LogClass: Log Anomaly Classification for Network Devices in Web Service Datacenters

Weibin Meng, *Student Member, IEEE,* Ying Liu, *Member, IEEE,* ShengLin Zhang, *Member, IEEE,* Dan Pei, *Senior Member, IEEE,* Yuzhi Zhang, Hui Dong, Lei Song, Xulong Luo, Ming Zhang

Abstract—Anomaly classification, namely, detecting whether a network device in datacenter networks is anomalous, and if yes, determining its anomaly category, plays a crucial role in troubleshooting. Compared to metrics (time series), device logs contain much more valuable information for anomaly classification. However, the commonly used regular expression technique cannot tackle the challenges lying in anomalous log classification in terms of device-agnostic vocabulary, partial labels and word weighting. To address these challenges, we propose LogClass, a framework to automatically and robustly detect and classify anomalous logs for diverse manufacturers/models of network devices in datacenter networks based on *partial labels*. LogClass combines a word representation method and the PU learning model to construct a device-agnostic vocabulary based on partial labels. In addition, we propose a novel Term Frequency Inverse Location Frequency (TF-ILF) method to properly weight the words of device logs in feature construction. We evaluate LogClass on more than 18 million real-world switch logs collected from 6547 switches (of 58 manufacturers/models) deployed in 10+ datacenters owned by a top global search engine over a 2-week period. Our experiments show that LogClass achieves very-close-to-one accuracy (measured by F1 score) in both log anomaly detection and classification, and TF-ILF improves the accuracy (measured by Macro-F1) from 93.31% to 99.57% compared to a state-of-the-art method.

Index Terms—web services, device log, anomaly detection, anomaly classification, troubleshooting

1 Introduction

With the rapid development of the Internet, web services have penetrated into all aspects of society. Datacenter networks are rapidly evolving technology architectures allowing users to tap into a variety of web services in an extremely effcient and cost effective manner. Datacenter network consists of network devices. With the explosion of the web services traffic in datacenter networks, the number of network devices including switches, routers and middleboxes (say VPNs, IDPS, and firewalls) have witnessed a dramatic increase [1]. Since network device anomalies can significantly impact the services provided by the datacenter, operators continuously monitor the status of network devices [2], [3], [4]. In general, operators have to classify anomalies in order to rapidly locate the root cause and mitigate the damage imposed by anomalies.

Although KPI (key performance indicator, *e.g.*, CPU utilization, memory utilization) curves can answer whether a network device is anomalous [5], they are of little help for anomaly classification, for the reason that KPI curves usually contain too little information. Specifically, an anomaly (say a level sift) in the curve of CPU utilization only indicates that the CPU utilization increases sharply, but it cannot tell why it happens. On the contrary, device logs describe a vast range of (anomalous) events, which are quite valuable for anomaly classification and troubleshooting. For example, when a switch generates a log of "System is rebooting now", operators can determine that the switch is anomalous, and classify this anomaly into the category of "SYSTEM_REBOOT".

Due to the fundamental role of device logs in anomaly classification, in practice operators usually classify device logs into *healthy logs* and (multiple categories of) *anomalous logs* using regular expressions. However, the regular expression based log classification suffers from low recall (*i.e.,* large amounts of anomalous logs remain undetected due to the incompleteness of regular expressions), low generality (*i.e.,* it is device type[1] specific), high labor intensity (*i.e.*, maintaining the large number of regular expressions consumes operators a lot of time) and computational inefficiency [6], [7]. Although in the literature device logs have been extensively studied for monitoring network status [8], understanding network events [9], detecting anomalies [10], [11], and predicting device failures [12], log based anomaly classification has not yet been investigated.

In this work, we try to *automatically and accurately* (detect and) classify anomalous logs for network devices into multiple categories in datacenter networks. However, it faces three great challenges as follows.

