Device-Agnostic Log Anomaly Classification with Partial Labels

Webin Meng†‡, Ying Liu†‡, Shenglin Zhang†*, Dan Pei†‡, Hui Dong§, Lei Song§, Xulong Luo§
†Tsinghua University ‡Nankai University §Baidu

Email: mwb16@mails.tsinghua.edu.cn, liuying@cernet.edu.cn, zhangsl@nankai.edu.cn, peidan@tsinghua.edu.cn, {dohui02, song_lei, luoxulong}@baidu.com

Abstract—Anomaly classification, i.e., detecting whether a network device is anomalous and determining its anomaly category if yes, plays a crucial role in troubleshooting. Compared to KPI curves, device logs contain too much more valuable information for anomaly classification. However, the regular expression based anomaly classification techniques cannot tackle the challenges lying in log anomaly classification.

We propose LogClass, a data-driven framework to detect and classify anomalies based on device logs. LogClass combines a word representation method and the PU learning model to construct device-agnostic vocabulary with partial labels. We evaluate LogClass on tens of millions of switch logs collected from several real-world datacenters owned by a top global search engine. Our results show that LogClass achieves 99.515% F1 score in anomalous log detection, 95.32% Macro-F1 and 99.74% Micro-F1 in anomalous log classification in a computationally efficient manner.

Index Terms—Device Logs, Anomaly Classification, TF-IDF

I. INTRODUCTION

With the explosion of the traffic in datacenter networks, the number of network devices including switches, routers and middleboxes (say VPNs, IDPS, and firewalls) [1] have witnessed a dramatic increase. Since network device anomalies can significantly impact on the services provided by datacenters [2], operators continuously monitor the status of network devices carefully, and intervene immediately after an anomaly occurs. In the literature, many anomaly detection methods have been proposed to detect anomalies in network devices by means of analyzing KPI (Key Performance Indicator, e.g., CPU utilization, memory utilization) curves [3].

In addition to detecting anomalies, operators also have to classify anomalies in order to rapidly locate the root cause and mitigate the damage imposed by anomalies. For example, if a severe anomaly occurs on a line card of a switch, operators will replace the line card to mitigate the loss. On the other hand, operators will reboot the system when a switch suffers from software crash. Although KPI curves can answer whether a network device is anomalous, anomaly classification barely benefits from KPI curves, for the reason that KPI curves usually contain too little information. Specifically, an anomaly (say a level sift) in the CPU utilization only indicates that the CPU utilization increases sharply, and it cannot tells why it happens. On the contrary, device logs describe a vast range of (anomalous) events, which are quite valuable for root cause analysis and log classification. For instance, when the switch log of “System is rebooting now” is generated, operators can determine that the switch is anomalous, and classify the anomaly into the category of “SYSTEM_REBOOT”. Device logs have been extensively studied in monitoring network status [4], understanding network events [5], [6], detecting anomalies [7], [8], and predicting device failures [9].

Due to the fundamental role of device logs in anomaly classification, operators usually classify device logs into healthy logs and (multiple types of) anomalous logs using regular expression. However, the regular expression based log classification suffers from low generality (i.e., it is device manufacturer/model specific), labor intensity (i.e., maintaining the large number of regular expressions is labor intensive) and computational inefficiency [10], [11].

In this paper, we propose LogClass, a data-driven system to detect and classify anomalies based on device logs. The key intuition is that most device logs are semi-structured texts “printf”-ed by device operating systems, and traditional methods in natural language processing can be applied to analyze device logs.

LogClass faces two interesting challenges as follows.

(1) Device-agnostic vocabulary. Since device logs are manufacture/model-specific, their formats vary among different manufacturers and device models. That is, it is very difficult, if possible, to train a single classification model for all device manufacturers/models. None of the existing log processing techniques can learn device-agnostic vocabulary for log classification. For example, FT-Tree [9] is only able to learn log templates one switch manufacturer/model by one.

