LogClass: Anomalous Log Identification and Classification With Partial Labels

Weibin Meng, *Graduate Student Member, IEEE*, Ying Liu, *Member, IEEE*, Shenglin Zhang, *Member, IEEE*, Federico Zaiter, Yuzhe Zhang, Yuheng Huang, Zhaoyang Yu, Yuzhi Zhang, Lei Song, Ming Zhang, and Dan Pei, *Senior Member, IEEE*

***Abstract*—Logs are imperative in the management process of networks and services. However, manually identifying and classifying anomalous logs is time-consuming, error-prone, and labor-intensive. Additionally, rule-based approaches cannot tackle the challenges underlying anomalous log identification and classification resulting from new types of logs and partial labels. We propose LogClass, a framework to automatically and robustly identify and classify anomalous logs for network and service based on *partial labels*. LogClass combines a word representation method, a positive and unlabeled learning (PU learning) model, and a machine learning classifier. Besides, we propose a novel Inverse Location Frequency (ILF) method to weight the words of logs in feature construction properly. We evaluate the performance of LogClass based on 18 million+ real-world switch logs and six public log datasets. It achieves 99.56% and 98% F1 scores in anomalous log identification on switch logs and publicly available supercomputer logs, respectively, and very-close-to-one F1 score in anomalous log classification. Moreover, we have conducted extensive experiments to demonstrate LogClass' superior performance in addressing partial labels and new types of logs.**

***Index Terms*—AIOps, service management, log analysis, anomaly identification, anomaly classification.**

Manuscript received July 27, 2020; revised November 19, 2020 and January 18, 2021; accepted January 25, 2021. Date of publication January 28, 2021; date of current version June 10, 2021. The work was supported by the National Key R&D Program of China (Grant No. 2018YFB1800405, 2019YFE0105500), National Natural Science Foundation of China (Grant No. 61902200, 61772307, and 62072264), Beijing National Research Center for Information Science and Technology (BNRist), The China Postdoctoral Science Foundation (2019M651015). The associate editor coordinating the review of this article and approving it for publication was N. Zincir-Heywood. *(Corresponding author: Shenglin Zhang.)*

Weibin Meng, Federico Zaiter, Zhaoyang Yu, and Dan Pei are with the Department of Computer Science, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Beijing 100084, China.

Ying Liu is with the Institute of Network Sciences and Cyberspace, Tsinghua University, Beijing 1000841, China, and also with the Beijing National Research Center for Information Science and Technology, Beijing 100084, China.

Shenglin Zhang, Yuzhe Zhang, and Yuzhi Zhang are with the College of Software, Nankai University, Tianjin 300457, China (e-mail: zhangsl@nankai.edu.cn).

Yuheng Huang is with the School of computer science, Beijing University of Posts and Telecommunications, Beijing 100084, China.

Lei Song is with Baidu, Inc., Beijing 100084, China.

Ming Zhang is with China Construction Bank, Beijing 100084, China.

Digital Object Identifier 10.1109/TNSM.2021.3055425

I. Introduction

With the rapid development of the Internet, Web services have penetrated all aspects of society. With the explosion of the number of Web services, the stability of network and service is becoming more and more important. Since network and service anomalies can significantly impact user experience and/or company income, operators continuously monitor their status [1]–[4]. Accurate and timely anomaly identification and classification can help operators quickly mitigate any outage and locate its root cause, which is crucial for network and service.

Logs (as shown in Table I) are one of the most valuable data for network and service management. Although key performance indicator (KPI, e.g., CPU utilization, memory utilization) curves can indicate whether a service is anomalous [1]–[4], they are of little help for anomaly classification and localization when they are used in isolation [5], [6], for the reason that KPI curves usually contain too little information. For example, an anomaly (say a level shift) in the curve of CPU utilization only indicates that the CPU utilization increases sharply, but it cannot tell why it happens. On the contrary, logs describe a vast range of (anomalous) events, which are quite valuable for anomaly classification and localization. For example, when a switch generates a log of "System is rebooting now", operators can determine that the switch is anomalous, classify this anomaly into the category of "SYSTEM_REBOOT", and conclude that this anomaly is caused by a system reboot.

The rich information and the pervasiveness of logs enable a wide variety of management and diagnostic tasks, such as monitoring service status [7], understanding network events [8], identifying anomalies [9] and predicting network device failures [10]. As for log-based anomaly identification, there are many categories, such as anomalous individual log identification, which aims to identify the anomalous individual logs, anomalous log sequence identification [7], [9], [11] tries to capture the sequential patterns of anomalous logs, quantity anomaly identification of a single type of logs [12], and anomalous log quantitative relationship identification [13], which is based on the natural quantitative relationship of logs. In this article, we focus on single log anomaly identification and classification.

Generally, operators identify and classify anomalous individual logs using rule-based approaches (e.g., regular expressions [14]), which are usually manually maintained by

1932-4537 © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE I
EXAMPLES OF LOGS

Source	Timestamp	Message Type	Detailed Message
Switch	Jun 12 19:03:27	SIF	Interface te-1/1/59, changed state to down
Supercomputer	Jun 13 20:22:03	ERROR	EDRAM 0x0157 identified and corrected over 27362 seconds
HDFS	Jun 18 05:21:03	INFO	dfs.DataNodePacketResponder: PacketResponder 1 for block blk_-1608999687919862906 terminating
Router	Jun 15 13:46:43	OSPF	Neighbour(rid:10.231.0.43, addr:10.231.39.61) on vlan23, changed state from Exchange to Loading

operators. They classify logs into *healthy logs* and (multiple categories of) *anomalous logs*. However, the increasing scale and complexity of modern network and service make the volume of logs explode, and tens of millions of logs are generated in a large datacenter. Rule-based techniques, suffering from inflexibility and labor intensiveness, are thus not effective, scalable and applicable to such a scale.

In this work, we aim to *automatically and accurately* identify and classify anomalous logs, as shown in Fig. 1. However, it faces three main challenges as follows.

1) *Partial labels:* In spite of those labor-intensive rule-based methods being used by operators, many anomalous logs remain unidentified because some anomalies can fly under operators' radar. Therefore, a large number of anomalous logs are unlabeled. In addition, operators usually do not label healthy logs because of their huge number and triviality. Consequently, traditional machine learning approaches, needing fully labeled samples as input, cannot address this partially labeled challenge.
2) *New types of logs:* Operators continuously conduct software/firmware upgrades on network and service to introduce new features, fix bugs or improve performance, which can generate new types of logs [15]. The manually maintained regular expressions, which are not updated frequently, cannot identify or classify these new types of logs.
3) *Feature construction:* In classical methods for constructing feature vectors from texts, e.g., bag-of-words, a log (text) is represented as a vector of its words. In general, each element in the vector denotes the estimated importance (weight) of a word [16]. Existing word weighting methods such as TF-IDF [17], assuming that frequently occurring words are not important, are not suitable in our scenario because the assumption is not always true for logs [18].

To address the above challenges, we propose LogClass, a framework to identify and classify anomalous logs based on *partial labels*. The core idea of LogClass is that most logs are semi-structured texts "printf"-ed by network and service, and logs of the same anomaly category share common patterns in terms of the word combination. In addition, the intuitions and methods in natural language processing can be applied or improved for anomalous log classification. LogClass consists of two main components, i.e., the offline learning component and the online identification and classification component. In the offline learning component, LogClass preprocesses logs

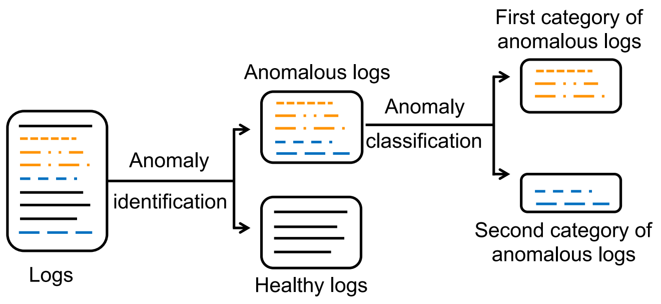


Fig. 1. Problem definition of anomalous log identification and classification.

and generates feature vectors weighting with TF-ILF (term frequency-inverse location frequency). Then LogClass applies positive and unlabeled learning (PU Learning), a classification model based on positive and unlabeled data [19], to train an anomaly identification model, and a machine learning classifier to train an anomaly classification model, respectively. Similarly, in the online identification component, using the trained anomaly identification/classification model, LogClass determines whether a new log is anomalous, and if so, it determines its anomaly category.

We evaluate LogClass based on both real-world switch logs collected from a top global search engine and public log datasets [20] collected from BGL (Blue Gene/L supercomputer). For anomalous log identification, the accuracies (in terms of F1 Score) of LogClass are 99.56% and 98% on switch logs and the BGL dataset, respectively. Moreover, LogClass achieves very-close-to-one accuracy (in terms of Micro-F1 Score and Macro-F1 Score) in anomalous log classification. Because the introduction of PU learning, the accuracy of LogClass remains stable as the ratio of unlabeled logs increases, demonstrating its superior performance in handling partial labels. In addition, LogClass achieves very-close-to-one F1 Score even when new types of logs appear. In the end, we have evaluated the importance of TF-ILF.

