electronics

MDPI

Article

Research on Log Anomaly Detection Based on Sentence-BERT

Caiping Hu [1,*], Xuekui Sun [2], Hua Dai [2], Hangchuan Zhang [1] and Haiqiang Liu [1]

[1] Department of Computer Engineering, Jinling Institute of Technology, Nanjing 211169, China; 19850303315@163.com (H.Z.); 00000005207@jit.edu.cn (H.L.)
[2] School of Computer Science, Nanjing University of Posts and Telecommunications, Nanjing 210023, China; kasonsun@foxmail.com (X.S.); daihua@njupt.edu.cn (H.D.)
* Correspondence: hucp@jit.edu.cn

Abstract: Log anomaly detection is crucial for computer systems. By analyzing and processing the logs generated by a system, abnormal events or potential problems in the system can be identified, which is helpful for its stability and reliability. At present, due to the expansion of the scale and complexity of software systems, the amount of log data grows enormously, and traditional detection methods have been unable to detect system anomalies in time. Therefore, it is important to design log anomaly detection methods with high accuracy and strong generalization. In this paper, we propose the log anomaly detection method LogADSBERT, which is based on Sentence-BERT. This method adopts the Sentence-BERT model to extract the semantic behavior characteristics of log events and implements anomaly detection through the bidirectional recurrent neural network, Bi-LSTM. Experiments on the open log data set show that the accuracy of LogADSBERT is better than that of the existing log anomaly detection methods. Moreover, LogADSBERT is robust even under the scenario of new log event injections.

Keywords: anomaly detection; log; deep learning; Sentence-BERT; semantic feature

Citation: Hu, C.; Sun, X.; Dai, H.; Zhang, H.; Liu, H. Research on Log Anomaly Detection Based on Sentence-BERT. *Electronics* **2023**, *12*, 3580. https://doi.org/10.3390/electronics12173580

Academic Editor: Grzegorz Dudek

Received: 23 July 2023
Revised: 17 August 2023
Accepted: 22 August 2023
Published: 24 August 2023

1. Introduction

Logs usually contain information about the operational status of a system, including operation records, fault information, security time, etc., which can provide a comprehensive view of the system's operational status [1]. Logs are time-series in nature; the information in the logs is recorded by time, which allows us to analyze it in order to gain insight into the operation of the system. Logs can provide a historical view—they collect all information about the application, and there are a lot of helpful insights that can be gleaned from an application's history record, including information about potential problems and benchmarks for determining when a process becomes an exception. Logs can monitor the behavior of a system, and in contrast to other data sources, they can go deeper into the system and track the actual behavior of the system as it runs. Log records contain information and trends during system operation. Analyzing and mining the log data can help detect and diagnose system anomalies.

With the expansion of software systems' scale, complexity, and application scope, the number of logs generated shows exponential growth, making it difficult for the traditional log anomaly detection methods based on rules and statistics. In order to adapt to the development of software systems, researchers have shifted their research focus to deep learning-based solutions, and, currently, log anomaly detection based on deep learning has become a hot spot in the field of anomaly detection [2]. Compared to the traditional methods based on rules and statistics, an anomaly detection method based on deep learning requires no human intervention and can quickly and accurately identify abnormal behaviors in logs. Moreover, traditional log anomaly detection methods are constrained by the limitations of algorithms and capacity, whereas a log anomaly detection method based on deep learning can process a large amount of data in parallel. It can efficiently solve the problems of

repeated sampling and information extraction. In addition, deep learning models can extract useful information from dozens of data metrics, which can better capture the details of the log data that reflect the anomalies of the system. This is because log data are usually in plain text format, and natural language processing is a specialized field for processing and analyzing text data. For example, chatbots [3] based on machine translation have become popular in recent years. Most of the log anomaly detection methods based on deep learning that have emerged so far are associated with natural language processing (NLP). NLP is used to extract the semantic features in log files, such as vocabulary, phrases, sentences, and grammatical structures. These features are useful for pattern recognition and classification in log anomaly detection, and the models built in this way can better process and analyze the log data, as well as predict abnormal behaviors and events.

In this paper, we propose the log anomaly detection method LogADSBERT. It uses the Sentence-BERT model [4] to extract the semantic features of log events and realizes the final anomaly detection using the recurrent neural network model Bi-LSTM [5]. LogADSBERT consists of two stages: the model training and the anomaly detection. In the model training stage, the log parser parses the original logs into log events and log triples. The log events are used as the corpus to train the Sentence-BERT model, and the log triples are used to construct a sliding window sequence of log event semantic vectors to train the Bi-LSTM neural network classification model, Bi-LSTM-ADM. In the anomaly detection stage, Bi-LSTM-ADM is used to detect anomalies in the log data. LogADSBERT can achieve anomaly detection with high accuracy and robustness.

The contributions of this paper can be summarized as follows:

1. We construct a log event semantic feature extraction model, T-SBERT, based on the Sentence-BERT model, which can convert log events into log event semantic feature representations. The Bidirectional Long Short-Term Memory Recurrent Neural Network model (Bi-LSTM) with an attention mechanism is adopted to generate an anomaly detection model.
2. We propose a log event semantic feature matching algorithm and an anomaly detection algorithm. The log event semantic matching dictionary is established, and the log anomaly detection method LogADSBERT, based on Sentence-BERT, is constructed. It is, to the best of our knowledge, the first to extract log event semantic features using the Sentence-BERT model.
3. In the scenario of new log event injection, LogADSBERT can ensure high accuracy and strong robustness of anomaly detection. Experiment results demonstrate the effectiveness of the proposed method.

This paper is structured as follows: Section 2 discusses the related work; Section 3 presents the preliminary knowledge of this paper; Section 4 presents the definitions related to the proposed method; Section 5 presents the framework of our anomaly detection method; Section 6 describes the experiments used to evaluate the effectiveness of the proposed method; and finally, the conclusion is provided in Section 7.

2. Related Work

The traditional log anomaly detection methods are based on rules and statistics [6–8] and generally need to analyze normal and abnormal behavior patterns using mathematical counting methods. They usually define a set of features, design response rules for each feature, and combine these rules into a complete system. In the testing stage, the newly generated logs are compared with the existing rules to determine the existence of anomalies. For example, Prewett et al. [7] proposed the log file analysis tool Logsurfer, which achieves anomaly detection by defining rules for the expected behavior of the system and then matching them using regular expressions. At the same time, Logsurfer can also update its rule set at runtime. Rouilard et al. [8] proposed the SEC simple temporal correlator to create feature rule sets by analyzing log sequences, which reduces the false alarm rate but is less automated and incurs higher labor costs. Due to the expansion and update of the scale of log data, the traditional log anomaly detection methods based on rules and statistics are

usually not effective in detecting complex and or unknow anomalies. Thus, researchers in the field have shifted their research direction to the area of machine learning and deep learning.