(2) Partial labels. Due to the huge number of device logs (tens of millions), operators have detected/classified only a small portion of anomalous logs, and thus most of the anomalous logs remains undetected and unclassified. This poses a great challenge to log detection and classification.

The challenges mentioned above are tackled by LogClass as follows.

(1) Our preliminary investigation demonstrates that device logs of the same anomaly category share some common patterns in terms of word combination. In natural language processing, the bag-of-words model is a popular and effective
word representation method [12]. In this paper, we use bag-of-words to represent the patterns of word combinations.

(2) LogClass applies PU Learning [13] to handle the partial labels. Specifically, it inputs anomalous (positive) logs and unlabeled logs into the PU learning model, which then outputs which logs are anomalous and which ones are healthy.

The overall design of LogClass is as follows. For all the devices of different manufacturers/models, in the offline learning procedure, LogClass pre-processes the device logs and generates bag-of-words vectors. Then LogClass applies PU Learning [13] and SVM to train an anomaly detection and classification model. Similarly, in the online detection procedure, LogClass determines whether a new log is anomalous and its anomaly category if yes.

We evaluate LogClass on tens of millions of switch logs collected from several real-world datacenters owned by a top global search engine. Our results show that LogClass achieves 99.515% F1 score in anomalous log detection, 95.32% Macro-F1 and 99.74% Micro-F1 in (anomalous) log classification in an computationally efficient manner.

The rest of the paper is organized as follows: Section II provides the background of our problem, and the design of LogClass is elaborated in Section III. The evaluation is shown in Section IV. Finally, we introduce the related works in Section V and conclude our work in Section VI.

II. BACKGROUND

At present, datacenter networks compris a variety of network devices, such as switches, routers [14]. Network operators are required to monitor network by monitoring all devices and detect even a slight device anomaly.

A. Device Logs

Many devices have similar log fields. In this paper, we take the switch logs as examples to depict the each field of device logs. Table I shows several examples of switch logs. As the table shows, a switch log message usually has a primitive structure containing several fields. Network operators usually care about the detailed messages. In this paper, we utilize detailed messages to classify device logs.

B. Anomalous Devices Logs Classification

We provide the definitions of logs and device anomaly, which are similar to the definition of device failure in [15].

- **Device anomaly.** When the service (such as traffic forwarding) provided by a device deviates from the correct service, a device anomaly occurs.

- **Anomaly category.** Each device anomaly belongs to a specific category, which is the cause of this anomaly. Typical categories include reboot, packet loss etc.

- **Anomalous logs.** The anomalous logs are associated with device anomalies. Each anomalous log belongs to a specific category (e.g., packet loss).

- **Healthy logs.** The logs are not associated with device anomalies. They usually describe the routine status.

![Fig. 1: Problem Definition](image1)

![Fig. 2: The Framework of LogClass](image2)

- **Unlabeled logs.** The logs that network operators do not know whether they relate to device anomalies.

In practice, network operators usually use keywords to mark anomalous logs, but many words have ambiguities. For instance, if a network device tries to PING another device, a *loss ratio* will be recorded in the syslog message as "packets : sent = 5, received = 5, lost = 0 (0% loss)". The popular method in most companies is the Regular Expression (RE). However, the Regular Expression approach has several drawbacks for log classification. First, a RE is rigorous. It cannot match text even if existing only an additional whitespace or symbol. Besides, configuring regular expressions need heavy workload. More especially, when new devices are used, network operators need to update regular expressions.

Finally, we define the problem of anomalous log classification. There are tens of millions of logs. It is infeasible to ask network operators to identify all anomalous logs. Some anomalous logs are labeled by network operators, but a few anomalous logs still mingle in rest logs. We aim to find the pattern of anomalous logs based on existing device logs and classify anomalous logs during runtime. Fig 1 shows the problem definition of anomalous log classification.

III. DESIGN OF LOGCLASS

In this section, we present the LogClass system.
because of their low utility for log classification. What’s more, an IP belongs to a specific device. We filter parameter words by templates. Therefore, preprocessing logs is necessary.