The contributions of this article can be summarized as follows:

1) We use the framework of PU learning to address the challenges introduced by partial labels.
2) LogClass automatically and robustly measures the similarity of the word combination between a given log and labeled anomalous logs, which successfully addresses the challenge of new types of logs.
3) We propose TF-ILF, a *novel, simple yet effective* method to properly weight the words of logs in feature vector construction.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE II
EXAMPLES OF RULES FOR LABELING ANOMALOUS LOGS

.*Power.*recovered.*
.* ISIS: Neighbor.*Down.*
.*neighbor.* Down Interface flap.*

4) We have performed extensive evaluation experiments on LogClass using real-world switch logs and public log datasets. Moreover, we have open-sourced[1] the implementation of LogClass, and hope that it will be useful for future researches.

The rest of this article is organized as follows. Section II provides the background and motivation of our problem. The design of LogClass is presented in Section III. The evaluation experiments are described in Section IV, followed by a brief review of related works in Section V. Finally, we conclude our work in Section VI.

II. BACKGROUND

In this section, we first describe the characteristics of logs in Section II-A, followed by the introduction of rule-based approaches in Section II-B. In the end, log identification and classification are formulated in Section II-C.

A. Logs

Logs are imperative in the management process of network and service. They record detailed runtime information that allows operators to monitor network and service status. Additionally, logs are so general that almost all network devices and services generate them. A log message is essentially a semi-structured text "printf"-ed by network and service. Table I lists four examples of log messages. As the table shows, a log message usually has a primitive structure containing at least four fields, including the source of logs, a timestamp indicating when this message is generated, a message type describing its rough characteristics, and a detailed message depicting the details of the event that this log representation. The message type field, on the one hand, is too rough and ambiguous to be used to identify and classify anomalous logs [10] accurately. On the other hand, the detailed message field describes the runtime status of network and service in detail, and thus operators pay the most attention to this field. In this article, we also focus on the detailed message field. Henceforth, we use "log" or "log message" for short, to denote the detailed message field of a log.

B. Rule-Based Approaches

Typically, operators need to manually define the rules of anomalous logs in order to automatically identify and classify anomalous logs in the future. The simplest rule is to match keywords including "loss", "lost", *etc*. However, this method can lead to false alarms. For instance, when a network device tries to PING another one, a log message, e.g., "packets : sent = 5, received = 5, lost = 0 (0% loss)", will be generated to record this event. Although this log message contains the word "loss", it is not anomalous. As listed in Table II, another popular way to add rules is to manually configure regular expressions that match with anomalous logs based on domain knowledge. However, it has several drawbacks as follows:

1) *Inflexibility:* Regular expressions are too rigorous for matching anomalous logs. That is, a regular expression cannot match an anomalous log when the log is not in its exact format, even if the difference is only a trivial word, a whitespace, or a symbol (see Table X for more details). Besides, the anomalous logs belonging to the same anomaly category but from different types of network devices or services are likely to have some minor differences. More specifically, the anomalous logs generated by two different types of devices are very similar in semantics but different in syntax [21]. Therefore, the similar anomaly events from different devices, although different in syntax, have similar pattern in semantics. Regular expression, however, cannot capture these patterns. Therefore, the rule-based method is not flexible and generic for anomalous log identification and classification. Note that the inflexibility is not a problem induced by regular expressions but resulting from regular expressions produced from partial labels of historical logs.
2) *Labor intensive:* All the regular expressions are *manually* configured and updated by operators. Because a large number of new types of anomalous logs (thousands of per day) are generated, and operators have to configure regular expressions for these logs, manually configuring regular expressions costs huge amounts of work. Besides, Kabinna *et al.* [22] found that 20%-45% of the logging statements in many applications change throughout their lifetime. Although some template extraction methods [20] can be applied to assist operators in generating regular expressions, the generated regular expressions should be further manually labelled (anomalous or not) by operators. Considering the large number of regular expressions, it is still labor expensive.

C. Log Identification and Classification

As mentioned above, there are three log anomaly detection/identification scenarios: anomalous individual log identification, anomalous log sequence identification [9] and

[1]LogClass is available on Github: https://github.com/NetManAIOps/LogClass.

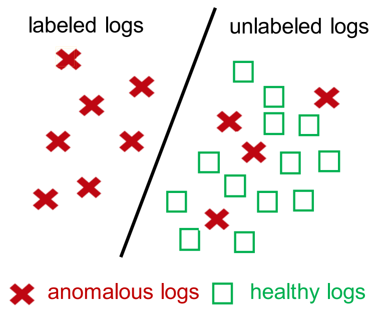


Fig. 2. Four types of logs: anomalous, healthy, labeled, and unlabeled logs.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE III
TWELVE EXAMPLES OF ANOMALY CATEGORY

FAN_RECOVERED	OSPF_NEIGHBOR_CHANGE	FAN_FAILED
BOARD_DISABLE	BGP_NEIGHBOR_CHANGE	POWER_DOWN
SYSTEM_REBOOT	INTERFACE_DOWN	PORT_DOWN
PROTOCOL_DOWN	OSPF_DOWN	MODEL_OUT

anomalous log quantitative relationship identification [13]. There are many types of anomalies in logs, such as sequential anomaly (LogAnomaly [7], LogRobust [11] and Deeplog [9]), quantitative relation anomaly (Invariants mining, [23], LogCluster [13]), quantity anomaly of a single template (ADELE [12]), and so on. Most of them ignore variables (parameter values), because the monitoring data of these variables can form metric streams, the anomaly of which can be detected by time series based anomaly detection methods, such as Donut [4] and Buzz [2]. In our paper, we focus on single log anomaly detection/identification, e.g., the logs with the pattern "** power down**" are anomalous and every time they appear represents a device anomaly.

In our scenario, there are four types of logs as shown in Fig. 2: anomalous logs, healthy logs, unlabeled logs and labeled logs, defined as follows.

- *Anomalous log:* An anomalous log indicates that an anomaly occurs on the network/service. Each anomalous log belongs to a specific anomaly category.
- *Healthy log:* A healthy log is the one that does not denote any anomaly. That is, it describes a healthy event or status.
- *Unlabeled log:* As aforementioned, operators are not able to label (identify) all the anomalous logs. The anomalous logs that remain unidentified together with the healthy logs constitute the set of unlabeled logs.
- *Labeled log:* The identified anomalous logs are the labeled logs. Generally, operators manually configure regular expressions to label logs. They do not configure regular expressions for healthy logs because it does not benefit detecting anomalous logs.

If a log is anomalous, it can be classified into a specific anomaly category, which is based on the event it represents. Table III shows 12 examples of anomaly categories. For example, the anomalous log "Interface te-1/1/59, changed state to down" (L_1 in Table I), which denotes that the interface is down, can be classified into the anomaly category "INTERFACE_DOWN" (in Table III).

Typically, operators label anomalous logs using regular expressions (see Table II). However, due to its inflexibility and labor-intensiveness as mentioned in Section II-B, operators cannot configure the regular expressions for all anomalous logs, and a large number of anomalous logs are flying under operators' radar. More specifically, only a fraction of anomalous (positive) logs are labelled, and neither the remaining anomalous (positive) logs nor the normal (negative) logs are labelled. PU leaning is designed for the scenario where only a fraction of positive samples are labeled, and the remaining positive samples as well as all the negative samples are unlabeled [24]–[28]. PU learning is different from supervised learning in that the later needs to obtain the labels of all the positive and negative samples of the training set. Furthermore, it is distinguished from semi-supervised learning by that the later needs a part of positive samples as well as a part of negative samples to be labeled [29]. Therefore, PU learning methods, rather than supervised learning nor semi-supervised learning methods, can be applied for our scenario. Therefore, we try to design an automatic and robust anomalous log identification and classification method to identify and classify *all* anomalous logs.

III. DESIGN OF LOGCLASS

In this section, we present the framework of LogClass. The main objective of LogClass is to *automatically and accurately* identify and classify anomalous logs for network and service using *partial labels*. We first introduce the design overview of LogClass in Section III-A, and then explain how to preprocess logs in Section III-B. Feature extraction of LogClass is elaborated in Section III-C, followed by the description of offline learning and online classification in Section III-D.

A. Design Overview

As aforementioned, a log is a semi-structured text "printf"-ed by network and service. After extensive investigations on real-world logs, we have the following observations:

1) Anomalous logs of the same anomaly category usually share common patterns. Specifically, they share a common *combination of "important" words*. If we can accurately obtain this combination, we can easily determine whether a log belongs to some anomaly category. If the log does not belong to any anomaly category, it is a healthy log.
2) To get the above combinations of "important" words, we need to identify which words in a log are "important" and assign higher weights to these words. If a word frequently appears in a log, it is likely an "important" word. However, if it appears in different logs *with different "locations"*, it is likely an article (say, "a", "the"), or a preposition (say, "in", "on"), or a conjunction (say, "and", "but"), and thus it is not an "important" word.

