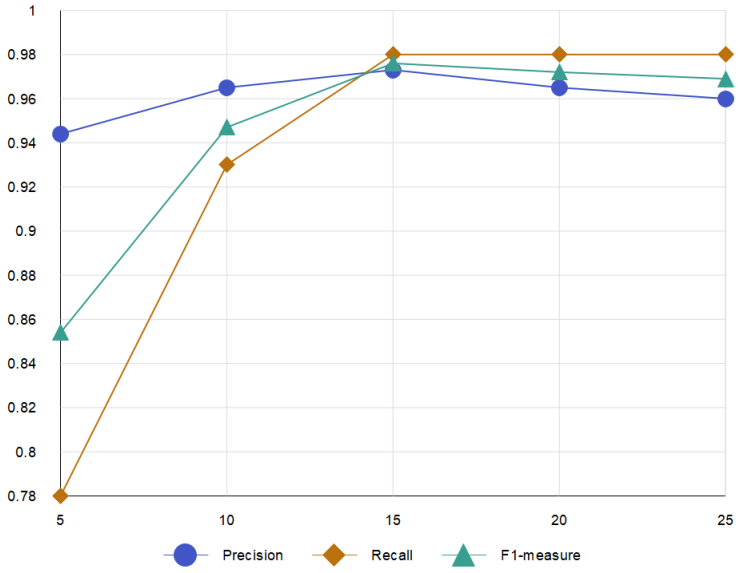

Figure 9. Effect of window size.

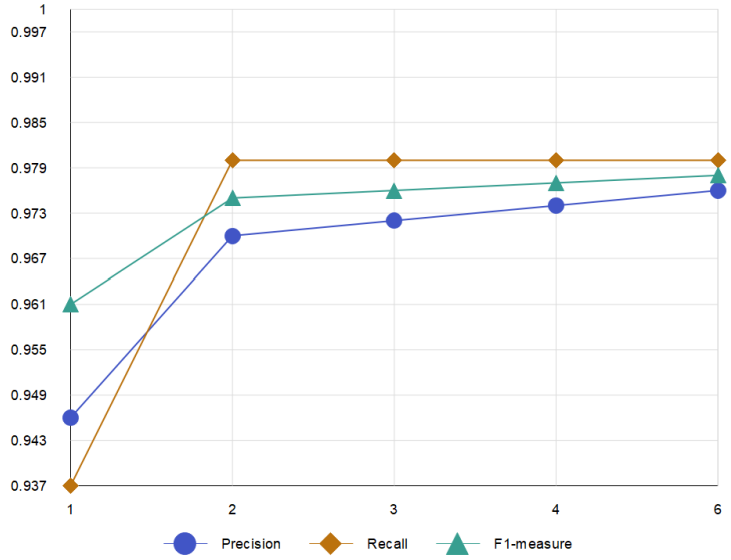

Figure 10. Influence of LSTM layers.

4.4. Ablation Study

In this study, we considered not only the semantic information of log messages and the sequential relationships between logs but also the component value parameters in logs. To verify the advantages of feature combinations, we evaluated the performance of LTAnomaly without component value parameters in the log. Without the semantic message of the log template sequence, LTAnomaly only models the accuracy of component value parameters. Therefore, ablation experiments were performed on the BGL and HDFS datasets. Table 4 shows the performances of different feature extraction techniques in two datasets, in which "-" and "O" indicate using and not using the technique, respectively, and W means that we should think about the semantic information of the log message and the sequential relationship between logs, with the highest score expressed in bold. More importantly, C represents the component value parameter that uses the log. We observe that LTAnomaly with no logging component parameters has a lower recall rate than that with logging component parameters, indicating that some exceptions caused by parameter value errors are ignored. The accuracy of LTAnomaly without log semantic and sequence information is relatively low; hence, this information is important for anomaly detection. The LTAnomaly model (with no component value parameters) and the LogRobust model ignore component value information. However, LTAnomaly outperforms LogRobust, which shows that Transformer with LSTM is more effective in logging exception detection.

In log anomaly detection, it is crucial to consider the values of log components because they provide information about the state of the application or system. Abnormal events usually cause abnormal changes in log component values. Therefore, analyzing log component values can effectively detect and identify abnormal events. When selecting an anomaly detection algorithm, various log component values must be taken into account, and attempts should be made to determine which values are most suitable for detecting anomalous events. Attention should also be paid to issues such as how to perform preprocessing and normalization and how to deal with missing values in the data to ensure the quality and reliability of the data. By considering these factors collectively, the proposed approach's performance results can be improved, and accurate log anomaly detection can be achieved.