Traditional machine learning log anomaly detection includes supervised and unsupervised machine learning methods. Supervised machine learning methods include Support Vector Classifier (SVM) [9,10], Linear Regression (LR) [11,12], Decision Tree (DT) [13], K-Neighborhood Algorithm (KNN) [14], etc. These are based on the log frequency statistics vector to record the frequency of occurrence of each log event within the log sequence, and they use the frequency statistics vector as input and dichotomous labels as the classification result. Unsupervised machine learning methods include Principal Component Analysis (PCA) [15] and clustering-based methods such as Isolated Forest (IF) [16], Invariant Mining (IM) [17], and Log Clustering (LC) [18]. These use unlabeled data for training, and unsupervised log anomaly detection can be achieved.

The deep learning-based log anomaly detection methods [19–21] usually have three steps: First, a log parser is used to split the system log data into two parts, the log event and the parameter. The log event describes the system or process behavior, and the parameter element records state information such as the timestamp and the process identifier. Second, the behavior sequence of the system or process is constructed using the timestamp and log event of the log record. Third, anomaly detection is performed based on the behavioral sequences. Researchers have been developing log anomaly detection methods based on recurrent neural networks. For example, Du et al. [19] trained LSTM based on log keys and parameters to obtain a log key anomaly detection model and a parameter value anomaly detection model. They combined two models to achieve anomaly detection. However, the log key is the index of the log event, which is not combined with the semantic features in the real sense. Log key-based detection requires knowledge of the size of the collection of log events before the detection, which may fail when the log events are updated or added. Meng et al. [20] proposed a template2vec-based method, LogAnomaly, that used the Bi-LSTM model with an attention mechanism to combine log event features and word features within the event to obtain the log event semantic feature space vector. When the log event is updated, the semantic feature vector of the log event is computed first, and then the existing log event is replaced by selecting the closest log event with the Euclidean distance. However, the performance drops sharply when more log events are added. Brown et al. [21] also proposed an LSTM-based approach for routine detection that incorporates multiple implementations of attention mechanisms into the LSTM model to extract log features and achieve eventual anomaly detection. Although the experiments show a high accuracy rate for this method on the LANL cyber security datasets, the experimental datasets are relatively limited, and high accuracy cannot be achieved on several publicly available and commonly used datasets. This method only focuses on discovering relationships hidden in system logs and the effectiveness of multiple attention mechanisms in log anomaly detection, which causes limitations in practical application scenarios. In addition, the BERT model and its derivative models, which have recently become popular in the field of natural language processing, have been used in the field of log anomaly detection. For example, Chen et al. [22] produced semantic log vectors by utilizing a pre-trained language BERT model and used the linear classification to detect anomalies. This method uses a single BERT implementation, which may lose semantic information in sequence feature extraction processing. Zhang et al. [23] adopted the SBERT model to extract the semantic representation of log events, which considers the semantic and word order relationship of each word in log events. They designed a GRU model for anomaly detection; however, as the content of exception log is diverse, including sequence pattern, frequency, correlation, etc., GRU can only capture one-way sequence information. Guo et al. [24] learned the patterns of normal log sequences using two novel, self-supervised training tasks: the masked log message prediction and volume of hypersphere minimization. Nevertheless, this work does not identify and train the semantic information of abnormal logs.

Currently, the log anomaly detection methods based on rules and statistics can no longer meet the rapid development of software systems, and machine learning-based log anomaly detection suffers from weak feature extraction ability, poor adaptability, large labor cost, and low accuracy rate compared to deep learning. Therefore, current log anomaly detection research focuses on the deep learning-based methods. However, the existing log anomaly detection methods based on deep learning still do not fully utilize the semantic information existing in the log data, as well as some other feature information such as frequency statistics, location embedding, etc. As a result, the accuracy rate of the methods does not reach the required standard, and the robustness of these methods to the addition of new logs needs to be further improved.

3. Preliminary Knowledge

3.1. Log Parser

System log data as semi-structured data are difficult to input directly into model training and detection, so processing semi-structured log data into structured log data is the first step of data processing and is crucial for subsequent anomaly detection. A system log data includes variable and constant parts. When generating a log, it is actually a process of combining constants and variables. The variable is the log parameters, which change dynamically depending on the type of log generated. The constants are usually fixed and unchanged log events that are the system log in the parameter part of the use of wildcard replacement to get the standard event. LogParser does exactly the opposite of the log generation process; the log parser must generate logs reverse-parsed into log events and parameters in order to better complete the anomaly detection—there are many open-source log parsers to choose from. Currently, log parsers [25–29] can be divided into two main groups: log parsers based on clustering and log parsers based on heuristic structures.

3.2. Self-Attention Mechanism

A self-attention approach was designed by Google in 2017 [30], which was an implementation of the original attention mechanism proposed in 2014 [31]. Early attention mechanisms need to use other neural networks to extract relevant features, compute intermediate states, and finally give different attention to each intermediate state through the attention mechanism. Now, the self-attention mechanism does not need to use other neural networks to extract sequence features. It directly uses the self-attention mechanism to learn sequence features, which solves the problem of other neural networks not being able to perform in parallel and long short-term dependence.

3.3. Sentence-BERT Model

The Sentence-BERT model [3] is a derivative of the pre-training model BERT that sheds the decoder of the Transformer model so that the construction of BERT is the encoder part of the Transformer. BERT has proved to be effective in a variety of NLP tasks, and with pre-training and fine-tuning, it can obtain better results. Sentence-BERT comes from a similar background. It was constructed based on the Siamese Network and Triplet Network [32]; it performs better in clustering and semantic-based retrieval tasks and can quickly and efficiently realize sentence semantic similarity computation and obtain sentence vector representations, etc. In this paper, the pre-training Sentence-BERT uses the log data for training and fine-tuning so that it can obtain better vector representations.

