Anomaly Detection of Unstable Log Data Based on Contrastive Learning

Lan Liu[1], Zhihao Huang[1], Jun Lin[2,3(✉)], Kangjian He[1], and Zhanfa Hui[1]

[1] Guangdong Polytechnic Normal University, Guangzhou 510655, China
hzhnan@qq.com
[2] Sun Yat–Sen University, Guangzhou 510006, China
[3] China Electronic Product Reliability and Environmental Testing Institute, Guangzhou 511370, China

Abstract. In today's large computer systems, it is almost impossible to locate errors using traditional manual methods in the face of abnormal conditions due to the unprecedented size and complexity of the system. As part of computer resources, log data records every step of the system operation process, making log data a popular method for detecting abnormal system status. However, in actual production environments, normal logs make up most of the log data types, and the log structure is constantly adjusted with each update for maintenance and upgrades. Therefore, overcoming the instability of logs caused by these issues has become a major concern for researchers in the field of log detection. In this paper, we propose a method to improve log instability and enhance model detection accuracy by using contrastive learning in the log vectorization and detection phases. Contrastive learning is used to train the vectorization and diagnostic models by aggregating and distinguishing classes in mathematical space, resulting in a more robust vectorization model and better generalization of the generated template vector data. This also improves the feature extraction ability of the diagnostic model and improves the final anomaly classification results. Experimental results show that our method improves the handling of unstable log data in anomaly detection and outperforms the baseline on HDFS and BGL datasets in terms of experimental performance.

Keywords: Anomaly Detection · Log Embedding · Unstable Log Data · Contrastive Learning · Deep Learning

1 Introduction

The way of diagnosing systems based on log data has evolved with the development of deep learning techniques in recent years. Before AlexNet [14] technology was proposed, using machine learning to analyze logs was almost the default approach in the academic community. In this context, approaches such as using PCA [30], SVM [16,27], clustering [18], rule-based system detection [2,8,23,32], and pattern recognition-based analysis methodss [19,29] were developed, amongothers. As deep learning gained popularity in industry and academia

J. Cai et al. (Eds.): SPNCE 2023, LNICST 525, pp. 147–160, 2025.
https://doi.org/10.1007/978-3-031-73699-5_10

due to its powerful feature extraction capabilities in space and time, more researchers began to explore using log data in a deep learning approach for diagnosing system states. Logs are generated sequentially over time and thus exhibit a high degree of correlation with each other in the temporal dimension. This makes techniques like RNN [4,21,22,33] and BERT [7], which are powerful at capturing features in the temporal space, well-suited for uncovering deep temporal features between logs and log sequences to determine the current system state. These techniques have achieved experimental results that far surpass those of machine learning.

Although log data is a valuable resource for recording system states, in practice the model's performance can be impacted by the instability of logs. The instability of log data manifests in two main ways: (1) Since the normal state of the system is much more common than the abnormal state in real-world environments, a small amount of abnormal data is often hidden among a large amount of normal data, resulting in an imbalance between categories. (2) Due to system maintenance and upgrades, the log template format changes, which can confuse the running detection model with unfamiliar log templates. Despite extensive research and progress in the field of log detection, the problem of log instability remains an important factor limiting the performance of log detection.

In recent years, pre-trained language models and contrastive learning have achieved good results in computer vision (CV) and natural language processing (NLP) respectively. In many well-known pre-trained models, researchers have adopted the transformer architecture [26], which allows for simultaneous observation of moments on either side of a moment in time. The BERT model, designed using the transformer architecture, has a larger number of parameters than other neural networks, allowing for a richer learning process during pre-training. Sentence-BERT [24], proposed by Nils Reimers and Iryna Gurevych, aims to train a generalized linguistic embedding model, using the model to vectorize character sentences in the form of vectors. This approach has achieved good results in terms of speed and accuracy. The core idea of the contrastive loss function, proposed in the context of contrastive learning, is to spatially partition different samples such that samples of the same kind are aggregated and samples of different categories are separated in the embedding space. This makes contrastive learning well-suited for classification tasks. With the efforts of academic researchers, contrastive learning-based techniques have achieved excellent results on both supervised [1,13] and unsupervised tasks [5,6,31].

