Human Interaction and Emerging Technologies (IHIET 2024), Vol. 157, 2024, 207–214
https://doi.org/10.54941/ahfe1005479

Survey of Research Issues and Proposed Solutions for Detecting Parameter Anomalies in System Logs

Hironori Uchida[1], Keitaro Tominaga[2], Hideki Itai[2], Yujle Li[1], and Yoshihisa Nakatoh[1]

[1]Kyushu Institute of Technology, 1–1 Sensuicho, Tobata-ku, Kitakyushu-shi, Fukuoka Prefecture, Japan
[2]Panasonic System Design Co., Ltd., 3-1-9, Shinyokohama, Kohoku-ku, Yokohama-shi, Kanagawa Prefecture, Japan

ABSTRACT

In the ever-evolving field of software development, the demand for automation of fault analysis that is time-consuming and expertise-requiring is growing. One solution to this challenge is the study of anomaly detection using text logs, which has seen numerous research efforts. However, despite the variety of patterns that system anomalies can exhibit, many studies have predominantly focused on sequence anomalies. This is largely attributed to the limited availability of datasets, with the commonly used Loghub data being oriented towards sequence anomalies. This research addresses the current challenges in anomaly detection models and proposes several new methods for detecting parameter anomalies. Initially, due to the lack of datasets of parameter anomalies, we prepared common parameter anomaly scenarios and compared them with existing sequence anomaly detection models (including DNN models for sequence anomalies and DNN models using semantic information), and with a variety of proposed methods. The prepared parameter anomaly patterns include four Integer types and three String types. For instance, a parameter within a certain range (-100 to 100) is considered normal, while parameters outside this range are deemed anomalies. Our proposed method begins by extracting parameters using LogParser and determining whether they are of Int or String type. For Int types, we use Z-Score, IQR, K-NN and DBSCAN for evaluation, while for String types, we use a Bert-based positive-negative classifier. The experimental results showed that the DNN model for sequence anomaly had an F1 Score of less than 0.5 for all patterns. In contrast, our proposed methods achieved F1 Scores exceeding 0.9 or 0.8 for almost all methods, except for one anomaly pattern. It was found that the proposed methods are effective for common parameter anomaly problems. Furthermore, since our methods do not require prior training, they are particularly advantageous for ad-hoc learning in the context of continuously updated software development.

Keywords: Anomaly detection, Software log, Log analysis, Deep learning, Log generator, Parameter anomaly detection

INTRODUCTION

In the constantly evolving field of software development, there is a pressing need for the automation of complex and time-consuming fault analysis tasks that require specialized knowledge. As one solution, anomaly detection using textual system logs has been extensively studied. However, despite the variety of anomaly patterns that can occur within systems, much of the research has focused on sequence anomalies. A significant reason for this focus is the limited number of dataset types available, with commonly used datasets like Loghub (He et al., 2020) being tailored for sequence anomalies. The issue of limited dataset variety has been reported in several studies (Le et al., 2022; Shilin et al., 2022, Uchida et al., 2023). Moreover, current log anomaly detection models face the issue of losing parameter information during preprocessing with LogParsers like Drain (He et al., 2017) and Brain (Yu et al., 2023), which template the logs. Consequently, some studies have explored the use of parameter sequence vectors in addition to sequence vectors (Zhu et al., 2020) and research utilizing sentence information without LogParsers (Le et al., 2021). While these studies focused on parameters, the datasets used were designed for sequence vectors and lacked specific anomaly labels for each parameter. In response to these challenges, this research aims to automatically generate datasets with typical parameter anomalies to create an environment suitable for evaluating parameter anomaly detection. Using these datasets, we will examine the current limitations of anomaly detection models and propose several new methods for detecting parameter anomalies. The parameter anomaly patterns prepared include four Integer types and three String types, such as considering parameters within a certain range (-100 to 100) as normal and those outside this range as anomalies.