3.4. Bi-LSTM Neural Network Model

Long Short-Term Memory (LSTM) [33] is a common recurrent neural network model that has a much longer memory. It solves the gradient vanishing and long-distance dependence problems that recurrent neural networks are prone to. It has been proved in recent years that LSTM shows good performance in several natural language processing tasks. The Bi-LSTM model [5] is used in the research methodology of this paper, which employs a bidirectional LSTM model. It is a combination of forward LSTM as well as reverse LSTM,

where the hidden output of the current layer is obtained by splicing the processed results of the forward inputs with the processed outputs of the reverse inputs. Bi-LSTM captures backward and forwards temporal correlation and can maximize the use of historical and future information through bi-directional propagation to achieve better performance.

4. Definitions of LogADSBERT

Assuming that the system log set is $L = \{l_1, l_2, \ldots, l_n\}$. After parsing the log set L using LogParser, we obtain a set of the log events $T = \{t_1, t_2, \ldots, t_m\}$ and a set of the log triples $P = \{p_1, p_2, \ldots, p_n\}$.

Definition 1 (Log Event (LE)). *A log event is a structured text information obtained by removing the variable parameter from the system logs l_i using the log parser, which is denoted as $t_i \in T$.*

Definition 2 (Log Triple (LT)). *A log triple is a structured log information obtained by parsing the system logs through the log parser, which is denoted as p_i = (id, t, ts), where id is the process ID, t is the log event, and ts is the timestamp of the log generation.*

Definition 3 (Log Event Semantic Vector (LE-SV)). *Taking the log events of T as the input of the T-SBERT model, the output is the log event semantic vector set $V = \{v_1, v_2, \ldots, v_m\}$.*

Definition 4 (Log Event Semantic Dictionary (LE-SD)). *The log event semantic dictionary is denoted as D, and D is initialized as the mapping set $t_i \rightarrow v_j$, that is $D = \{ t_i \rightarrow v_j \big| t_i \in T, v_j \in V \}$. When a new type of log appears, the log event semantic vector of the new log is obtained by the log event semantic matching algorithm based on the T-SBERT model, and the new mapping $t_i \rightarrow v_j$ is added to the log event semantic dictionary.*

Definition 5 (Log Event Semantic Vector Sliding Window (LE-SV-SW)). *Assuming that h is the size of a sliding window, $T_i = \{e_1, e_2, \ldots, e_q\}$ is the sequence of the log event, and $T_i \subseteq T$ is the sequence of the log event, the semantic matching algorithm of the log event based on the T-SBERT model converts the log event sequence T_i into the log event semantic vector sequence $S_i = <v_{e_1}, v_{e_2}, \ldots, v_{e_q}>$. Given $v_{e_{j+1}} \in S_i$, the corresponding sliding window is denoted as $W(S_i, v_{e_j})$ which is generated according to the following rules.*

1. *If $(h \leq j < q)$, then $W(S_i, v_{e_j}) = <v_{e_{j-h+1}}, v_{e_{j-h+2}}, \ldots, v_{e_j}>$;*
2. *Else, $W(S_i, v_{e_j}) = \varnothing$.*

In addition, for the log event semantic vector sequence S_i that meets the first requirements, the window set of S_i is $W_{s_i} = \{W(S_i, v_{e_j}) \big| v_{e_j} \in S_i \wedge j \in [h,q)\}$, and the number of items in W_{s_i} is q-h. The corresponding log event semantic vector set is $V_{e_{j+1}} = \{ v_{e_{j+1}} \big| v_{e_{j+1}} \in S_i \wedge j \in [h,q)\}$.

Definition 6 (Log Sequence Anomaly Detection (LSAD)). *Assuming that the log event sequence is $T_i = \{e_1, e_2, \ldots, e_q\}$, the log event semantic vector window set is $W_{s_i} = \{W(S_i, v_{e_j}) \mid v_{e_j} \in S_i \wedge j \in [h, q)\}$, and the corresponding set of log event semantic vector $v_{e_{j+1}}$ is $V_{e_{j+1}} = \{v_{e_{j+1}} \mid v_{e_{j+1}} \in S_i \wedge j \in [h,q)\}$, the result vector set predicted by the Bi-LSTM-ADM with inputting W_{s_i} is $R_{e_{j+1}} = \{r_{e_{j+1}} \mid j \in [h,q)\}$. Given the threshold ξ, the log sequence anomaly detection is performed as follows.*

1. *For each $v_{e_{j+1}} \in V_{e_{j+1}}$ and $\forall\, r_{e_{j+1}} \in R_{e_{j+1}}$, if the similarity between $v_{e_{j+1}}$ and $r_{e_{j+1}}$ is greater than the threshold ξ, it can be determined that the log event sequence T_i is normal;*
2. *Otherwise, the log event sequence T_i is abnormal.*

5. Algorithms of LogADSBERT

The proposed LogADSBERT consists of two stages: the model training and the anomaly detection. The specific implementation process of these two stages is described as follows.

Model training stage: The log parser parses the logs into a set of log events and a set of log triples. The set of log events is used as training data for Sentence-BERT and is trained to generate the T-SBERT log event vector generation model based on the *TSBERTTrain* algorithm (Algorithm 1). While the log triples are ordered according to the time stamp *ts* and transformed into a sequence of log event semantic vectors using the log event semantic matching algorithm based on T-SBERT model (Algorithm 2), they are converted into sequences of log event semantic vectors, then the sliding window mechanism is utilized and sliding window training data are constructed based on the log event semantic vector sequences. The Bi-LSTM model is trained to generate the Bi-LSTM-ADM model using the BILSTMADMTrain algorithm (Algorithm 3).

Anomaly detection stage: The logs to be detected are first transformed into a set of log triples using the log parser, then the log event semantic matching algorithm is used to obtain a log event semantic vector sequence. Finally, the log event semantic vector sequence is used to complete the log anomaly detection by the *LogADSBERTDetect* algorithm (Algorithm 4).

The framework of the proposed log anomaly detection method LogADSBERT is shown in Figure 1.