Table 4. The performance of different feature extraction technology combinations in datasets.

Dataset	Techniques		Precision	Recall	F1-Measure
	W	C			
BGL	-	O	0.95	0.91	0.929
	O	-	0.79	0.86	0.823
	-	-	**0.97**	**0.98**	**0.975**
HDFS	-	O	0.97	0.98	0.975
	O	-	0.83	0.89	0.859
	-	-	**0.98**	**0.99**	**0.985**

This study uses Transformer with LSTM for log exception detection for the first time. We investigated whether the fusion of LSTM and Transformer can improve the accuracy of detection. Ablation experiments on the HDFS and BGL datasets were performed to analyze the impact of each component on the overall model. Table 5 shows the performance of different anomaly detection techniques in two datasets, with the best results highlighted in bold. Table 5 confirms that adding LSTM modules to Transformer provides better exception detection performance, with the highest score expressed in bold.

Table 5. The performance of different anomaly detection technology combinations in datasets.

Dataset	Techniques	Precision	Recall	F1-Measure
BGL	LSTM	0.92	0.86	0.889
	Transformer	0.95	0.91	0.929
	LTAnomaly	**0.97**	**0.98**	**0.975**
HDFS	LSTM	0.96	0.98	0.970
	Transformer	**0.99**	0.97	0.979
	LTAnomaly	0.98	**0.99**	**0.985**

4.5. Explainability

LTAnomaly employs an LSTM-based Transformer architecture that compresses the input log sequence into a fixed-length vector representation in the encoder and converts this vector representation back to the original log sequence in the decoder. This structure is intuitive and is easy to understand and interpret. Additionally, LTAnomaly can determine which features are most important for anomaly detection by analyzing the attention mechanism in the model. This feature importance analysis can help users gain a deeper understanding of the causes and characteristics of exceptions, enabling them to handle exceptions more effectively. The model can label all items or subsets of the input data and observe the model's response to these specific data points, allowing the user to better understand where the model made incorrect predictions or accurate predictions. Visualization techniques can be used to visualize the inner workings of the model. For example, drawing a diagram between inputs, outputs, and internal states can help provide better understanding of how the model processes inputs and generates outputs. This is how models contribute to explainability.

4.6. Robustness

To account for the differences in various environments, such as different computer, server, or network conditions, we need to evaluate the stability of technical performance. To achieve this, we conducted tests on multiple hardware devices and found that the differences in test results were not significant. However, based on past experience, we recognize that data sets significantly impact technical performance. Therefore, during the testing process, we paid special attention to using data sets of different types, sizes, and qualities. It is gratifying to note that the proposed technique shows good applicability and generalization. One thing to note is that certain technologies may depend on specific parameter settings. Therefore, during the test, we attempted multiple parameter combinations to determine the optimal setting and verified the robustness of its adjusted results. We use metrics such as accuracy and F1-measure to evaluate the technology's performance and cross-validation methods to verify the reliability of the results.

5. Conclusions

We designed a log-based abnormal detection model: LTAnomaly. The semantic information of log sequences and component-valued parameters in logs were utilized to represent logs. We applied Transformer with LSTM to implement anomaly detection. The model offers a more accurate representation of logs and various anomaly detection methods to improve detection efficiency. The experimental results demonstrated that wider applications increased detection accuracy. The model's ability to accurately represent logs using semantic information in log sequences and component-valued parameters in logs can enable wider applications and improve detection efficiency. One potential scenario for the use of this model is in detecting anomalies in large-scale distributed systems, such as cloud computing infrastructures. The LTAnomaly model can automate the process of detecting anomalies, thereby reducing the time required to identify and resolve issues. Furthermore, the LTAnomaly model can help detect anomalous activities within network logs and alert security personnel to take immediate action to prevent a security breach.

However, a limitation of our work is that variable parameters in log messages cannot be well handled. In the future, we aim to represent logs in a simpler and more comprehensive manner and detect anomalies more accurately and quickly. A direction of our future work is to be able to forecast anomalies and take effective measures to protect against them, hence reducing their associated damages.