We expect system state detection models to have high detection accuracy, be able to handle unstable log data, and exhibit high robustness. Building on the previously proposed problem, this work proposes the following two contributions as a solution to the problem.

(1) To address the problem of accuracy degradation caused by the model's difficulty in dealing with unknown template types introduced by the development and maintenance process, we propose a Bert-based log template embedding model. We obtain positive example samples in contrastive learning by two random dropouts, and finally update the model parameters in an unsuper-

vised way to obtain semantically more abstract and more generalized vector data.

(2) In the log data detection phase, we propose a novel detection model that learns the system state information implicit in the log data and infers the current system state information accordingly. The model uses LSTM and CNN modules in parallel to capture temporal and attentional information, respectively, and the results of both modules are early fused before input to the MLP projection.

The rest of this paper is organized as follows: Sect. 2 presents the current state of research on system detection models based on log data and summarizes related methods for handling unstable logs. Section 3 describes our proposed method in detail. Section 4 compares and validates the related algorithms, and analyzes and discusses the results. Section 5 is the conclusion section, which summarizes the research methods proposed in this paper and discusses potential future research directions.

2 Related Work

In this section, we summarize the algorithms and research work related to this study.

2.1 Log Anomaly Detection

Over the past few years, the popularity of deep learning has allowed it to start replacing traditional statistical and rule-based machine learning in log detection. Most of these methods use neural networks to learn feature information in log sequences to determine the current state of the system and have achieved good prediction results in practice. Lu et al. [20] used convolutional neural networks (CNNs) to extract feature information from log data by convolution, which improved the accuracy of prediction results. The DeepLog model proposed by Du et al. [4] used the LSTM model, which is better at capturing timing information, compared to CNNs, and split the log information into two parts, log keys and log variables, to be modeled separately, which effectively improved the effectiveness of the detection model compared to other published methods. Meng et al. [22] proposed a novel log vectorization approach to address the shortcomings of previous studies on log preprocessing, which more fully exploits the semantic information of log data, and the model shows stronger robustness in processing than previous approaches when facing unfamiliar logs. Wang et al. [28] proposed a multiscale detection model that uses CNNs for feature extraction of both global and local information and combines features at multiple scales to complement the expressiveness of feature information. Guo et al. proposed a novel training approach based on the idea of DeepLog by taking advantage of the transformer's ability to simultaneously take into account information from each time period, which improved the detection accuracy of the model. [7]

2.2 Unstable Log Data

In practice, logs evolve as the system is maintained and updated, so they are not static, and we can assume that the vast majority of environments have far more normal states than abnormal states, which creates an imbalance in log data in terms of categories. Although many works have achieved good results in experiments, this is often based on a closed-world assumption, which is often affected by unstable log data in real environments. Therefore, Zhang et al.'s LogRobust [33] proposes a novel vectorization technique that effectively mitigates the negative effects of sentence structure and character transformations by mapping log data onto the data space. The vectorization scheme is further refined in the LogAnomaly [22] method proposed by Meng et al. by introducing "template2vec", which takes into account the synonyms and antonyms of the characters in the log, making the generated vectors more robust, and also uses the word Term Frequency-Inverse Document Frequency (TF-IDF) method to assign weights to different characters to balance their influence on the final results, ultimately achieving higher accuracy. In terms of both accuracy and robustness, the final results are excellent. In solving the class imbalance problem of log data in a practical environment, Shayan et al. proposed a detection model based on a Siamese network, which is effective for similarity learning and can learn from both anomalous and non-anomalous data without relying on balanced training data, achieving good results in dealing with the class imbalance of log data [9].

3 Method

In this section, we propose a Bert-based Embedding model for log data and a novel detection model to improve the problem of unstable log data in log detection systems.