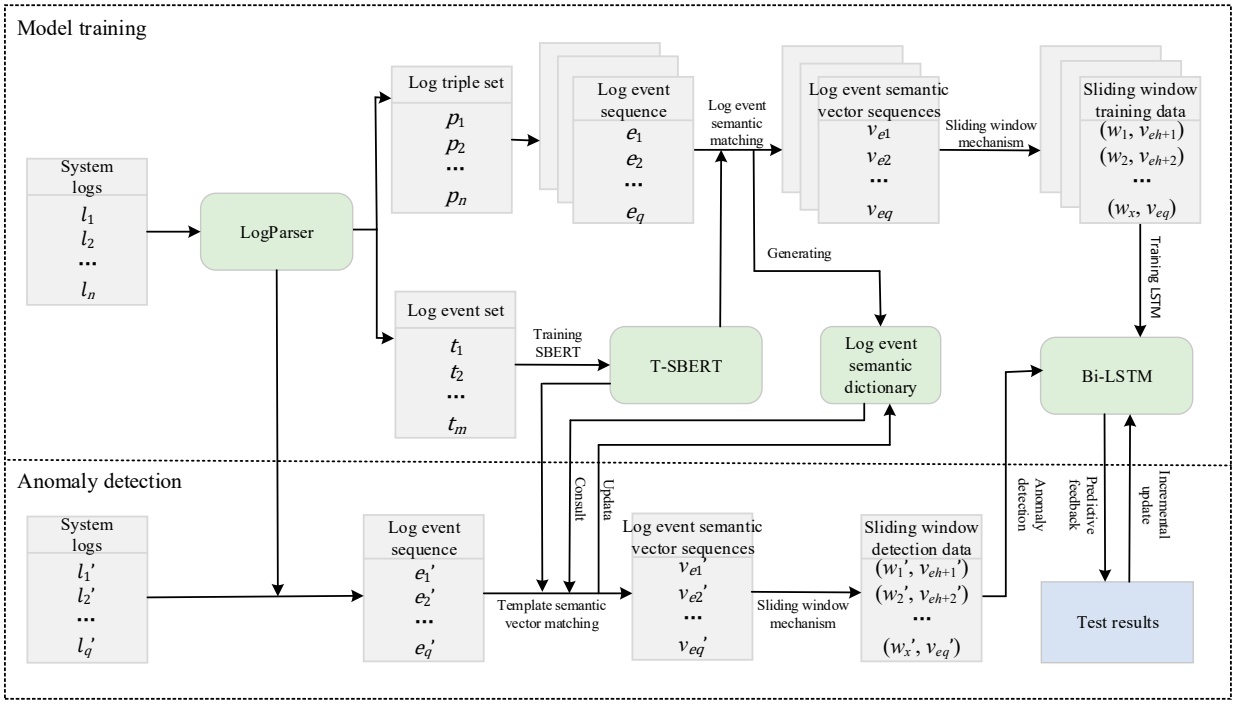

Figure 1. The framework of the LogADSBERT anomaly detection method.

5.1. Sentence-BERT Training Algorithm

In the model training stage, the Sentence BERT model is trained to convert log events into log event semantic vectors, and then the Bi-LSTM model is trained.

TSBERTTrain(*T*): The log event semantic vector generation model T-SBERT is generated based on the Sentence-BERT model using the log event dataset *T*. First, the text corpus (*TC*) is initialized to be empty, the log event set *T* is preprocessed to obtain the text corpus, and *TC* is fed into the Sentence-BERT model to generate the T-SBERT model. The specific process of T-SBERT model generation is shown in Algorithm 1. The log event semantic dictionary *D* is initialized to be empty and is used to store the mapping relationship between log events and log event semantic vectors.

Algorithm 1: TSBERTTrain(T)

Input: Log event set T
Output: Log event semantic vector generation model T-SBERT
(1) Initialize the text corpus $TC = \emptyset$;
(2) Initialize log event semantic dictionary $D = \emptyset$;
(3) Initialize the Sentence-BERT model instance;
(4) **FOR** $t_i \in T$ **DO**
(5) Split t_i into word lists *WL*;
(6) **FOR EACH** *word* **IN** *WL* **DO**
(7) *word* = *lowerCase*(*word*);
(8) **IF** *word* is a stop-words or no semantic identifiers **THEN**
(9) Remove *word* from *WL*;
(10) **END IF**
(11) **END FOR**
(12) Add the corresponding *WL* of the processed sentence to the *corpus*;
(13) Add the *corpus* to the *TC*;
(14) **END FOR**
(15) Train Sentence-BERT model to get T-SBERT using text library *TC*;
(16) **RETURN** T-SBERT;

5.2. Log Event Semantic Matching Algorithm

Before the Bi-LSTM model training and anomaly detection, each log event in a sequence of log events needs to be converted into a log event semantic vector.

LESVMatch (t_i, T-SBERT): Log Event Semantic Vector Matching Algorithm based on T-SBERT implements the process of transforming log events to log event topics in the training and detection stage. For log event t_i, the log event semantic dictionary D is first queried to see if there exists a mapping relationship for $t_i \rightarrow v_j$. If there is no mapping relation for $t_i \rightarrow v_j$, then t_i is processed with the log event processing described in Algorithm 1 and inputted into the log event semantic vector model, and the corresponding log event semantic vector $v_j{}'$ is obtained and returned. At the same time, the new mapping relationship for $t_i{}' \rightarrow v_j{}'$ is added to the log event semantic dictionary D. If there exists a mapping for $t_i \rightarrow v_j$, then the corresponding log event semantic vector v_j is returned. The specific algorithm of log event semantic vector matching is shown in Algorithm 2.

Algorithm 2: LESVMatch(t_i, T-SBERT)

Input: Log Event t_i, Log event semantic vector model T-SBERT
Output: Log event semantic vector v_j
(1) **IF** k_i **IN** D **THEN**
(2) **RETURN** $D(k_i)$;
(3) **END IF**;
(4) Split k_i into the word list *WL*;
(5) **FOR EACH** *word* **IN** *WL* **DO**
(6) *lowerCase*(*word*);
(7) **IF** *word* is a stop-words or no semantic identifiers **THEN**
(8) Remove *word* from *WL*;
(9) **END IF**
(10) **END FOR**
(11) Add the corresponding *WL* of the processed sentence to the *corpus*;
(12) v_j = *T-SBERT* (*corpus*);
(13) The mapping $\{t_i \rightarrow v_j\}$ is added to the log event semantic dictionary D;
(14) **RETURN** v_j;

5.3. Bi-LSTM Training Algorithm

After the T-SBERT training is completed, the Bi-LSTM also needs to be trained for learning the normal log behavior patterns.

BILSTMADMTrain(S, h): The log event prediction model training algorithm uses the sliding window training pairs generated from the sequence of log event semantic vectors (Definition 5) to train the Bi-LSTM model to obtain the log event prediction model Bi-LSTM-ADM. The initial sliding window length is h. The log event sequence $T_i = \{e_1, e_2, \ldots, e_q\}$ will be converted into the log event semantic vector sequence $S_i = <v_{e_1}, v_{e_2}, \ldots, v_{e_q}>$ by Algorithm 2. Sliding with the size of the sliding window h to construct the training data pair (*TDP*), the sliding window is denoted as $W(S_i, v_{e_j})$. The training data pair *TDP* constructed by $v_{e_{j+1}}$ is denoted as $(w_i, v_{e_{j+1}})$, and the training data pair *TDP* is stored in the list to form the training data pair list (*TDPL*). The Bi-LSTM model is trained with *TDPL* to obtain the log event prediction model Bi-LSTM-ADM, which is then used for log event prediction for further anomaly detection. The specific process of Bi-LSTM training to generate Bi-LSTM-ADM is shown in Algorithm 3.