Author Contributions: Conceptualization, M.L.; Funding acquisition, D.H.; Methodology, M.S.; Supervision, M.L. and Q.C.; Validation, D.H; Writing—original draft, M.S.; Writing—review and editing, M.S.; Data curation, Q.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Key Research and Development Program of China (2022YFB4004401) and the Pilot Project for Integrated Innovation of Science, Education, and Industry of Qilu University of Technology (Shandong Academy of Sciences) (2022JBZ01-01).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Datasets can be accessed upon request by the corresponding author.

Acknowledgments: The authors appreciate all reviewers for their insightful comments and constructive suggestions.

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

References

1. Khan, S.; Parkinson, S. Eliciting and Utilising Knowledge for Security Event Log Analysis: An Association Rule Mining and Automated Planning Approach. *Expert Syst. Appl.* **2018**, *113*, 116–127. [CrossRef]
2. Meng, W.; Liu, Y.; Zhu, Y.; Zhang, S.; Pei, D.; Liu, Y.; Chen, Y.; Zhang, R.; Tao, S.; Sun, P.; et al. Loganomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. In Proceedings of the 28th International Joint Conference on Artificial Intelligence, IJCAI'19, Macao, China, 10–16 August 2019; AAAI Press: Washington, DC, USA, 2019; pp. 4739–4745.
3. Gao, P.; Xiao, X.; Li, D.; Li, Z.; Jee, K.; Wu, Z.; Kim, C.H.; Kulkarni, S.R.; Mittal, P. SAQL: A Stream-Based Query System for Real-Time Abnormal System Behavior Detection. In Proceedings of the 27th USENIX Conference on Security Symposium, SEC'18, Berkeley, CA, USA, 15–17 August 2018; USENIX Association: Baltimore, MD, USA, 2018; pp. 639–656.
4. Gao, P.; Xiao, X.; Li, Z.; Jee, K.; Xu, F.; Kulkarni, S.R.; Mittal, P. AIQL: Enabling Efficient Attack Investigation from System Monitoring Data. In Proceedings of the 2018 USENIX Conference on Usenix Annual Technical Conference, USENIX ATC '18, Boston, MA, USA, 11–13 July 2018; USENIX Association: Boston, MA, USA, 2018; pp. 113–125.
5. Xu, W.; Huang, L.; Fox, A.; Patterson, D.; Jordan, M.I. Detecting Large-Scale System Problems by Mining Console Logs. In Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, SOSP '09, New York, NY, USA, 11–14 October 2009; Association for Computing Machinery: New York, NY, USA, 2009; pp. 117–132. [CrossRef]
6. Lou, J.G.; Fu, Q.; Yang, S.; Xu, Y.; Li, J. Mining Invariants from Console Logs for System Problem Detection. In Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference, USENIXATC'10, Berkeley, CA, USA, 23–25 June 2010; USENIX Association: Boston, MA, USA, 2010; p. 24.
7. He, P.; Zhu, J.; He, S.; Li, J.; Lyu, M.R. Towards Automated Log Parsing for Large-Scale Log Data Analysis. *IEEE Trans. Dependable Secur. Comput.* **2018**, *15*, 931–944. [CrossRef]
8. 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, CCS '17, Dallas, TX, USA, 30 October–3 November 2017; Association for Computing Machinery: New York, NY, USA, 2017; pp. 1285–1298. [CrossRef]
9. Zhang, X.; Xu, Y.; Lin, Q.; Qiao, B.; Zhang, H.; Dang, Y.; Xie, C.; Yang, X.; Cheng, Q.; Li, Z.; 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, ESEC/FSE, Tallinn, Estonia, 26–30 August 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 807–817. [CrossRef]
10. Liang, Y.; Zhang, Y.; Xiong, H.; Sahoo, R. Failure Prediction in IBM BlueGene/L Event Logs. In Proceedings of the Seventh IEEE International Conference on Data Mining (ICDM 2007), Omaha, NE, USA, 28–31 October 2007; pp. 583–588. [CrossRef]
11. Zhang, K.; Xu, J.; Min, M.R.; Jiang, G.; Pelechrinis, K.; Zhang, H. Automated IT system failure prediction: A deep learning approach. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; pp. 1291–1300. [CrossRef]
12. Huang, S.; Liu, Y.; Fung, C.; He, R.; Zhao, Y.; Yang, H.; Luan, Z. HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Trans. Netw. Serv. Manag.* **2020**, *17*, 2064–2076. [CrossRef]