3.1 Data Pre-processing

Raw log data is typically unstructured. In order to convert log data into a form that can be easily processed by the model, we use the Drain method [10] to pre-process log data and convert unstructured text into structured text. Drain uses a fixed-depth parse tree that encodes parsing rules specifically designed to guide the parsing of log data. It is worth noting that although Drain's performance is excellent and demonstrates its capability in log parsing, some noise will inevitably still be introduced into the parsing results during the parsing process. However, our approach is sufficient to handle this problem.

Drain is an online log parsing method based on a fixed-depth tree. When a new raw log message arrives, Drain preprocesses it using simple regular expressions based on domain knowledge. Drain then searches for log groups (i.e., leaf nodes of the tree) according to special design rules encoded in the nodes inside the tree. If a suitable log group is found, the log message is matched to a log event stored in that log group. Otherwise, a new log group is created based on the log message.

3.2 BERT-Based Embedding Model

This method of vectorizing log data is based on the BERT model [3], which can grasp the semantics globally compared to Word2ver's vectorization method.BERT benefits from the Transformer architecture, which can take into account the character information at each point in time when processing sentence information, making the quantized sentences characterize stronger abstract information. It has stronger robustness and generalization ability when facing unstable log data.

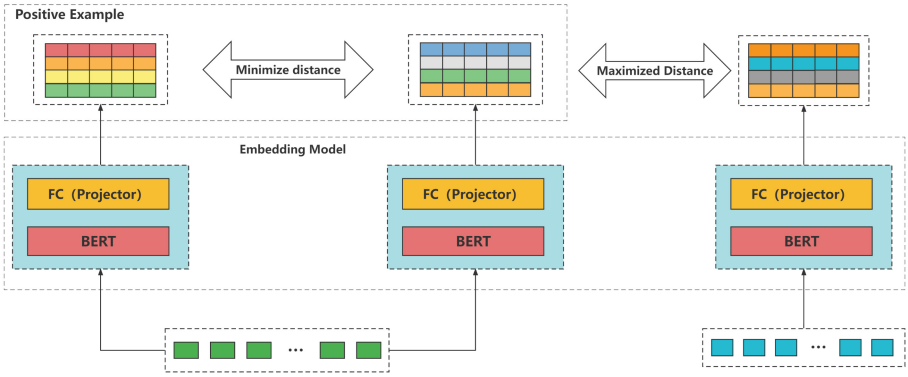

Fig. 1. Embedding model training framework

To train the model, we use an unsupervised contrast learning method to fine-tune a pre-trained BERT model. A fully connected layer is attached to the output of the BERT model for vector projection. The structure and training process of the model are shown in Fig. 1. The log data is first input to the BERT model, and the first CLS sentence vector is taken as the vector representation of the global data. The data is then input to the fully connected layer to obtain the projection of the data on the vector space. By using two random dropouts, different positive examples of the same example are obtained. In the loss training, unsupervised contrastive learning is used to calculate the loss of the results, to minimize the distance between positive examples of samples in the vector space, while maximizing the distance between negative samples. This guides the model to update its parameters according to the loss results, so that the vector generated by the final model has a stronger degree of aggregation within the category and a more obvious distinction between different categories, resulting in stronger generalization in the results.

3.3 Detection Model Based on Log Data

For log data, it is difficult to manually augment the data while ensuring semantic invariance. In SimCSE [5] and R-Drop [17], we use a method that doesn't involve

manual processing, but rather leverages the properties of neural networks to perform data augmentation. This method uses the properties of randomly deactivated neurons in Dropout to augment a piece of data, and it produces good results. Therefore, we also use a simple "Dropout twice" method to construct a supervised contrastive learning method for positive samples. The samples are passed through the feature extraction model twice to obtain different feature vectors for the same input.