Based on the above observations, we design LogClass as shown in Fig. 3. LogClass consists of two main components: an offline learning component and an online classification component. In the offline learning component, LogClass first preprocesses logs and filters parameters (Section III-B). Then LogClass constructs features using the bag-of-words model, which is weighted by our novel TF-ILF method (Section III-C). Finally, LogClass trains a PU learning based binary classifier, and an SVM-based multiclass classifier (Section III-D). Similarly, in the online classification component, LogClass preprocesses real-time logs and extracts features, after which LogClass determines whether a log is anomalous using the trained binary classifier, and if so, classifies it into an anomaly category using the multiclass classifier.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

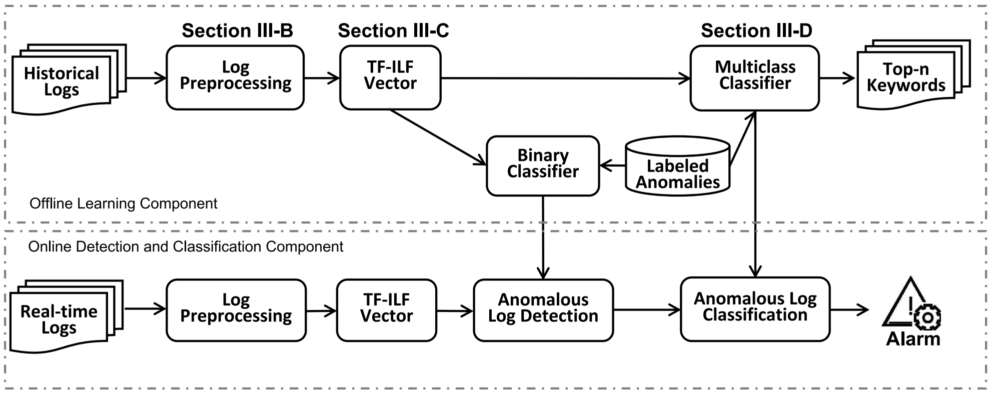


Fig. 3. The Framework of LogClass.

TABLE IV
FORMAT RULES OF TEXT WORDS AND PARAMETER WORDS

Category	Format rules	Example
Text	Letters	change
	Symbols and letters	Vlan-interface
Parameter	Symbols	(@ &
	Numbers	1512028952
	Numbers and symbol	192.168.64.107
	Numbers and letters	vlan23

B. Log Preprocessing

In general, simple log preprocessing using domain knowledge, e.g., removing IP addresses, is able to improve the performance of log parsing and in turn increases anomalous log identification and classification [30]. Therefore, we preprocess logs before extracting features from them.

As is with [31], we classify the words of logs into *text words* and *parameter words*. Text words are those depicting the events occurring on network and service. On the contrary, parameter words are those variables (say, IP address, interface ID, device ID) that are usually vary from one log to another. It is quite difficult for the log-based anomaly detection/identification/classification methods to capture patterns from these parameter words. Consequently, in this article we ignore these parameter words following popular log-based anomaly detection methods, e.g., Invariants Mining [23], PCA [32], LogRubust [11], and LogAnomaly [7].

We classify text words and parameter words using simple format rules based on operators' domain knowledge. Table IV lists the format rules and some examples of text words and parameter words, respectively. Based on these rules, we filter parameter words and preserve text words for each log. Note that operators can change these rules based on their own domain knowledge.

C. Feature Construction

Since logs are semi-structured texts, they cannot be directly input into machine learning-based classifiers. In this work, we apply the bag-of-words model [16], one of the most popular methods in natural language processing, to construct features from documents. In this model, a log is represented as a vector of its words, and the value of each element in the vector denotes the estimated importance (weight) of a word [16]. Given that the classical weighting method, TF-IDF, is not suitable for our scenario, we propose a novel method, TF-ILF, to properly weight the words of logs based on domain knowledge as follows.

1) TF-ILF: TF-IDF, a simple yet effective weight model, is the most widely used weighting scheme for the bag-of-words representation [33]. As the term implies, TF refers to term frequency. The IDF (inverse document frequency) is a measure of how much information the word provides, namely, whether the term is common or rare across all documents. Formally, the IDF value of word w is $\mathrm{IDF}_w = \log(\frac{N}{n_w})$, where N is the total number of documents, and n_w is the number of documents in which w appears. This model suggests that the more documents (logs in this case) a word appears in, the less important this word is, which is inconsistent with the second observation mentioned in Section III-A. For example, both L_1 and L_4 in Fig. 4 indicate the same event – an interface of the switch changes its state to down. If this type of events occur frequently in a period of time (this often happens because of interface flap), it will generate lots of logs like L_1 and L_4. As a result, a large number of the words "Interface" will be generated, and in turn, the word "Interface" will be assigned a low weight in TF-IDF. However, the word "Interface" is a significant word for log classification based on operators' domain knowledge.

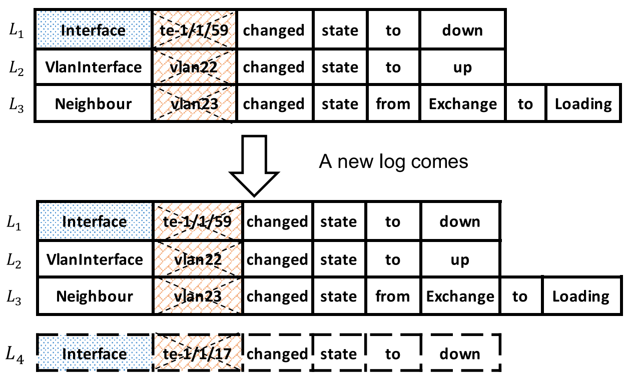
Motivated by the above observations, we propose a novel method, TF-ILF, to weight every word of device logs for the bag-of-words representation. It consists of TF and ILF as follows.

TF refers to term frequency. The more times a word appears in a log, the more significant this word is estimated to be.

ILF measures how many different locations a word appears in. Specifically, a *location* $l_k \in \mathcal{L}$ of a word, where $\mathcal{L}$ is the longest length of all the logs, is defined as the ordinal position (the kth word) where this word appears. We define the ILF of word w as

$$\mathrm{ILF}_w = \log\left(\frac{\mathcal{L}}{\sum_{k=1}^{\mathcal{L}} w \diamond l_k}\right) \quad (1)$$

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.


L_1 Interface te-1/1/59 changed state to down
L_2 VlanInterface vlan22 changed state to up
L_3 Neighbour vlan23 changed state from Exchange to Loading
A new log comes
L_1 Interface te-1/1/59 changed state to down
L_2 VlanInterface vlan22 changed state to up
L_3 Neighbour vlan23 changed state from Exchange to Loading
L_4 Interface te-1/1/17 changed state to down

# total logs (rows)	# logs "Interface" appearing in	IDF value of "Interface"
3	1 (L_1)	log(3/1)

# total locations (columns)	# locations "Interface" appearing in	ILF value of "Interface"
7	1 (the first position)	log(7/1)

# total logs (rows)	# logs "Interface" appearing in	IDF value of "Interface"
4	2 (L_1, L_4)	log(4/2)

# total locations (columns)	# locations "Interface" appearing in	ILF value of "Interface"
7	1 (the first position)	log(7/1)

Fig. 4. An example of adding a new log in offline learning.

where $w \Diamond l_k$ is

$$w \Diamond l_k = \begin{cases} 1, & \text{word } w \text{ appears in } l_k \text{ of some log} \\ 0, & w \text{ does not appear in } l_k \text{ of any log} \end{cases} \quad (2)$$

As shown in the bottom of Fig. 4, there are four logs containing the word "to". It is the fifth word in L_1, L_2 and L_4 ("to" $\Diamond l_5 = 1$), and the seventh word in L_3 ("to" $\Diamond l_7 = 1$). That is, the word "to" appears in two different locations. On the contrary, the word "Interface" appears in the *first* location of all the logs ("Interface" $\Diamond l_1 = 1$). Intuitively, the fewer number of different locations where a word appears, the more important it is to anomaly identification/classification. We get this argument based on our investigation on tens of thousands of real-world device logs and it is confirmed by experienced operators. Therefore, the weight of the word *w* in a log message is

$$\mathrm{TF} - \mathrm{ILF}_w = \mathrm{TF}_w * \mathrm{ILF}_w \quad (3)$$

where TF_w is the frequency of word *w* in a log.

As discussed in [34], although a large number of new types of logs can be generated at running time, the new words in these new types of logs are mainly variable word, which are filtered out in the preprocessing procedure. Therefore, the TF-IDF model, which is applied after the preprocessing procedure, does not need to calculate features for many new words at running time, making sure that LogClass is scalable to the large number of new types of logs.

2) Why Does TF-ILF Work Better Than TF-IDF?: For a given word, *IDF* cares about the number of different *logs* it appears in, while *ILF* counts the number of different *locations* where it appears within logs. In the training procedure, for an "important" text word (see Section III-B for definition), its IDF value will become smaller when a new log containing it is added, while its ILF value will likely remain unchanged. Fig. 4 shows an example of how the IDF and ILF value of an important text word changes when a new log is added. At first, the longest length of all the logs is $\mathcal{L} = 7$ (after preprocessing logs and filtering parameter words), and the word "Interface" appears *only* in the first position of L_1. Therefore, the IDF value of the word "Interface" is $\log(\frac{3}{1})$, and the ILF value is $\log(\frac{7}{1})$. When L_4 is added, we find that the word "Interface" appears in the first position of L_4. As a result, the IDF value of the word "Interface" changes to IDF $= \log(\frac{4}{2})$ which is smaller than $\log(\frac{3}{1})$, while the ILF value remains unchanged. As more and more logs with the same format of L_1 and L_4 are added, the IDF value will become smaller and smaller. Eventually, TF-IDF will assign a very small weight to the word "Interface", which is inconsistent with the fact that this word is really important to anomalous log identification/classification (based on operators' domain knowledge). On the contrary, the ILF value of the word "Interface" remains unchanged when new logs (with the same format of L_1 and L_4) are added. Therefore, TF-ILF always assigns an important weight to this word.

We can see that the TF-ILF approach considers the words and structures of log messages. Although it does not consider abbreviations, or words with similar semantics, it performs really well in practice, as shown in Section IV-E. It demonstrates that, in real-world logs, words and structures, rather than abbreviations or semantics, are important to the log-based anomaly identification and classification.