1) *Device-agnostic vocabulary*. Since device logs are type-specific, their formats vary across different manufacturers and device models. That is, it is very difficult, if possible, to train a single classification

- *S. Zhang is the correspondence author.*
- *W. Meng, Y. Liu and D. Pei are with Tsinghua University and Beijing National Research Center for Information Science and Technology (BNRist), Beijing 100084, China. E-mail: mwb16@mails.tsinghua.edu.cn, liuying@cernet.edu.cn, peidan@tsinghua.edu.cn.*
- *S. Zhang and Y. Zhang are with Nankai University, Tianjin, 300457, China. E-mail: {zhangsl, zyz}@nankai.edu.cn.*
- *L. Song, X. Luo are with Baidu, Inc., Beijing 100085, China. E-mail: {donghui02, song_lei, luoxulong}@baidu.com.*
- *M. Zhang is with China Construction Bank. E-mail: zhangming1.zh@ccb.com*

1. Henceforth, we term the network devices of a given model produced by a specific manufacturer "a type of network devices".

model for all device types. None of the existing log parsing techniques for network devices, including Signature Tree [13], Statistical Template Extraction (STE) [9], LogSimilarity [14], and FT-tree [12], can learn device-agnostic vocabulary for log classification. That is because they are designed to learn log templates for a single switch type.

2) *Partial labels*. Generally, operators use regular expressions to label anomalous logs. Device manufacturers usually provide regular expressions to help operators detect anomalous logs. However, these regular expressions are far from complete [4]. That is, many anomalous logs cannot be detected using these regular expressions. Therefore, operators have to manually maintain these regular expressions, namely, adding new regular expressions of anomalous logs when new types of device anomalies occur. In spite of these labor-intensive works, many anomalous logs remain undetected for the reason that some switch anomalies can fly under operators' radar [1]. Consequently, a large number of anomalous logs are unlabeled. In addition, operators do not label healthy logs because of their huge number and triviality. Training using the labeled and unlabeled anomalous logs as well as the unlabeled healthy logs is really challenging.
3) *Word weighting*. To extract features from logs, we try to construct a feature vector for each log. In classical methods of constructing feature vectors for texts, *e.g.,* bag-of-words, a log (text) is represented as a vector (multiset) of its words in order to keep its multiplicity. In general, each element in the vector denotes the estimated importance (weight) of a word [15]. However, existing word weighting methods such as Term Frequency Inverse Document Frequency (TF-IDF) [16] perform badly in our scenario, because they usually believe that frequently occurring words are not important, but it is not always true for device logs.

We propose LogClass, a framework to *automatically and robustly* detect and classify anomalous logs for *diverse types of* network devices in datacenter networks based on *partial labels*. The core idea of LogClass is that most device logs are semi-structured texts "printf"-ed by device operating systems (OSes), and the intuitions and methods in natural language processing can be applied or improved for anomalous log classification. To the best of our knowledge, this is the first work to investigate the challenges in anomalous log classification for network devices, and LogClass is the first framework to tackle these challenges as elaborated in the following.

1) Our preliminary investigation demonstrates that device logs of the same anomaly category share common patterns in terms of word combination. In natural language processing, the bag-of-words model is a typical word representation method that effectively represents word combination [17]. In this paper, we also adopt bag-of-words to extract the common patterns.
2) LogClass applies positive unlabeled (PU) learning [18] to learn patterns from partial labels. Specifically, it inputs labeled logs (anomalous logs having been detected) and unlabeled logs (anomalous logs having not yet been detected and healthy logs) into the PU learning model, which then determines which logs are anomalous and which are healthy. To the best of our knowledge, this is *the first time that PU learning is used for log based anomaly detection/classification*.
3) To weight the words of logs in feature vector construction, we propose a *novel, simple yet effective* Term Frequency Inverse Location Frequency (TF-ILF) method. In this method, we weight the words based on not only word frequency, but also the number of word positions. It is more robust and efficient than existing methods such as TF-IDF.