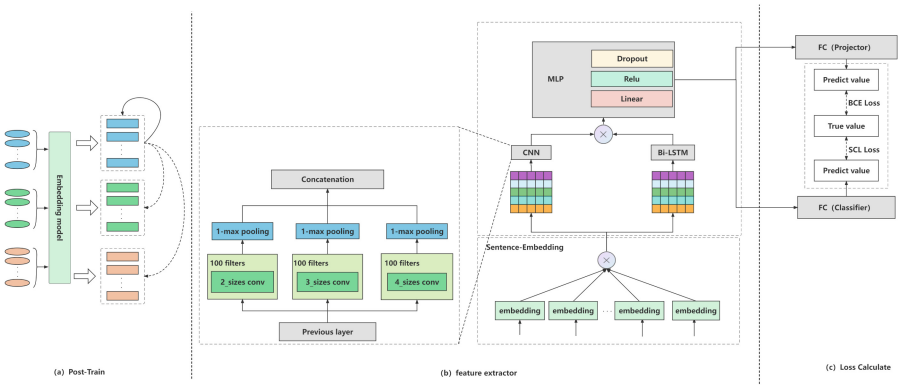

Fig. 2. Detection model structure diagram

In this section, we propose a novel model for log data detection. The architecture of the model is shown in Fig. 2, and the whole model consists of three major parts, which are log template embedding, feature extraction of vector information, and loss calculation during the training update stage. The information on log template vectorization can be know in detail from Sect. 3.2. For the feature extraction of vector information, we use Long Short Term Memory Network (LSTM) [12] and Convolutional Neural Network (CNN) [15] in parallel, LSTM has the advantage of capturing the temporal sequence of long sequence information, which can uncover the implicit semantic expressions hidden between the sequence information, while CNN model has the advantage of giving different attentions to uncover the key information, and the output of these two models is The outputs of these two models are ealry fused and the fused results are fed into a Multilayer Perceptron (MLP) to obtain two semantically consistent vectors with different expressions using the randomness of dropout in the MLP. The role of the projection network is to map the feature-extracted vectors into final vectors for loss computation. In this paper, the projection network uses a 256-dimensional fully-connected layer, uses a unit hypersphere for vectorization. And the classification network will map the dimensionality to the number of label categories through a fully connected layer.The final model parameters will be updated based on the Supervised Contrastive Learning(SCL) loss values and Binary Cross Entropy(BCE) loss values on the projection and classification layers.

Algorithm 1: Anomaly detection model training

input : A batch T, which contains the log sequence x_i and its corresponding label y_i, uses $e(\cdot)$ to represent the function of the embedding layer, $f(\cdot)$ to represent the function of the projection calculation, and $g(\cdot)$ to represent the function of the classification calculation

output: Updated anomaly detection model

foreach $(x_i, y_i) \in T$ **do**
  Vectorize the log sequence $E(x_i) = e_i$;
  Input e_i twice to the feature extraction model and to the projection and classification layers, respectively, using $f^*(\cdot)$ and $g^*(\cdot)$ to represent the computational process when the sample is input to the model a second time;
  Data is output from the projection layer:$f(e_i) = f_i$,$f^*(e_i) = f_i^*$;
  Data is output from the classification layer:$g(e_i) = g_i$,$g^*(e_i) = g_i^*$;
  if $y_i == 1$ **then**
    $\ell_{BCE} = (f_i, 1)\, and\, (f_i^*, 1)$;
    $\ell_{SCL} = (g_i, g_i^*, g_{j \neq i}, 1)$;
    $\ell = \lambda \ell_{BCE} + (1 - \lambda)\, \ell_{SCL}$;
  end
  if $y_i == 0$ **then**
    $\ell_{BCE} = (f_i, 0)\, and\, (f_i^*, 0)$;
    $\ell_{SCL} = (g_i, g_i^*, g_{j \neq i}, 0)$;
    $\ell = \lambda \ell_{BCE} + (1 - \lambda)\, \ell_{SCL}$;
  end
  Update anomaly detection models using loss functions;
end

