Empirical Software Engineering (2023) 28:6
https://doi.org/10.1007/s10664-022-10214-6

An empirical study of the impact of log parsers on the performance of log-based anomaly detection

Ying Fu[1,2] · Meng Yan[1,2] · Zhou Xu[1,2] · Xin Xia[3] · Xiaohong Zhang[1,2] · Dan Yang[1,2]

Accepted: 20 July 2022

Abstract

Log-based anomaly detection plays an essential role in the fast-emerging Artificial Intelligence for IT Operations (AIOps) of software systems. Many log-based anomaly detection methods have been proposed. Due to the variety and unstructured characteristics of logs, log parsing is the first necessary step for parsing logs into structured ones in log-based anomaly detection methods. Prior studies have found that the effectiveness of log parsing will impact the performance of log-based anomaly detection. However, few studies comprehensively investigate whether better log parsing implies better anomaly detection. In this paper, we conduct a comprehensively empirical study to investigate the impact of six state-of-the-art log parsers belonging to four categories (including heuristic-based, frequency-based, clustering-based, and subsequence-based) on six state-of-the-art log-based anomaly detection methods (including machine-learning-based and deep-learning-based methods). Experimental results on three public datasets show that (1) High parsing accuracy does not definitely imply high anomaly detection performance. Both parsing accuracy and the number of parsed event templates should be considered when choosing log parsers for anomaly detection. (2) The log parsers have an impact on the efficiency of anomaly detection methods. With the increase in the number of parsed event templates, the efficiency of anomaly detection decreases. In detail, the heuristic-based parsers have less impact on the efficiency of anomaly detection methods, followed by frequency-based parsers. (3) All the anomaly detection methods perform more effectively and efficiently with the heuristic-based log parsers. Thus, the heuristic-based log parsers are recommended for a new practitioner on anomaly detection. We believe that our work, with the evaluation results and the corresponding findings, can help researchers and practitioners better understand the impact of log parsers on anomaly detection and provide guidelines for choosing a suitable log parser for their anomaly detection method.

Keywords Log parser · Anomaly detection · Empirical study

Communicated by: Philipp Leitner

✉ Meng Yan
mengy@cqu.edu.cn

✉ Dan Yang
dyang@cqu.edu.cn

Extended author information available on the last page of the article.

Published online: 08 November 2022

1 Introduction

Modern software systems have evolved to provide 24/7 h of online services. The system breaks down can lead to severe economic loss in the real world. For example, the loss of one-hour downtime for Amazon on Prime Day in 2018 is up to $100 million. If the anomalies can be detected before the system breaks down, the quality and reliability of the system can be effectively improved. Therefore, anomaly detection plays an essential role in the fast-emerging Artificial Intelligence for IT Operations (AIOps) (Dang et al. 2019; Lin et al. 2018; He et al. 2018a; El-Sayed et al. 2017; Huang et al. 2018).

Since the logs are used to record the detailed running status information of the system when the system is running, it is widely used for anomaly detection (i.e., Liu et al. 2019; Xia et al. 2020; Nandi et al. 2016; Breier and Branišová 2015), failure diagnosis (i.e., Chen 2019; Yuan et al. 2010; Babenko et al. 2009; Jia et al. 2017), and failure prediction (i.e., Berrocal et al. 2014; Chen et al. 2019; Zhou et al. 2019). As logs are too massive to examine manually, many semi-automatic or automatic log-based anomaly detection methods have been proposed. According to the adopted technique, log-based anomaly detection methods can be categorized into keyword-searching-based methods, rule-based methods, machine-learning-based methods (i.e., Chen et al. 2004; Lou et al. 2010; Lin et al. 2016), and deep-learning-based methods (i.e., Zhang et al. 2019; Du et al. 2017; Meng et al. 2019). The limitations of the keyword-searching-based methods are inaccuracy and insufficiency. And the limitations of rule-based methods are that they require the operator to have domain knowledge and involve the operator in making the rules and their limited coverage. To overcome the limitations of keyword-searching-based methods and rule-based methods, many machine-learning-based methods and deep-learning-based methods are proposed. We focus on the impact of log parsers on the performance of machine-learning-based and deep-learning-based methods.

These methods extract features from logs as model input. The raw logs generated by the system are semi-structured and cannot be directly used for feature extraction. So, it needs to be parsed and converted into structured data. Automatic log parsers are widely used in the data preprocessing stage of anomaly detection. Automatic log parsers can be divided into source code-based log parsers (i.e., Nagappan et al. 2009; Xu et al. 2009) and data-driven log parsers according to the used objects. Because some source code is not easily accessible, such as commercial components, data-driven log parsers are often used. The current data-driven log parsers can be divided into four categories according to the technology adopted, heuristic-based (i.e., He et al. 2017; Makanju et al. 2011), frequency-based (i.e., Dai et al. 2020; Nagappan and Vouk 2010; Hamooni et al. 2016), clustering-based (i.e., Shima 2016; Tang et al. 2011), and others.

Due to the importance of automatic log parsing and anomaly detection in AIOps, several studies have attempted to propose new parsers (Du and Li 2018; He et al. 2017; Dai et al. 2020 and anomaly detection methods Du et al. 2017; Zhang et al. 2019; Meng et al. 2019), aiming to improve the effectiveness of anomaly detection. At the same time, some studies are devoted to the empirical evaluation of automatic log parsers and anomaly detection methods. He et al. (2016b) evaluated the performance of six machine-learning-based anomaly detection methods on two public datasets. Their study focused on evaluating the effectiveness of different anomaly detection methods and did not study the impact of log parsers on different anomaly detection methods. In their subsequent study (He et al. 2016a), they evaluated the performance of four log parsers on five datasets and the impact of three log parsers on the effectiveness of one anomaly detection method. However, the impact of the follow-up state-of-the-art log parsers (i.e., Drain (He et al. 2017), Spell (Du and Li

2018), Logram (Dai et al. 2020)) on the effectiveness of supervised machine-learning-based (i.e., Logistic Regression (Bodik et al. 2010) and Decision Tree (Chen et al. 2004)) and deep-learning-based (i.e., Deeplog (Du et al. 2017) and LogRobust (Zhang et al. 2019)) anomaly detection methods has never been investigated. Unlike the work of He et al. (2016a), we comprehensively study the impact of six log parsers on six log-based anomaly detection methods and the impact of parsing errors on the effectiveness of anomaly detection methods, then make a suggestion on log parser selection. Zhu et al. (2019) studied the performance of log parsers and made their datasets publicly available.

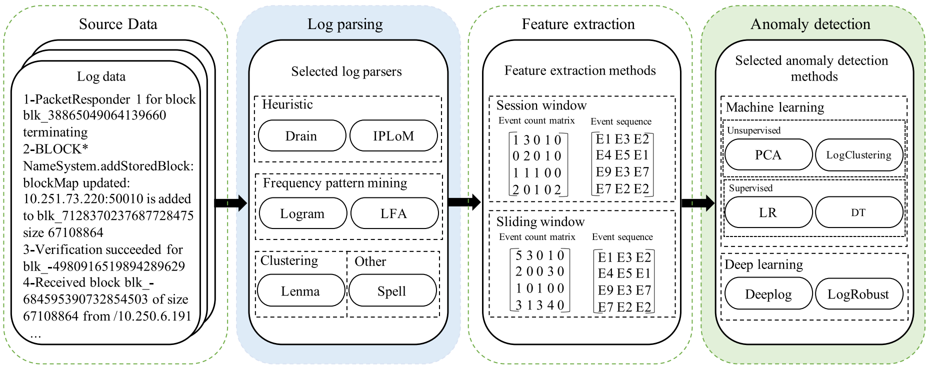
Since the input features to the anomaly detection method are extracted from the log parsing results, the effect of log parsing can impact the effectiveness of the anomaly detection method. In this article, we comprehensively evaluate the impact of log parsers on log-based anomaly detection methods to explore whether better log parsing implies better anomaly detection? If not, what is the impact of log parsing errors on anomaly detection, and what are the guidelines for choosing a suitable log parser for different types of anomaly detection methods? To this end, we conduct a comprehensive empirical study to investigate the impact of six state-of-the-art log parsers (including two heuristic-based, two frequency-based, one clustering-based, and one subsequence-based) on six anomaly detection methods (including four traditional machine-learning-based and two deep-learning-based). We public our replication package for follow-up works.[1] We believe that our work can benefit researchers and practitioners in the following two aspects: the one is to help them better understand the impact of the log parsers on anomaly detection; the other is to provide guidelines for choosing a suitable log parser for different anomaly detection methods.

In summary, the main contributions of this paper are as follows:

- We conduct a comprehensive evaluation to investigate the impact of four types of log parsers on the **effectiveness** of machine-learning-based and deep-learning-based anomaly detection methods. We find that the heuristic-based parsers are more effective for anomaly detection than other types of parsers. Additionally, high parsing accuracy does not definitely lead to high anomaly detection performance. The performance of anomaly detection is impacted by both parsing accuracy and the number of parsed event templates.
- We conduct a comprehensive evaluation to investigate the impact of four types of log parsers on the **efficiency** of machine-learning-based and deep-learning-based anomaly detection methods. We find that the log parsers have an impact on the efficiency of anomaly detection methods. With the increase in the number of parsed event templates, the efficiency of anomaly detection decrease. The efficiency of anomaly detection methods is higher on the heuristic-based parsers parsed data, followed by frequency-based parsers parsed data.

Paper organization. Section 2 reviews the log parsers, feature extraction methods, and anomaly detection methods selected in our study. Section 3 presents the experimental setup of research questions, selected datasets, evaluation setting, and evaluation metrics. Section 4 details the experimental results of each research question, respectively. Section 5 presents the threats to the validity of our work. Section 6 reviews the related studies. Section 7 concludes this paper.

[1] https://github.com/cqu-isse/Impact_evalution



```
1-PacketResponder 1 for block
blk_38865049064139660
terminating
2-BLOCK*
NameSystem.addStoredBlock:
blockMap updated:
10.251.73.220:50010 is added
to blk_7128370237687728475
size 67108864
3-Verification succeeded for
blk_-4980916519894289629
4-Received block blk_-
684595390732854503 of size
67108864 from /10.250.6.191
...
```

Fig. 1 The overview of our empirical study

2 Methodology

In this section, we introduce the methods we use in each step of our evaluation study. The overview framework of our evaluation study is presented in Fig. 1. In the log parsing stage, we introduce six log parsers (Drain (He et al. 2017), IPLoM (Makanju et al. 2011), Logram (Dai et al. 2020), LFA (Nagappan and Vouk 2010), Lenma (Shima 2016) and Spell (Du and Li 2018)) which belong to four categories. The summary information of six log parsers is presented in Table 1. We introduce two log segmentation methods (session window and sliding window) that we use in the feature extraction. In the anomaly detection stage, we introduce six anomaly detection methods, including four traditional machine-learning-based methods (PCA (Xu et al. 2009), LogClustering (Lin et al. 2016), Logistic Regression (Bodik et al. 2010), and Decision Tree (Chen et al. 2004)), and two deep-learning-based methods (Deeplog (Du et al. 2017) and LogRobust (Zhang et al. 2019)).

2.1 Log Parsers

The purpose of log parsing is to convert semi-structured log data into structured ones. Generally, the structured log data contain the fields such as timestamp, event level, event template, and dynamic variables. The timestamp, the event template, and the dynamic variables are often used for anomaly detection. The event template refers to the part of the log that is identified as static texts by the log parser. The dynamic variable refers to the part that is identified as the variable by the log parser. There are four categories of log parsers: heuristic-based, frequency-based, clustering-based, and others. In our study, we select one

Table 1 Summary of log parsers

Parser	Category	Technology	Year
Drain	Heuristic	Heuristic	2017
IPLoM	Heuristic	Heuristic	2012
LFA	Frequency	Frequency pattern mining	2010
Logram	Frequency	Frequency pattern mining	2020
Lenma	Clustering	Clustering	2016
Spell	Subsequence	Longest common subsequence	2016

or two log parsers with relatively high accuracy and efficiency in prior studies (i.e., Dai et al. 2020; Zhu et al. 2019) from each category to conduct our evaluation study.

Heuristic-Based Log Parsers They mainly use the characteristics of log data and the conditions defined by experts to partition log data. The classic heuristic-based log parsers Drain (He et al. 2017) and IPLoM (Makanju et al. 2011) are selected in our study. IPLoM uses the number of tokens in the content of log data as the first step partition condition and the token position as the second step partition condition. It partitions log data by searching map in the third step, then generates event templates. Drain uses a fixed-depth tree to represent the hierarchical relationship between log data. Similar to IPLoM, firstly, the length of the content and the beginning position token are used as the partition conditions of the first and second layers of the fixed-depth tree, respectively. Secondly, it calculates the similarity between the log data and the log event of each log group in the third layer. Thirdly, it generates event templates from each group. Heuristic-based parsers have the characteristics of high efficiency (Zhu et al. 2019).

Frequency-Based Log Parsers They mainly construct frequent itemsets to group log data. The classic frequency-based log parser LFA (Nagappan and Vouk 2010) and the latest frequency-based log parser Logram (Dai et al. 2020) are selected in our study. Firstly, LFA builds a frequency table with the number of times that a word occurs in a specific position of the log data. Secondly, it uses the frequency of a word in a specific position of the log data to generate the event templates. Logram leverages n-grams to parse log data. Firstly, Logram builds n-gram dictionaries. Secondly, it parses log data by the n-gram dictionaries. Frequency-based log parsers also have the characteristics of high efficiency (Dai et al. 2020; Zhu et al. 2019).

Clustering-Based Log Parsers The efficiency of clustering-based log parsers is lower than heuristic-based log parsers and frequency-based log parsers (Zhu et al. 2019). Most of them cannot parse the datasets that we use in a reasonable time. So, we use the classic Lenma (Shima 2016), which is more efficient than other clustering-based log parsers in our study. Lenma creates a word length vector and a word vector of the new message for clustering.