Algorithm 3: BILSTMADMTrain (S, h)

Input: Sliding window length h, Log event semantic vector sequence set $S = \{<v_{e_1,1}, v_{e_1,2}, \ldots, v_{e_1,q_1}>, <v_{e_2,1}, v_{e_2,2}, \ldots, v_{e_2,q_2}>, \ldots, <v_{e_f,1}, v_{e_f,2}, \ldots, v_{e_f,q_f}>\}$
Output: Log prediction model Bi-LSTM-ADM
(1) Initialize the $TPDL=\emptyset$;
(2) Initialize the Bi-LSTM model;
(3) **FOR** $S_i \in S \bigwedge i \in [1, f]$ **DO**
(4) **FOR** $j=h, h+1, \ldots, q-1$ **DO**
(5) According to Definition 5 to generate the log event semantic vector sliding window $W(S_i, v_{e_j})$;
(6) **IF** $W(S_i, v_{e_j})=\emptyset$ **THEN**
(7) **CONTINUE**;
(8) **END IF**
(9) Generate the $TDP = (w_i, v_{e_{j+1}})$ and add the *TDPL*;
(10) **END FOR**
(11) **END FOR**
(12) Using *TPDL* as training datasets to train Bi-LSTM to generate Bi-LSTM-ADM;
(13) **RETURN** Bi-LSTM-ADM;

5.4. Anomaly Detection Algorithm

In the anomaly detection stage, the log will be detected using the T-SBERT model, log event semantic matching algorithm, and Bi-LSTM-ADM model.

LogADSBERTDetect(S_i, h, ξ, Bi-LSTM-ADM): In the anomaly detection implementation algorithm, for the sequence of log events $T_i = \{e_1, e_2, \ldots, e_q\}$ to be detected, the sliding windows set of log event semantic vectors W_{s_i} is generated using Algorithms 1 and 2. The set consisting of semantic vectors of log events corresponding to W_{s_i} is denoted as $V_{e_{j+1}}$. The prediction set of result vectors obtained from the input of W_{s_i} to the Bi-LSTM-ADM is $R_{e_{j+1}} = \{r_{e_{j+1}} \mid j \in [h,q)\}$. Given the threshold ξ, the sequence anomaly determination method is as follows: for $\forall v_{e_{j+1}} \in V_{e_{j+1}}$ and $r_{e_{j+1}} \in R_{e_{j+1}}$, if the similarity between $v_{e_{j+1}}$ and $r_{e_{j+1}}$ is greater than ξ, then it is determined that there is no anomaly in T_i; otherwise, there is an anomaly in T_i. The specific process of log anomaly detection algorithm implementation is shown in Algorithm 4.

Algorithm 4: LogADSBERTDetect (S_i, h, ξ, Bi-LSTM-ADM)

Input: Sequence of log event semantic vectors $S_i = <v_{e_1}, v_{e_2}, \ldots, v_{e_q}>$, Sliding window size h, Threshold value ξ, Bi-LSTM-ADM model
Output: TRUE-normal/FALSE-abnormal
(1) **FOR** $j = h, h+1, \ldots, q-1$ **DO**
(2) Generate the event semantic vector sliding window $W(S_i, v_{e_j})$ and add the value $v_{e_{j+1}}$ into $V_{e_{j+1}}$;
(3) **IF** $W(S_i, v_{e_j}) = \emptyset$ **THEN**
(4) **CONTINUE**;
(5) **END IF**
(6) Input $W(S_i, v_{e_j})$ into Bi-LSTM-ADM to obtain the prediction vector $r_{e_{j+1}}$;
(7) Add $r_{e_{j+1}}$ to the set of prediction result vectors $R_{e_{j+1}}$;
(8) **IF** $Similarity(v_{e_{j+1}}, r_{e_{j+1}}) < \xi$ **THEN**
(9) **RETURN FALSE**;
(10) **END IF**
(11) **END FOR**
(12) **RETURN TRUE**;

6. Evaluation

In this section, we evaluate the proposed LogADSBERT by conducting experiments on the real log datasets. We implement the LogADSBERT together with the existing log anomaly detection methods based on deep learning, such as DeepLog [19] and LogAnomaly [20].

6.1. Experimental Setting

6.1.1. Evaluation Metrics

The evaluation metrics for this experiment are the false positive, false negative, precision, recall, and F1-Score.

1. False positive: the number of normal log sequences marked as abnormal, which are denoted as FP.
2. False negative: the number of abnormal log sequences marked as normal, which are denoted as FN.
3. Precision: the proportion of log sequences with real anomalies that are correctly marked out; the computation of precision is shown in Equation (1).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{1}$$

4. Recall: the proportion of log sequences with real anomalies that are successfully marked; the computation of recall is shown in Equation (2).

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

5. F1-Score: the reconciliation average of the detection result accuracy and detection result completeness, which is denoted as F1-Score; the calculation of F1-Score is shown in Equation (3).

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{3}$$

6.1.2. Environment and Hyperparameters

The operating system of the experimental equipment is Windows 10 64-bit, the memory size is 32 GB, the CPU is AMD Ryzen 5 3600 4.2 Ghz six cores and twelve threads, and the GPU is Nvidia GTX 1660S. The IDE is PyCharm 2021 with python 3.6. The Sentence-BERT model, Bi-LSTM model, and Self-Attention mechanism were constructed based on the framework of Pytorch 1.4. The experimental comparison method is DeepLog, LogAnomaly.

The experimental parameters were set according to the characteristics of the log data, the structure of the model, and the final experimental results. We tried a variety of different parameter combinations and found that the following parameters can achieve the best detection results. Table 1 shows the specific hyperparameter Settings.

Table 1. Experimental hyperparameters.