■**Log Format**

<Label> <Timestamp> <Date> <Node> <Time> <NodeRepeat> <Type> <Component> <Level> <Content>

■**Explanation of How to Create Log Formats**

Format	How to Create	Format	How to Create
Label	"-" or "INTERR" or "STRERR"	NodeRepeat	Constant Value: "R02-M1-N0-C:J12-U11"
Timestamp	Constant Value: "1111111111"	Type	Constant Value: "RAS"
Date	Constant Value: Date Created	Component	Constant Value: "KERNEL"
Node	Constant Value: "R02-M1-N0-C:J12-U11"	Level	Constant Value: "INFO"
Time	16 lines of logging per 0.5 second	Content	Automatic Generation

■**Example of Generated Logs (only the variable portion is excerpted)**

- 1111111111 2024-03-10-11.46.46.050249 7wBfcbp2 r7fzP bknkj OKsGrT
- 1111111111 2024-03-10-11.46.46.050249 pAxmWZ MKV7wpz VlBQuR jlT7NC 2UrYobZZaP eqv60i
- 1111111111 2024-03-10-11.46.46.050249 c9JdJrsoS y5xiwbE AOuo4 KeMwOHtL Oe2fHLR HmGYI
- 1111111111 2024-03-10-11.46.46.050249 7wBfcbp2 r7fzP bknkj OKsGrT
- 1111111111 2024-03-10-11.46.46.050249 y7mM eYgD9d0X cv5RRwva FRvI5V UmU2rxY
INTERR 1111111111 2024-03-10-11.46.46.050249 jPShEtW E3bPwaKYw OukP9NlR RXcupA oEnf -8159
- 1111111111 2024-03-10-11.46.50.050249 jPShEtW E3bPwaKYw OukP9NlR RXcupA oEnf -36
- 1111111111 2024-03-10-11.47.03.050249 jPShEtW E3bPwaKYw OukP9NlR RXcupA oEnf -53

Figure 1: Description of the log format used for log generation and some examples.

This research seeks to clarify the following:

RQ1: Accuracy of current anomaly detection models against typical parameter anomalies.

RQ2: Proposals for new methods tailored to parameter anomaly detection.

DATASET FOR PARAMETER ANOMALY DETECTION

Due to the absence of readily available datasets for parameter anomaly detection, we created a custom algorithm for automatic generation. Please refer to Github for specific implementations https://github.com/hiro877/LogGenerator/tree/ihiet2024.

Table 1. Parameters used to generate logs.

Parameter	Description
Total number of logs	40000+1200(parameter logs)
Number of words of content	Range: 3∼6
Number of words in content	Range: 4∼10
Number of logs per second	15
Number of log types	16+1(parameter log type)
Histogram of logs	[14483, 1742, 1738, 1736, 1727, 1726, 1726, 1721, 1702, 1700, 1687, 1686, 1682, 1660, 1644, 1640]
Number of parameter logs	1200: Approx. 3% of total number of logs
Number of parameter logs	240: Approx. 20% of parameter logs

Automatic Log Generation Algorithm

Figure 1 illustrates the log format used for automatic generation and provides examples of the generated logs. The log format is based on the BGL (Blue Gene/L supercomputer log) L from Loghub. Considering that only the content part of the logs is important for a dataset aimed at parameter anomaly detection, this log generator was designed to create logs with variables for labels and content parts.

Next, we introduce the log generation system. Table 1 presents the key variable parameters used within the log generation system. These parameters were selected based on research that investigated differences in complexity between research datasets and actual logs in development environments (Uchida et al., 2024). The parameters used in this study are as shown in Table 1, which are approximately 1/10th of the corresponding parameters in the BGL within Loghub.

The strings for the log content part were randomly generated.

Parameter Log Pattern

In this experiment, we prepared seven commonly considered parameter anomaly patterns, including three Integer types and four String types. Figure 2 shows the pattern of parameter anomalies in this experiment.