We have three observations for the device logs. First, there are many parameters in logs, such as IP, interface etc. In addition, most device logs are semi-structured natural languages provided by device developers and produced by “print” program. Finally, network operators cannot label all anomalous logs. According to above observations, we proposed LogClass, a data-driven system to detect and classify anomalies based on device logs. Fig. 2 gives an overview of LogClass. To achieve our goal, the system first preprocesses logs (Section III-B). It then calculates log feature vectors (Section III-C). Because of partial labels, we utilize PU Learning to detect anomalous logs (Section III-D). Finally, LogClass train a multiclass classifier (Section III-D). We now give a detailed description for each component.

### B. Log Preprocessing

From Table I, we can see that there are many parameters in logs, such as IP, interface etc.. Different device manufacturers and device types have no unified format for log messages. [9], [16] proposed methods to extract per-device type log templates. However, their methods need to learn templates for each device type. It results in that we cannot filter parameters by templates. Therefore, preprocessing logs’ is necessary.

Intuitively, the text features in the log are mainly nouns, verbs etc. The parameters with numerical words or specific symbols, such as IPs, have little effect on log classification. Although several prior works [9], [16]–[18] have been proposed to extract log template (also known as “log key”), they all have some drawbacks. For example, [18] utilizes source code of generating logs to extract log templates, but not all source codes can be acquired.

In this paper, we classify the words into two categories, text words and parameter words. The text words are similar to the template words, which are mentioned in [9], [16]. The detailed definition of classification of words is described in Table II. Parameter words are usually related to specific devices (e.g., an IP belongs to a specific device). We filter parameter words because of their low utility for log classification. What’s more, filtering parameter words can reduce vocabulary. After log preprocessing, we get device-agnostic words from logs.

### C. Feature Vectors

After log preprocessing, LogClass attempts to construct a feature vector for each log. The universal method to construct a text feature vector is the bag-of-words expression, which is used in the natural language processing area [19]. Generally, each vector component is assigned a value related to estimated importance (also known as weight) of words in logs [20].

**TF-IDF** is the most popular weighting method used to describe documents [20]. Regarding bag-of-words representation, each feature vector component of each document relates to a word of the vocabulary. The TF-IDF method weights each vector component on the following basis. First, it involves the word frequency in the document, which is represented by TF (the term frequency). The more a word appears in a document, the more it is estimated to be significant in this document. Simultaneously, IDF measures how infrequent a word is in the total collected documents. This IDF value is estimated using the whole training text collection at hand. If a word appears in the total collected documents frequently, it is not regarded to be especially representative of these documents. For instance, stop words (e.g., a, and, or) nearly appear in all documents, but they are of little help to classification. In other words, if a word is infrequent in the total documents, it is believed to be very relevant for a particular category of documents. In this scenario, documents are logs.

### D. Machine Learning based Training and Classifying

After generating feature vectors, we adopt PU Learning [13] to handle the partial labels and detect the anomalous logs from all unlabeled logs. Then, LogClass utilizes a multiclass classifier to classify the anomalous logs into each specific category they belong to. We introduce PU Learning method and classifiers we adopt as follows.

1) **PU Learning:** Let $x$ be an example and let $y \in \{0, 1\}$ be a binary label. Let $s = 1$ if the example $x$ is labeled, and let $s = 0$ if $x$ is unlabeled. Only positive examples are labeled. So $y = 1$ is certain when $s = 1$, but when $s = 0$, then either $y = 1$ or $y = 0$ may be true, which can be stated formally by the equation

$$p(s = 1|x, y = 0) = 0$$

Device anomalies occur randomly and anomalous logs are randomly labeled. Stated formally, the assumption is that

$$p(s = 1|x, y = 1) = p(s = 1|y = 1)$$

The goal is to learn a function $f(x)$ such that $f(x) = p(y = 1|x)$ as closely as possible, which is a traditional probabilistic
classifier. PU Learning also yields a function $g(x)$ such that $g(x) = p(s = 1|x)$ approximately, which a nontraditional classifier. Their core result is the following equation that shows how to obtain a traditional classifier $f(x)$ from $g(x)$.