Subsequence-Based Log Parsers In addition to the above three categories of log parsers, we also use subsequence-based Spell (Du and Li 2018) in our study. The Spell uses the longest common subsequence based algorithm to parse log data. It computes the longest common subsequence of two log data to generate the event templates.

2.2 Feature Extraction

The purpose of feature extraction is to generate the numerical or sequence features of log data from the parsed log data. The first step of feature extraction is to segment the parsed log data into event subsequences by different segmentation methods (He et al. 2016b). The segmentation methods include the session window segmentation method, fixed window segmentation method, and sliding window segmentation method. According to the characteristics of the dataset, we use the session window segmentation method and the sliding window segmentation method.

Session Window It segments the log data by the special identifiers in them. The log data that contain the same identifier is segmented into the same session window. For example, HDFS log data use block IDs to record the different operations of a block. The block IDs are used to segment the log data into different session windows, a life cycle of a block.

Sliding Window Unlike the session window segmentation method, the sliding window segmentation method uses the window size and the step size to segment the log data into event subsequences. The window size and the step size can be either a period of time or the number of log events. For example, 6 h window sliding every half hour or 20 log events window sliding every 10 log events. Generally, the step size is smaller than the window size in the sliding window segmentation method.

After segmentation, the log data need to be converted into an event count matrix as the input of traditional machine-learning-based anomaly detection methods. In this process, we count the occurrence times of different log events in the segmented event subsequences to generate an event count matrix. For example, the event count vector [3, 0, 2] means that event 1 occurs three times, event 2 occurs zero times, and event 3 occurs two times in this segmented event subsequences. For deep-learning-based anomaly detection methods, log event subsequences can be used directly as the model input. For example, the event subsequence [E1, E5, E2] means that event 5 happens after event 1, and event 2 happens after event 5.

2.3 Anomaly Detection Methods

The current anomaly detection methods can be divided into machine-learning-based and deep-learning-based methods. We study the impact of six state-of-the-art log parsers on the effectiveness of machine-learning-based and deep-learning-based methods respectively for the following reasons: 1) Due to the two kinds of methods use different feature processing methods, they cannot be fairly compared in the effectiveness of anomaly detection; 2) Since the two kinds of methods use different computing resources, they cannot be fairly compared in efficiency.

1) *Machine-learning-based Methods*

   Principal Component Analysis (PCA). PCA is a classic dimension reduction method. It transforms the original high-dimension data into low-dimension data that can preserve the major characteristics of the original data. Xu et al. (2009) proposed to apply PCA for anomaly detection based on their four key insights. They used PCA to capture dominant patterns in the event subsequences to construct a low dimensional normal space. PCA-based anomaly detection, firstly, transforms the segmented event subsequences into event count vectors as the input of PCA. Secondly, PCA is applied to detect non-dominant patterns in the event subsequences that are considered as anomaly event patterns. All parameters in PCA can be either chosen automatically or tuned easily. The most important parameter is the threshold of squared prediction error.

   LogClustering. LogClustering is a clustering-based method provided by Lin et al. (2016). Firstly, in the training phase, LogClustering converts each segmented event subsequence into an event count vector and assigns a weight to each event in the vector. Secondly, it clusters event subsequences by the agglomerative hierarchical clustering method based on the calculated similarity between two event subsequences. It is worth mentioning that LogClustering only uses normal event count vectors in the training phase. Thirdly, it selects an event subsequence with the minor score with the centroid of a cluster to represent the cluster of event subsequences.

In the testing phase, LogClustering calculates the distance between the new event subsequence and the representative event subsequence. If the minimum distance between the new event subsequence and the representative event subsequences is greater than the threshold, the new event subsequence is reported to be anormal.

Logistic Regression (LR). LR is a classic classification method. It predicts the probability that an event subsequence is anormal (Bodik et al. 2010). Both the event count vectors and their labels are the input of the model training. After the training stage, the model can predict the anomaly probability of a new event subsequence. In the anomaly detection stage, a probability threshold is set to determine whether a new event subsequence is anormal. We set the probability threshold to be 0.5, consistent with the setting in the work of He et al. (2016a). When the anormal probability of a new event subsequence predicted by the model is greater than 0.5, the new event subsequence is reported to be anormal; otherwise, it is reported to be normal.

Decision Tree (DT). Chen et al. (2004) applied DT to failure diagnosis on the web request log data. The event count vectors, which are converted from the event subsequences together with their labels, are used as the training data of the model. In the training stage, a decision tree can be built. Each path of the decision tree represents a type of event subsequence pattern. The leaf nodes of the decision tree represent the status of a type of event subsequence pattern, i.e., normal or anormal. The decision tree is traversed according to the new event subsequence in the anomaly detection stage. The state of the new event subsequence is obtained from the leaf node of the decision tree at the end of the traverse.

2) *Deep-learning-based Methods*

Deeplog. Since the log data are produced by log statements of the program that follow a rigorous set of logic and control flows and are very much like natural language, the log data can be viewed as elements of a sequence that follows specific patterns and grammar rules. Du et al. (2017) proposed Deeplog to model the pattern of event subsequences as a natural language sequence based on Long Short Term Memory (LSTM) model. They defined the anomaly detection task as a multiple classification problem. Firstly, Deeplog transforms the segmented event subsequences into event index subsequences. Secondly, it uses a small fraction normal event subsequence to train the LSTM model. Then the model can be used to recognize normal event subsequences. If the new event subsequence is the same as the model predicts, it is considered as normal; else anormal.

LogRobust. Since there are semantic characteristics in log data, some studies (i.e., Zhang et al. 2019; Meng et al. 2019) used the NLP technology for anomaly detection. Zhang et al. (2019) proposed LogRobust for anomaly detection, which uses the semantic characteristic in log data. They defined the anomaly detection task as a supervised binary classification problem. LogRobust firstly uses word vectors to present the words in the log events. The word vectors are pre-trained on the Common Crawl Corpus dataset, and the dimension of word vectors is 300. Secondly, it uses the TF-IDF weight to sum up all word vectors in a log event to generate the log event vector. At the same time, the event subsequences, which are the output of the previously mentioned log segmentation method, are converted into a semantic vector list. The length of the semantic vector list is the window size used by segmentation. Thirdly, LogRobust uses the attention-based Bi-LSTM neural network model to classify the log subsequences. The attention layer is used to automatically

Table 2 Summary of datasets

Dataset	Category	Data size	Time span	Original logs	Anomalies	Anomalies (%)
HDFS	Distributed system logs	1.47GB	38.7 h	11,175,629	16,838(blocks)	2.93
BGL	Supercomputer logs	667.3MB	214.7 days	4,747,693	348,460	7.33
ThunderBird	Supercomputer logs	708.76MB	244 days	3,992,351	162,953	4.08

learn the weight of the log events by LogRobust because log events have different contributions to the status of event subsequence. LogRobust can reduce the impact of noisy log events because noisy events tend to have a small weight.

3 Experimental Setup

This section presents our research questions, selected datasets, evaluation setting, and evaluation metrics.

3.1 Research Questions

We plan to investigate the following three Research Questions (RQ):

- RQ1: What is the impact of log parsers on the effectiveness of anomaly detection methods?
- RQ2: What is the impact of log parsing errors and the number of parsing event templates on anomaly detection?
- RQ3: How do log parsers impact the efficiency of anomaly detection methods?

3.2 Datasets

We employ three open-source datasets to evaluate the impact of log parsers on anomaly detection methods. The basic information of the datasets is provided in Table 2.

HDFS data[2] are distributed system logs that were collected from more than 200 Amazon's EC2 nodes and labeled by Hadoop domain experts (Xu et al. 2009). Among 11,197,945 log messages, 16,383 log messages are labeled as anomalies, and anormal log messages account for 2.93%.

BGL data[3] are supercomputer logs that were collected from the BlueGene/L supercomputer system at Lawrence Livermore National Labs (Oliner and Stearley 2007). Among 4,747,693 log messages, 348,460 are identified as failures, accounting for 7.33%.

ThunderBird data[4] are supercomputer logs that were collected from the ThunderBird supercomputer system at Sandia National Labs (Oliner and Stearley 2007). There are more than two hundred million log messages in the original dataset. Since two hundred million log

[2]https://zenodo.org/record/3227177/files/HDFS_2.tar.gz?download=1
[3]https://zenodo.org/record/3227177/files/BGL.tar.gz?download=1
[4]https://zenodo.org/record/3227177/files/Thunderbird.tar.gz?download=1

messages are highly time-consuming for the clustering-based approach, we choose the top four million log messages that we can handle in a reasonable time, as Yin et al. (2020). The ThunderBird data are identified as alert and non-alert log messages by the tags contained in the logs. There are 162,953 log messages identified as alert log messages, accounting for 4.08%.

3.3 Evaluation Setting

We run all our experiments on a Linux server with Intel Xeon E5-2650 v4 CPU and 256GB DDR3, on which 64-bit Debian GNU/Linux 9.13 with Linux kernel 4.9.0 is running. Each anomaly detection method is run five times, and the average result is reported.

In the feature extraction stage, the anomaly detection methods use different segment methods on different datasets. For the machine-learning-based methods on HDFS data, the logs are segmented by the session window. The data of the same block ID is segmented into the same window, consistent with the setting in the work of He et al. (2016a). For BGL data, the sliding window segment method is adopted with window size = 6 h and step size = 1h, consistent with the setting in the work of He et al. (2016a). For ThunderBird data, we also use the sliding window segment method to segment the logs. We experiment with the segment settings, a comprehensive better segment setting (window size = 0.1 h and step size = 0.03 h) is selected. For deep-learning-based methods on HDFS data, we also use the session window segment method, consistent with the setting in the work of Zhang et al. (2019). At the same time, the number of events is used as the window size and step size of the sliding segment method on BGL data and ThunderBird data. Similar to the existing work of Yin et al. (2020), we set window size = 18 and step size = 10 on both BGL and ThunderBird data.

Different anomaly detection methods use different ways to split data for training and test in the model training. Similar to the existing work (He et al. 2016a), 80% of event subsequences are used as the training data for all supervised machine-learning-based methods, and the remaining 20% of event subsequences are used as the testing data. Since there is no training stage in unsupervised machine-learning-based methods, 20% of event subsequences are used for testing directly. For deep-learning-based methods, since Deeplog and LogRobust defined the anomaly detection task as a different problem, the number of event subsequences used for model training is different. Deeplog only uses 1% of normal event subsequences for the model training on HDFS, and the remaining event subsequences are used for the model testing (Du et al. 2017). On BGL and ThunderBird data, Deeplog uses 80% of normal event subsequences for the model training, and the remaining event subsequences are used for the model testing, consistent with the setting in the work of Yin et al. (2020). Since LogRobust uses balanced data for model training, we use 6000 normal event subsequences and 6000 anormal event subsequences for model training on HDFS data. The remaining event subsequences are used for model testing as Zhang et al. (2019). Similar to the split setting of HDFS, for LogRobust on ThunderBird, we use 20,000 normal event subsequences and 20,000 anormal event subsequences for model training, and the remaining event subsequences are used for testing. For LogRobust on BGL, we use 28,000 normal event subsequences (about seven times the number of anormal event sequences used for training) and 3,995 anormal event subsequences (80% anormal event subsequences) for model training, and the remaining event subsequences are used for model testing. We use unbalanced data for LogRobust training on BGL data because the anormal logs are concentrated. We experiment with the split settings; a better split setting is selected.

3.4 Evaluation Metrics

We use parsing accuracy (PA), which is defined by Zhu et al. (2019), to evaluate the accuracy of log parsers. As shown below, the logs are considered to be parsed correctly if the logs that are grouped together indeed belong to the same event template, and all the logs that indeed belong to this event template are in this group.

$$PA = \frac{Correctly\ parsed\ log\ messages}{The\ total\ number\ of\ log\ messages} \tag{1}$$

We use precision (P), recall (R), and F-measure (F1) which are widely used metrics as the evaluation metrics of anomaly detection methods. As shown below, TP is the number of anormal logs that are detected as anormal logs by model. FP is the number of normal logs that are detected as anormal logs by the model. FN is the number of anormal logs that are detected as normal logs by the model.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

$$F\text{-}measure = \frac{2 * Precision * Recall}{Precision + Recall} \tag{4}$$

4 Evaluation Results

4.1 RQ1: What is the Impact of Log Parsers on the Effectiveness of Anomaly Detection Methods?

Motivation Current log parsers use parsing accuracy as the key performance metric. Parsing accuracy is also an important indicator for researchers and industrial practitioners to choose which log parser to preprocess data in anomaly detection methods. However, different anomaly detection methods have different working principles and feature processing methods. Does higher parsing accuracy imply higher anomaly detection performance? In this question, we investigate the impact of log parsing accuracy on four machine-learning-based (including two unsupervised and two supervised) and two deep-learning-based anomaly detection methods, respectively. And we verify the correlation between the parsing accuracy and the effectiveness of anomaly detection methods to evaluate whether higher parsing accuracy implies higher anomaly detection performance.

Methods First, we implement the six selected log parsers, namely Drain, IPLoM, Logram, LFA, Lenma, and Spell, following their description in the prior studies (He et al. 2017; Makanju et al. 2011; Dai et al. 2020; Nagappan and Vouk 2010; Shima 2016; Du and Li 2018), and evaluate their parsing accuracy on three public log parsing datasets (Zhu et al. 2019). We follow the evaluation method of Zhu et al. (2019). Each of the three public log parsing datasets contains 2,000 manually labeled logs as the ground truth. Second, we use the six log parsers to parse the described datasets in Section 3. Third, we implement the selected machine-learning-based (ML-based) anomaly detection methods, namely DT, LR, LogClustering, and PCA, and deep-learning-based (DL-based) anomaly detection methods, namely LogRobust and Deeplog, following their description from the prior studies (Chen et al. 2004; Bodik et al. 2010; Lin et al. 2016; Xu et al. 2009; Zhang et al. 2019; Du et al.