EVALUATED MODELS

In this experiment, we utilized NeauralLog and PLELog as representative models of sequence anomaly detection due to their high accuracy. The reason for their selection was based on the assessment that they have a higher potential for addressing parameter anomalies compared to other models. Both models were chosen because their use of semantic information was deemed likely to enable the detection of parameter anomalies, in contrast to other sequence vectors that exclude parameters. Furthermore, we proposed four methods capable of detecting parameter anomalies and compared the accuracy of each model. The following sections will introduce each model in detail.

NeauralLog

NeauralLog is a novel approach that does not necessitate log parsing. It directly converts log messages into semantic vectors and employs a Transformer model to detect anomalies. This method extracts the semantic meaning of log messages using BERT and identifies anomalies through a Transformer-based classification model. However, there is a process that deletes data if the split strings are numerical during the operation of splitting the logs. Therefore, it is anticipated that NeauralLog will perform poorly in detecting integer type parameter anomalies in this experiment.

Int Type	Normal	Anomaly
Param 1	Range: -100 ~ 100	Out of Range: -10000 ~ -101 || 101 ~ 10000
Param 2	Range such as Registration ID	Out of Range: 30 ~ 10000
Param 3	Random Specific Value: 30 values in the range 0 ~ 10000	Non-Specific Values in the Range 0 ~ 10000
Param 4	Non-Specific Values in the Range 0 ~ 10000	Random Specific Value: 30 values in the range 0 ~ 10000

String Type	Normal	Anomaly	Note
Param 5	"true"	"false", "null", "error"	Words for which positive negatives
Param 6	"connected"	"disconnected"	can be determined
Param 7	"MODE_0", "MODE_1", …, "MODE_29")	"MODE_30" ~ "MODE_10000"	Words for which a positive negative cannot be determined
Param8	World Countries	World Cities	Same as above

Figure 2: Anomaly parameter patterns.

PLELog

PLELog adopts a semi-supervised approach that uses only the labels of known normal log sequences to estimate the labels of mixed anomalous and normal log sequences through probabilistic label estimation. This allows for leveraging the advantages of a supervised approach while saving time on manual labeling. Additionally, it deals with unstable log data by using semantic embeddings and an attention-based (Gated Recurrent Unit) GRU neural network to efficiently and effectively detect anomalies. The steps for creating semantic embeddings include 1) log parsing, 2) word embedding,

and 3) aggregation based on TF-IDF. During the word embedding process, non-text tokens (such as delimiters, operators, punctuation, and numbers) are removed. Therefore, it is predicted that PLELog will perform poorly in detecting integer type parameter anomalies in this experiment.

Proposed Methods

Initially, parameters are extracted from raw logs using Drain. Next, it is determined whether the extracted parameter strings are of type integer or string. If the type is Integer, anomaly detection is performed using Z-score, Interquartile range (IQR), k-Nearest Neighbour (k-NN), and Density-Based Spatial Clustering of Applications with Noise (DBSCAN). For String types, anomaly detection is conducted using a combination of BERT and Transformer for positive-negative judgment.

$$\mathrm{Z-score} = (\mathrm{X} - \mu)/\sigma \tag{1}$$

X represents an individual data point, μ is the mean of the dataset, and σ is the standard deviation of the dataset. Data is considered anomalous when the Z-Score exceeds a threshold of 2.

IQR: The Interquartile Range (IQR) is a measure of variability based on dividing a dataset into quartiles. The IQR is defined as Q3-Q1, and data points lying outside Q3+1.5×IQR or Q1–1.5×IQR are considered outliers.

Table 2. Experimental results.

F1-Score	PLELog	NeauralLog	Z Score	IQR	k-NN	DBSCAN
Param 1	0.385	0.000	0.655	0.998	0.821	0.947
Param 2	0.356	0.000	0.617	1.000	0.779	0.902
Param 3	0.177	0.000	0.000	0.000	0.818	0.912
Param 4	0.345	0.000	0.000	0.000	0.000	0.000
F1-Score	PLELog		NeauralLog		DistilBert	
Param 5	0.256		0.000		1.000	
Param 6	0.320		0.000		1.000	
Param 7	0.384		0.000		0.834	
Param 8	0.343		0.000		0.061	

k-NN: For each data point, the distance to the second nearest neighbour is obtained. If this distance is greater than twice the average distance to the second nearest neighbour, the data point is considered anomalous.