13. Zhou, J.; Qian, Y.; Zou, Q.; Liu, P.; Xiang, J. DeepSyslog: Deep Anomaly Detection on Syslog Using Sentence Embedding and Metadata. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3051–3061. [CrossRef]
14. Cheng, S.W.; Pei, D.; Wang, C.J. Error Log Clustering of Internet Software. *J. Chin. Comput. Syst.* **2018**, *39*, 865–870.
15. Vaarandi, R. A data clustering algorithm for mining patterns from event logs. In Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764), Kansas City, MO, USA, 1–3 October 2003; pp. 119–126. [CrossRef]
16. Makanju, A.; Zincir-Heywood, A.N.; Milios, E.E. A Lightweight Algorithm for Message Type Extraction in System Application Logs. *IEEE Trans. Knowl. Data Eng.* **2012**, *24*, 1921–1936. [CrossRef]
17. Du, M.; Li, F. Spell: Streaming Parsing of System Event Logs. In Proceedings of the 2016 IEEE 16th International Conference on Data Mining (ICDM), Barcelona, Spain, 12–15 December 2016; pp. 859–864. [CrossRef]
18. Du, M.; Li, F. Spell: Online Streaming Parsing of Large Unstructured System Logs. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 2213–2227. [CrossRef]
19. He, P.; Zhu, J.; Zheng, Z.; Lyu, M.R. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In Proceedings of the 2017 IEEE International Conference on Web Services (ICWS), Honolulu, HI, USA, 25–30 June 2017; pp. 33–40. [CrossRef]
20. Zhang, S.; Meng, W.; Bu, J.; Yang, S.; Liu, Y.; Pei, D.; Xu, J.; Chen, Y.; Dong, H.; Qu, X.; et al. 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), Vilanova i la Geltru, Spain, 14–16 June 2017; pp. 1–10. [CrossRef]
21. Studiawan, H.; Sohel, F.; Payne, C. Automatic Event Log Abstraction to Support Forensic Investigation. In Proceedings of the Australasian Computer Science Week Multiconference, ACSW '20, Melbourne, Australia, 4–6 February 2020; Association for Computing Machinery: New York, NY, USA, 2020. [CrossRef]
22. Studiawan, H.; Payne, C.N.; Sohel, F. Automatic Graph-Based Clustering for Security Logs. In Proceedings of the Advanced Information Networking and Applications (AINA), Matsue, Japan, 27–29 March 2019.
23. Zhu, J.; He, S.; Liu, J.; He, P.; Xie, Q.; Zheng, Z.; Lyu, M.R. Tools and Benchmarks for Automated Log Parsing. In Proceedings of the 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP), Montréal, QC, Canada, 27 May 2019; pp. 121–130. [CrossRef]
24. Doreswamy, H.; Hooshmand, M.K.; Gad, I. Feature Selection Approach Using Ensemble Learning for Network Anomaly Detection. *Caai Trans. Intell. Technol.* **2020**, *5*, 283–293. [CrossRef]
25. Xu, H.; Pang, G.; Wang, Y.; Wang, Y. Deep Isolation Forest for Anomaly Detection. *IEEE Trans. Knowl. Data Eng.* **2023**, 1–14. [CrossRef]
26. Zeng, C.; Jiang, Y.; Zheng, L.; Li, J.; Li, L.; Li, H.; Shen, C.; Zhou, W.; Li, T.; Duan, B.; et al. FIU-Miner: A fast, integrated, and user-friendly system for data mining in distributed environment. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013.
27. Mondal, T.; Pramanik, P.; Bhattacharya, I.; Boral, N.; Ghosh, S. Analysis and Early Detection of Rumors in a Post Disaster Scenario. *Inf. Syst. Front.* **2018**, *20*, 961–979. [CrossRef]
28. Troudi, A.; Zayani, C.A.; Jamoussi, S.; Amor, I.A. A New Mashup Based Method for Event Detection from Social Media. *Inf. Syst. Front.* **2018**, *20*, 981–992. [CrossRef]
29. Shukla, A.K.; Srivastav, S.; Kumar, S.; Muhuri, P.K. UInDeSI4.0: An efficient Unsupervised Intrusion Detection System for network traffic flow in Industry 4.0 ecosystem. *Eng. Appl. Artif. Intell.* **2023**, *120*, 105848. [CrossRef]
30. Malki, A.; Atlam, E.S.; Gad, I. Machine learning approach of detecting anomalies and forecasting time-series of IoT devices. *Alex. Eng. J.* **2022**, *61*, 8973–8986. [CrossRef]
31. Chen, M.; Zheng, A.; Lloyd, J.; Jordan, M.; Brewer, E. Failure diagnosis using decision trees. In Proceedings of the International Conference on Autonomic Computing, New York, NY, USA, 17–18 May 2004; pp. 36–43. [CrossRef]
32. Farshchi, M.; Schneider, J.G.; Weber, I.; Grundy, J. Experience report: Anomaly detection of cloud application operations using log and cloud metric correlation analysis. In Proceedings of the 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), Gaithersbury, MD, USA, 2–5 November 2015; pp. 24–34. [CrossRef]
33. Zhang, S.; Liu, Y.; Meng, W.; Luo, Z.; Bu, J.; Yang, S.; Liang, P.; Pei, D.; Xu, J.; Zhang, Y.; et al. PreFix: Switch Failure Prediction in Datacenter Networks. *Proc. ACM Meas. Anal. Comput. Syst.* **2018**, *2*, 1–29. [CrossRef]
34. Bertero, C.; Roy, M.; Sauvanaud, C.; Trédan, G. Experience report: Log mining using natural language processing and application to anomaly detection. In Proceedings of the 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), Toulouse, France, 23–26 October 2017; pp. 351–360.
35. Yang, R.; Qu, D.; Qian, Y.; Dai, Y.; Zhu, S. An online log template extraction method based on hierarchical clustering. *Eurasip J. Wirel. Commun. Netw.* **2019**, *2019*, 135. [CrossRef]
36. Ruff, L.; Vandermeulen, R.A.; Görnitz, N.; Binder, A.; Müller, E.; Müller, K.R.; Kloft, M. Deep semi-supervised anomaly detection. *arXiv* **2019**, arXiv:1906.02694.
37. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep Learning for Anomaly Detection: A Review. *ACM Comput. Surv.* **2021**, *54*, 1–38. [CrossRef]
38. Brown, A.; Tuor, A.; Hutchinson, B.; Nichols, N. Recurrent Neural Network Attention Mechanisms for Interpretable System Log Anomaly Detection. In Proceedings of the First Workshop on Machine Learning for Computing Systems, MLCS'18, Tempe, AZ, USA, 12 June 2018; Association for Computing Machinery: New York, NY, USA, 2018. [CrossRef]