3.4 Loss Function

For the training of the embedding model, we use an Unsupervised Contrastive Learning approach. We represent the log sequences as $\{x_i\}_i^m$.Simultaneous use x_i and x_i^+ to represent different samples between the same positive cases.Think of z as the random mask used by the embedding model in random dropot.We treat equation $h_i^z = f_\theta(x_i, z)$ as the vector result obtained by the model when processing the log sequence x_i and the random mask z.For the same sequence, we input it into the model twice and use different random masks to get two different vectorization results. In a batch of m log sequences, the loss function will be expressed as the following eq. (1),$\tau > 0$ is an adjustable scalar temperature parameter that controls the separation of classes;

$$\ell_{CL} = -\log \frac{e^{\mathrm{sim}\left(\mathbf{h}_i^{z_i}, \mathbf{h}_i^{z_i'}\right)/\tau}}{\sum_{j=1}^{m} e^{\mathrm{sim}\left(\mathbf{h}_i^{z_i}, \mathbf{h}_j^{z_j'}\right)/\tau}} \tag{1}$$

The BCE loss function in Eq. (2) calculates the cross-entropy loss for each sample in the batch. The cross-entropy loss is commonly used for binary classification tasks, where the goal is to predict the probability that a sample belongs

to the positive class. In this case, the BCE loss function is applied to both the original sample x_i and its corresponding positive example x_i^+.

The SCL loss function in Eq. (3) uses supervised contrastive learning to calculate the loss for each sample in the batch. Supervised contrastive learning is a technique used to improve the representation learning of a model by contrasting the features of similar and dissimilar samples. In this case, the SCL loss function compares the features of each sample with those of every other sample in the batch, and applies a log loss to the contrastive terms.

The total loss is the weighted average of the BCE and SCL loss functions, where the weight of each loss is determined by the hyperparameter λ. By setting λ to different values, it is possible to adjust the relative importance of the BCE and SCL loss functions in the final loss calculation.

$$\ell_{BCE} = \frac{1}{2N}\sum_{i} -\Big[y_i \cdot \log\left(p_i\right) + \left(1-y_i\right)\cdot \log\left(1-p_i\right) + y_i^+ \cdot \log\left(p_i\right) + \left(1-y_i^+\right)\cdot \log\left(1-p_i\right)\Big] \tag{2}$$

$$\ell_{SCL} = \sum_{i=1}^{2N} -\left[\frac{1}{2N_{y_i}-1}\sum_{j=1}^{2N}\mathbf{1}_{i\neq j}\mathbf{1}_{y_i=y_j}\cdot \log\frac{\exp\left(f\left(x_i\right)\cdot f\left(x_j\right)/\tau\right)}{\sum_{k=1}^{2N}\mathbf{1}_{i\neq k}\exp\left(f\left(x_i\right)\cdot f\left(x_k\right)/\tau\right)}\right] \tag{3}$$

$$\ell = (1-\lambda)\mathcal{L}_{BCE} + \lambda\mathcal{L}_{SCL} \tag{4}$$

4 Experiments

4.1 Benchmark Dataset

In this thesis, we use the log data provided by the open-source loghub project [11] for our experiments. Loghub is a well-known dataset for system anomaly detection based on log data, and has been used by many seminal and groundbreaking works in the field.

Loghub provides 17 real-world log datasets collected from a variety of systems, including distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software. In our experiments, we use two of these datasets, HDFS and BGL, as the data for our experiments.

The BGL dataset is generated by the Blue Gene/L supercomputer and consists of 4,747,963 logs, each of which has been manually labeled as normal or abnormal. The HDFS dataset contains 11,175,629 logs collected from over 200 Amazon EC2 nodes. The dataset includes 575,061 log blocks, of which 16,838 blocks have been labeled as abnormal by Hadoop domain experts.

In this experiment, we use Drain [10] to parse the logs. Drain is a fast and accurate parser that can parse logs in a streaming and timely manner. To improve

Table 1. Details of the datasets.

Datasets	Description	#Durain	#Messages
HDFS	Hadoop distributed file system	38.7 h	11,175,629
BGL	BlueGene/L Supercomputer log	7 months	4,747,963

its performance, Drain uses a fixed-depth parse tree with specially designed parsing rules. This allows it to achieve both high accuracy and fast parsing speeds, making it a superior choice compared to other parsers.

Table 2. Details of the processed datasets.