2017). Fourth, we run the two kinds of anomaly detection methods on three datasets parsed by the six selected log parsers. We also run the two kinds of anomaly detection methods on the Ground Truth of HDFS and BGL which are the exactly correct parsed results. The Ground Truth of HDFS and BGL is provided by prior studies (He et al. 2016b). To fairly compare the impact of different types of log parsers on the anomaly detection methods, all their parameters remain the same on the same dataset which is parsed by six log parsers. Fifth, to evaluate whether higher parsing accuracy implies higher anomaly detection performance, we verify the correlation between the parsing accuracy and the effectiveness of anomaly detection methods with the Spearman rank correlation test (Zar 2005). The Spearman rank correlation test was used to verify whether there is a close correlation between the parsing accuracy and the effectiveness of anomaly detection methods at the 95% confidence level if the p-value < 0.05. Sixth, we use the Wilcoxon rank-sum test (Wilcoxon 1992) with a Bonferroni correction (Abdi and et al 2007) at the 95% confidence level (p-value < 0.05) to analyze the statistically significant difference in the effectiveness of anomaly detection methods on different log parsers parsed data.

Results Table 3 shows the parsing accuracy of six log parsers on three datasets. Table 4 shows the effectiveness of ML-based anomaly detection methods on six log parser parsed data, and Table 6 shows the effectiveness of two DL-based anomaly detection methods on six log parser parsed data. Table 5 shows the Spearman correlation between PA and the effectiveness of ML-based methods. Table 7 shows the Spearman correlation between parsing accuracy and the effectiveness of DL-based methods. Tables 8 and 11 show the average effectiveness of ML-based methods and DL-based, respectively. Tables 9 and 10 show the statistical significance of the difference between the effectiveness of unsupervised ML-based methods and supervised ML-based methods on IPLoM parsed data and other parsers parsed data, respectively. Table 12 shows the statistical significance of the difference between the effectiveness of Deeplog on IPLoM parsed data and other parsers parsed data, and the statistical significance of the difference between the effectiveness of LogRobust on Drain parsed data and other parsers parsed data. Figure 2 shows the effectiveness distribution of two unsupervised ML-based methods. Figure 3 shows the effectiveness distribution of two supervised ML-based methods, and Fig. 4 shows the effectiveness distribution of two DL-based methods across different log parsers parsed data. From the results shown in Tables 3, 4, 5, 6, 7, 8, 9, 10, 11, and 12, Figs. 2, 3, and 4, we make the following observations:

(1) As shown in Table 3, the average parsing accuracy of the heuristic-based Drain (97.17%) is the best, followed by the clustering-based Lenma (87.67%) and the subsequence-based Spell (87.67%). The average parsing accuracy of frequency-based Logram and LFA is lower than that of the other three types of log parsers. Specifically,

Table 3 Parsing accuracy (%) of the log parsers on three datasets

Datasets	Drain	IPLoM	LFA	Logram	Lenma	Spell
HDFS	99.75	**100.00**	<u>88.50</u>	99.75	99.75	**100.00**
BGL	**96.25**	93.90	85.40	<u>63.95</u>	68.95	78.65
ThunderBird	**95.50**	66.30	<u>64.85</u>	65.40	94.30	84.35
Average	**97.17**	86.73	79.58	<u>76.37</u>	87.67	87.67

The parsing accuracy highlighted in bold font is the best among the six log parsers and highlighted in underline is the worst among the six log parsers

Table 4 The effectiveness of ML-based anomaly detection methods on six log parsers parsed logs

Dataset	Log parser	Unsupervised machine learning-based methods											
		PCA			LogClustering			LR			DT		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
HDFS	Ground Truth	99.80	72.62	84.07	100.00	76.80	86.88	100.00	99.41	99.70	100.00	99.81	99.90
	Drain	99.78	67.34	80.41	100.00	76.80	86.88	100.00	99.41	99.70	100.00	99.81	**99.90**
	IPLoM	99.80	72.62	**84.07**	100.00	77.05	**87.04**	100.00	99.41	99.70	100.00	99.81	**99.90**
	LFA	99.79	70.86	82.85	92.72	76.57	83.87	89.21	99.18	93.93	88.99	99.51	93.96
	Logram	83.69	12.65	21.98	6.75	76.38	12.41	99.94	99.67	**99.81**	100.00	99.81	**99.90**
	Lenma	99.78	67.34	80.41	100.00	76.80	86.88	100.00	99.41	99.70	100.00	99.81	**99.90**
	Spell	99.78	67.34	80.41	100.00	77.05	**87.04**	100.00	99.41	99.70	100.00	99.81	**99.90**
BGL	Ground Truth	98.50	44.56	61.36	37.75	94.92	54.01	81.50	63.13	71.14	75.28	62.93	68.53
	Drain	65.85	45.92	54.11	37.02	91.99	52.80	87.16	61.70	**72.20**	73.45	64.42	**68.63**
	IPLoM	95.62	44.56	**60.79**	37.43	93.41	**53.45**	86.26	61.02	71.45	74.47	63.27	68.41
	LFA	43.22	46.60	44.84	37.31	88.23	52.44	70.11	59.05	64.08	67.38	58.57	62.66
	Logram	28.44	98.30	44.12	36.18	83.62	50.51	87.22	26.60	40.75	43.30	50.48	46.59
	Lenma	28.52	100.00	44.38	36.82	87.67	51.86	54.81	17.76	26.82	51.12	51.09	51.10
	Spell	27.56	94.22	42.65	36.88	89.85	52.30	92.53	42.86	58.57	72.87	57.28	64.13
ThunderBird	Drain	95.38	64.71	77.11	62.00	96.59	**75.52**	97.09	95.56	96.32	95.54	96.92	96.23
	IPLoM	95.52	66.89	**78.68**	61.46	96.65	75.14	97.84	96.45	97.15	96.60	96.62	**96.61**
	LFA	95.75	36.92	53.29	60.64	96.77	74.56	98.20	96.81	97.50	96.27	96.05	96.16
	Logram	96.39	25.47	40.30	60.89	97.12	74.85	97.47	93.43	95.41	96.75	91.77	94.19
	Lenma	97.33	49.72	65.82	61.75	96.24	75.23	97.08	96.08	96.58	96.95	95.07	96.00
	Spell	77.97	53.54	63.49	60.16	96.77	74.19	98.70	96.89	**97.86**	94.02	98.07	96.00

The effectiveness on the Ground Truth is highlighted in grey color. The effectiveness highlighted in bold font is the best among the six log parsers and highlighted in underline is the worst among the six log parsers

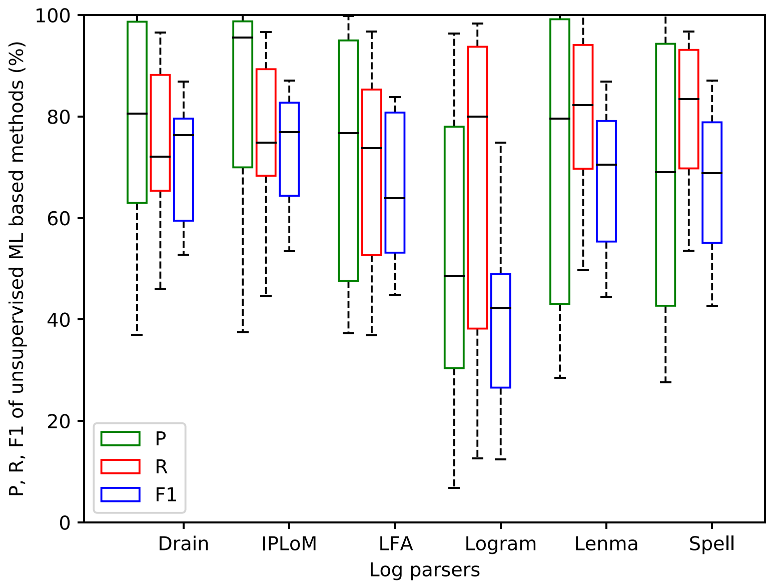


Fig. 2 Precision (P) distribution, Recall (R) distribution, and F-measure (F1) distribution of unsupervised ML-based methods across different log parsers

all log parsers present high parsing accuracy on HDFS data (88.5% at least). Drain presents the highest parsing accuracy on both BGL and ThunderBird data (96.25% and 95.50%, respectively).

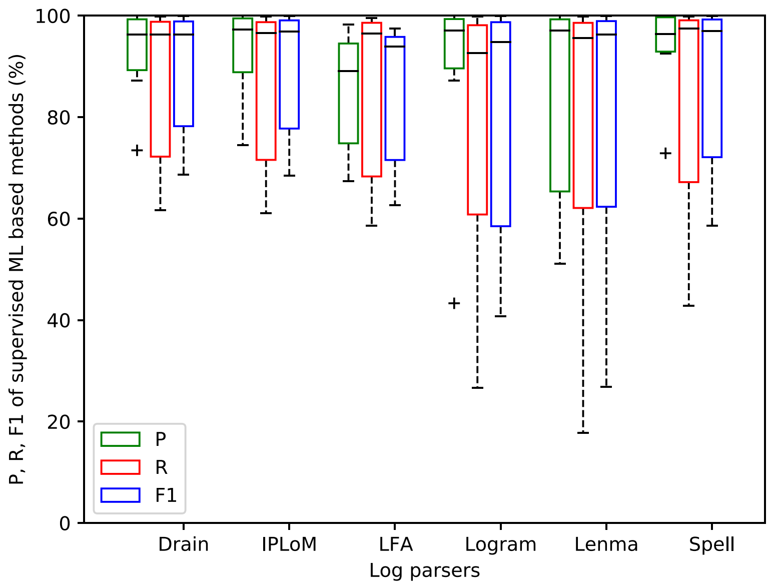


Fig. 3 Precision (P) distribution, Recall (R) distribution, and F-measure (F1) distribution of supervised ML-based methods across different log parsers

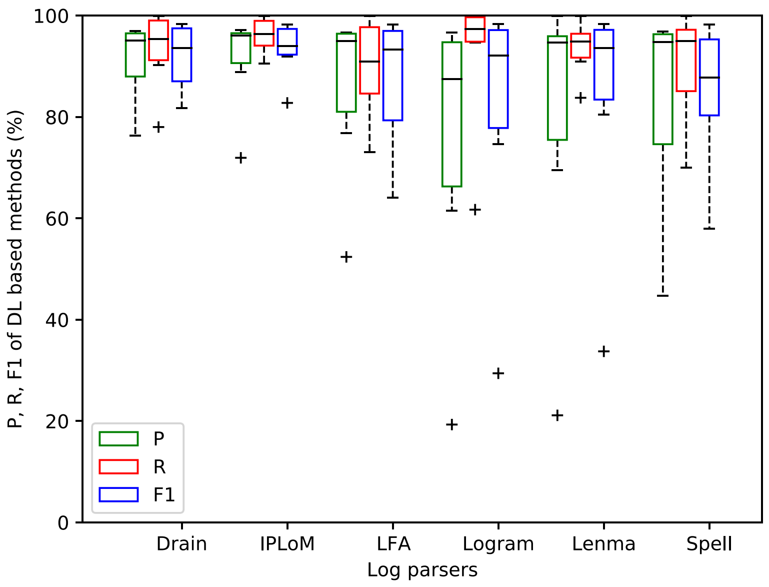


Fig. 4 Precision (P) distribution, Recall (R) distribution, and F-measure (F1) distribution of DL-based method across different log parsers

(2) Table 4 shows the effectiveness of ML-based methods. We can observe that ML-based anomaly detection methods do not always perform the best on the data parsed by the parser with the highest parsing accuracy. For example, although Drain presented the highest parsing accuracy on both BGL and ThunderBird data (96.25% and 95.50%,

Table 5 Spearman correlation coefficient (Spearman $|\rho|$) between parsing accuracy (PA) and the effectiveness of ML-based anomaly detection methods

The effectiveness of	PA of HDFS		PA of BGL		PA of ThunderBird	
ML-based methods	Spearman $\lvert\rho\rvert$	p-value	Spearman $\lvert\rho\rvert$	p-value	Spearman $\lvert\rho\rvert$	p-value
Precision of PCA	0.131	>0.05	0.771	>0.05	0.257	>0.05
Precision of LogClustering	0.584	>0.05	0.486	>0.05	0.657	>0.05
Precision of LR	0.730	>0.05	0.086	>0.05	0.771	>0.05
Precision of DT	0.707	>0.05	0.886	**<0.05**	0.143	>0.05
Recall of PCA	0.131	>0.05	0.886	**<0.05**	0.543	>0.05
Recall of LogClustering	0.826	**<0.05**	0.886	**<0.05**	0.754	>0.05
Recall of LR	0.365	>0.05	0.943	**<0.05**	0.200	>0.05
Recall of DT	0.707	>0.05	1.000	**<0.05**	0.429	>0.05
F-measure of PCA	0.131	>0.05	0.771	>0.05	0.600	>0.05
F-measure of LogClustering	0.826	**<0.05**	0.943	**<0.05**	0.657	>0.05
F-measure of LR	0.365	>0.05	0.943	**<0.05**	0.200	>0.05
F-measure of DT	0.707	>0.05	0.943	**<0.05**	0.174	>0.05

The p-value<0.05 indicates that the PA is close correlated with the effectiveness of anomaly detection. The p-value is highlighted in grey color and p-value<0.05 is highlighted in bold font

Table 6 The effectiveness of DL-based anomaly detection methods on six log parsers parsed logs