DBSCAN: In DBSCAN, points that do not belong to any cluster are considered anomalous.

DistilBert: DistilBERT is a smaller, faster, and lighter Transformer model based on the BERT architecture. We used the DistilBERT model fine-tuned on the Stanford Sentiment Treebank (SST-2) dataset for sentiment analysis tasks. Items judged as Negative by this model are considered anomalies.

EXPERIMENTAL METHOD

Sequence Anomaly Detection Models Input

The input utilized sequence data with Window = 20 and Slide = 1.

Dataset Split Ratio

For NeauralLog and PLELog, which are learning-based methods, the dataset was divided into Training = 0.6, Validation = 0.3, and Test = 0.1.

On the other hand, the proposed method, which does not use learning, created distances and classes based on all logs and detected anomalous data.

Accuracy Evaluation Method

In the accuracy comparison, the accuracy of anomaly detection on the test data is verified using each model after training. Each model is evaluated for classification performance using the F-measure value; The F-measure is an evaluation index that indicates the balance between detection accuracy and the number of anomaly detections. Here, the F-measure is computed as follows.

$$\begin{aligned} Precision &= \frac{TP}{TP+FP} \\ Recall &= \frac{TP}{TP+FN} \\ F-measure &= \frac{2 \cdot Precision \cdot Recall}{Precision+Recall} \end{aligned} \tag{2}$$

Where,

TP: Anomaly instances correctly classified by the model
TN: Normal instances correctly classified by the model
FP: Normal instances misclassified by the model
FN: Abnormal instances misclassified by the model

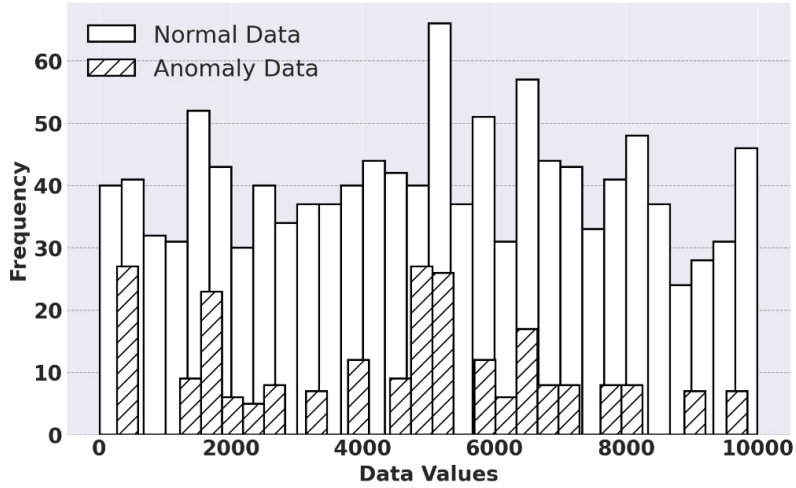

Figure 3: The histograms for the datasets used in Param 4.

EXPERIMENTAL RESULTS

We conducted an accuracy evaluation of NeauralLog, PLELog, and the proposed method on the dataset with each parameter anomaly pattern shown in Figure 2. The experimental results are presented in Table 2. Looking at the results, it is evident that both NeauralLog and PLELog have low accuracy for both Int and String types. On the other hand, the proposed method shows high-accuracy results, except for the experiment with Param 4. Particularly, DBSCAN shows high accuracy for Integer types, and Bert shows high accuracy for String types. It is understandable that NeauralLog and PLELog would exhibit decreased accuracy due to their processes that exclude numerical parameters. However, the low accuracy with String-type data, which are not excluded, indicates that these models struggle with detecting parameter anomalies for each line. It is important to note, though, that these models are specialized for sequence anomalies.