Datasets	#Log Template	#Log Sequence	#Sequence Anomaly
HDFS	48	575061	16838
BGL	1000	37315	3018

For the HDFS logs, we vectorized all of the templates and used them to build our diagnostic system. However, to test the robustness of the system, we only used 100 of the 1000 available templates for BGL logs. The remaining 900 templates were vectorized using a vectorization model, and their similarity to the selected 100 templates was measured using cosine similarity. The categories of the unfamiliar templates were then matched to the most similar ones among the selected templates. This allowed us to evaluate the performance of the diagnostic system on both familiar and unfamiliar templates.

4.2 Evaluation Criteria

We use Prediction, Recall, and F1-measure to evaluate the performance of the experimental model. These evaluation criteria reflect the performance of the model for detecting the system state.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{6}$$

$$F1 \text{ measure} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{7}$$

4.3 Implementation Details

We used 80% of the dataset for training our model and 20% for testing the results. We trained the model using the Adam optimizer with a learning rate of 1e-2, a uniform batch size of 1024, and a dropout of 0.3 for a total of 500 epochs. We selected Deeplog [4], LogRobust [33], CNNLog [20], LogBERT [7], LogCluster [25], and LogAnomaly [22] as the baseline models for comparison.

4.4 Evaluation of The Overall Performance

In this section, we compare our method with the baseline method on both the HDFS and BGL datasets. Additionally, in our proposed method, we compare the use of the Contrastive loss with versions using only the BCE loss and a mixed loss of BCE and SCL to verify the effectiveness of the Contrastive loss.

Table 3. Independent test performance metrics of different methods on the dataset (Precision, Recll, and F1-measure)

Datasets	Method	Precision	Recall	F1-measure
HDFS	LogCluster	0.87	0.74	0.80
	LogBERT	0.87	0.78	0.82
	Deeplog	0.95	0.96	0.96
	LogRobust	0.98	1.00	0.99
	LogAnomaly	0.96	0.94	0.95
	CNNLog	0.97	0.99	0.98
	Ours(BCE)	**0.99**	0.96	0.98
	Ours(SCL+BCE)	**0.99**	0.99	**0.99**
BGL	LogCluster	0.42	0.87	0.57
	LogBERT	0.89	0.92	0.90
	Deeplog	0.90	0.96	0.93
	LogAnomaly	0.97	0.94	0.96
	Ours(BCE)	**0.97**	0.93	0.94
	Ours(SCL+BCE)	**0.97**	0.94	**0.96**

Performance on Log Anomaly Detection. In Table 3, we can see that on the HDFS dataset, our method using SCL+BCE achieves the best F1 score, while using only BCE yields the best precision. SCL+BCE performs slightly worse than BCE in terms of recall and F1. On the BGL dataset, our method using SCL+BCE achieves the best F1 and precision scores, while the method using only BCE performs slightly worse in terms of both recall and F1. It should be noted that for the BGL template data, we only used one-tenth of it for training and prediction. Despite this, our model still achieved the best results, demonstrating its robustness in low-quality data environments.

4.5 Visualization Contribution of Contrastive Learning to Category Clustering

Figures 3, 4, and 5 show images of the BGL data projected by the projection layer and then reduced in dimension using PCA. Figure 3 shows the results

of training our model using only the BCE method, while Figs. 4 and 5 show the results of training our model using the BCE+SCL method with different temperature hyperparameters. In Fig. 4, τ was set to 0.09, and in Fig. 5, τ was set to 0.2.