D. Binary and Multiclass Classification

In the offline learning procedure, after obtaining the feature vectors, we apply the PU learning model [19] to train a binary classifier based on partial labels (namely, unlabeled anomalous logs and healthy logs, as well as labeled anomalous logs). After that, we train a multiclass classifier to classify anomalous logs into different categories in an interpretable manner. Similarly, in the online classification procedure, we use the trained binary classifier to determine whether a real-time log is anomalous. If yes, we then apply the trained multiclass classifier to infer its category.

In this section, we first introduce how PU learning is used in our scenario, after which we depict the binary and multiclass classifiers.

1) PU Learning-Based Binary Classification: Under the assumption that labeled examples are selected randomly from positive examples, PU learning is trained on positive and unlabeled examples, and predicts the probabilities of being positive, which differ by only a constant factor from the true conditional probabilities [35].

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE V
DETAILS OF THE SWITCH LOG DATASET

Duration	# switches	# switch types	# labeled (anomalous) logs	# unlabeled (anomalous and healthy) logs
2 weeks	6574	58	$1,758,458$	$16,702,547$

Let x be a log and let $y \in \{1, 0\}$ be a binary label (anomalous or not). Let $s = 1$ if x is labeled, and $s = 0$ if not. Only anomalous logs are labeled, and thus $s = 0$ (an unlabeled log) is certain when $y = 0$ (a healthy log), which can be stated formally as

$$p(s = 1|x, y = 0) = 0 \tag{4}$$

We have

$$p(s = 1|x, y = 1) = p(s = 1|y = 1) \tag{5}$$

For a traditional probabilistic classifier, the goal is to learn a function $f(x)$ such that $f(x) = p(y = 1|x)$ as closely as possible. Similarly, PU Learning needs to learn a function $g(x)$ such that $g(x) = p(s = 1|x)$ approximately. After that, a traditional classifier $f(x)$ can be learned from $g(x)$ as

$$f(x) = \frac{g(x)}{c} \tag{6}$$

where $c = p(s = 1|y = 1)$ is the constant probability that an anomalous log is labeled.

Let V be such a validation set that follows the distribution $p(x, y, s)$, and P be the subset of logs in V that are labeled. As shown in [36], a simple yet effective estimator of c is

$$c = \frac{1}{n} \sum_{x \in P} g(x) \tag{7}$$

where n is the number of different logs in P.

So far, we transform a traditional binary classifier to a PU Learning classifier, which is trained based on labeled (anomalous) and unlabeled (anomalous and healthy) logs.

In general, the number of labeled logs is much smaller than that of unlabeled logs (see Table V for more information). As discussed in [10], [37], since random forest makes decisions based on the outcome of the majority voting by its many decision trees, the imbalance of positive and negative samples has little impact on the accuracy of random forest. Therefore, we apply random forest to train the binary classifier.

2) Multiclass Classification: In the online classification procedure, when the above binary classifier determines a real-time log to be anomalous, the multiclass classifier will then decide which anomaly category this log belongs to. As shown in Fig. 3, the multiclass classifier is trained based on the labeled anomalous logs that have been well investigated and classified by operators. Note that any robust machine learning classifier can be used to train the multiclass classifier. We compare the performance of four classifiers, i.e., Support Vector Machine (SVM) [38], Decision Tree (DT) [39], Naive Bayes (NB) [40], and Logistic Regression (LR) [41], in Table VIII, and find that LogClass is robust to multiple machine learning classifiers.

In addition to which category an anomalous log belongs to, operators are also eager to understand why it happens. Therefore, we can get the coefficient assigned to each feature (word) from the trained machine learning classifier. Consequently, the top n important words (*keywrods*) can be obtained by sorting these coefficients. Operators can further understand every category by its top n important words (see Table IX for more details).

IV. EVALUATION

In this section, we first describe the details of experiment settings in Section IV-A, followed by the introduction of how the overall performance of LogClass is evaluated in Section IV-B. The evaluation of LogClass' performance in addressing partial labels and new types of logs are respectively depicted in Section IV-C and Section IV-D. In the end, we demonstrate the importance of TF-ILF in Section IV-E.

A. Experiment Setting

1) Datasets: To evaluate the performance of LogClass, we use a real-world switch log dataset and six public log datasets. The switch log dataset is collected from the datacenters of a top global search engine. At first, considering the huge number of *all* switch logs (millions per day), we randomly selected 6574 switches (of 58 types) deployed in 10+ datacenters. These switches are of three manufacturers (Huawei, H3C, and Cisco). Then, we collect *all* the switch logs of the above switches over *a 2-week period*. Table V lists the detailed information of the dataset, including its duration, as well as the number of switches/switch types/labeled anomalous logs/unlabeled logs. These anomalous logs belong to 12 different anomaly categories as listed in Table III. Regarding to the labeled anomalous logs, they are all labeled and classified using the regular expressions manually configured and confirmed by experienced operators (see Section II-B for more details). As Table V shows, a large number of logs remain unlabeled, which brings great challenges to LogClass.

In addition to switch logs, we also use six public datasets [20], which are the logs collected from BGL (Blue Gene/L supercomputer), HDFS (Hadoop distributed file system), Proxifier (Proxifier software), Zookeeper (ZooKeeper service), Hadoop (Hadoop map/reduce job), HPC (High performance cluster), respectively. All the six datasets have released their classification results of logs, which are used as the ground truth for log classification. Besides, each log of the BGL dataset is labeled as anomalous or not, which is suitable to evaluate the performance of anomalous log identification.

2) Evaluation Metrics: For anomalous log identification, let the set of labeled anomalous logs be L, the set of unlabeled logs be U. Apparently, U consists of the set of unlabeled

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

healthy logs H, and the set of unlabeled anomalous logs A. That is, $U = H + A$. Therefore, LogClass is trained based on L and U, and aims to classify the whole dataset into $L + A$ (anomalous logs) and H (healthy logs). We use precision, recall, F1 score to evaluate the performance of the PU learning-based binary classifier. We label LogClass's outcome as true positive (*TP*), true negative (*TN*), false positive (*FP*) and false negative (*FN*). True positives are the anomalous logs (belonging to L or A) that are accurately determined as such by the method. True negatives are the healthy logs (belonging to H) that are accurately determined. If the method determines a log as anomalous (belonging to L or A), but in fact it is healthy (belonging to H), then this outcome is a false positive. The rest are false negatives. We calculate precision, recall and F1 score as follows: $\text{precision} = \frac{TP}{TP+FP}$, $\text{recall} = \frac{TP}{TP+FN}$, $\text{F1score} = \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$, all metrics are calculated on the testing dataset.

A multiclass classification method's capability is usually assessed by Micro-F1 Score and Macro-F1 Score [42]. For anomalous log classification, we also use Micro-F1 Score and Macro-F1 Score to evaluate its performance. For a given anomaly category i, we label an outcome as TP_i, TN_i, FP_i and FN_i. TP_i is a log belonging to i that is accurately determined as such by the method. TN_i is a log not belonging to i that is accurately determined. If the method determines a log belonging to i, but in fact it does not, we label the outcome as FP_i. The rest are FN_is. Based on TP_i, TN_i, FP_i and FN_i, we then calculate precision_i and recall_i as $\text{precision}_i = \frac{TP_i}{TP_i+FP_i}$, $\text{recall}_i = \frac{TP_i}{TP_i+FN_i}$. We then obtain precision_{macro} and recall_{macro} as

$$\text{precision}_{macro} = \frac{1}{n}\sum_{i=1}^{n}\text{precision}_i \tag{8}$$

$$\text{recall}_{macro} = \frac{1}{n}\sum_{i=1}^{n}\text{recall}_i \tag{9}$$

After that, we calculate Macro-F1 as

$$\text{Macro} - \text{F1} = \frac{2\times\text{precision}_{macro}\times\text{recall}_{macro}}{\text{precision}_{macro}+\text{recall}_{macro}} \tag{10}$$

Similarly, we first calculate

$$\text{precision}_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \tag{11}$$

and

$$\text{recall}_{micro} = \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \tag{12}$$

Then we get

$$\text{Micro} - \text{F1} = \frac{2\times\text{precision}_{micro}\times\text{recall}_{micro}}{\text{precision}_{micro}+\text{recall}_{micro}} \tag{13}$$

From the above definitions, we can conclude that Macro-F1 assigns an equal weight to each class, while Micro-F1 weights a class based on its number of samples (logs). The combination of Micro-F1 and Macro-F1 provides us a global view of a classification method's accuracy.

3) Baselines: For anomalous log identification, we compare LogClass with five log-based anomaly identification methods, i.e., principal component analysis (PCA) [43], Invariant Mining (IM) [23], LogCluster [13], Deeplog [9], LogAnomaly [7] and Labeled-LDA(L-LDA).

4) Experimental Setup: We conduct all the experiments on a Linux server with Intel Xeon 2.40 GHz CPU and 64G memory. We implement LogClass, DeepLog, LogAnomaly, L-LDA, and FT-tree with Python3. In order to help researchers further understand LogClass, we have open-sourced its implementation.[2] As for LogCluster, PCA and IM, we use a popular open-source toolkit implemented in [44].

B. Evaluation of The Overall Performance

Recall that the objective of LogClass is to automatically and accurately identify and classify anomalous individual logs. Here we evaluate the performance of LogClass in anomalous identification and classification based on switch logs and public datasets.