Next, we consider the experiment with Param4. The histograms for the datasets used in Param 4 are shown in Figure 3. While the parameters for Normal and Anomaly in Param4 do not overlap, they are randomly selected from the same range of values. As a result, it is thought that classification based on clustering or distance did not function effectively.

Moreover, it is understandable why Param5 and Param6, which conform to positive-negative judgment, would have high accuracy. Additionally, it makes sense that Param8, which does not adhere to positive-negative judgment, shows low accuracy. Contrarily, the high accuracy observed in Param7 can be attributed to the influence of the value following "MODE_". However, it is necessary to note that in this instance, since anomalies are defined as cases with high values, the method will not function in situations that oppose this definition. However, it should be noted that this time, since high values were defined as anomalies, the approach will not work in cases that contradict this definition.

CONCLUSION

In this study, we created a dataset for parameter anomaly detection using our custom log generator and conducted an investigation into the challenges of parameter anomaly detection for current anomaly detection models, which had not been thoroughly explored before, and examined proposed methods. As a result, we found that sequence anomaly detection models struggle to detect parameter anomalies on a per-line basis. On the other hand, we demonstrated the potential to detect anomalies by extracting parameters and using parameter-specific machine learning methods, as done with the proposed methods. However, we also discovered anomaly patterns that cannot be addressed with simple machine learning methods, indicating the need to explore various approaches, including specialized DNN models for parameter anomalies. We hope that there will be an increase in research using parameter anomaly log generators like the one we have created.

ACKNOWLEDGMENT

This work is supported by a grant from Panasonic System Design.

This work was supported by JST, Kyutech Research Fellowship, Grant Number JPMJFS2133.

REFERENCES

Bei Zhu, Jing Li, Rongbin Gu, and Liang Wang., (2020) "An Approach to Cloud Platform Log Anomaly Detection Based on Natural Language Processing and LSTM", ACAI '20: Proceedings of the 2020 3rd International Conference on Algorithms, Computing and Artificial Intelligence, pp. 1–7, https://doi.org/10.1145/3446132.3446415.

He. P, Zhu. J, Zheng. Zm and Lyu. M, (2017) "Drain: An Online Log Parsing Approach with Fixed Depth Tree", 2017 IEEE International Conference on Web Services (ICWS).

He. S, Zhu. J, He. P, R. M, and Lyu. M, (2020) "Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics", Arxiv Website: https://arxiv.org/pdf/2008.06448.pdf.

Le. V and Zhang. H, (2022) "Log-based anomaly detection with deep learning: how far are we?", ICSE '22: Proceedings of the 44th International Conference on Software Engineering, pp. 1356–1367.

Le. V.-H., Zhang. H., (2021) "Log-based Anomaly Detection Without Log Parsing", in Proceedings of the 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp. 492–504, doi: 10.1109/ASE51524.2021.9678773.

Shilin. H., Xu. Z., Pinjia. H., Young. X., Liqun. L., Yu. K., Minghua. M., Yuning. W., Yngnong. D., Sarabanakumar. R., & Qingwei. L., (2022 November 9) "An empirical study of log analysis at Microsoft", ESEC/FSE 2022: Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 1465–1476, https://doi.org/10.1145/3540250.3558963.

Tree, IEEE Transactions on Service Computing, vol. 16, issue 5, pp. 3224–3237, doi: 10.1109/TSC.2023.3270566.

Uchida. H, Tominaga. K, Itai. H, Li. Y, and Nakatoh. Y., (2023) "Verification of Generalizability in Software Log Anomaly Detection Models", Anomaly Detection - Recent Advances, AI and ML Perspectives and Applications, doi: 10.5772/intechopen.111938.

Uchida. H, Tominaga. K, Itai. H, Li. Y, and Nakatoh. Y., (2024) "Differences between research log datasets and development field logs and creation of complexity evaluation index", Pertanika Journal, under review.

Yu. S, He. P, Chen. N, and Wu. Y, (2023) "Brain: Log Parsing with Bidirectional Parallel".