The overall design of LogClass is as follows. For diverse types of devices, in the offline learning procedure, LogClass preprocesses device logs and generates bag-of-words vectors using TF-ILF. Then LogClass respectively applies PU Learning [18] and SVM to train the anomaly detection and classification model. Similarly, in the online detection procedure, using the trained anomaly detection/classification model, LogClass determines whether a new log is anomalous, and if yes, decides its anomaly category.

We evaluate LogClass on more than 18 million real-world switch logs collected from 6547 switches (of 58 switch types) deployed in 10+ datacenters owned by a top global search engine over a 2-week period. Our results show that LogClass achieves 99.56% F1 score in anomalous log detection, and performs superiorly in addressing the challenges of device-agnostic vocabulary and partial labels. In addition, it shows a superior performance in (anomalous) log classification compared to Labeled-LDA (L-LDA) [19], with 99.57% Macro-F1 and 99.96% Micro-F1. Finally, TF-IDF improves the accuracy (measured by Macro-F1 score) of TF-IDF from 93.31% to 99.57%.

The rest of this paper is organized as follows. Section 2 provides the background and motivation of our problem. The design of LogClass is presented in Section 3. The evaluation experiments are described in Section 4, followed by a brief review related works in Section 5. Finally, we conclude our work in Section 6.

2 Background and Motivation

In this section, we first introduce cloud datacenter networks in Section 2.1. Then, we use switch logs as examples to depict the format of device logs in Section 2.2. Finally, we describe anomalous log detection and classification in Section 2.3.

2.1 Datacenter Network

At present, severs in a datacenter are connected via high-speed datacenter networks, which comprise a variety of network devices, such as switches, routers, and middleboxes (*e.g.,* VPNs, IDPS, and firewalls) [20], [21], [22]. With the explosion of the web services traffic in datacenter networks, the number of network devices have witnessed a dramatic

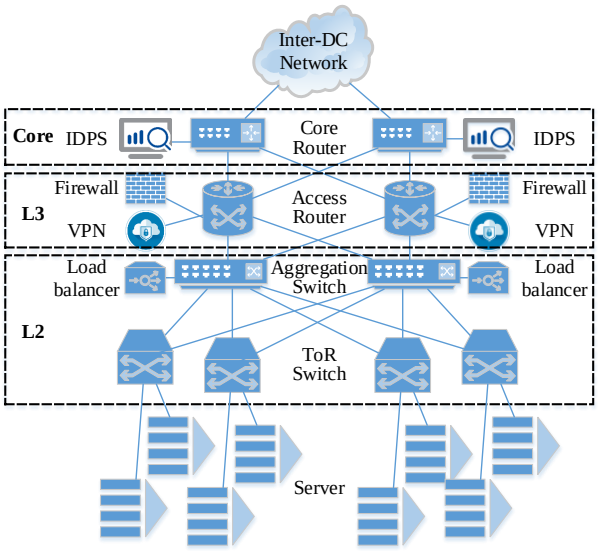


Fig. 1: The typical datacenter network architecture

increase. Figure 1 shows a typical architecture of cloud datacenter network [1], [23]. As the number of network devices increases, an increasing number of anomalies occur in today's datacenter networks [1]. Operators need to carefully monitor network devices, in order to detect and localize device anomalies soon after they happen. That is because even a slight anomaly of network devices may lead to a severe problem in the Internet services provided by the cloud, which in turn impact user experience and revenue.

2.2 Device Logs

Among the network devices in datacenters, switches and routers are the most important and they dominate in the total number and the number/duration of failures [1], and both of them generate large amounts of logs. These logs are the most valuable data source for operators to carefully monitor network status, because they not only indicate whether a network device is anomalous, but also directly tell or indirectly imply why an anomaly occurs. Therefore, in this work we mainly study the logs of network devices *including switches and routers*.