Considering the huge number (16,702, 547) of unlabeled logs, manually classifying them into healthy and anomalous logs is infeasible. We thus request operators to label 0.1% of randomly sampled unlabeled logs, which are still 16,702 logs. Each anomalous log is then manually classified into a specific anomaly category serving as the ground truth for anomalous log classification. Among these 16,702 sampled unlabeled logs, 12 logs are anomalous, and the rest 16,691 are healthy. Note that although the sample ratio is relatively small, manually classifying more than 16,000 logs indeed consumes operators a lot of time. To construct the testing set, we also randomly sample 0.1% of labeled anomalous logs (i.e., 1,758 logs). Moreover, although there are only 12 anomalous logs in the sampled unlabeled dataset, there are 1770 anomalous logs in total in the training set, which combines the sampled unlabeled dataset with the sampled labeled anomalous dataset. The sampling ratio is suggested by operators. Therefore, the 1,758 sampled labeled logs together with the 16,702 sampled unlabeled logs constitute the testing set (we call the testing set $\mathbb{E}$ henceforth). The training set (we call the training set $\mathbb{T}$ henceforth) consists of the entire dataset of switch logs, including the labeled logs and the unlabeled ones, as listed in Table V.

The second row of Table VI (Experiment#1) lists the experiment results of anomalous log identification in terms of precision, recall and F1 score. LogClass achieves very-close-to-one precision and recall, demonstrating its superior performance in anomalous log identification.

We also apply the BGL dataset, which has released the anomalous label for every log, to demonstrate LogClass' performance in anomalous log identification. Specifically, we compare LogClass with PCA, IM, LogCluster, DeepLog and LogAnomaly, as shown in Fig. 5. Overall, LogClass achieves the best F1 score among the six methods, achiving an F1 score of 0.98. Although these baseline methods have good performance on anomalous log sequential and/or quantitative relationship identification [7], they do not perform as good as

[2]LogClass is available on github: https://github.com/federicozaiter/LogClass.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE VI
THE EVALUATION RESULTS OF LOGCLASS IN ANOMALOUS LOG IDENTIFICATION

No.	Training set	Testing set	Precision	Recall	F1 score
1	Training set $\mathbb{T}$	Testing set $\mathbb{E}$	99.13%	99.99%	99.56%
2	Logs of 53 types of switches in the $\mathbb{T}$	Logs of 5 types of switches in the $\mathbb{E}$	99.25%	99.18%	99.21%

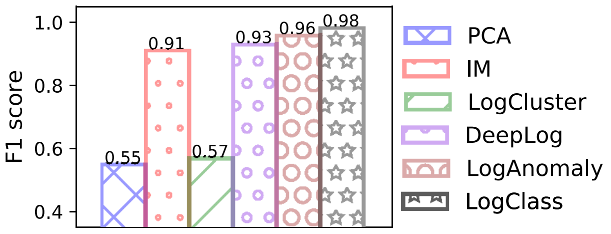


Fig. 5. The evaluation results of LogClass and baseline methods in anomalous log identification on the BGL dataset.

TABLE VII
THE EVALUATION RESULTS OF LOGCLASS AND L-LDA IN ANOMALOUS LOG CLASSIFICATION ON SWITCH LOGS

Methods	Macro-F1	Micro-F1	Training Time(s)	Classification Time(s)
LogClass	99.57%	99.96%	216.2	8.442×10^{-2}
L-LDA	89.68%	93.53%	4436	0.5280

LogClass in anomalous individual log identification. In addition, they cannot be used for anomalous log classification. Therefore, they are not suitable for our scenario.

After identifying anomalous logs, LogClass then classifies these logs into different categories. Firstly, we compare LogClass with L-LDA [45] based on $\mathbb{T}$ and $\mathbb{E}$. Table VII lists the comparison results on switch logs. LogClass achieves better performance than L-LDA in terms of both Macro-F1 and Micro-F1. LogClass improves 9.89% and 6.87% over L-LDA in Macro-F1 Score and Micro-F1 Score, respectively. In addition, the training time on $\mathbb{T}$ (containing 1,758,458 labeled logs and 16,702,547 unlabeled logs), and the classification time on $\mathbb{E}$ (containing 1,758 labeled logs and 16,703 unlabeled logs), of LogClass and L-LDA, are also listed in Table VII, respectively. Compared to L-LDA, LogClass respectively improves the training and classification efficiency by 19.51 and 5.25 times. Operators are quite satisfactory with the computational efficiency of LogClass in online anomalous log classification.

The experiments on BGL logs, as listed in Table VIII, also demonstrate LogClass' superior performance in accuracy and efficiency. Moreover, we show the performance of LogClass using four machine learning methods, i.e., SVM [38], DT [39], NB [40], and LR [41], as multiclass classifiers in Table VIII. The results show that all the four classifiers achieve similar accuracy, and LogClass is general to different multiclass classifiers.

As described in Section III-D, LogClass extracts the top-*n* important words (keywords) for each anomaly category, in oder to help operators better understand every anomaly category. In Table IX, we list the top 5 important words of the

TABLE VIII
THE EVALUATION RESULTS OF LOGCLASS WITH DIFFERENT MULTICLASS CLASSIFIERS AND L-LDA IN ANOMALOUS LOG CLASSIFICATION ON THE BGL DATASET. T.T. DENOTES TRAINING TIME (S), AND C.T. DENOTES CLASSIFICATION TIME(S)

Methods	Macro-F1	Micro-F1	T.T.	C.T.
LogClass w/ SVM	69.21%	99.30%	0.224	0.303
LogClass w/ NB	69.24%	98.63%	0.097	0.366
LogClass w/ LR	69.28%	99.28%	0.110	0.304
LogClass w/ DT	68.20%	99.21%	0.188	0.180
L-LDA	59.16%	94.20%	517.060	3.767

TABLE IX
THE TOP 5 IMPORTANT WORDS OF THE CATEGORY "INTERFACE_DOWN"

Logs	Interface TenGigabitEthernet 1/0/30 is down. Interface te-1/1/56, changed state to down GigabitEthernet 1/0/22: changed status to down
Top-5 words	"interface", "down", "state" "GigabitEthernet", "link"

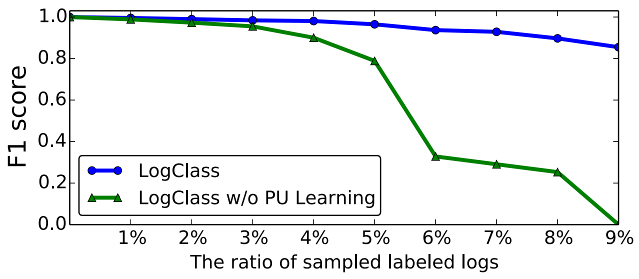


Fig. 6. The evaluation results of LogClass with/without PU learning as the ratio of sampled labeled logs increases on switch logs.

anomaly category "INTERFACE_DOWN" (see Table III for more details). Based on an on-site interview with operators, we find that the top-*n* important words really help operators further comprehend every anomaly category.

C. Evaluation of Address Partial Labels

LogClass adopts the PU learning model to address the challenge of partial labels. In order to show PU learning's performance, we adjust the training set as follows. For the training set $\mathbb{T}$, we randomly sample some fixed ratio (say 5%) of labeled anomalous logs, and move them to the unlabeled logs. That is, we "pretend" that the sampled labeled (anomalous) logs are unlabeled. Then, we train LogClass using the new training set, and test it based on $\mathbb{E}$.

Fig. 6 compares the accuracy (in terms of F1 score) of LogClass with and without PU learning as the ratio of sampled

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

TABLE X
AN ACTUAL CASE OF ANOMALOUS LOG IDENTIFICATION BY LOGCLASS

Log No.	Switch log	Labeled or not
$L1$	Interface TenGigabitEthernet 1/0/12 is link down.	unlabeled
$L2$	Interface TenGigabitEthernet 1/0/30 is protocol down.	labeled (anomalous)

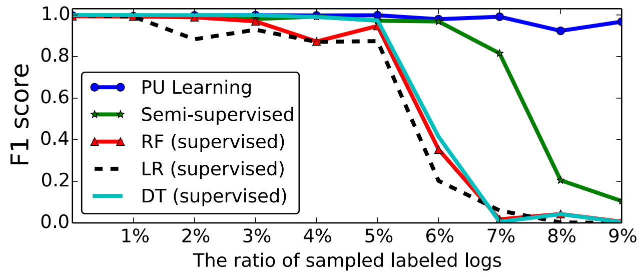


Fig. 7. The comparison results of LogClass with PU learning, semi-supervised learning, and three supervised learning methods as the ratio of sampled labeled logs increases on the BGL dataset.

labeled switch logs increases. The accuracy of LogClass with PU learning remains stable as the ratio increases. That is because the PU learning model properly transforms the traditional binary classifier to a classifier that is trained based on labeled (anomalous) logs and unlabeled (anomalous and healthy) logs. Without the transformation, the accuracy of LogClass without PU learning drops sharply when the ratio is larger than 4%, and becomes 0.0 when 9% of labeled logs are sampled and moved to the unlabeled dataset. This experiment demonstrates that PU learning greatly improves LogClass's accuracy.