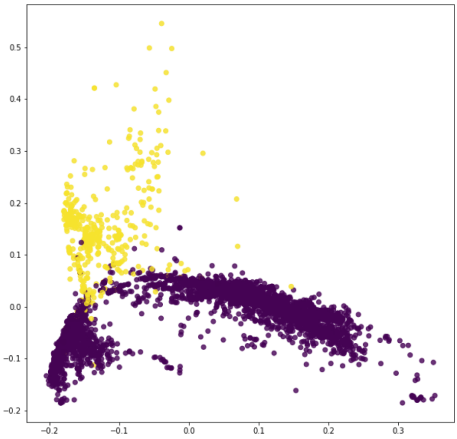

Fig. 3. Vector projection map using only the BCE loss function

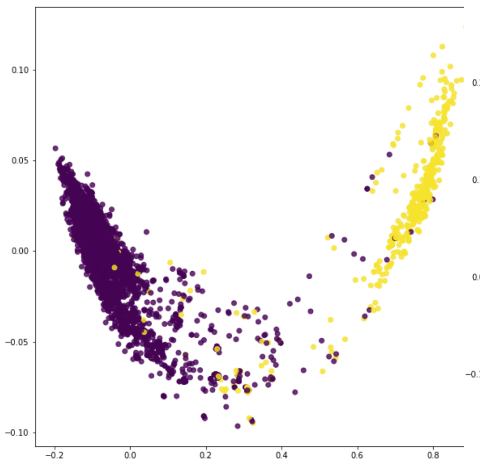

Fig. 4. τ is set to 0.09

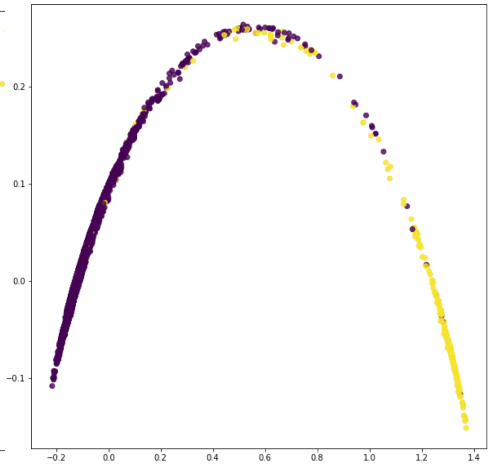

Fig. 5. τ is set to 0.2

Projection Results. It can be seen that when using only the BCE method, the normal and abnormal data are less tightly clustered in spatial distribution,

and similar data are more severely separated from each other. In contrast, when using the contrastive loss function, the data is more densely clustered within the same category and more clearly separated between different categories. In Figs. 4 and 5, where τ is set to 0.09 and 0.2, respectively, it can be seen that using a larger temperature hyperparameter results in better clustering and separation of the data.

5 Conclusion and Future Work

In this paper, we propose a new approach to address the common challenges of log template updates and data type imbalance in industrial environments. Our approach uses a log template embedding model and a novel log detection model based on contrastive learning techniques. Through experiments on HDFS and BGL datasets, we show that our approach yields small improvements over the baseline, while being robust to poor data quality. We attribute the success of our approach to the use of contrastive learning and data augmentation through simple dropout. The embedding model produces data with stronger generalization, while the detection model benefits from the aggregation of data between categories and the separation of data between different categories, resulting in improved detection accuracy.

In future work, we plan to explore the potential of extending our model from supervised to self-supervised and unsupervised settings. Another area of interest for us is to investigate how to train detection models on small or unlabeled data, and reduce the need for manual data labeling. To this end, we propose to study the feature extraction of data and discuss how to improve its representation in the context of category imbalance, with the goal of better distinguishing between data categories and achieving more accurate judgments of system status.

6 Conflict of Interest Statement

The authors have no financial or personal relationships that could have influenced the research. This study did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Acknowledgments. This work was supported in part by Scientific Research Projects of Guangdong Provincial Education Department (No. 2022ZDZX1015).

References

1. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International Conference on Machine Learning, pp. 1597–1607. PMLR (2020)
2. Cinque, M., Cotroneo, D., Pecchia, A.: Event logs for the analysis of software failures: a rule-based approach. IEEE Trans. Software Eng. **39**(6), 806–821 (2012)