Dataset	Log parser	Deeplog			LogRobust		
		P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
HDFS	Ground_truth	95.94	95.60	95.77	98.92	99.83	99.37
	Drain	95.98	94.11	95.04	96.92	99.83	**98.35**
	IPLoM	95.81	93.68	94.73	97.11	99.47	98.28
	LFA	96.48	91.22	93.78	96.67	99.92	98.27
	Logram	94.36	99.27	**96.75**	94.87	99.81	97.27
	Lenma	95.93	94.10	95.01	95.93	99.98	97.91
	Spell	95.71	93.11	94.39	93.93	97.35	95.61
BGL	Ground_truth	87.74	91.78	89.72	76.39	96.36	85.22
	Drain	85.88	78.01	81.76	76.36	96.63	**85.31**
	IPLoM	88.90	95.25	**91.96**	71.94	97.44	82.77
	LFA	76.79	73.02	74.86	52.39	82.62	64.12
	Logram	19.31	61.71	29.42	61.49	94.68	74.62
	Lenma	21.13	83.78	33.75	69.47	95.69	80.50
	Spell	44.70	82.42	57.96	68.19	96.77	80.00
ThunderBird	Drain	94.27	90.23	92.21	96.67	99.97	**98.30**
	IPLoM	96.30	90.50	**93.30**	96.60	99.96	98.25
	LFA	96.70	90.60	92.12	96.27	99.88	98.04
	Logram	80.68	95.36	87.42	96.67	99.97	**98.30**
	Lenma	93.50	90.90	92.20	99.96	96.66	98.29
	Spell	96.82	69.97	81.23	96.53	99.97	98.22

The effectiveness on the Ground Truth is highlighted in grey color. The effectiveness highlighted in bold font is the best among the six log parsers and highlighted in underline is the worst among the six log parsers

Table 7 Spearman correlation coefficient (Spearman $|\rho|$) between parsing accuracy (PA) and the effectiveness of DL-based anomaly detection methods

The effectiveness of	PA of HDFS		PA of BGL		PA of ThunderBird	
DL-based methods	Spearman $|\rho|$	p-value	Spearman $|\rho|$	p-value	Spearman $|\rho|$	p-value
Precision of Deeplog	0.617	>0.05	0.943	**<0.05**	0.143	>0.05
Precision of LogRobust	0.062	>0.05	0.600	>0.05	0.609	>0.05
Recall of Deeplog	0.000	>0.05	0.314	>0.05	0.429	>0.05
Recall of LogRobust	0.802	>0.05	0.429	>0.05	0.152	>0.05
F-measure of Deeplog	0.000	>0.05	0.943	**<0.05**	0.314	>0.05
F-measure of LogRobust	0.157	>0.05	0.600	>0.05	0.493	>0.05

The p-value<0.05 indicates that the PA is close correlated with the effectiveness of anomaly detection. The p-value is highlighted in grey color and p-value<0.05 is highlighted in bold font

Table 8 The average effectiveness of ML-based methods on three datasets parsed by different parsers

Parser	PCA			LogClustering			LR			DT		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Drain	84.94	59.32	70.54	**66.34**	88.46	71.73	94.75	85.56	89.41	89.66	**87.05**	88.25
IPLoM	**96.98**	61.36	**74.51**	66.30	**89.04**	**71.88**	94.70	**85.63**	**89.43**	**90.36**	86.57	**88.31**
LFA	79.59	51.46	60.33	63.56	87.19	70.29	85.84	85.01	85.17	84.21	84.71	84.26
Logram	69.51	45.47	35.47	34.61	85.71	45.92	94.88	73.23	78.66	80.02	80.69	80.23
Lenma	75.21	**72.35**	63.54	66.19	86.90	71.32	83.96	71.08	74.37	82.69	81.99	82.33
Spell	68.44	71.70	62.18	65.68	87.89	71.19	**97.08**	79.72	85.38	88.96	85.05	86.68

The average effectiveness highlighted in bold font is the best among the six log parsers and highlighted in underline is the worst among the six log parsers

respectively), the precision and F-measure of PCA on BGL are best on IPLoM parsed data (95.62% and 60.79%). The F-measure of LR on ThunderBird is best on Spell parsed data (97.86%). Table 5 shows the Spearman correlation between parsing accuracy (PA) and the effectiveness of ML-based methods. On HDFS data, only the recall and F-measure of LogClustering are correlated with PA at the 95% confidence level (p-value<0.05). This indicates that the recall and F-measure of LogClustering are closely correlated with PA, and the Spearman correlation coefficient is 0.826. The effectiveness of other ML-based methods is not closely correlated with PA on HDFS data, and the Spearman correlation coefficients range from 0.131 to 0.730. On BGL data, the recall and F-measure of all ML-based methods are closely correlated with PA at the 95% confidence level (p-value<0.05), except the F-measure of PCA, with Spearman correlation coefficients ranging from 0.886 to 1. The precision of all ML-based methods, except DT, are not closely correlated with PA, with Spearman correlation coefficients ranging from 0.086 to 0.771. On ThunderBird data, the effectiveness of all ML-based methods is not closely correlated with PA, with Spearman correlation coefficients ranging from 0.143 to 0.771. As we know that the events recorded by BGL data are more complex and diverse than those recorded by HFDS data. This indicates

Table 9 The statistical significance of the difference between the effectiveness of unsupervised ML-based methods on data parsed by IPLoM vs. by other parsers

Log parser	PCA			LogClustering		
	p-value on P	p-value on R	p-value on F1	p-value on P	p-value on R	p-value on F1
IPLoM vs Drain	<0.05	<0.05	<0.05	>0.05	<0.05	> 0.05
IPLoM vs LFA	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Logram	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Lenma	<0.05	>0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Spell	<0.05	>0.05	<0.05	<0.05	<0.05	<0.05

The p-value > 0.05 is highlighted in underline. The p-value < 0.05 indicates that the effectiveness of the detection method on the IPLoM parsed data significantly better than other parsers. The p-value > 0.05 indicates that the effectiveness of the detection method on the IPLoM parsed data does not significantly better than other parsers

Table 10 The statistical significance of the difference between the effectiveness of supervised ML-based methods on data parsed by IPLoM vs. by other parsers

Log parser	LR			DT		
	p-value on P	p-value on R	p-value on F1	p-value on P	p-value on R	p-value on F1
IPLoM vs Drain	>0.05	>0.05	> 0.05	<0.05	>0.05	>0.05
IPLoM vs LFA	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Logram	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Lenma	<0.05	<0.05	<0.05	<0.05	<0.05	<0.05
IPLoM vs Spell	>0.05	<0.05	<0.05	<0.05	>0.05	<0.05

The p-value > 0.05 is highlighted in underline. The p-value < 0.05 indicates that the effectiveness of the detection method on the IPLoM parsed data significantly better than other parsers. The p-value > 0.05 indicates that the effectiveness of the detection method on the IPLoM parsed data does not significantly better than other parsers

Table 11 The average effectiveness of DL-based methods on three datasets parsed by different parsers

Parser	Deeplog			LogRobust		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
Drain	92.04	87.45	89.67	**89.98**	98.81	**93.98**
IPLoM	**93.67**	**93.14**	**93.33**	88.55	**98.96**	93.10
LFA	89.99	84.95	87.41	81.78	94.14	86.81
Logram	64.78	85.45	71.20	84.34	98.15	90.06
Lenma	70.19	89.59	73.65	88.45	97.44	92.23
Spell	79.08	81.83	77.86	86.22	98.03	91.28

The average effectiveness highlighted in bold font is the best among the six log parsers and highlighted in underline is the worst among the six log parsers

Table 12 The statistical significance of the difference between the effectiveness of Deeplog on data parsed by IPLoM vs. by other parsers, and the statistical significance of the difference between the effectiveness of LogRobust on data parsed by Drain vs. by other parsers

Log parser	Deeplog			Log parser	LogRobust		
	p-value on P	p-value on R	p-value on F1		p-value on P	p-value on R	p-value on F1
IPLoM vs Drain	<0.05	<0.05	<0.05	Drain vs IPLoM	<0.05	>0.05	<0.05
IPLoM vs LFA	>0.05	<0.05	<0.05	Drain vs LFA	<0.05	<0.05	<0.05
IPLoM vs Logram	<0.05	<0.05	<0.05	Drain vs Logram	<0.05	<0.05	<0.05
IPLoM vs Lenma	<0.05	> 0.05	<0.05	Drain vs Lenma	<0.05	<0.05	<0.05
IPLoM vs Spell	<0.05	<0.05	<0.05	Drain vs Spell	<0.05	<0.05	<0.05

The p-value > 0.05 is highlighted in underline. For Deeplog, the p-value < 0.05 indicates that the effectiveness of detection methods on the IPLoM parsed data significantly better than other parsers. The p-value > 0.05 indicates that the effectiveness of the detection method on the IPLoM parsed data does not significantly better than other parsers. For LogRobust, the p-value < 0.05 indicates that the effectiveness of the detection method on the Drain parsed data significantly better than other parsers. The p-value larger than 0.05 indicate the effectiveness of the detection method on the Drain parsed data does not significantly better than other parsers that the PA correlated closely with the effectiveness of ML-based methods on complex data (BGL) than it does on simple data (HDFS). In general, although the effectiveness of the ML-based methods is correlated with PA, almost all of them are not linear correlation. This indicates that high parsing accuracy does not definitely lead to the high effectiveness of ML-based anomaly detection methods.

(3) Table 6 shows the effectiveness of DL-based methods. We can observe that the precision, recall, and F-measure of Deeplog on BGL data are best on the IPLoM parsed data (88.90%, 95.25%, and 91.96%, respectively). The recall and F-measure of Deeplog on HDFS data are best on Logram parsed data (99.27% and 96.75%, respectively). However, Drain presents better parsing accuracy on BGL data than IPLoM and the same parsing accuracy as Logram on HDFS data. Table 7 shows the Spearman correlation between parsing accuracy (PA) and the effectiveness of DL-based methods. On HDFS and ThunderBird data, the effectiveness of all DL-based methods is not closely correlated with PA, with Spearman correlation coefficients ranging from 0.000 to 0.802. On BGL data, the precision and F-measure of Deeplog are closely correlated with PA at the 95% confidence level (p-value$<$0.05), with Spearman correlation coefficients ranging from 0.750 to 0.893. The precision, recall, and F-measure of LogRobust are not closely correlated with PA, with Spearman correlation coefficients ranging from 0.429 to 0.600. These results indicate that PA has a closer correlation with Deeplog than LogRobust on BGL. That's probably because that LogRobust can handle the unstable log events caused by parsing errors. In general, the effectiveness of the DL-based methods is not linearly correlated with PA. This indicates that high parsing accuracy does not definitely lead to the high effectiveness of DL-based anomaly detection methods.

(4) Table 8 shows the average effectiveness of ML-based methods. Table 9 shows the results of the Wilcoxon rank-sum test with a Bonferroni correction on unsupervised ML-based methods. We verify whether IPLoM is significantly better than other log parsers for unsupervised ML-based methods by the Wilcoxon rank-sum test with a Bonferroni correction. Tables 8 and 9 show that the average precision and F-measure of PCA on IPLoM parsed data (96.98% and 74.51%, respectively) are significantly better than other log parsers at the 95% confidence level (p-value $<$ 0.05). The average recall and F-measure of LogClustering on IPLoM parsed data (89.04% and 71.88%, respectively) are significantly better than other log parsers at the 95% confidence level (p-value $<$ 0.05), except Drain. It is worth noting that the average effectiveness of LogClustering on IPLoM parsed data is not significantly better than Drain. In general, IPLoM is better than other log parsers for PCA based on the average effectiveness of PCA. Both Drain and IPLoM are better than other log parsers for LogClustering based on the average effectiveness of LogClustering. Figure 2 shows the effectiveness distribution of unsupervised ML-based methods. We can observe that the precision and F-measure of unsupervised ML-based methods are the most robust on the heuristic-based IPLoM and Drain parsed data. The recall of unsupervised ML-based methods is the most robust on the subsequence-based Spell parsed data. Considering the average effectiveness and robustness of effectiveness, IPLoM is more suitable for PCA, and both IPLoM and Drain are more suitable for LogClustering.

(5) Table 10 shows the results of the Wilcoxon rank-sum test with a Bonferroni correction on supervised ML-based methods. We verify whether IPLoM is significantly better than other parsers for supervised ML-based methods by the Wilcoxon rank-

sum test with a Bonferroni correction. Tables 8 and 10 show the average recall and F-measure of LR on IPLoM parsed data (85.63% and 89.43%, respectively) are significantly better than other log parsers at 95% confidence level (p-value < 0.05), except Drain. The average precision and F-measure of DT on IPLoM parsed data (90.36% and 88.31%, respectively) are significantly better than other log parsers at the 95% confidence level (p-value < 0.05), except Drain. The average effectiveness of LR and DT on IPLoM parsed data is not significantly better than Drain. This indicates that both Drain and IPLoM are better than other log parsers for supervised ML-based methods according to the average effectiveness of supervised ML-based methods. Figure 3 shows the effectiveness distribution of supervised ML-based methods. We can observe that the precision, recall, and F-measure of supervised ML-based methods are the most robust on the heuristic-based IPLoM and Drain parsed data, and that is the least robust on the clustering-based Lenma parsed data, followed by the frequency-based Logram. Considering the average effectiveness and robustness of effectiveness, both IPLoM and Drain are more suitable for supervised ML-based methods.

(6) Table 12 shows the results of the Wilcoxon rank-sum test with a Bonferroni correction on DL-based methods. We verify whether IPLoM is significantly better than other parsers for Deeplog and Drain is significantly better than other parsers for LogRobust by the Wilcoxon rank-sum test with a Bonferroni correction. Tables 11 and 12 show the average precision, recall, and F-measure of Deeplog on IPLoM parsed data (93.67%, 93.14%, and 93.33%, respectively) are significantly better than other parsers at the 95% confidence level (p-value < 0.05). The average precision and F-measure of LogRobust on Drain parsed data (89.98% and 93.98%, respectively) are significantly better than other parsers at the 95% confidence level (p-value < 0.05). This indicates that IPLoM is better than other log parsers for Deeplog, and Drain is better than other log parsers for LogRobust according to the average effectiveness of DL-based methods. As shown in Fig. 4, we can observe that the precision of DL-based methods is the most robust on the heuristic-based Drain parsed data, followed by IPLoM. That is the least robust on the frequency-based Logram parsed data, followed by the clustering-based Lenma. The recall of DL-based methods is the most robust on the heuristic-based IPLoM parsed data, followed by clustering-based Lenma. The robustness of the F-measure of DL-based methods on Drain and IPLoM parsed data are comparable. For the average effectiveness and robustness, IPLoM is more suitable for Deeplog, and Drain is more suitable for LogRobust.