A device log is a semi-structured text "printf"-ed by the OS of the device. Since switches and routers share the same log structure, we here take switch logs as examples to depict the structure of device logs. Table 1 lists six examples of switch log messages. As the table shows, a switch log message usually has a primitive structure containing five fields, including a timestamp indicating when this message is generated, a switch ID identifying the switch that generates the message, a message type describing the rough characteristics of the logs, and a detailed message depicting the details of the event. Unfortunately, the message type field is ambiguous [4], and thus it cannot be used to accurately detect and classify anomalous logs. On the other hand, the detailed message field describes (anomalous) event in detail, and thus operators pay the most attention to this field. Therefore, in this paper, we detect and classify anomalous logs based on the detailed message field. *Henceforth, we use "log" or "log message" to denote the detailed message field for short.*

2.3 Anomalous Device Log Detection and Classification

TABLE 2: 12 examples of device anomaly categories

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

As is with [1], a *device anomaly* denotes when the service (such as traffic routing or forwarding) provided by this device deviates from the correct service. Each device anomaly belongs to a specific anomaly category, which indicates the cause of this anomaly. An anomaly category is usually predefined by operators. Typical device anomaly categories include system reboot, power down, *etc*. Table 2 shows 12 examples of anomaly categories in the studied cloud datacenters. We believe that these categories are generic to other cloud datacenters.

Typically, if a log message is anomalous, it can be classified into a specific anomaly category, which is based on the event this log message represents. For example, the anomalous log message "Vlan-interface vlan22, changed state to down", which denotes that the interface is down, can be classified into the anomaly category "INTERFACE_DOWN".

Generally, there are four types of device logs, *i.e.*, anomalous logs, healthy logs, unlabeled logs and labeled logs, defined as follows.

- *Anomalous log*. An anomalous log indicates that an anomaly occurs on the device. 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 device's healthy status.
- *Unlabeled log*. As aforementioned, operators are not able to label (detect) all the anomalous logs because regular expressions are incomplete. The anomalous logs that remain undetected and healthy logs constitute a large number of unlabeled logs.
- *Labeled log*. The anomalous logs having been detected are the labeled logs.

Manufacturers of network devices usually provide some patterns of anomalous logs, which however cannot detect *all* device anomalies [4], [14]. Therefore, operators need to manually add a new rule (pattern) of anomalous logs when a new type of anomaly occurs, in order to automatically detect this type of anomalies in the future.

Apparently, the simplest way to add the above rules 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.

TABLE 1: Examples of Switch Logs

Log No.	Vendor ID	Timestamp	Message Type	Detailed Message
L_1	V1	Jun 12 19:03:27 2017	SIF	*Interface* te-1/1/59, changed state *to* down
L_2	V2	Jun 13 20:22:03 2017	-	Vlan-interface vlan22, changed state *to* down
L_3	V1	Jun 13 20:22:03 2017	SIF	*Interface* te-1/1/17, changed state *to* up
L_4	V3	Jun 18 05:21:03 2017	SIF	*Interface* te-1/1/19, changed state *to* up
L_5	V3	Jun 15 13:46:43 2017	OSPF	Neighbour vlan23, changed state from Exchange *to* Loading
L_6	V3	Jun 15 13:46:43 2017	OSPF	PVID mismatch discovered on Ten-GigabitEthernet 6/0/10 , *to* S12516XAF-38.Int Ten-GigabitEthernet 3/0/17

TABLE 3: Examples of regular expressions for detecting and classifying anomalous device logs

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

Therefore, operators usually apply regular expressions to add the above rules. Specifically, when a new type of device anomaly is detected, operators will *manually* add a regular expression based on domain knowledge. After that, when this type of device anomaly occurs again, operators will be able to use the newly added regular expression to timely detect and classify this anomaly and mitigate the loss. Table 3 lists three examples of regular expressions. Nevertheless, the regular expression based approach has several drawbacks as follows.