To demonstrate the performance of PU learning, we replace PU learning of LogClass with Self Training [46] (a semi-supervised learning method) and three supervised learning methods including Random Forest (RF) [10], (Logistic Regression) LR [41], and (Decision Tree) DT [39]. Fig. 7 shows the comparison results on the BGL dataset. The ratio of the number of anomalous logs to that of healthy logs is 1:8. In the beginning, we suppose that all the healthy logs are unlabeled. Then, we gradually sample some ratio of anomalous logs and move them to the unlabeled set. For the self-training model, we kept the number of positives in its training set consistent with the number of other experiments, and label 50% of the unlabeled samples as negatives. As the anomalous logs takes a larger part of the unlabeled dataset and the number of labeled logs decreases, LogClass with PU learning maintains its superior accuracy while the performances of LogClass with semi-supervised/supervised decrease dramatically.

To intuitively demonstrate how LogClass performs in addressing partial labels, we randomly sampled 1,000,000 unlabeled logs from $\mathbb{T}$ (because manually investigating all the anomalous logs among the unlabeled logs of $\mathbb{E}$ will consume operators too much time), and apply LogClass to identify anomalous logs from the above unlabeled dataset. 710 anomalous logs are identified by LogClass, all of which are manually investigated and confirmed by operators. Table X shows a case of anomalous log identified by LogClass. $L1$ is an unlabeled log in the unlabeled dataset, and it is determined as anomalous by LogClass. We find that LogClass believes $L1$ is anomalous because a similar log in the training set $\mathbb{T}$, $L2$, is labeled anomalous. LogClass extracts the similarities between $L1$ and $L2$, and applies these similarities to determine whether $L1$ is anomalous.

D. Evaluation of Addressing New Types of Logs

To demonstrate how LogClass performs in addressing the challenge of new types of logs, we randomly select 53 types of switches and use their logs in the training set $\mathbb{T}$ to train LogClass.[3] That is, 1,582,612 labeled logs and 15,032,292 unlabeled logs constitute the new training set. Moreover, we use the logs that are collected from the *rest five* types of switches in the original testing set $\mathbb{E}$ to construct the new testing set, which consists of 175 labeled and 1,670 unlabeled logs. The third row of Table VI (experiment #2) lists the precision, recall and F1 score. The LogClass trained using the logs of 53 types of switches achieves a very-close-to-one F1 score in identifying anomalous logs for the other five types of switches. This demonstrates LogClass's good performance for new types of logs.

Then, we prove that using rules (i.e., regular expressions/log templates) and their similarities to identify and classify anomalous logs cannot address the challenges imposed by new types of logs. As mentioned above, the performance of LogClass in anomalous log identification and classification is superior even when the training set and testing set have different types of logs. It is because the PU learning of LogClass can well learn the patterns of anomalous logs even when the labeled logs and unlabeled logs are of different types, i.e., having different "rules". To demonstrate the above point, similar to the experiment introduced in Section IV-C, we first suppose that all the healthy logs are unlabeled in the BGL dataset. We then gradually (based on logs' generation order) sample the anomalous logs covering some ratio (say 50%) of anomalous categories (anomaly category coverage ratio), and move them to the unlabeled dataset. Fig. 8 shows the F1 Scores (blue and orange curves), and the ratio of anomalous logs to unlabeled logs (green curves), when the anomaly category coverage ratio increases from 0% to 90%. As the ratio increases, the accuracy of LogClass remains stable. However, the LogClass without PU learning suffers from a sharp drop when the anomaly category coverage ratio becomes higher than 86% and the ratio of anomalous logs to unlabeled logs reaches 3.2%. Consequently, PU learning helps LogClass successfully deal with new anomaly categories.

[3]Different types of switches usually generate different types of logs that are different in syntax.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

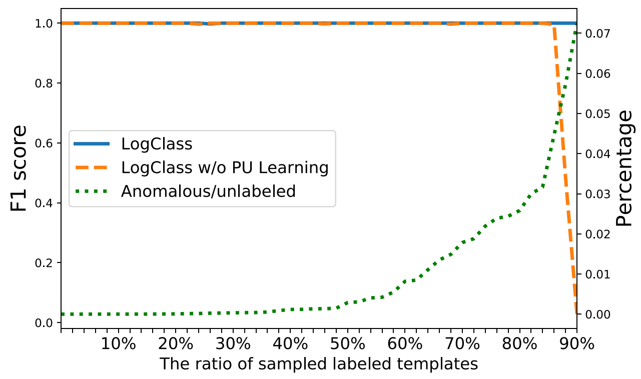


Fig. 8. The evaluation results of LogClass with/without PU learning as the ratio of sampled anomalous categories increases on the BGL dataset.

TABLE XI
THE EVALUATION RESULTS OF LOGCLASS WITH TF-ILF, WITH TF-IDF SWITCH LOGS

Methods	Macro-F1	Micro-F1	# of FP
LogClass with TF-ILF	99.57%	99.96%	72
LogClass with TF-IDF	96.32%	99.74%	615

E. Evaluation of TF-ILF

As described in Section III-C, we propose a novel, simple yet effective method, TF-ILF, to assign a weight for each word in feature construction. TF-ILF is motivated by TF-IDF – it preserves "term frequency (TF)" and improves from "document frequency" to "location frequency". To show the performance of TF-ILF, we compare the accuracy (measured by Micro-F1 and Macro-F1) of LogClass (with TF-ILF) in anomalous log (multiclass) classification, with that of LogClass with TF-IDF. Note that *the parameters of the later two methods are set best for Macro-F1*. Similarly, we train the above methods using the training set $\mathbb{T}$, and test them using the testing set $\mathbb{E}$.

Table XI shows the comparison results of the above methods. Although LogClass with TF-ILF and that with TF-IDF achieve comparable Micro-F1s, the former one has a higher Macro-F1. Specifically, TF-ILF improves the Macro-F1 of LogClass from 96.32% (with TF-IDF) to a very-close-to-one value. That is, the false alarm rate (roughly denoted as 1 – Macro-F1) of TF-IDF (3.68%) is 7.6 times larger than that of TF-ILF (0.43%), demonstrating that TF-ILF greatly reduces the false alarm rate of LogClass. That is because TF-IDF is not robust to all anomaly categories, while the domain knowledge-based TF-ILF method fits for every anomaly category.

F. How Does Log Preprocessing Impact The Performance of LogClass?

To show how log preprocessing impacts the performance of LogClass, we compare the performance of LogClass with that of LogClass without log preprocessing. We use 10% and 50% logs of the BGL dataset to train LogClass, and use the remaining logs in the dataset to test the performance, respectively. Table XII lists the comparison results in terms of accuracy

TABLE XII
THE ANOMALY IDENTIFICATION ACCURACY OF LOGCLASS WITH/WITHOUT PREPROCESSING ON THE BGL DATASET

Training logs	Preprocessing	F1 score	Training time
10%	w/o preprocessing	0.9911	11.70
	w/ preprocessing	0.9918	3.27
50%	w/o preprocessing	0.9999	32.34
	w/ preprocessing	0.9999	8.55

(measured by F1 score) and training efficiency (measured by training time). We can see that log preprocessing impacts little on the accuracy of LogClass. Therefore, the log preprocessing rules, which are predefined by operators based on their domain knowledge, have little effect on the accuracy of LogClass. However, log preprocessing does improve training efficiency and shorten training time. More specifically, the preprocessing procedure improves the training efficiency by 258% to 278%. Consequently, preprocessing logs is vitally important to improve training efficiency, and there is no need to worry about the quality of preprocessing rules impacting the accuracy of LogClass.

V. RELATED WORK

Many works have been proposed to apply logs to detect/identify/classify anomalies for network and service management [7], [8], [11], [13], [21], [23], [43], [47], network security [9], [48]–[51], and process control (SCADA systems) [52]–[54], *etc.* Existing log-based anomaly identification can be classified into many categories, i.e., anomalous log sequence identification methods [7], [9], [11], and anomalous log quantitative relationship identification methods [13]. For the first category, both DeepLog [9] and LogRobust [11] proposed to apply long short-term memory (LSTM) to capture the sequential patterns of normal logs and use the pattern to identify anomalous log sequences. However, neither DeepLog [9] nor LogRobust [11] can be used to detect individual anomalous logs. For those approaches designed to identify anomalous log quantitative relationship, a time or session window is defined, and then the count of each template index (regardless of sequence) within the window is used as the basis for anomaly identification. To achieve this goal, [43] applied PCA, LogCluster [13] and Log3C [8] used clustering algorithms, and Invariants Mining [23] identified whether some mined invariants hold true within the window. Moreover, Shang *et al.* [55] used the development history to measure the amount of log churn for every software component. LogLens [56] try to detect stateful anomalies and stateless anomalies simultaneously. None of the above methods, however, can accurately perform anomaly detection/identification when new types of anomalous logs appear. In addition, Syer *et al.* [5] proposed the first approach on anomalous log identification. It is an automated approach that combines performance counters and execution logs to diagnose memory-related issues in load tests.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

A collection of automatic template extraction approaches have been proposed, which can be classified into four categories [20]. The first category include the longest common subsequence-based methods. For example, Spell [57] used the longest common subsequence algorithm to parse logs in a stream. The second category consists of the frequent item mining-based methods. In these methods, e.g., FT-tree [58], [59], log templates are seen as a set of constant tokens that occur frequently in logs. The methods using heuristics to extract templates constitute the third category. Compared with general text data, log messages have some unique characteristics. Consequently, works such as IPLoM [60] and Drain [61] proposed heuristics-based log parsing methods. Next category, Logram [62] and LogParse [63] leverages n-gram dictionaries to achieve efficient log parsing. The last category, which believes that a log template forms the natural pattern of a group of log messages, applies cluster-based methods to extract templates. For example, LogSig [64] modeled template extraction as a clustering problem. Although the template extraction methods can automatically extract templates from logs, they are not suitable for anomalous log identification or classification.