Finding 1: High parsing accuracy does not imply high anomaly detection effectiveness. Heuristic-based log parsers are more suitable for anomaly detection than other types of log parsers. In detail, IPLoM is more suitable for PCA. Both IPLoM and Drain are suitable for LogClustering and supervised ML-based methods (LR and DT). For the DL-based method, IPLoM is more suitable for Deeplog, while Drain is more suitable for LogRobust.

4.2 RQ2: What is the Impact of Log Parsing Errors and the Number of Parsing Event Templates on Anomaly Detection?

Motivation As shown in Section 4.1, we observe that high parsing accuracy does not imply high anomaly detection effectiveness. At the same time, we also found that some log parsers have the same parsing incorrect event, but they parse the same event into different numbers of event templates. For example, Drain parses the E1 of BGL to 686 event templates, while LFA parses it to 416 event templates; IPLoM parses the E164 of BGL to 337 event templates, while Spell parses it to 381 event templates. Therefore, we explore the impact of parsing errors and the number of parsing event templates on the effectiveness of anomaly detection methods. Do parsing errors definitely have a negative impact on the effectiveness of anomaly detection methods? We explore the impact caused by parsing errors.

Methods To explore the impact of parsing errors on the effectiveness of anomaly detection methods, we analyze the detection results of all detection methods on the six log parsers parsed HDFS and BGL data. We treat the detection results on Ground Truth of HDFS and BGL data as the baseline. In the case that the dataset and the detection method are consistent, it can be considered that the inconsistency between the detection results of the detection method on the parser parsed data and the detection results on the Ground Truth is caused by parsing errors. Therefore, the **impact** can be defined as the ratio of the number of inconsistent detection results caused by parsing errors to the number of event subsequences in test data. The more inconsistent the detection results, the greater the impact of parsing errors. To explore the impact caused by parsing errors, we deeply analyze the inconsistency between the detection results of the detection method on the parser parsed data and the detection results on the Ground Truth. If an event subsequence contains parsing errors, while the event subsequence is detected correctly, it can be considered that the positive impact is caused by parsing errors. Therefore, the **positive impact** can be defined as the ratio of the number of correct detection results caused by parsing errors to the number of inconsistent detection results caused by parsing errors. If an event subsequence contains parsing errors and the event subsequence is detected incorrectly, it can be considered that the negative impact is caused by parsing errors. Therefore, the **negative impact** can be defined as the ratio of the number of incorrect detection results caused by parsing errors to the number of inconsistent detection results caused by parsing errors.

To explore the number of parsing event templates on the effectiveness of anomaly detection methods, we run four ML-based methods and two DL-based methods on the processed HDFS and BGL data. Firstly, we find the log events that is parsed to the most event templates by the six log parsers in the Ground Truth of HDFS and BGL. Secondly, we merged the event templates belonging to the same event into two event templates without improving the parsing accuracy according to the definition of parsing accuracy, called processed parsed data. Thirdly, we run four ML-based methods and two DL-based methods on the processed HDFS and BGL data. At last, we conduct the Wilcoxon rank-sum test at the 95% confidence level ($p\text{-value} < 0.05$) to analyze the statistically significant difference in the effectiveness of anomaly detection methods on the processed data and the original parsed data.

Results Tables 13 and 14 show the impact of parsing errors on ML-based and DL-based methods, respectively. Figures 5 and 6 show the positive impact and negative impact of parsing errors on ML-based and DL-based methods, respectively. Tables 15 and 16 show the summary of the impact of the number of parsed event templates on the effectiveness of

Table 13 The impact of parsing errors of six log parsers on ML-based anomaly detection methods

Impact on method	Drain (%)	IPLoM (%)	LFA (%)	Logram (%)	Lenma (%)	Spell (%)	Average (%)
Impact on PCA	0.12	0.00	0.15	1.41	0.51	0.50	0.45
Impact on LogClustering	0.01	0.01	0.66	14.64	0.03	0.04	2.56
Impact on LR	0.02	0.02	0.24	0.10	0.11	0.05	0.09
Impact on DT	0.06	0.06	0.26	0.15	0.12	0.05	0.12
Average	0.05	0.02	0.33	4.08	0.19	0.16	–

Table 14 The impact of parsing errors of six log parsers on DL-based anomaly detection methods

Impact on method	Drain (%)	IPLoM (%)	LFA (%)	Logram (%)	Lenma (%)	Spell (%)	Average (%)
Impact on Deeplog	1.88	1.55	3.00	13.66	15.00	5.20	6.71
Impact on LogRobust	0.09	0.12	0.22	0.23	0.17	0.20	0.17
Average	0.98	0.84	1.61	6.94	7.59	2.70	–

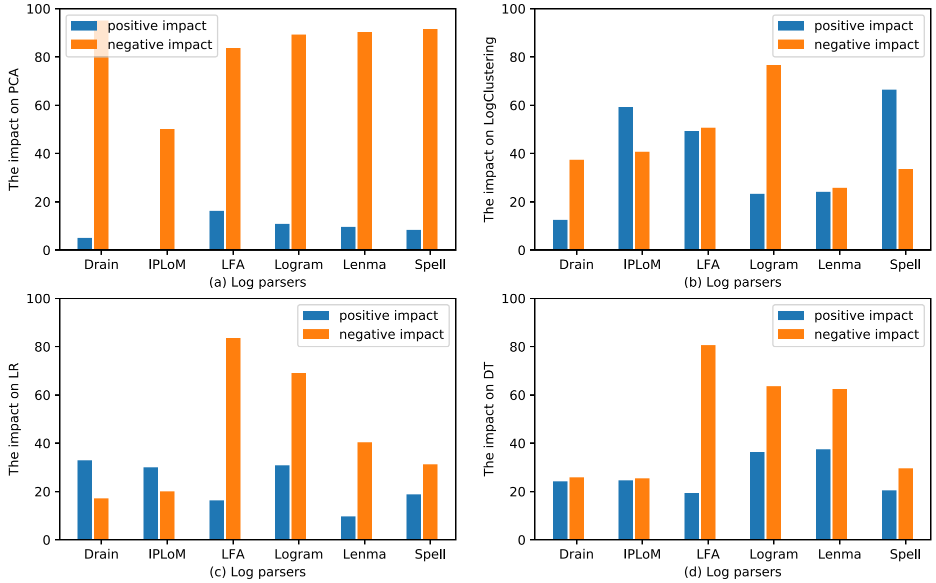


Fig. 5 The impact of parsing errors on the effectiveness of ML-based methods

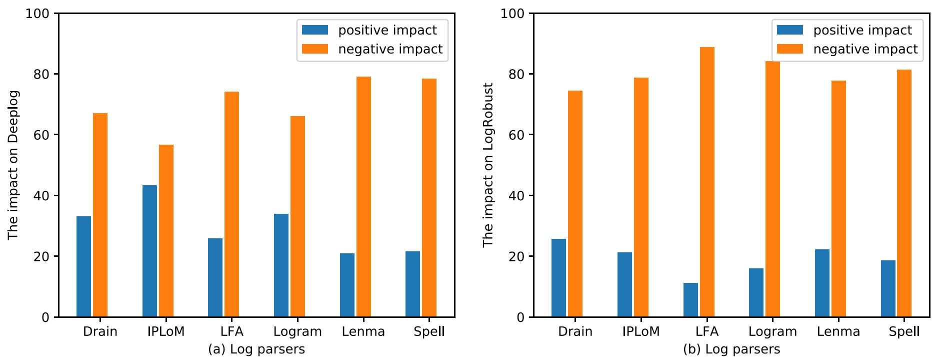


Fig. 6 The impact of parsing errors on the effectiveness of DL-based methods

ML-based methods and DL-based methods, respectively. More detailed results are shown in Appendix A. From the results shown in Tables 13, 14, 15, 16, Figs. 5, and 6, we make the following observations:

(1) Table 13 shows the average inconsistency percentage of ML-based methods on log parsers parsed data (HDFS and BGL). The greater the average inconsistency percentage of an anomaly detection method is, the more significant the impact of the parsing errors on the effectiveness of the anomaly detection method is. It can be observed from Table 13, which shows that IPLoM has the smallest average impact (0.02%) on ML-based methods, followed by Drain (0.05%). Logram has the greatest average impact (4.08%) on ML-based methods, followed by LFA (0.33%). These indicate that heuristic-base log parsers (IPLoM and Drain) have the least impact on the ML-based anomaly detection methods, and frequency-based log parsers (Logram and LFA) have the greatest impact on the ML-based anomaly detection methods. The average impact of six log parsers on unsupervised ML-based methods (PCA (0.45%) and LogClustering (2.56%)) is greater than that on supervised ML-based methods (LR (0.09%) and DT (0.12%)). This indicates that supervised ML-based methods are more robust to parsing errors than unsupervised ML-based methods.

(2) Table 14 shows the average inconsistency percentage of DL-based methods on log parsers parsed data (HDFS and BGL). It can be observed from Table 14 that IPLoM has the smallest average impact (0.84%) on DL-based methods, followed by Drain (0.98%). Lenma has the greatest average impact (7.59%) on DL-based methods, followed by Logram (6.94%). These indicate that heuristic-based log parsers (IPLoM and Drain) have the smallest impact on the DL-based anomaly detection methods, and clustering-based log parser (Lenma) has the greatest impact on the DL-based anomaly detection methods. The average impact of six log parsers on Deeplog (6.71%) is greater than that on LogRobust (0.17%). This indicates that LogRobust is more robust to parsing errors than Deeplog. That's probably because that LogRobust can handle unstable log events, including the unstable log events caused by parsing errors.

(3) We can observe from Fig. 5, which shows that parsing errors have not only a negative impact but also a positive impact on the effectiveness of ML-based methods. And the parsing errors of different log parsers have a different impact on the effectiveness of

Table 15 Summary of the impact of the number of parsed event templates on the effectiveness of ML-based methods

Parser	ML on HDFS			ML on BGL		
	Significant different in P	Significant different in R	Significant different in F1	Significant different in P	Significant different in R	Significant different in F1
Drain	0	0	0	2	2	3
IPLoM	0	0	0	3	4	4
LFA	3	3	3	4	4	3
Logram	3	3	2	4	4	3
Lenma	0	0	0	3	4	4
Spell	0	0	0	1	2	1

The value presents the number of ML-based methods that the effectiveness on the processed parsed data is significantly different to that on the original parsed data

ML-based methods. In most cases, the negative impact of parsing errors on the effectiveness of ML-based methods is greater than the positive impact. However, in only a few cases, the positive impact of parsing errors on the effectiveness of ML-based methods is greater than the negative impact. For example, the impact of parsing errors of IPLoM on LogClustering (The positive and negative impacts are 59.23% and 40.77%, respectively) and the impact of parsing errors of Drain on LR (The positive and negative impacts are 32.86% and 17.14%, respectively). This is why the effectiveness of ML-based methods is better on the log parser data than on Ground Truth in a few cases. The parsing errors of all log parsers have a significantly greater negative impact than positive impact on PCA. This indicates that the parsing errors are more likely to degrade the effectiveness of PCA. The positive impact caused by the parsing errors of IPLoM and Drain is close to or slightly higher than the negative impact on supervised ML-based methods. As Table 13 shows that heuristic-based log parsers (IPLoM and Drain) have the smallest impact on the supervised ML-based methods. This is why heuristic-based log parsers are more suitable for supervised ML-based methods.

Table 16 Summary of the impact of the number of parsed event templates on the effectiveness of DL-based methods

Parser	DL on HDFS			DL on BGL		
	Significant different in P	Significant different in R	Significant different in F1	Significant different in P	Significant different in R	Significant different in F1
Drain	1	1	1	2	1	2
IPLoM	0	0	0	2	1	2
LFA	1	1	1	2	2	2
Logram	1	2	2	2	2	2
Lenma	1	2	1	2	2	2
Spell	0	0	0	2	2	2

The value presents the number of DL-based methods that the effectiveness on the processed parsed data is significantly different to that on the original parsed data

(4) We can observe from Fig. 6, which shows that parsing errors have not only a negative impact but also a positive impact on the effectiveness of DL-based methods. The parsing errors of different log parsers have a different impact on the effectiveness of anomaly detection. The negative impact of parsing errors on the effectiveness of DL-based methods is greater than the positive impact. The positive impact caused by the parsing errors of IPLoM is closer to the negative impact on Deeplog than other log parsers. Table 14 shows that IPLoM has the smallest impact on Deeplog. This is why IPLoM is more suitable for Deeplog. Similarly, Drain has the smallest impact on LogRobust, and the positive impact caused by the parsing errors of Drain is closer to the negative impact on LogRobust than other log parsers. This is why Drain is more suitable for LogRobust.

Finding 2: Parsing errors of log parsers have not only a negative impact but also a positive impact on the effectiveness of anomaly detection methods.