1) *Low recall*. As aforementioned, only after a new type of anomaly is detected, operators will add its corresponding regular expression. Therefore, the anomalies that fly under operators' radar, and those having not yet occurred, cannot be detected using regular expression. That is, the regular expression based method has a relatively low recall.
2) *Non-generic*. Regular expression is too rigorous for matching anomalous logs. That is, it 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 7 for more details). However, for the anomalous logs belonging to the same anomaly category, but from different types of devices, they are likely to have some minor differences. Therefore, the regular expression based method is not generic to diverse types of devices.
3) *Labor intensive*. All the regular expressions except for those provided by manufacturers are *manually* configured by operators. Considering the large number of new types of anomalous device logs (thousands of these logs per day), manually configuring regular expressions consumes huge amounts of work.
4) *Computationally inefficient*. Large cloud datacenters generate millions of device logs per day. Since regular expression is not computationally efficient [6], [7], using it to detect and classify anomalies consumes too much computational resources.

Consequently, the non-generic, labor intensive and computationally inefficient regular expression based method has detected only *part of* anomalous device logs, and large amounts of anomalous logs remain undetected. In other words, as Figure 2 shows, only part of anomalous device logs have been *labeled* (detected), and the anomalous logs that have not yet been detected and healthy logs constitute the set of unlabeled logs.

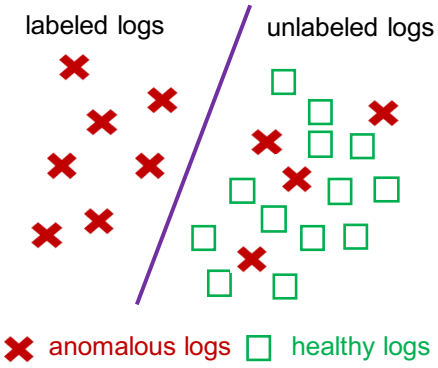


Fig. 2: Examples of anomalous logs, healthy logs and unlabeled logs

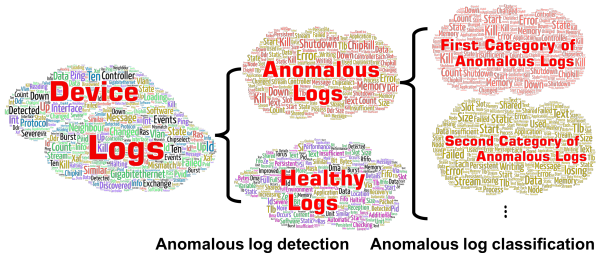


Fig. 3: Problem definition of anomalous device log detection and classification

In this work, we aim to *automatically* detect and classify anomalous logs based on *labeled (anomalous) logs and unlabeled (anomalous and healthy) logs*. Specifically, as shown in Figure 3, we try to detect anomalous logs (namely, divide all the device logs into anomalous and healthy ones), after which we classify anomalous logs into different anomaly categories.

Note that in this work, we focus on whether a single log message is anomalous or not, rather than the underlying relationship of log messages or the irregular patterns of log messages, which [14], [24] focus on.

3 Design of LogClass

In this section, we present the the framework of LogClass. The main objective of LogClass is to *automatically and accurately* detect and classify anomalous logs for *diverse types* of network devices in cloud using *partial labels*. We first

introduce the design overview of LogClass in Section 3.1, and then depict how to preprocess device logs in Section 3.2. Feature extraction of LogClass is elaborated in Section 3.3, followed by the description of offline learning and online classification in Section 3.4.

3.1 Design Overview

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

1) Anomalous logs of the same anomaly category share common patterns, even if they are collected from different types of network devices. Specifically, they share a common *combination of "important" words*. If we can accurately obtain this combination of "important" words, we can easily determine whether a device log belongs to this anomaly category. Obviously, 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 appears frequently 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 cannot be an "important" word.
3) The anomalous logs are detected (labeled) randomly, namely, the labeled and unlabeled anomalous logs follow the same distribution. That is because: (1) The anomalous logs detected by the regular expressions provided by device manufacturers are randomly distributed. (2) An unlabeled device anomaly occurs randomly, after which operators produce a regular expression to detect the anomalous logs denoting this anomaly. The anomalous logs detected by this regular expression is also randomly distributed.