Hyperparameters	Value
Learning rate	0.001
Batch size	2048
Epoch	300
l (Neural network layers)	2
α (Hide layer cell size)	64
h (Sliding window size)	10

6.1.3. Experimental Datasets

The log datasets used in this experiment come from Hadoop Distributed File System (HDFS) [34] and OpenStack [35]. The HDFS log dataset comes from more than 200 of Amazon's EC2 nodes and contains 11,175,629 log entries. The OpenStack log datasets platform project contains 1,335,318 log entries. We selected some of the data that have been processed by domain experts for our experiments. The duplicate logs were removed from the log dataset. The log data needed to be further parsed and processed before it could be used for our experiments. We used the open-source log parser LogParser to parse logs. According to relevant research in the field, the unsupervised or semi-supervised learning methods can avoid data imbalance and data noise to a certain extent using normal log data as training data, and they can improve the accuracy and efficiency of detection. Therefore, we chose normal logs as the training data. The specific information of the log sequence is shown in Table 2.

Table 2. Setup of log datasets.

Log Datasets	Number of Sessions			Number of Log Events
	Training Data	Normal	Abnormal	
HDFS	7333	14,296	4251	46
OpenStack	514	4904	425	40

6.2. Result

1. Precision, Recall, and F1-Score

Figure 2 shows the precision, recall, and F1-Score of LogADSBERT on the HDFS dataset. It indicates that LogADSBERT is better than DeepLoog and LogAnomaly in all performance metrics. In the F1-Score, LogADSBERT improves by 7.0% and 4.3% compared to DeepLog and LogAnomaly, respectively. There are improvements in both precision and recall for LogADSBERT. Specifically, LogADSBERT improves 8.8% and 5.1% more than DeepLog in precision and recall, respectively. Moreover, LogADSBERT improves 5.5% and 3.0% more than LogAnomaly in precision and recall, respectively.

Figure 3 illustrates the precision, recall, and F1-score of the three methods on the OpenStack dataset. The performance of LogADSBERT compared to DeepLog and LogAnomaly on the OpenStack dataset is more pronounced than on the HDFS dataset. There is already a more pronounced gap between LogADSBERT and the better-performing method, LogAnomaly, in terms of precision and F1-Score, with a difference of 7.1% and 7.0%, respectively. In addition, LogADSBERT achieves 100% in terms of recall performance, whereas the other methods achieve more than 90%.

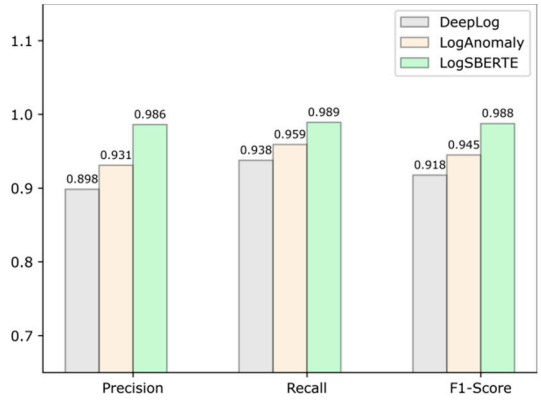

Figure 2. Accuracy on HDFS.

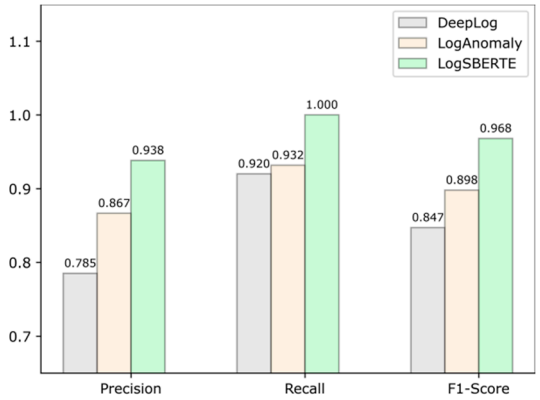

Figure 3. Accuracy on OpenStack.

According to the above analysis, LogADSBERT is superior to DeepLog and LogAnomaly in precision, recall and F1-Score. The reason is that LogADSBERT based on the Sentence-BERT model can capture more important log semantic features, and Bi-LSTM with an attention mechanism can enhance the extraction of the logs' semantic features to improve the accuracy of the anomaly detection.

2. Statistics of FP and FN

Tables 3 and 4 show the number of FP and FN of LogADSBERT, DeepLog, and LogAnomaly on the data set HDFS and OpenStack, respectively.

Table 3. Statistics on the number of FP and FN on the HDFS dataset.

Evaluation Metrics	DeepLog	LogAnomaly	LogADSBERT
False positive (FP)	451	303	59
False negative (FN)	265	174	47

Table 4. Statistics of the number of FP and FN on the OpenStack dataset.

Evaluation Metrics	DeepLog	LogAnomaly	LogADSBERT
False positive (FP)	107	61	28
False negative (FN)	34	29	0

Table 3 shows the number of FP and FN of the three methods on the HDFS dataset. The FP and FN of DeepLog and LogAnomaly are both significantly higher than those of LogADSBERT. Compared to the worst-performing method DeepLog, the FP and FN of

LogADSBERT are reduced by 244 and 127, respectively, which means that LogADSBERT makes an 80.5% and 80.0% improvement in the FP and FN, respectively.

Table 4 shows the number of FP and FN of the three methods on the OpenStack dataset. The result is similar to that shown in Table 3. The number of FP and FN in LogADSBERT is obviously less than that of DeepLog and LogAnomaly. It indicates that LogADSBERT outperforms DeepLog and LogAnomaly in the FP and FN metrics on the OpenStack dataset.

3. Effects of different parameters on LogADSBERT

The experiments on the effect of different parameters on the precision, recall, and F1-Score of LogADSBERT needed to be carried out using a control variable. For simplicity, the more commonly used HDFS dataset was adopted in the experiment. The results of the experiment are shown in Figures 4–7.

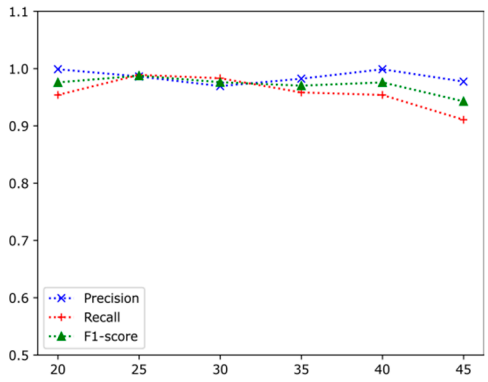

Figure 4. Number of log events: t.