(5) Table 15 shows the summary of the impact of the number of parsed event templates on the effectiveness of ML-based methods. The column of ML-based methods on HDFS shows the number of ML-based methods that the effectiveness on processed HDFS data is significantly different from the original HDFS data. It is worth noting that the processed HDFS data only reduces the number of templates belonging to the same event without changing the parsing accuracy. The column of ML-based methods on BGL is similar to the column of ML-based methods on HDFS. We can observe from Table 15, which shows that there are at least two ML-based methods' effectiveness on the processed parsed HDFS data of LFA and Logram significantly different from the effectiveness on original parsed HDFS data of them. There is no ML-based method' effectiveness on the processed parsed HDFS data of the other four log parsers (Drain, IPLoM, Lenma, and Spell) significantly different from the effectiveness on original parsed HDFS data of them. This may be because that the number of templates parsed by these four log parsers on HDFS data is close to the Ground Truth of HDFS, so the processing of merged templates has no significant impact on the effectiveness of ML-based methods. There is at least one ML-based method's effectiveness on the processed parsed BGL data of six log parsers significantly different from the effectiveness on original parsed BGL data of them. The impact of the number of parsed event templates on the effectiveness of ML-based methods is more evident on BGL than that on HDFS. This may be because the difference in the number of parsing event templates on BGL is greater than that of the HDFS. The results of the impact of the number of parsed event templates on the effectiveness of ML-based methods indicate that the number of parsed templates has an impact on the effectiveness of ML-based methods, especially on complex data.

(6) Table 16 shows the summary of the impact of the number of parsed event templates on the effectiveness of DL-based methods. We can observe from Table 16, which shows that there are one or two DL-based methods' effectiveness on the processed parsed HDFS data of four log parsers (Drain, LFA, Logram, and Lenma) is significantly different from the effectiveness on original parsed HDFS data of them. The effectiveness of two DL-based methods on the processed parsed BGL data of all six log parsers significantly different from the effectiveness on original parsed BGL data of them, except the recall of Deeplog on Drain and IPLoM parsed BGL data. The impact of the number of parsed event templates on the effectiveness of DL-based methods is more evident on BGL than that on HDFS, the same as ML-based methods. The results of the impact

of the number of parsed event templates on the effectiveness of DL-based methods indicate that the number of parsed templates has an impact on the effectiveness of DL-based methods, especially on complex data.

(7) As shown in Section 4.1, we observe that higher parsing accuracy does not imply higher effectiveness of anomaly detection methods. Such observation is also similar to the findings of He et al. (2016a). In the study of He et al., they also find that two log parsers with comparable parsing accuracy lead to the different performance of PCA, and two log parsers with different parsing accuracy lead to the comparable performance of PCA. Unlike their work, we further explore the impact of parsing errors on the effectiveness of anomaly detection methods. As shown in Figs. 5 and 6, we observe that parsing errors of log parsers have not only a negative impact but also a positive impact on the effectiveness of anomaly detection methods. This indicates that the parsing errors do not definitely lead to performance degradation. This is why high parsing accuracy does not imply the high effectiveness of anomaly detection methods. At the same time, we also found that some log parsers have the same parsing incorrect event, but they parse the same event into different numbers of event templates. We further explore the impact of the number of parsed event templates on the effectiveness of anomaly detection methods. As shown in Tables 15 and 16, the results of the impact of the number of parsed event templates on the effectiveness of ML-based methods and DL-based methods indicate that the number of parsed event templates has an impact on the effectiveness of anomaly detection methods.

Finding 3: Both parsing accuracy and the number of parsed event templates should be considered when choosing log parsers for anomaly detection.

4.3 RQ3: How do Log Parsers Impact the Efficiency of Anomaly Detection Methods?

Motivation With the generation of massive log data, the efficiency of anomaly detection methods becomes a key performance indicator. Anomaly detection methods are required to detect anomalies accurately and as quickly as possible to ensure online service quality. Do different log parsers have an impact on the efficiency of anomaly detection methods? In this question, we investigate the impact of log parsers on the efficiency of four machine-learning-based and two deep-learning-based anomaly detection methods, respectively.

Methods To ensure the consistency of the running environment when the anomaly detection method runs on all six log parsers parsing data, we run them simultaneously. Each anomaly detection method on different log parser parsed data is run five times, and the average training and testing time is reported as their efficiency.

Results Table 17 shows the number of event templates parsed by six log parsers on three datasets. Figure 7 shows the efficiency of ML-based methods, and Fig. 8 shows the efficiency of DL-based methods. From the results shown in Table 17, Figs. 7, and 8, we make the following observations:

(1) As Table 17 shown, the number of event templates parsed by the heuristic-based log parsers is more stable than that parsed by the other kinds of log parsers across different datasets. The number of event templates parsed by the clustering-based Lenma on Thunderbird data and BGL data is significantly larger than that parsed by the other

Table 17 The number of parsed event templates

Datasets	Drain	IPLoM	Lenma	Logram	LFA	Spell
HDFS	48	41	45	97	47	37
BGL	1,848	793	135,127	7,615	1,326	24,444
ThunderBird	1,008	1,002	20,630	2,077	877	692

kinds of log parsers, especially that on BGL (135,127). The number of event templates parsed by Spell and Logram on BGL data (24,444 and 7,615, respectively) is significantly larger than that of heuristic-based log parsers.

(2) As Fig. 7 shown, the number of parsed event templates impacts the efficiency of ML-based methods. With the increase in the number of parsed event templates, the efficiency of ML-based methods decreases. Since the increase in the number of parsed event templates increases the dimension of the event count matrix, the efficiency of the model decreases. The efficiency of all ML-based methods is the lowest on the data that have the largest number of parsed event templates. For example, they are the least efficient on the clustering-based Lenma parsed ThunderBird and BGL data. It is noticed that the efficiency of LogClustering on HDFS data is not shown in Fig. 7(a) due to the running time being about 500 times of other methods. It is because the efficiency of LogClustering scales non-linearly with log size.

(3) As Fig. 8 shown, the number of parsed event templates has less impact on the efficiency of DL-based methods, except Deeplog on BGL data. Deeplog handled the anomaly detection task as a multi-classification problem and used the event template index as input. When the number of parsed event templates increases within 10 times, the number of parsed event templates has little impact on the efficiency of the model. But when the number of parsed event templates increases more than 100 times, the impact of the number of parsed event templates on the efficiency of Deeplog increases significantly. This is because Deeplog performed fully connected calculations in the

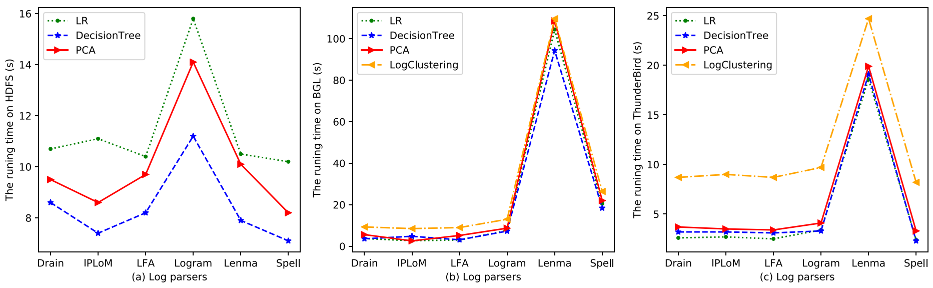


Fig. 7 The running time of ML-based anomaly detection methods (the running time of LogClustering on HDFS data is not shown in (a) as the running time of it is about 500 times of other methods)

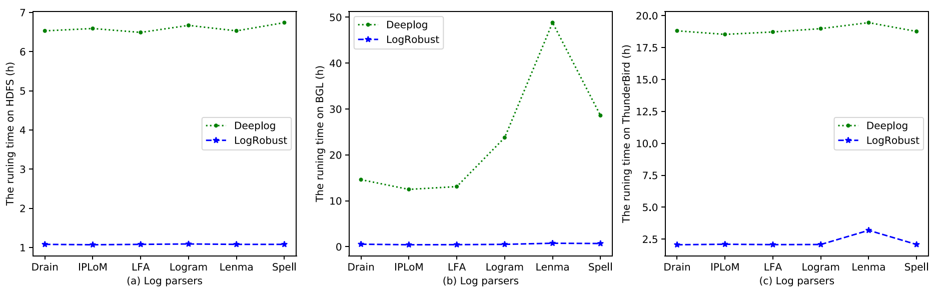


Fig. 8 The running time of DL-based anomaly detection methods

output layer. When the number of event templates increases beyond a specific range, its efficiency decreases significantly. For LogRobust, it converted the event subsequence into an event template vector list as input. Since the vector dimension of the event template and the length of the event template vector list are fixed, the efficiency of LogRobust is less affected by the number of parsed event templates.

Finding 4: The log parsers have an impact on the efficiency of anomaly detection methods. With the increase in the number of parsed event templates, the efficiency of anomaly detection decreases. The heuristic-based parsers have less impact on the efficiency of anomaly detection methods, followed by frequency-based parsers.

5 Discussion

5.1 Threats to Validity

Threats to Internal Validity Six log parsers and six anomaly detection methods are used in our study. Threats to internal validity relate to potential errors in our implementation. In order to reduce the mistakes in methods implementation, we carefully review the description of the methods in prior studies. And the implemented methods have been tested and double-checked by two graduate students to ensure the correctness of the implemented methods.

Threats to External Validity In this work, we evaluate the impact of the log parsers on anomaly detection methods on three public datasets. Although these three datasets represent two types of systems logs (Distributed system logs and Supercomputer logs), the evaluation results and findings may still be limited due to the diversity of the datasets. And our results may not be generalized to other types of datasets.

5.2 Implications

Correlation The effectiveness of anomaly detection methods is not linearly correlated with PA, and most of their effectiveness is not closely correlated whit PA. This indicates that high PA does not imply the high effectiveness of anomaly detection methods.

Impact of Parsing Errors The parsing errors of log parsers have an impact on the effectiveness of anomaly detection methods. And the parsing errors of log parsers have not only a negative impact but also a positive impact on the effectiveness of anomaly detection methods. However, in most cases, the negative impact is more significant than the positive impact. An end-to-end anomaly detection method that does not require a log parser to parse log data can be a potential improvement to avoid the impact of a log parser.

Impact of the Number of Parsed Event Templates The number of parsed event templates should be considered when evaluating the effectiveness of the log parser. Since the effectiveness of anomaly detection is impacted by both parsing accuracy and the number of parsing event templates, the previous studies of log parsers only evaluated parsing accuracy without considering the number of parsing event templates. A more comprehensive log parsing performance evaluation metric is needed in log parsing or log mining studies.

Efficiency The log parsers have an impact on the efficiency of anomaly detection methods. With the increase in the number of parsed event templates, the efficiency of anomaly detection decreases. The impact of log parsers on the efficiency of anomaly detection methods should also be considered when selecting a suitable log parser.

Recommended Log Parser All the anomaly detection methods perform more effectively and efficiently with the heuristic-based log parsers. Thus, heuristic-based log parsers are recommended for anomaly detection methods.

Others There are two main reasons why it is difficult for most anomaly detection methods, especially machine-learning-based methods, to achieve high performance: 1) Training data after cutting is highly unbalanced due to the concentration of anomaly logs; 2) log events recorded by BGL are very complex. Data sample or data synthesis technology can be adopted to improve the effectiveness of anomaly detection methods.

6 Related Work

6.1 Log Parsing

Log parsing is an important preprocessing step for automatic log analysis. It converts the raw semi-structured log messages into structured data which is used for feature extraction. Current log parsing methods can be divided into rule-based, source-code-based, and data-driven-based methods. Rule-based methods need practitioners or researchers to define heuristic rules manually (Vaarandi 2006; Damasio et al. 2002; Hansen and Atkins 1993). Rule-based methods have some limits, such as difficulty in maintaining rules and limited coverage of rules. Although some management tools were designed to support practitioners to specific customized rules (loggly 2021; Logstash 2021; Splunk 2021), they still cannot completely overcome the deficiencies of rule-based methods. Source-code-based methods require source code as input (Nagappan et al. 2009; Xu et al. 2009). For example, Xu et al. (2009) used static source code analysis technology to extract event templates in the log statement. However, source code is not always available, such as commercial components. Unlike the source-code-based methods, the data-driven methods take the log messages generated during system running as input. The current data-driven methods can be classified into four categories according to the technology it adopts: heuristic-based, frequency-based, clustering-based, and others. The heuristic-based methods (AEL (Jiang et al. 2008), Drain (He et al. 2017), and IPLoM (Makanju et al. 2011)) were mainly based on the characteristics of the log messages to define the partition conditions to group the logs, such as the length of log messages content. The frequency-based methods (SLCT (Vaarandi 2003), LFA (Nagappan and Vouk 2010), and LogCluster (Vaarandi and Pihelgas 2015)) generated log templates based on tokens frequently appearing in the log messages. Clustering-based methods (LKE (Fu et al. 2009), LogSig (Tang et al. 2011), SHISO (Mizutani 2013), Lenma (Shima 2016), and LogMine (Hamooni et al. 2016)) mainly used clustering algorithms to group logs and then generated log templates. In addition, Spell (Du and Li 2018) was based on the longest common subsequence algorithms to parse logs, and MoLFI (Messaoudi et al. 2018) was based on evolutionary algorithms to parse logs. It is worth noting that these log parsers all used log parsing accuracy as the key performance metric. However, does high parsing accuracy imply the high effectiveness of log mining? As far as we know, few of the log parser research mentioned above, except Drain, evaluated the impact of log parser on log mining tasks in their research.

At the same time, some studies are devoted to evaluating the log parsers, i.e., He et al. (2016a) evaluated four log parsers on five datasets, and Zhu et al. (2019) comprehensively evaluated 13 log parsers on 16 datasets and open-sourced the toolkit and benchmarks.