In natural language processing, the *topic model* is a popular unsupervised method for text classification. LDA, which identifies latent topic information in document collections, is a typical machine learning-based method of topic model [65]. In LDA, each document is represented as a probability distribution over some topics. In addition, each topic is represented as a probability distribution over a number of words. Ramage *et al.* proposed Labeled LDA, a supervised version of LDA [45]. Credit attribution is an inherent problem because although most documents have labels, the tags do not always apply with equal specificity across the whole document. Solving the credit attribution problem requires associating each word in a document with the most appropriate labels and vice versa. Labeled LDA constrains LDA by defining a one-to-one correspondence between LDA's latent topics and user labels. In log analysis domain, [66] applies LDA to study software logs. However, it is not suitable for short text and thus cannot be used in the anomalous log identification.

A preliminary six-page version of this article was published in [67]. This TNSM submission differs a lot from the conference version (less than 9% of the TNSM version is overlapped with the conference version), which can be summarized as follows.

1) We propose a novel, simple yet effective method, TF-ILF, to accurately weight the words of logs in feature vector construction. After a careful investigation on real-world device logs, we observe that both frequency and position are important to assign a weight to a word. Therefore, we design TF-ILF, which does not only count a word's frequency within a log, but also considers the number of positions a word appears in across all logs.
2) We identify a new challenge lying in anomalous log detection and classification, and evaluate how LogClass address it. Specifically, network and service continuously generate new types of logs at runtime because of software/firmware upgrades. The manually maintained rules, which are not updated frequently, cannot detect or classify these new types of logs. LogClass learns the patterns of anomalous individual logs, and uses the similarity of a given log and the above patterns to determine whether this log is anomalous or not. In this way, the above challenge is addressed. We apply switch logs and public log datasets to demonstrate the performance of LogClass in addressing this challenge.
3) We use more datasets and more baseline methods to evaluate the performance of LogClass. In addition to switch logs, we also evaluate LogClass using the public logs collected from BGL (Blue Gene/L supercomputer). To demonstrate LogClass' performance in anomalous log detection, we compare LogClass with existing log-based anomaly detection methods including PCA, Invariants Mining, LogCluster, DeepLog, and LogAnomaly.
4) We have tried our best to improve the presentation quality of this article. After a careful and thorough revision of the paper, we tried our best to improve the paper's organization and clarity.

VI. Conclusion and Future Work

Logs describe a vast range of events, which are extremely valuable for network and service management. However, it is quite difficult to do anomalous log identification and classification because of partial labels and new types of logs. We propose LogClass, a framework to automatically and accurately identify and classify anomalous logs using partial labels. To address the challenge posed by partial labels, LogClass introduces PU learning to identify anomalous logs. LogClass applies a novel, simple yet effective method, TF-ILF, to weight the words of logs. Extensive experiments using real-world switch logs and six public log datasets demonstrate that LogClass achieves superior performance in anomalous log identification and classification in terms of accuracy. For future work, we will compare LogClass with enterprise softwares (e.g., Splunk Enterprise, ANODOT). Moreover, we plan to use word embedding to represent logs and apply the improved work for intrusion identification.

Acknowledgment

The authors thank the anonymous reviewers for their valuable feedback. We also thank Yicheng Sui, Shimin Tao for their helpful suggestions and thank Juexing Liao for proofreading this article.

References

[1] X. Zhang *et al*, "Cross-dataset time series anomaly detection for cloud systems," in *Proc. USENIX Annu. Tech. Conf. (USENIXATC)*, 2019, pp. 1063–1076.
[2] W. Chen *et al.*, "Unsupervised anomaly detection for intricate kpis via adversarial training of VAE," in *Proc. IEEE INFOCOM IEEE Conf. Comput. Commun.*, 2019, pp. 1891–1899.
[3] N. Zhao, J. Zhu, R. Liu, D. Liu, M. Zhang, and D. Pei, "Label-less: A semi-automatic labelling tool for KPI anomalies," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2019, pp. 1882–1890.
[4] H. Xu *et al*, "Unsupervised anomaly detection via variational auto-encoder for seasonal KPIS in Web applications," in *Proc. World Wide Web Conf.*, 2018, pp. 187–196.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

[5] M. D. Syer, Z. M. Jiang, M. Nagappan, A. E. Hassan, M. Nasser, and P. Flora, "Leveraging performance counters and execution logs to diagnose memory-related performance issues," in *Proc. IEEE Int. Conf. Softw. Mainten.*, 2013, pp. 110–119.

[6] W. Meng *et al*, "Summarizing unstructured logs in online services," 2020. [Online]. Available: arXiv:2012.08938.

[7] W. Meng *et al*, "Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th Int. Joint Conf. Artif. Intell. (IJCAI)*, vol. 7, 2019, pp. 4739–4745.

[8] S. He, Q. Lin, J.-G. Lou, H. Zhang, M. R. Lyu, and D. Zhang, "Identifying impactful service system problems via log analysis," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 60–70.

[9] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf.*, 2017, pp. 1285–1298.

[10] S. Zhang *et al.*, "Prefix: Switch failure prediction in datacenter networks," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 2, no. 1, p. 2, 2018.

[11] X. Zhang *et al*, "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2019, pp. 807–817.

[12] S. Khatuya, N. Ganguly, J. Basak, M. Bharde, and B. Mitra, "ADELE: Anomaly detection from event log empiricism," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2018, pp. 2114–2122.

[13] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, 2016, pp. 102–111.

[14] Z. Fu, S. Zhou, and J. Li, "BITFA: A novel data structure for fast and update-friendly regular expression matching," in *Proc. SIGCOMM Posters Demos*, 2017, pp. 130–132.

[15] S. Zhang *et al.*, "FUNNEL: Assessing software changes in Web-based services," *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 34–48, Jan./Feb. 2018.

[16] H. C. Wu, R. W. P. Luk, K. F. Wong, and K. L. Kwok, "Interpreting TF-IDF term weights as making relevance decisions," *ACM Trans. Inf. Syst.*, vol. 26, no. 3, p. 13, 2008.

[17] P. Soucy and G. W. Mineau, "Beyond TFIDF weighting for text categorization in the vector space model," in *Proc. Int. Joint Conf. Artif. Intell.*, 2005, pp. 1130–1135.

[18] R. Vaarandi, B. Blumbergs, and M. Kont, "An unsupervised framework for detecting anomalous messages from syslog log files," in *Proc. IEEE/IFIP Netw. Oper. Manag. Symp. (NOMS)*, 2011, pp. 1–6.

[19] G. Niu, M. C. D. Plessis, T. Sakai, Y. Ma, and M. Sugiyama, "Theoretical comparisons of positive-unlabeled learning against positive-negative learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1199–1207.

[20] J. Zhu, S. He, J. Liu, P. He, and Q. Xie, "Tools and benchmarks for automated log parsing," in *Proc. 41st Int. Conf. Softw. Eng. Softw. Eng. Practice*, 2019, pp. 121–130.

[21] R. Chen, S. Zhang, D. Li, Y. Zhang, and Y. Liu, "LogTransfer: Cross-system log anomaly detection for software systems with transfer learning," in *Proc. IEEE 31st Int. Symp. Softw. Rel. Eng. (ISSRE)*, 2020, pp. 37–47.

[22] S. Kabinna, C.-P. Bezemer, W. Shang, M. D. Syer, and A. E. Hassan, "Examining the stability of logging statements," *Empirical Softw. Eng.*, vol. 23, no. 1, pp. 290–333, 2018.

[23] J. G. Lou, Q. Fu, S. Yang, Y. Xu, and J. Li, "Mining invariants from console logs for system problem detection," in *Proc. Usenix Atc*, 2010, pp. 231–244.

[24] J. Bekker and J. Davis, "Learning from positive and unlabeled data: A survey," *Mach. Learn.*, vol. 109, pp. 719–760, Apr. 2020.

[25] H. Ju, D. Lee, J. Hwang, J. Namkung, and H. Yu, "PUMAD: PU metric learning for anomaly detection," *Inf. Sci.*, vol. 523, pp. 167–183, Jun. 2020.

[26] Y. Kwon, W. Kim, M. Sugiyama, and M. C. Paik, "Principled analytic classifier for positive-unlabeled learning via weighted integral probability metric," *Mach. Learn.*, vol. 109, pp. 513–535, Nov. 2019.

[27] Y. Luo, S. Cheng, C. Liu, and F. Jiang, "PU learning in payload-based Web anomaly detection," in *Proc. IEEE 3rd Int. Conf. Security Smart Cities Ind. Control Syst. Commun. (SSIC)*, 2018, pp. 1–5.

[28] J. Zhang, Z. Wang, J. Yuan, and Y.-P. Tan, "Positive and unlabeled learning for anomaly detection with multi-features," in *Proc. 25th ACM Int. Conf. Multimedia*, 2017, pp. 854–862.

[29] T. Sakai, M. C. Plessis, G. Niu, and M. Sugiyama, "Semi-supervised classification based on classification from positive and unlabeled data," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2998–3006.

[30] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu, "An evaluation study on log parsing and its use in log mining," in *Proc. IEEE/IFIP Int. Conf. Depend. Syst. Netw.*, 2016, pp. 654–661.

[31] T Kimura, A. Watanabe, T. Toyono, and K. Ishibashi, "Proactive failure detection learning generation patterns of large-scale network logs," In *Proc. CNSM*, 2015, pp. 8–14.

[32] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. SOSP*, 2009, pp. 37–46.