39. Catillo, M.; Pecchia, A.; Villano, U. AutoLog: Anomaly detection by deep autoencoding of system logs. *Expert Syst. Appl.* **2022**, *191*, 116263. [CrossRef]
40. Li, S.; Jin, X.; Xuan, Y.; Zhou, X.; Chen, W.; Wang, Y.X.; Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 1–11.
41. Rae, J.W.; Potapenko, A.; Jayakumar, S.M.; Lillicrap, T.P. Compressive transformers for long-range sequence modelling. *arXiv* **2019**, arXiv:1911.05507.
42. Gulati, A.; Qin, J.; Chiu, C.C.; Parmar, N.; Zhang, Y.; Yu, J.; Han, W.; Wang, S.; Zhang, Z.; Wu, Y.; et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv* **2020**, arXiv:2005.08100.
43. He, P.; Zhu, J.; He, S.; Li, J.; Lyu, M.R. An Evaluation Study on Log Parsing and Its Use in Log Mining. In Proceedings of the 2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Toulouse, France, 28 June–1 July 2016; pp. 654–661. [CrossRef]
44. Lin, Q.; Zhang, H.; Lou, J.G.; Zhang, Y.; Chen, X. Log Clustering Based Problem Identification for Online Service Systems. In Proceedings of the 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), Austin, TX, USA, 14–22 May 2016; pp. 102–111.
45. Wang, X.; Cao, Q.; Wang, Q.; Cao, Z.; Zhang, X.; Wang, P. Robust log anomaly detection based on contrastive learning and multi-scale MASS. *J. Supercomput.* **2022**, 78, 17491–17512. [CrossRef]
46. Chen, Y.; Luktarhan, N.; Lv, D. LogLS: Research on System Log Anomaly Detection Method Based on Dual LSTM. *Symmetry* **2022**, *14*, 454. [CrossRef]