6.2 Log-Based Anomaly Detection

Log-based anomaly detection is an important application branch of automatic log analysis, and many methods have been proposed in recent years. These methods can be classified into four categories: keyword-searching-based, rule-based, traditional machine-learning-based, and deep-learning-based. In system development, developers define the log severity level when they use the log to record the detailed running status information of the system, such as “info”, “warning”, and “error”. Jiang et al. (2009) used a keyword-searching-based method to find critical events for abnormal detection. Shang et al. (2013) combined log abstraction with a keyword-searching-based method to find the abnormal log. Traditional machine-learning-based methods can be further divided into supervised and unsupervised according to whether they need labeled data for training. In the real world, the logs generated by the system are unlabeled, so unsupervised machine-learning-based methods (PCA (Xu et al. 2009), LogClustering (Lin et al. 2016), Frequent pattern Mining (Lu et al. 2018a), and Log3C (He et al. 2018b)) have more practical application significance. For example, Lin et al. (2016) proposed LogClustering and successfully applied it to many Microsoft online service systems. Supervised machine learning (Decision Tree (Chen et al. 2004) and SVM (Liang et al. 2007)) need to label the data manually or record the failure in the log as a label. Liang et al. (2007) applied SVM to predict failure events. At the same time, He et al. (2016b) evaluated six machine-learning-based anomaly detection methods on two public datasets. With the development of deep learning technology in recent years, deep-learning-based methods (Deeplog (Du et al. 2017), LogRobust (Zhang et al. 2019), LogC (Yin et al. 2020), LogAnormal (Meng et al. 2019), and CNN-based approach (Lu et al. 2018b)) have also been used for anomaly detection. Deep-learning-based methods mainly define the anomaly detection task as a multi-classification or binary classification problem and use RNN-based methods for learning and prediction. For example, Deeplog defined the anomaly detection task as a multi-classification problem, and LogRobust defined the anomaly detection task as a binary classification problem. It is worth noting that these log-based anomaly detection methods do not evaluate the impact of different log parsers on their methods.

7 Conclusion

In this paper, we explore the impact of log parsers on anomaly detection. To that end, we conduct a comprehensive study to investigate the impact of four kinds of log parsers on two kinds of anomaly detection methods on three public datasets. Firstly, we evaluate the impact of the four kinds of log parsers on the effectiveness of two kinds of anomaly detection methods. And we conduct a Spearman rank correlation test to verify the correlation between the parsing accuracy and the effectiveness of anomaly detection methods. The results show that although the effectiveness of anomaly detection methods is correlated with parsing accuracy, almost all of them are not linear correlations. This indicates that high parsing accuracy does not imply the high effectiveness of anomaly detection methods. Secondly, we evaluate the impact of parsing errors on two kinds of anomaly detection methods. We find that the parsing errors have not only a negative impact but also a positive impact on the effectiveness of anomaly detection methods. This indicates that the parsing errors do not definitely lead to performance degradation. Thirdly, we evaluate the impact of the number of parsed event templates on the effectiveness of anomaly detection methods. We find that the number of parsed event templates has an impact on the effectiveness of anomaly detection methods. This indicates that both parsing accuracy and the number of parsed event templates should be considered when choosing log parsers for anomaly detection. Fourthly, we evaluate the impact of the log parsers on the efficiency of the two kinds of anomaly detection methods. We found that the number of parsed event templates has an impact on the efficiency of anomaly detection. With the increase in the number of parsed event templates, the efficiency of anomaly detection decrease. In detail, the heuristic-based parsers have less impact on the efficiency of anomaly detection methods, followed by frequency-based parsers. Through the results of the impact of log parsers on the effectiveness and efficiency of the anomaly detection methods, we find that the heuristic-based parsers perform more effectively and efficiently on average of all the anomaly detection methods. In detail, the heuristic-based IPLoM is more suitable for unsupervised ML-based method PCA and DL-based method Deeplog. Both heuristic-based IPLoM and Drain are more suitable for unsupervised ML-based LogClustering and supervised ML-based methods. Heuristic-based Drain is more suitable for DL-based LogRobust. We expect our work could help more researchers and practitioners on log mining research and practical applications.

We plan to enrich our study with more available system logs in the future. We expect to conduct the evaluation on more diverse data to provide more valuable findings for researchers and practitioners. Besides, more newly proposed anomaly detection methods will be considered.

Appendix A: The Impact of the Number of Parsed Event Templates on Anomaly Detection Methods' Effectiveness

Table 18 The impact of the number of parsed event templates on the ML based methods' effectiveness (on HDFS)

Parser	Original parsed HDFS data			Processed parsed HDFS data					
	PCA			PCA			p-value of Wilcoxon test		
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	99.78	67.34	80.41	99.78	67.34	80.41	>0.05	>0.05	>0.05
IPLoM	99.80	72.62	84.07	99.80	72.62	84.07	>0.05	>0.05	>0.05
LFA	99.79	70.86	82.85	99.78	69.00	81.58	>0.05	**<0.05**	**<0.05**
Logram	83.69	12.65	21.98	90.55	24.47	38.52	**<0.05**	**<0.05**	**<0.05**
Lenma	99.78	67.34	80.41	99.78	67.34	80.41	>0.05	>0.05	>0.05
Spell	99.78	67.34	80.41	99.78	67.34	80.41	>0.05	>0.05	>0.05
Parser	LogClustering			LogClustering					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	100.00	76.80	**86.88**	100.00	76.80	**86.88**	>0.05	>0.05	>0.05
IPLoM	100.00	77.05	87.04	100.00	76.90	87.04	>0.05	>0.05	>0.05
LFA	92.72	76.57	83.87	93.80	78.51	85.47	**<0.05**	**<0.05**	**<0.05**
Logram	6.75	76.38	12.41	38.10	74.92	50.51	**<0.05**	**<0.05**	**<0.05**
Lenma	100.00	76.80	86.88	100.00	76.80	86.88	>0.05	>0.05	>0.05
Spell	100.00	77.05	87.08	100.00	77.05	87.08	>0.05	>0.05	>0.05
Parser	LR			LR					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	100.00	99.41	99.70	100.00	99.41	99.70	>0.05	>0.05	>0.05
IPLoM	100.00	99.41	99.70	100.00	99.41	99.70	>0.05	>0.05	>0.05
LFA	89.21	99.18	93.93	88.69	99.08	93.60	**<0.05**	**<0.05**	**<0.05**
Logram	99.94	99.67	**99.81**	100.00	99.08	99.02	**<0.05**	**<0.05**	>0.05
Lenma	100.00	99.41	99.70	100.00	99.41	99.70	>0.05	>0.05	>0.05
Spell	100.00	99.41	99.70	100.00	99.41	99.70	>0.05	>0.05	>0.05
Parser	DT			DT					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	100.00	99.81	**99.90**	100.00	99.81	99.90	>0.05	>0.05	>0.05
IPLoM	100.00	99.81	**99.90**	100.00	99.81	99.90	>0.05	>0.05	>0.05
LFA	88.99	99.51	93.96	88.99	99.52	93.96	**<0.05**	>0.05	>0.05
Logram	100.00	99.81	**99.90**	100.00	99.81	99.90	>0.05	>0.05	>0.05
Lenma	100.00	99.81	**99.90**	100.00	99.81	99.90	>0.05	>0.05	>0.05
Spell	100.00	99.81	**99.90**	100.00	99.81	99.90	>0.05	>0.05	>0.05

The p-value < 0.05 is highlighted in bold. The p-value < 0.05 indicates that the effectiveness of detection method on the processed parsed data is significantly different to that on the original parsed data

Table 19 The impact of the number of parsed event templates on the ML based methods' effectiveness (on BGL)

Parser	Original parsed BGL data			Processed parsed BGL data			p-value of Wilcoxon test		
	PCA			PCA					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	65.85	45.92	54.11	66.18	45.92	54.22	**<0.05**	>0.05	**<0.05**
IPLoM	95.62	44.56	60.79	95.62	44.56	60.79	**<0.05**	**<0.05**	**<0.05**
LFA	43.22	46.60	44.84	92.05	47.28	62.47	**<0.05**	**<0.05**	**<0.05**
Logram	28.44	98.30	44.12	28.94	94.22	44.28	**<0.05**	**<0.05**	**<0.05**
Lenma	28.52	100.00	44.38	29.56	91.50	44.68	**<0.05**	**<0.05**	**<0.05**
Spell	27.56	94.22	42.65	27.56	94.22	42.65	>0.05	>0.05	>0.05
Parser	LogClustering			LogClustering					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	37.02	91.99	**52.80**	37.76	93.70	**53.83**	**<0.05**	**<0.05**	**<0.05**
IPLoM	37.43	93.41	53.45	36.75	91.77	52.48	**<0.05**	**<0.05**	**<0.05**
LFA	37.31	88.23	52.44	37.16	87.55	52.17	**<0.05**	**<0.05**	**<0.05**
Logram	36.18	83.62	<u>50.51</u>	35.70	87.87	<u>50.77</u>	**<0.05**	**<0.05**	**<0.05**
Lenma	36.82	87.67	51.86	36.99	91.07	52.61	**<0.05**	**<0.05**	**<0.05**
Spell	36.88	89.85	52.30	37.87	92.23	53.69	**<0.05**	**<0.05**	**<0.05**
Parser	LR			LR					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	87.16	61.70	72.20	87.26	62.93	73.12	>0.05	**<0.05**	**<0.05**
IPLoM	86.26	61.02	71.45	88.32	59.80	71.29	**<0.05**	**<0.05**	**<0.05**
LFA	70.11	59.05	<u>64.08</u>	69.27	60.32	64.48	**>0.05**	**<0.05**	**>0.05**
Logram	87.22	26.60	**40.75**	49.49	47.30	48.36	**<0.05**	**<0.05**	**<0.05**
Lenma	54.81	17.76	26.82	100.00	36.39	53.37	**<0.05**	**<0.05**	**<0.05**
Spell	92.53	42.86	58.57	92.20	42.82	58.48	>0.05	>0.05	>0.05
Parser	DT			DT					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	73.45	64.42	**68.63**	74.03	64.96	69.20	>0.05	>0.05	>0.05
IPLoM	74.47	63.27	**68.41**	74.88	62.28	67.99	>0.05	**<0.05**	**<0.05**
LFA	67.38	58.57	<u>62.66</u>	66.45	57.22	61.49	**<0.05**	**<0.05**	**<0.05**
Logram	43.30	50.48	**46.59**	47.06	58.69	52.23	**<0.05**	**<0.05**	**<0.05**
Lenma	51.12	51.09	**51.10**	51.03	59.77	55.06	>0.05	**<0.05**	**<0.05**
Spell	72.87	57.28	**64.13**	72.67	56.58	63.62	>0.05	**<0.05**	>0.05

The p-value < 0.05 is highlighted in bold. The p-value < 0.05 indicates that the effectiveness of detection method on the processed parsed data is significantly different to that on the original parsed data

Table 20 The impact of the number of parsed event templates on the DL based methods' effectiveness (on HFDS)

Parser	Original parsed HDFS data			Processed parsed HDFS data			Wilcoxon test		
	Deeplog			Deeplog					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	95.98	94.11	95.03	96.01	93.86	94.92	>0.05	**<0.05**	>0.05
IPLoM	95.81	93.68	94.73	95.94	93.96	94.94	>0.05	>0.05	>0.05
LFA	96.48	91.22	93.78	96.90	90.75	93.72	**<0.05**	>0.05	>0.05
Logram	94.36	99.27	96.75	94.18	98.16	96.13	**<0.05**	**<0.05**	**<0.05**
Lenma	95.93	94.10	95.01	95.57	94.75	95.16	>0.05	**<0.05**	>0.05
Spell	95.71	93.11	94.39	95.75	93.15	94.43	>0.05	>0.05	>0.05
Parser	LogRobust			LogRobust					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	96.92	99.83	**98.35**	96.52	99.88	**98.17**	**<0.05**	>0.05	**<0.05**
IPLoM	97.11	99.47	98.28	96.71	99.86	98.26	>0.05	>0.05	>0.05
LFA	96.67	99.92	98.27	96.88	97.82	97.34	>0.05	**<0.05**	**<0.05**
Logram	94.87	99.81	<u>97.27</u>	95.76	99.63	<u>97.66</u>	**<0.05**	>0.05	**<0.05**
Lenma	95.93	99.98	97.91	97.30	99.78	98.52	**<0.05**	**<0.05**	**<0.05**
Spell	93.93	97.35	95.61	93.95	97.35	95.62	>0.05	>0.05	>0.05

The p-value < 0.05 is highlighted in bold. The p-value < 0.05 indicates that the effectiveness of detection method on the processed parsed data is significantly different to that on the original parsed data

Table 21 The impact of the number of parsed event templates on the DL based methods' effectiveness (on BGL)

Parser	Original parsed BGL data			Processed parsed BGL data			p-value of Wilcoxon test		
	Deeplog			Deeplog					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	85.88	78.01	81.76	88.72	78.12	83.08	**<0.05**	>0.05	**<0.05**
IPLoM	88.90	95.25	91.96	89.60	95.17	92.30	**<0.05**	>0.05	**<0.05**
LFA	76.79	73.02	74.86	79.33	72.25	75.62	**<0.05**	**<0.05**	**<0.05**
Logram	19.31	61.71	29.42	22.83	81.11	35.63	**<0.05**	**<0.05**	**<0.05**
Lenma	21.13	83.78	33.75	22.84	75.79	35.10	**<0.05**	**<0.05**	**<0.05**
Spell	44.70	82.42	57.96	52.66	83.01	64.44	**<0.05**	**<0.05**	**<0.05**
Parser	LogRobust			LogRobust					
	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	p-value on P	p-value on R	p-value on F1
Drain	76.36	96.63	85.31	87.58	86.12	86.85	**<0.05**	**<0.05**	**<0.05**
IPLoM	71.94	97.44	82.77	69.68	98.25	81.53	**<0.05**	**<0.05**	**<0.05**
LFA	52.39	82.62	64.12	60.80	74.31	66.88	**<0.05**	**<0.05**	**<0.05**
Logram	61.49	94.68	74.62	61.98	95.92	75.30	**<0.05**	**<0.05**	**<0.05**
Lenma	69.47	95.69	80.50	72.27	97.94	83.17	**<0.05**	**<0.05**	**<0.05**
Spell	68.19	96.77	80.00	72.21	97.71	83.03	**<0.05**	**<0.05**	**<0.05**

The p-value < 0.05 is highlighted in bold. The p-value < 0.05 indicates that the effectiveness of detection method on the processed parsed data is significantly different to that on the original parsed data

Acknowledgments This work was supported in part by the National Key Research and Development Project (No. 2018YFB2101200), the Fundamental Research Funds for the Central Universities (No. 2022CDJKYJH001), the Natural Science Foundation of Chongqing (No. cstc2021jcyj-msxmX0538), the National Natural Science Foundation of China (No. 62002034) and the Postdoctoral Foundation of Chongqing (No. 2020LY13).