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

(3)

Where $c = p(s = 1|y = 1)$ is the constant probability that a positive example is labeled. Let $V$ be such a validation set that is drawn from the overall distribution $p(x,y,s)$ in the same manner as the nontraditional training set. Let $P$ be the subset of examples in $V$ that are labeled. [13] has mentioned many estimators for the constant $c$, one estimator of them is as follows

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

(4)

So far, we can transform a traditional binary classifier to a PU Learning classifier, which is trained by anomalous logs and unlabeled logs.

2) Binary and Multiclass classifiers: Firstly, we try to choose algorithm for binary classification. In general, the number of anomalous logs is smaller than that of unlabeled logs. We need to choose a binary classification algorithm for the imbalanced problem. RandomForest (RF) is a well-known powerful ensemble learning algorithm. The RF constructs a multitude of decision trees, and learns patterns and makes decisions based on voting, and thus the imbalance problem of samples impacts little on RF [2]. Then, the anomalous logs which are detected by binary classifier need be classified into their categories. We choose SVM as our multi-classification algorithm. When the SVM model has been trained, we can get the coefficients assigned to the features (also known as weights in the primal problem). The $coef$, for instance, an attribute of the scikit-learn package [21] is the coefficient for trained model. Moreover, the top $n$ important words can be extracted by sorting these coefficients. Network operators may comprehend each category by its top $n$ important words.

As shown in Fig.2, in the offline training component, we train the PU binary classifier and multiclass classifier. In the online component, when a new device log comes, LogClass will generate feature vectors for PU classifier. If this log is considered as anomalous, the multiclass classifier will find the category it belongs to.

IV. Evaluation

In this section, we conduct experiments on a real-world switch logs to demonstrate the effectiveness of LogClass. Switches forward traffic from servers to higher level routers, and play a fundamental role in the datacenter networks [9]. All the experiments were conducted on a Linux server with Intel Xeon 2.40 GHz CPU and 64G memory. We implemented LogClass with Python 2.7. We take traditional RandomForest and Labeled-LDA (L-LDA) [22] as our baseline methods. L-LDA is a typical text classification technique. For Labeled LDA, we used the open-source implementation MALLET [23].

A. Data sets

In cooperation with network operators, we collected switch logs over 2 weeks period from 58 types of switches across more than 10 datacenters owned by a top global search engine. This data set consists of logs and anomaly labels. These datacenters use the RE to classify anomalous switch logs.

TABLE III: Detailed information for the switch log dataset

<table>
<thead>
<tr>
<th>Duration</th>
<th># switch types</th>
<th># anomalous device logs</th>
<th># unlabeled device logs</th>
<th># anomaly categories</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 weeks</td>
<td>58</td>
<td>1,758,458</td>
<td>16,702,547</td>
<td>12</td>
</tr>
</tbody>
</table>

Table III shows the detailed information of this data set.

TABLE IV: Names for Anomaly Categories

<table>
<thead>
<tr>
<th>PROTOCOL</th>
<th>BOARD_DISABLE</th>
<th>SYSTEM_REBOOT</th>
<th>INTERFACE_DOWN</th>
<th>PORT_DOWN</th>
<th>PROTOCOL_DOWN</th>
<th>OSPF_DOWN</th>
<th>OSPF_NEIGHBOR_CHANGED</th>
<th>OSPFQed_MEMBER_CHANGED</th>
<th>OSPFNeighbor_CHANGED</th>
<th>FAN_RECOVERED</th>
<th>FAN_FAILED</th>
<th>BOARD_DISABLE</th>
<th>BOARD_DOWN</th>
<th>SYSTEMUTDOWN</th>
<th>INTERFACE_DOWN</th>
<th>PORT_DOWN</th>
<th>PROTOCOL_DOWN</th>
<th>OSPF_DOWN</th>
<th>OSPF_NEIGHBOR_CHANGED</th>
<th>OSPFQed_MEMBER_CHANGED</th>
<th>OSPFNeighbor_CHANGED</th>
<th>FAN_RECOVERED</th>
<th>FAN_FAILED</th>
</tr>
</thead>
</table>