3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: pre-training of deep bidirectional transformers for language understanding. ArXiv preprint arxiv: abs/1810.04805 (2019)
4. Du, M., Li, F., Zheng, G., Srikumar, V.: DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298 (2017)
5. Gao, T., Yao, X., Chen, D.: SimCSE: simple contrastive learning of sentence embeddings. arXiv preprint arXiv:2104.08821 (2021)
6. Giorgi, J., Nitski, O., Wang, B., Bader, G.: DeCLUTR: deep contrastive learning for unsupervised textual representations. arXiv preprint arXiv:2006.03659 (2020)
7. Guo, H., Yuan, S., Wu, X.: LogBERT: log anomaly detection via BERT. In: 2021 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2021)
8. Hansen, S.E., Atkins, E.T.: Automated system monitoring and notification with swatch. In: LISA, vol. 93, pp. 145–152. Monterey, CA (1993)
9. Hashemi, S., Mäntylä, M.: Detecting anomalies in software execution logs with siamese network. ArXiv preprint arxiv: abs/2102.01452 (2021)
10. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 33–40 (2017)
11. He, S., Zhu, J., He, P., Lyu, M.R.: Loghub: a large collection of system log datasets towards automated log analytics. ArXiv preprint arxiv: abs/2008.06448 (2020)
12. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**, 1735–1780 (1997)
13. Khosla, P., et al.: Supervised contrastive learning. Adv. Neural. Inf. Process. Syst. **33**, 18661–18673 (2020)
14. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. Commun. ACM **60**, 84–90 (2012)
15. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proc. IEEE **86**, 2278–2324 (1998)
16. Li, K.L., Huang, H.K., Tian, S.F., Xu, W.: Improving one-class SVM for anomaly detection. In: Proceedings of the 2003 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 03EX693), vol. 5, pp. 3077–3081. IEEE (2003)
17. Liang, X., et al.: R-drop: regularized dropout for neural networks. In: NeurIPS (2021)
18. Lin, Q., Zhang, H., Lou, J.G., Zhang, Y., Chen, X.: Log clustering based problem identification for online service systems. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 102–111 (2016)
19. Lou, J.G., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: 2010 USENIX Annual Technical Conference (USENIX ATC 10) (2010)
20. Lu, S., Wei, X., Li, Y., Wang, L.: Detecting anomaly in big data system logs using convolutional neural network. In: 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), pp. 151–158 (2018)
21. Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., Shroff, G.: LSTM-based encoder-decoder for multi-sensor anomaly detection. arXiv preprint arXiv:1607.00148 (2016)
22. Meng, W., et al.: Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, vol. 19, pp. 4739–4745 (2019)

23. Prewett, J.E.: Analyzing cluster log files using logsurfer. In: Proceedings of the 4th Annual Conference on Linux Clusters. Citeseer (2003)
24. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using siamese BERT-networks. arXiv preprint arXiv:1908.10084 (2019)
25. Vaarandi, R., Pihelgas, M.: Logcluster - a data clustering and pattern mining algorithm for event logs. In: 2015 11th International Conference on Network and Service Management (CNSM), pp. 1–7 (2015). https://doi.org/10.1109/CNSM.2015.7367331
26. Vaswani, A., et al.: Attention is all you need. ArXiv preprint arxiv: abs/1706.03762 (2017)
27. Wang, Y., Wong, J., Miner, A.: Anomaly intrusion detection using one class SVM. In: Proceedings from the Fifth Annual IEEE SMC Information Assurance Workshop, 2004, pp. 358–364. IEEE (2004)
28. Wang, Z., Chen, Z., Ni, J., Liu, H., Chen, H., Tang, J.: Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining (2021)
29. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.: Online system problem detection by mining patterns of console logs. In: 2009 ninth IEEE International Conference on Data Mining, pp. 588–597. IEEE (2009)
30. 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, pp. 117–132 (2009)
31. Yan, Y., Li, R., Wang, S., Zhang, F., Wu, W., Xu, W.: ConSERT: a contrastive framework for self-supervised sentence representation transfer. arXiv preprint arXiv:2105.11741 (2021)
32. Yen, T.F., et al.: Beehive: large-scale log analysis for detecting suspicious activity in enterprise networks. In: Proceedings of the 29th Annual Computer Security Applications Conference, pp. 199–208 (2013)
33. Zhang, X., 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, pp. 807–817 (2019)