[33] A. Sun, "Short text classification using very few words," in *Proc. ACM 35th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2012, pp. 1145–1146.

[34] M. Weibin *et al.*, "A semantic-aware representation framework for online log analysis," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2020, pp. 1–7.

[35] G. Ward, T. Hastie, S. Barry, J. Elith, and J. R. Leathwick, "Presence-only data and the EM algorithm," *Biometrics*, vol. 65, no. 2, pp. 554–563, 2009.

[36] C. Elkan and K. Noto, "Learning classifiers from only positive and unlabeled data," in *Proc. ACM SIGKDD*, 2008, pp. 213–220.

[37] D. Liu *et al.*, "Opprentice: Towards practical and automatic anomaly detection through machine learning," in *Proc. Internet Meas. Conf.*, 2015, pp. 211–224.

[38] S. Suthaharan, "Support vector machine," in *Machine Learning Models and Algorithms for Big Data Classification*, Boston, MA, USA: Springer, 2016, pp. 207–235.

[39] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[40] K. P. Murphy *et al*, *Naive Bayes Classifiers*. Univ. British Columbia, Vancouver, BC, Canada, 2006.

[41] R. E. Wright, *Logistic Regression*. Washington, DC, USA: Amer. Psychol. Assoc., 1995.

[42] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li, "RCV1: A new benchmark collection for text categorization research," *J. Mach. Learn. Res.*, vol. 5, no. 2, pp. 361–397, 2004.

[43] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan, "Largescale system problem detection by mining console logs," in *Proc. SOSP*, 2009.

[44] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE)*, 2016, pp. 207–218.

[45] D. Ramage, D. Hall, R. Nallapati, and C. D. Manning, "Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora," in *Proc. EMNLP*, 2009, pp. 248–256.

[46] C. Rosenberg, M. Hebert, and H. Schneiderman, "Semi-supervised self-training of object detection models," in *Proc. WACV/MOTION*, 2005, pp. 29–36.

[47] W. Xu, L. Huang, A. Fox, D. A. Patterson, and M. I. Jordan, "Mining console logs for large-scale system problem detection," in *Proc. SysML*, vol. 8, 2008, p. 4.

[48] R. Yang, D. Qu, Y. Gao, Y. Qian, and Y. Tang, "nLSALog: An anomaly detection framework for log sequence in security management," *IEEE Access*, vol. 7, pp. 181152–181164, 2019.

[49] M. E. DeYoung, R. C. Marchany, and J. G. Tront, *Network Security Data Analytics Architecture for Logged Events*, Virginia Tech, Blacksburg, VA, USA, Jan. 2017.

[50] Z. Luo, Z. Qu, T. Nguyen, H. Zeng, and Z. Lu, "Security of HPC systems: From a log-analyzing perspective," *EAI Endorsed Trans. Security Safety*, vol. 6, no. 21, p. e5, 2019.

[51] R. Tang *et al.*, "ZeroWall: Detecting zero-day Web attacks through encoder–decoder recurrent neural networks," in *Proc. INFOCOM*, 2020, pp. 2479–2488.

[52] A. Rahman, Y. Xu, K. Radke, and E. Foo, "Finding anomalies in scada logs using rare sequential pattern mining," in *Proc. Int. Conf. Netw. Syst. Security*, 2016, pp. 499–506.

[53] Y. Cui, P. Bangalore, and L. B. Tjernberg, "An anomaly detection approach based on machine learning and scada data for condition monitoring of wind turbines," in *Proc. IEEE Int. Conf. Probabil. Methods Appl. Power Syst. (PMAPS)*, 2018, pp. 1–6.

[54] A. Rahman, "Rare sequential pattern mining of critical infrastructure control logs for anomaly detection," Ph.D. dissertation, Sci. Eng. Faculty, Queensland Univ. Technol., Brisbane, QLD, Australia, 2019.

[55] W. Shang, "Bridging the divide between software developers and operators using logs," in *Proc. IEEE 34th Int. Conf. Softw. Eng. (ICSE)*, 2012, pp. 1583–1586.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

[56] B. Debnath *et al.*, "LogLens: A real-time log analysis system," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, 2018, pp. 1052–1062.
[57] M. Du and F. Li, "SPELL: Streaming parsing of system event logs," in *Proc. IEEE 16th Int. Conf. Data Min. (ICDM)*, 2016, pp. 859–864.
[58] S. Zhang *et al*, "Syslog processing for switch failure diagnosis and prediction in datacenter networks," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS)*, 2017, pp. 1–10.
[59] S. Zhang *et al*, "Efficient and robust syslog parsing for network devices in datacenter networks," *IEEE Access*, vol. 8, pp. 30245–30261, 2020.
[60] A. A. O. Makanju, A. N. Zincir-Heywood, and E. E. Milios, "Clustering event logs using iterative partitioning," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Disc. Data Min.*, 2009, pp. 1255–1264.
[61] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "DRAIN: An online log parsing approach with fixed depth tree," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, 2017, pp. 33–40.
[62] H. Dai, H. Li, W. Shang, T.-H. Chen, and C.-S. Chen, "LogRam: Efficient log parsing using N-GRAM dictionaries," 2020. [Online]. Available: arXiv:2001.03038.
[63] W. Meng *et al.*, "LogParse: Making log parsing adaptive through word classification," in *Proc. IEEE 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, 2020, pp. 1–9.
[64] L. Tang, T. Li, and C.-S. Perng, "LogSig: Generating system events from raw textual logs," in *Proc. 20th ACM Int. Conf. Inf. Knowl. Manag.*, 2011, pp. 785–794.
[65] P. Gupta, Y. Chaudhary, F. Buettner, and H. Schütze, "Document informed neural autoregressive topic models with distributional prior," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 6505–6512.
[66] H. Li, T.-H. P. Chen, W. Shang, and A. E. Hassan, "Studying software logging using topic models," *Empirical Softw. Eng.*, vol. 23, no. 5, pp. 2655–2694, 2018.
[67] W. Meng *et al.*, "Device-agnostic log anomaly classification with partial labels," in *Proc. Qual. Service (IWQoS)*, 2018, pp. 1–10.

Weibin Meng (Graduate Student Member, IEEE) received the B.S. degree in software engineering from Jilin University, Changchun, China, in 2016. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University. He is also with the Beijing National Research Center for Information Science and Technology. His research interests include anomaly detection, log analysis, and failure prediction in datacenter networks.

Ying Liu (Member, IEEE) received the B.S. degree in information engineering, the M.S. degree in computer science and the Ph.D. degree in applied mathematics from Xidian University in 1995, 1998, and 2001, respectively. She was a Postdoctoral Researcher with the Department of Computer Science and Technology, Tsinghua University from 2001 to 2003, where she is a Professor with the Institute for Network Sciences and Cyberspace, Tsinghua University. She is also with the Beijing National Research Center for Information Science and Technology. Her research interests include multicast routing, network architecture and router design and implementation.

Shenglin Zhang (Member, IEEE) received the B.S. degree in network engineering from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2012 and the Ph.D. degree in computer science from Tsinghua University, Beijing, China, in 2017. He is currently an Assistant Professor with the College of Software, Nankai University, Tianjin, China. His current research interests include failure detection, diagnosis, and prediction in data center networks.

Federico Zaiter received the B.S. degree in software systems engineering from Universidad ORT Uruguay, Montevideo, Uruguay, in 2017. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University. He is also with the Beijing National Research Center for Information Science and Technology. His current research interest includes syslogs analysis for anomaly detection and troubleshooting within AIOps.

Yuzhe Zhang is currently pursuing the undergraduate degree with the Collage of Software, Nankai University, Tianjin, China. His current research interests include anomaly detection, deep learning, and NLP.

Yuheng Huang is currently pursuing the undergraduate degree with the School of Computer science, Beijing University of Posts and Telecommunications, Beijing, China. His current research interests including deep learning, NLP, and data analysis.

Zhaoyang Yu is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Tsinghua University. He is also with the Beijing National Research Center for Information Science and Technology. His current research interests include machine learning, time series anomaly detection and automatic log summary extraction. He is currently an intern with Tsinghua NetMan Lab.

Yuzhi Zhang received the B.S. and M.S. degrees in computer science from the Department of Computer Science and Technology, Tsinghua University in 1985 and 1987, respectively, and the Ph.D. degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences in 1991. He is currently the Dean of the College of Software, Nankai University, and is also an Distinguished Professor. His research interests include deep learning and other aspects in artificial intelligence.

Lei Song received the B.S. degree in information security from the School of Computer Science, Wuhan University, Wuhan, China, in 2009. He is currently a Senior Engineer with Baidu, Inc.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.

Ming Zhang received the B.S. degree in computer software from the School of Computer Science, Beijing University of Technology, Beijing, China, in 2001. He is currently an Engineer with China Construction Bank.

Dan Pei (Senior Member, IEEE) received the B.E. and M.S. degree in computer science from the Department of Computer Science and Technology, Tsinghua University in 1997 and 2000, respectively, and the Ph.D. degree in computer science from the Computer Science Department, University of California, Los Angeles, Los Angeles, CA, USA, in 2005. He is currently an Associate Professor with the Department of Computer Science and Technology, Tsinghua University. He is also with the Beijing National Research Center for Information Science and Technology. His research interests include network and service management in general. He is an ACM Senior Member.

Authorized licensed use limited to: NANKAI UNIVERSITY. Downloaded on June 23,2021 at 06:57:21 UTC from IEEE Xplore. Restrictions apply.