B. Evaluation on Anomalous Log Detection

The first step of LogClass is anomaly detection. According to [13], we define four sets. The anomalous logs labeled by operators belong to the set of anomalous examples $P$. The set of unlabeled examples is $U$. The actual anomalous examples inside $U$ are called subset $Q$. Finally, let $N = U - Q$. A classification method’s capability is usually assessed by three metrics that have intuitive interpretations, i.e., Precision, Recall, F1 score. For each method, we label its outcome as a true positive ($TP$), true negative ($TN$), false positive ($FP$), and false negative ($FN$). True positives are the logs belonging to $P$ that are accurately determined. The true negatives are logs belonging to $N$ that are accurately determined. If the method determines a log as an anomalous log, but in fact it belongs to $N$, we then labeled the outcome as a false positive. The rest are false negatives. We calculated Precision, Recall, and F1 score as follows: $Precision = \frac{TP}{TP+FP}$, $Recall = \frac{TP}{TP+FN}$, $F1 score = \frac{2 \times Precision \times Recall}{Precision + Recall}$.

First, we use LogClass to do anomaly detection based on the switch logs. We used a 10-fold cross-validation model to evaluate these methods [24]. The first experiment in Table V shows 10-fold cross-validation results of anomalous log detection for whole dataset, from which we can see that the LogClass can detect the anomalous logs accurately.

To demonstrate LogClass can handle device-agnostic logs, we assume that 5 types of switches\(^1\) are newly added switches. Then, we use LogClass to detect anomalous logs from these switches. The second experiment in Table V shows the results anomalous log detection for newly added switches. We observe that even logs of new-type switches do not appear before, LogClass can still get great anomaly detection performance.

Table VI shows an actual detection case of LogClass. L2 is

\(^1\)Ten percent of all logs are generated by these types of switches.
TABLE V: The results of switch logs for anomalous log detection

<table>
<thead>
<tr>
<th>Experiment No.</th>
<th>Training Set</th>
<th>Testing Set</th>
<th>Precision</th>
<th>Recall</th>
<th>F1 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10-fold cross validation for all logs</td>
<td>99.048%</td>
<td>99.988%</td>
<td>99.515%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>53 types of switches (90% of all logs)</td>
<td>99.081%</td>
<td>99.132%</td>
<td>99.106%</td>
<td></td>
</tr>
</tbody>
</table>

TABLE VI: An actual anomalous log detection case of the LogClass

<table>
<thead>
<tr>
<th>Log No.</th>
<th>Switch Log</th>
<th>Historical Label</th>
<th>LogClass Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>L1</td>
<td>Interface TenGigabitEthernet 1/0/30 is protocol down.</td>
<td>anomalous</td>
<td>anomalous</td>
</tr>
<tr>
<td>L2</td>
<td>Interface TenGigabitEthernet 1/0/12 is link down.</td>
<td>unlabeled</td>
<td>anomalous</td>
</tr>
</tbody>
</table>

A multi-class classification method’s capability is usually assessed by Micro-F1 and Macro-F1 [25]. The Micro-F1 allows larger classes to dominate its results, the Macro-F1 assigns an equal weight to all classes, providing us complementary information. We also score each method based on training time and classifying time. While training time and classifying time are important for the model update and operation.