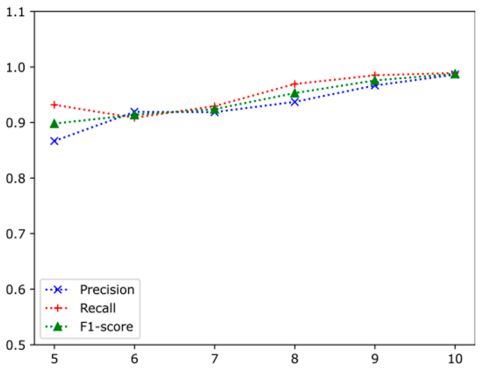

Figure 5. Sliding window size: h.

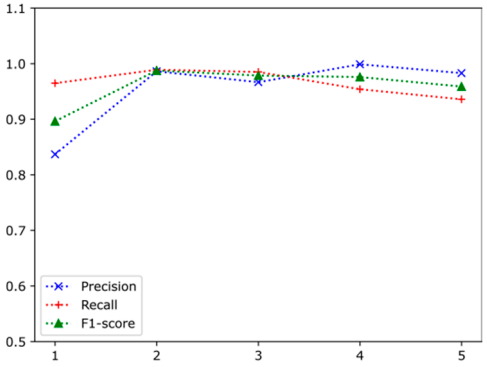

Figure 6. Number of neural network layers: l.

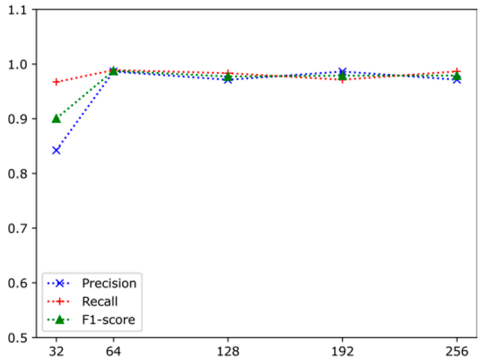

Figure 7. Hide layer unit size: α.

Figure 4 shows the effect of t on the three performance metrics of LogADSBERT. When t = 40, the performance of LogADSBERT is optimal, and when t = 45, the performance of the method decreases, but overall, the effect is not significant. Figure 5 shows the effect of the sliding window size h on the three performance metrics of LogADSBERT, where the accuracy of LogADSBERT is gradually improved as h increases. As shown in Figures 6 and 7, the effects of the number of neural network layers l and the hidden layer unit size α on the LogADSBERT's precision, recall, and F1-Score all reach the highest rate at l = 2 and α = 64. In summary, under the conditions of different hyperparameters of the number of log events t, sliding window size h, the number of neural network layers l, and the size of the hidden layer unit α, LogADSBERT can ensure the stability of the overall performance and obtain a high accuracy, which means that LogADSBERT is robust. In this way, it can cope with the various uncertainties and complex factors that need to be faced in the actual network system application scenario to achieve accurate and stable anomaly detection.

4. Performance comparison of new log event injection

In order to further validate the robustness and effectiveness of LogADSBERT, we conducted experiments involving the addition of new log events on the HDFS dataset. We once again used precision, recall, and F1-Score as the performance metrics, and the comparison methods employed DeepLog and LogAnomaly. The set of log events in the training stage covers the system log datasets and contains 13 log events, and the number of newly added log events was 33. DeepLog does not provide a solution for newly added log events, and here, it was set to mark the log sequence as abnormal when the newly added log events were detected. The results of the experiments are shown in Table 5.

Table 5. Experimental results of new log event injection.

Evaluation Metrics	DeepLog	LogAnomaly	LogADSBERT
Precision	0.409	0.552	0.937
Recall	0.976	0.932	0.928
F1-Score	0.577	0.694	0.932

Table 5 shows that for LogADSBERT, two of the evaluation metrics, precision and F1-Score, were significantly better than for the other two methods. In particular, the F1-Score reached 93.2%, which is 23.8% higher than LogAnomaly. Since LogADSBERT is based on the semantic features of log events for log anomaly detection, the new log events will be matched by a T-SBERT-based log event semantic matching algorithm to obtain the most similar log event semantic representations, so it can vastly reduce the impact of new log events on the anomaly detection results. Additionally, in the experiments, DeepLog was set to detect all log sequences of the new log events as abnormal log sequences, which would certainly lead to a significantly better DeepLog detection rate compared with the other methods, but this setting made the number of FP too high and, consequently, both

the precision and F1-Score were much lower than for the other methods. The solution strategy of LogAnomaly for new log events is to replace the log events by calculating the Euclidean distance with the already determined log events; however, this method does not represent the new log events well, and when the number of new log events is too large, the overall performance decreases rapidly. In summary, LogADSBERT, a log anomaly detection method based on Sentence-BERT, maintains strong robustness in the scenario of adding new log events.

7. Conclusions

In this paper, to solve the existing problems of log anomaly detection methods based on deep learning, we proposed a Sentence-BERT-based log anomaly detection method, LogADSBERT. The proposed anomaly detection model trained by inputting the log event corpus not only extracts the log event information containing semantic features, but also obtains the most relevant log event semantic information based on the log event semantic matching algorithm for the newly added log events. The proposed method shows improved accuracy compared to the existing anomaly detection methods, and it also shows robustness when new log events are added.

With the rapid development of software systems, log anomaly detection needs to be updated and iterated to meet new requests. In the future, the following aspects should be focused on: (1) optimizing the preprocessing of log data to improve the efficiency of anomaly detection; and (2) realizing multimodal log anomaly detection, where log anomaly detection integrates multiple types of log data to conduct joint analysis and processing to improve the accuracy and robustness of anomaly detection.

Author Contributions: Conceptualization, C.H. and H.D.; methodology, C.H. and X.S.; software, C.H. and X.S.; validation, H.D., H.Z. and C.H.; formal analysis, X.S. and H.D.; investigation, H.Z.; resources, X.S. and C.H.; data curation, H.L.; writing—original draft preparation, C.H., H.Z. and X.S.; writing—review and editing, C.H. and H.D.; visualization, X.S.; supervision, H.D. and C.H.; project administration, C.H. and H.D.; funding acquisition, C.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by Jinling Institute of Technology High-level Talent Research Start-up Project (JIT-RCYJ-202102), Key R&D Plan Project of Jiangsu Province (BE2022077), Jinling Institute of Technology Science and Education Integration Project (2022KJRH18), and Jiangsu Province College Student Innovation Training Program Project (202313573080Y, 202313573081Y).

Data Availability Statement: This research employed publicly available datasets for its experimental studies.