Declarations

Conflict of Interest The authors have no conflict of interest.

References

Abdi H et al (2007) Bonferroni and šidák corrections for multiple comparisons. Encyclopedia of Measurement and Statistics 3:103–107

Babenko A, Mariani L, Pastore F (2009) Ava: automated interpretation of dynamically detected anomalies. In: Proceedings of the eighteenth international symposium on software testing and analysis, pp 237–248

Berrocal E, Yu L, Wallace S, Papka ME, Lan Z (2014) Exploring void search for fault detection on extreme scale systems. In: 2014 IEEE International conference on cluster computing (CLUSTER). IEEE, pp 1–9

Bodik P, Goldszmidt M, Fox A, Woodard DB, Andersen H (2010) Fingerprinting the datacenter: automated classification of performance crises. In: Proceedings of the 5th European conference on computer systems, pp 111–124

Breier J, Branišová J (2015) Anomaly detection from log files using data mining techniques. In: Information science and applications. Springer, pp 449–457

Chen AR (2019) An empirical study on leveraging logs for debugging production failures. In: 2019 IEEE/ACM 41st international conference on software engineering: companion proceedings (ICSE-C). IEEE, pp 126–128

Chen M, Zheng AX, Lloyd J, Jordan MI, Brewer E (2004) Failure diagnosis using decision trees. In: International conference on autonomic computing, 2004. Proceedings. IEEE, pp 36–43

Chen Y, Yang X, Lin Q, Zhang H, Gao F, Xu Z, Dang Y, Zhang D, Dong H, Xu Y et al (2019) Outage prediction and diagnosis for cloud service systems. In: The world wide web conference, pp 2659–2665

Dai H, Li H, Chen CS, Shang W, Chen TH (2020) Logram: efficient log parsing using n-gram dictionaries. IEEE Trans Softw Eng

Damasio CV, Fröhlich P, Nejdl W, Pereira LM, Schroeder M (2002) Using extended logic programming for alarm-correlation in cellular phone networks. Appl Intell 17(2):187–202

Dang Y, Lin Q, Huang P (2019) Aiops: real-world challenges and research innovations. In: 2019 IEEE/ACM 41st international conference on software engineering: companion proceedings (ICSE-C). IEEE, pp 4–5

Du M, Li F (2018) Spell: online streaming parsing of large unstructured system logs. IEEE Trans Knowl Data Eng 31(11):2213–2227

Du M, Li F, Zheng G, Srikumar V (2017) 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

El-Sayed N, Zhu H, Schroeder B (2017) Learning from failure across multiple clusters: a trace-driven approach to understanding, predicting, and mitigating job terminations. In: 2017 IEEE 37th international conference on distributed computing systems (ICDCS). IEEE, pp 1333–1344

Fu Q, Lou JG, Wang Y, Li J (2009) Execution anomaly detection in distributed systems through unstructured log analysis. In: 2009 Ninth IEEE international conference abstracting log lines to log event types for mining software system logsce on data mining. IEEE, pp 149–158

Hamooni H, Debnath B, Xu J, Zhang H, Jiang G, Mueen A (2016) Logmine: fast pattern recognition for log analytics. In: Proceedings of the 25th ACM international on conference on information and knowledge management, pp 1573–1582

Hansen SE, Atkins ET (1993) Automated system monitoring and notification with swatch. In: LISA, vol 93, pp 145–152

He P, Zhu J, He S, Li J, Lyu MR (2016a) An evaluation study on log parsing and its use in log mining. In: 2016 46th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, pp 654–661

He S, Zhu J, He P, Lyu MR (2016b) Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE), IEEE, pp 207–218

He P, Zhu J, Zheng Z, Lyu MR (2017) Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International conference on web services (ICWS). IEEE, pp 33–40

He S, Lin Q, Lou JG, Zhang H, Lyu MR, Zhang D (2018a) Identifying impactful service system problems via log analysis. In: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 60–70

He S, Lin Q, Lou JG, Zhang H, Lyu MR, Zhang D (2018b) Identifying impactful service system problems via log analysis. In: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 60–70

Huang P, Guo C, Lorch JR, Zhou L, Dang Y (2018) Capturing and enhancing in situ system observability for failure detection. In: 13th Symposium on operating systems design and implementation (OSDI), pp 1–16

Jia T, Chen P, Yang L, Li Y, Meng F, Xu J (2017) An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services. In: 2017 IEEE International conference on web services (ICWS). IEEE, pp 25–32

Jiang ZM, Hassan AE, Flora P, Hamann G (2008) Abstracting execution logs to execution events for enterprise applications (short paper). In: 2008 The eighth international conference on quality software. IEEE, pp 181–186

Jiang W, Hu C, Pasupathy S, Kanevsky A, Li Z, Zhou Y (2009) Understanding customer problem troubleshooting from storage system logs. In: Proccedings of the 7th conference on file and storage technologies, pp 43–56

Liang Y, Zhang Y, Xiong H, Sahoo R (2007) Failure prediction in ibm bluegene/l event logs. In: Seventh IEEE international conference on data mining (ICDM 2007). IEEE, pp 583–588

Lin Q, Zhang H, Lou JG, Zhang Y, Chen X (2016) Log clustering based problem identification for online service systems. In: 2016 IEEE/ACM 38th international conference on software engineering companion (ICSE-c). IEEE, pp 102–111

Lin Q, Hsieh K, Dang Y, Zhang H, Sui K, Xu Y, Lou JG, Li C, Wu Y, Yao R et al (2018) Predicting node failure in cloud service systems. In: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 480–490

Liu F, Wen Y, Zhang D, Jiang X, Xing X, Meng D (2019) Log2vec: a heterogeneous graph embedding based approach for detecting cyber threats within enterprise. In: Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pp 1777–1794

loggly (2021) [EB/OL]. https://www.loggly.com/blog/

Logstash (2021) [EB/OL]. https://logz.io

Lou JG, Fu Q, Yang S, Xu Y, Li J (2010) Mining invariants from console logs for system problem detection. In: USENIX Annual technical conference, pp 1–14

Lu J, Li F, Li L, Feng X (2018a) Cloudraid: hunting concurrency bugs in the cloud via log-mining. In: Proceedings of the 2018 26th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 3–14

Lu S, Wei X, Li Y, Wang L (2018b) Detecting anomaly in big data system logs using convolutional neural network. In: 2018 IEEE 16th international conference on dependable, autonomic and secure computing, 16th international conference on pervasive intelligence and computing, 4th international conference on big data intelligence and computing and cyber science and technology congress (DASC/picom/datacom/cyberscitech). IEEE, pp 151–158

Makanju A, Zincir-Heywood AN, Milios EE (2011) A lightweight algorithm for message type extraction in system application logs. IEEE Trans Knowl Data Eng 24(11):1921–1936

Meng W, Liu Y, Zhu Y, Zhang S, Pei D, Liu Y, Chen Y, Zhang R, Tao S, Sun P et al (2019) Loganomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, vol 7, pp 4739–4745

Messaoudi S, Panichella A, Bianculli D, Briand L, Sasnauskas R (2018) A search-based approach for accurate identification of log message formats. In: 2018 IEEE/ACM 26th international conference on program comprehension (ICPC). IEEE, pp 167–16,710

Mizutani M (2013) Incremental mining of system log format. In: 2013 IEEE International conference on services computing. IEEE, pp 595–602

Nagappan M, Vouk MA (2010) Abstracting log lines to log event types for mining software system logs. In: 2010 7th IEEE working conference on mining software repositories (MSR 2010). IEEE, pp 114–117

Nagappan M, Wu K, Vouk MA (2009) Efficiently extracting operational profiles from execution logs using suffix arrays. In: 2009 20th International symposium on software reliability engineering (ISSRE). IEEE, pp 41–50

Nandi A, Mandal A, Atreja S, Dasgupta GB, Bhattacharya S (2016) Anomaly detection using program control flow graph mining from execution logs. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp 215–224

Oliner A, Stearley J (2007) What supercomputers say: a study of five system logs. In: 37th Annual IEEE/IFIP international conference on dependable systems and networks (DSN). IEEE, pp 575–584

Shang W, Jiang ZM, Hemmati H, Adams B, Hassan AE, Martin P (2013) Assisting developers of big data analytics applications when deploying on hadoop clouds. In: 2013 35th International conference on software engineering (ICSE). IEEE, pp 402–411

Shima K (2016) Length matters: clustering system log messages using length of words. arXiv:161103213

Splunk (2021) [EB/OL]. https://docs.splunk.com/Documentation/Splunk/7.3.1/_Knowledge/AboutSplunkregularexpressions/

Tang L, Li T, Perng CS (2011) Logsig: generating system events from raw textual logs. In: Proceedings of the 20th ACM international conference on Information and knowledge management, pp 785–794

Vaarandi R (2003) A data clustering algorithm for mining patterns from event logs. In: Proceedings of the 3rd IEEE workshop on IP operations & management (IPOM). IEEE, pp 119–126

Vaarandi R (2006) Simple event correlator for real-time security log monitoring. Hakin9 Magazine 1(6):28–39

Vaarandi R, Pihelgas M (2015) Logcluster-a data clustering and pattern mining algorithm for event logs. In: 2015 11th International conference on network and service management (CNSM). IEEE, pp 1–7

Wilcoxon F (1992) Individual comparisons by ranking methods. In: Breakthroughs in statistics. Springer, pp 196–202

Xia B, Bai Y, Yin J, Li Y, Xu J (2020) Loggan: a log-level generative adversarial network for anomaly detection using permutation event modeling. Inf Syst Front 1–14

Xu W, Huang L, Fox A, Patterson D, Jordan MI (2009) Detecting large-scale system problems by mining console logs. In: Proceedings of the ACM SIGOPS 22nd symposium on operating systems principles, pp 117–132

Yin K, Yan M, Xu L, Xu Z, Li Z, Yang D, Zhang X (2020) Improving log-based anomaly detection with component-aware analysis. In: 2020 IEEE International conference on software maintenance and evolution (ICSME). IEEE, pp 667–671

Yuan D, Mai H, Xiong W, Tan L, Zhou Y, Pasupathy S (2010) Sherlog: error diagnosis by connecting clues from run-time logs. In: Proceedings of the fifteenth international conference on architectural support for programming languages and operating systems, pp 143–154

Zar JH (2005) Spearman rank correlation. Encyclopedia of biostatistics 7

Zhang X, Xu Y, Lin Q, Qiao B, Zhang H, Dang Y, Xie C, Yang X, Cheng Q, Li Z et al (2019) 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

Zhou X, Peng X, Xie T, Sun J, Ji C, Liu D, Xiang Q, He C (2019) Latent error prediction and fault localization for microservice applications by learning from system trace logs. In: Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, pp 683–694

Zhu J, He S, Liu J, He P, Xie Q, Zheng Z, Lyu MR (2019) Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st international conference on software engineering: software engineering in practice (ICSE-SEIP). IEEE, pp 121–130

Ying Fu received the B.S. and M.S. degrees from Chongqing University, China, where she is currently pursuing the Ph.D. degree at the School of Big Data & Software Engineering. Her current research focuses on data mining and analyzing.

Meng Yan is now a Research Professor at the School of Big Data & Software Engineering, Chongqing University, China. He received Ph.D. degree in June 2017 under the supervision of Prof. Xiaohong Zhang from Chongqing University, China. His current research focuses on how to improve developers' productivity, how to improve software quality, and how to reduce the effort during software development by analyzing rich software repository data.

Zhou Xu is currently an Assistant Professor at the School of Big Data and Software Engineering, Chongqing University, Chongqing, China. He received the dual Ph.D. degree from the School of Computer Science, Wuhan University, China, and the Department of Computing, The Hong Kong Polytechnic University, Hong Kong. His research interests include feature engineering and data mining.

Xin Xia is the director of the software engineering application technology lab, Huawei, China. Prior to joining Huawei, he was an ARC DECRA Fellow and a lecturer at Monash University, Australia. Xin received his Ph.D in computer science from Zhejiang University in 2014. To help developers and testers improve their productivity, his current research focuses on mining and analyzing rich data in software repositories to uncover interesting and actionable information. More information at: https://xinxia.github.io/.

Xiaohong Zhang is currently a Professor and the Vice Dean of the School of Big Data and Software Engineering, Chongqing University. He received the M.S. degree in applied mathematics and the Ph.D. degree in computer software and theory from Chongqing University, China, in 2006. His current research interests include data mining of software engineering, topic modeling, image semantic analysis, and video analysis.

Dan Yang is currently the President of Southwest Jiaotong University. He is also a Professor with the School of Big Data and Software Engineering, Chongqing University. He received the B.S. degree in automation, the M.S. degree in applied mathematics, and the Ph.D. degree in machinery manufacturing and automation from Chongqing University, Chongqing. From 1997 to 1999, he held a postdoctoral position at the University of ElectroCommunications, Tokyo, Japan. His research interests include computer vision, image processing, pattern recognition, software engineering, and scientific computing.

Affiliations

Ying Fu[1,2] · **Meng Yan**[1,2] · **Zhou Xu**[1,2] · **Xin Xia**[3] · **Xiaohong Zhang**[1,2] · **Dan Yang**[1,2]

Ying Fu
fuying@cqu.edu.cn

Zhou Xu
zhouxullx@cqu.edu.cn

Xin Xia
xin.xia@acm.org

Xiaohong Zhang
xhongz@cqu.edu.cn

1 Key Laboratory of Dependable Service Computing in Cyber Physical Society (Chongqing University), Ministry of Education, Chongqing, China

2 School of Big Data and Software Engineering, Chongqing University, Chongqing, China

3 Software Engineering Application Technology Lab, Huawei, Hangzhou, China