Table VII shows the comparison between LogClass, L-LDA and RE. We observe that the Macro-F1 and Micro-F1 of LogClass are larger than L-LDA’s, especially for Macro-F1. The low Macro-F1 means that L-LDA cannot classify logs accurately for several categories. We also observe that the training time and classifying time of L-LDA are 17.91 and 5.91 times longer than LogClass obtained respectively. We used the 150 real-world regular expressions\(^2\) to classify logs in 419.47 seconds, which is 86.74 times longer than LogClass's classifying time. Therefore, the overheads of L-LDA and RE are much larger than LogClass.

TABLE VII: The Macro-F1, Micro-F1, Training Time and Classifying Time for LogClass, L-LDA and RE

<table>
<thead>
<tr>
<th>Methods</th>
<th>Macro-F1</th>
<th>Micro-F1</th>
<th>Training Time(s)</th>
<th>Classifying Time(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LogClass</td>
<td>95.32%</td>
<td>99.74%</td>
<td>247.73</td>
<td>4.836</td>
</tr>
<tr>
<td>L-LDA</td>
<td>89.68%</td>
<td>93.53%</td>
<td>4436.4</td>
<td>28.59</td>
</tr>
<tr>
<td>RE</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>419.47</td>
</tr>
</tbody>
</table>

Finally, as described in Section III-D2, LogClass can extract top-n important words for each anomaly category. In Table VIII, we shows top 5 important words of several anomaly categories to display the interpretability of LogClass. These words were manually confirmed by network operators, and considered as accurate words.

TABLE VIII: Top 5 important words for INTERFACE_DOWN category

<table>
<thead>
<tr>
<th>Logs</th>
<th>Top-5 words</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interface TenGigabitEthernet 1/0/30 is down.</td>
<td>interface, down, state, GigabitEthernet, link</td>
</tr>
<tr>
<td>Interface te-1/1/56, changed state to down</td>
<td>GigabitEthernet 1/0/22: changed status to down</td>
</tr>
</tbody>
</table>

\(^2\)In our dataset, there are 150 regular expressions for each type of switch on average.
V. RELATED WORK

Log parsing techniques for network devices including routers and switched have been well studied in [5], [9], [16], [26]. Specifically, Qiu et al. proposed a log template extraction technique [16]. For log messages that belong to a given message type, the technique constructs a signature tree. However, the Signature Tree is not incrementally retrainable. More recently, Zhang et al. proposed FT-Tree technique [9]. The FT-Tree is more incrementally retrainable for extracting log template. Kimura et al. presented an STE approach that extracts log templates using a statistical clustering algorithm [5].

The high-level idea is that template words appear more frequently than parameter words. In NLP scenario, the Topic Model is a popular method for text classification. LDA is a technique of Topic Model which identifies latent topic information in document collections [27]. Then, Ramage et al. proposed Labeled LDA (L-LDA), a supervised version of LDA [22]. Credit attribution is an inherent problem corporate because most documents have labels, but the tags do not always apply with equal specificity across the whole document.

VI. CONCLUSION

Devices logs describe a vast range of events, which are extremely valuable in network device management (say anomaly detection and troubleshooting). However, it is quite difficult to apply device logs in anomaly detection and classification because of the huge amount of logs, diverse manufacturers/models of devices, and partial labels. We propose LogClass, a data-driven framework to classify anomalies based on device logs. LogClass applies the bag-of-word model to represent the word combinations of device logs. To address the challenge posed by partial labels, LogClass introduces PU learning to detect anomalous logs. Extensive experiments using real-world data has demonstrated that LogClass achieves excellent performance.

VII. ACKNOWLEDGMENT

We thank the anonymous reviewers for their valuable feedback. We also thank Yu Sa, Daqing Wu and Yuanye Zhu for their helpful suggestions. The work was supported by National Natural Science Foundation of China (NSFC) under grant No.61402257, No. 61472214 and No. 61472210, the National Key Basic Research Program of China (973 program) under grant No. 2013CB329105, the Global Talent Recruitment (Youth) Program, and the Cross-disciplinary Collaborative Teams Program for Science, Technology and Innovation, of Chinese Academy of Sciences Network and system technologies for security monitoring and information interaction in smart grid.

REFERENCES