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

References

1. Lam, H.; Russell, D.; Tang, D.; Munzner, T. Session viewer: Visual exploratory analysis of web session logs. In Proceedings of the2007 IEEE Symposium on Visual Analytics Science and Technology, Sacramento, CA, USA, 30 October–1 November 2007; IEEE: Piscataway, NJ, USA, 2007; pp. 147–154.
2. Yadav, R.B.; Kumar, P.S.; Dhavale, S.V. A survey on log anomaly detection using deep learning. In Proceedings of the 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), Noida, India, 4–5 June 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 1215–1220.
3. Anastasiou, D.; Ruge, A.; Ion, R.; Segărceanu, S.; Suciu, G.; Pedretti, O.; Gratz, P.; Afkari, H. A machine translation-powered chatbot for public administration. In Proceedings of the 23rd Annual Conference of the European Association for Machine Translation, Ghent, Belgium, 1–3 June 2022; pp. 327–328.
4. Reimers, N.; Gurevych, I. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Hong Kong, China, 3–7 November 2019; pp. 3982–3992.
5. Jang, B.; Kim, M.; Harerimana, G.; Kang, S.U.; Kim, J.W. Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. *Appl. Sci.* **2020**, *10*, 5841. [CrossRef]

6. Roy, S.; König, A.C.; Dvorkin, I.; Kumar, M. Perfaugur: Robust diagnostics for performance anomalies in cloud services. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1167–1178.
7. Prewett, J.E. Analyzing cluster log files using logsurfer. In Proceedings of the 4th Annual Conference on Linux Clusters, St. Petersburg, Russia, 2–4 June 2003; Citeseer: State College, PA, USA, 2003; pp. 1–12.
8. Rouillard, J.P. Real-time Log File Analysis Using the Simple Event Correlator (SEC). *LISA* **2004**, *4*, 133–150.
9. 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; IEEE: Piscataway, NJ, USA, 2007; pp. 583–588.
10. Wang, Y.; Wong, J.; Miner, A. Anomaly intrusion detection using one class SVM. In *Proceedings of the Fifth Annual IEEE SMC Information Assurance Workshop, West Point, NY, USA, 10–11 June 2004*; IEEE: Piscataway, NJ, USA, 2004; pp. 358–364.
11. Breier, J.; Branišová, J. Anomaly detection from log files using data mining techniques. In *Information Science and Applications*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 449–457.
12. 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.* **2017**, *15*, 931–944. [CrossRef]
13. Chen, M.; Zheng, A.X.; Lloyd, J.; Jordan, M.I.; Brewer, E. Failure diagnosis using decision trees. In Proceedings of the International Conference on Autonomic Computing, New York, NY, USA, 17–19 May 2004; IEEE: Piscataway, NJ, USA, 2004; pp. 36–43.
14. Ying, S.; Wang, B.; Wang, L.; Li, Q.; Zhao, Y.; Shang, J.; Huang, H.; Cheng, G.; Yang, Z.; Geng, J. An improved KNN-based efficient log anomaly detection method with automatically labeled samples. *ACM Trans. Knowl. Discov. Data (TKDD)* **2021**, *15*, 1–22. [CrossRef]
15. 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, Big Sky, MT, USA, 11–14 October 2009; pp. 117–132.
16. Xu, D.; Wang, Y.; Meng, Y.; Zhang, Z. An improved data anomaly detection method based on isolation forest. In Proceedings of the 2017 10th International Symposium on Computational Intelligence and Design (ISCID), Hangzhou, China, 9–10 December 2017; IEEE: Piscataway, NJ, USA, 2017; Volume 2, pp. 287–291.
17. Lou, J.G.; Fu, Q.; Yang, S.; Xu, Y.; Li, J. Mining Invariants from Console Logs for System Problem Detection. In Proceedings of the USENIX Annual Technical Conference, Virtual, 14–16 July 2010; pp. 1–14.
18. Vaarandi, R.; Pihelgas, M. Logcluster-a data clustering and pattern mining algorithm for event logs. In Proceedings of the 2015 11th International Conference on Network and Service Management (CNSM), Barcelona, Spain, 9–13 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–7.
19. 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, Dallas, TX, USA, 30 October–3 November 2017; pp. 1285–1298.
20. 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. *IJCAI* **2019**, *19*, 4739–4745.
21. 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, Tempe, AZ, USA, 12 June 2018; pp. 1–8.
22. Chen, S.; Liao, H. Bert-log: Anomaly detection for system logs based on pre-trained language model. *Appl. Artif. Intell.* **2022**, *36*, 2145642. [CrossRef]
23. Zhang, M.; Chen, J.; Liu, J.; Wang, J.; Shi, R.; Sheng, H. LogST: Log semi-supervised anomaly detection based on sentence-BERT. In Proceedings of the 2022 7th International Conference on Signal and Image Processing (ICSIP), Suzhou, China, 20–22 July 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 356–361.
24. Guo, H.; Yuan, S.; Wu, X. Logbert: Log anomaly detection via bert. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–8.
25. Mizutani, M. Incremental mining of system log format. In Proceedings of the 2013 IEEE International Conference on Services Computing, Santa Clara, CA, USA, 28 June–3 July 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 595–602.
26. Shima, K. Length matters: Clustering system log messages using length of words. *arXiv* **2016**, arXiv:1611.03213.
27. Hamooni, H.; Debnath, B.; Xu, J.; Zhang, H.; Jiang, G.; Mueen, A. Logmine: Fast pattern recognition for log analytics. In Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, Indianapolis, IN, USA, 24–28 October 2016; pp. 1573–1582.
28. 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; IEEE: Piscataway, NJ, USA, 2017; pp. 33–40.
29. 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.* **2011**, *24*, 1921–1936. [CrossRef]

30. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention is all you need. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: New York, NY, USA; pp. 6000–6010.
31. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
32. Hoffer, E.; Ailon, N. Deep metric learning using triplet network. In Proceedings of the Similarity-Based Pattern Recognition: Third International Workshop, SIMBAD 2015, Copenhagen, Denmark, 12–14 October 2015; Springer International Publishing: Berlin/Heidelberg, Germany, 2015; pp. 84–92.
33. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
34. Shvachko, K.; Kuang, H.; Radia, S.; Chansler, R. The hadoop distributed file system. In Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), Lake Tahoe, NV, USA, 3–7 May 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1–10.
35. Sefraoui, O.; Aissaoui, M.; Eleuldj, M. OpenStack: Toward an open-source solution for cloud computing. *Int. J. Comput. Appl.* **2012**, *55*, 38–42. [CrossRef]