Based on the above observations, we design LogClass as shown in Figure 4. 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 3.2). Then it constructs features using the bag-of-words model which is weighted by our novel TF-ILF method (Section 3.3). Finally, it trains a PU learning based binary classifier, and a SVM based multiclass classifier (Section 3.4). Similarly, in the online classification component, LogClass preprocesses real-time logs and extracts features, after which it determines whether a log is anomalous using the binary classifier, and if yes, classifies it to an anomaly category using the multiclass classifier. Both the binary and the multiclass classifier are learned in the offline learning procedure.

3.2 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 detection and classification [25]. Therefore, we preprocess logs before extracting features from them.

TABLE 4: Format rules and examples 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

As is with [14], we classify the words of logs into *text words* and *parameter words*. Text words are those depicting the events occurring on network devices. On the contrary, parameter words are those parameters (say, IP address, interface ID, device ID) that may appear with a low frequency. Parameter words are of little help in training for anomalous log detection and classification because they are usually distinct from one another.

We classify text words and parameter words using simple format rules based on operators' domain knowledge. Table 4 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.

3.3 Feature Construction

Since logs are semi-structured texts, they cannot be directly input into machine learning based classifiers such as SVM and decision tree. In the literature, one popular way to transfer a log to a structured representation is to learn templates from logs and then match logs to templates [13], [12], [9], [14], [25]. However, templates are learned and matched per device type. Therefore, they cannot be used for device-agnostic anomaly detection/classification.

In this work, we apply the bag-of-words model [15], one of the most classical methods in natural language processing, to construct features from documents. Henceforth, a document is a log after preprocessing in this work. In this model, a log is represented as a vector (multiset) of its words in order to keep its multiplicity. In general, the value of each element in the vector denotes the estimated importance (weight) of a word [15]. We propose a novel method, TF-ILF, to properly weight the words of device logs based on domain knowledge as follows.

3.3.1 The TF-ILF Model

TF-IDF, a simple yet effective weight model, is the most widely used weighting scheme for the bag-of-words representation [26]. As the term implies, TF refers to term frequency. The IDF (inverse document frequency) is a measure of how much information the word provides, that is, whether the term is common or rare across all documents (formally, the IDF value of word w is $\text{IDF}_w = \log(\frac{N}{n_w})$, where N is the total number of documents, and n_w is the number of logs in which w appears). This model believes

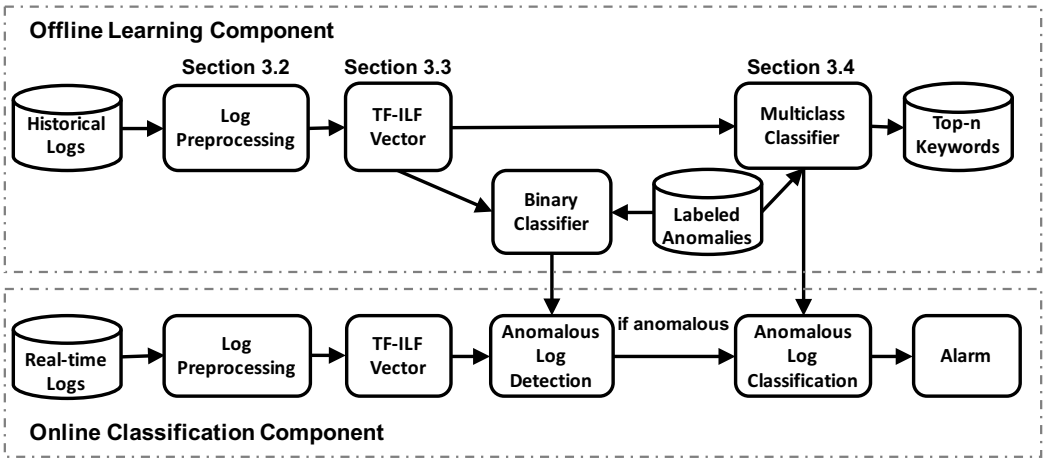


Fig. 4: The Framework of LogClass

that the more documents (logs in this work) a word appearing in, the less important this word is, which is inconsistent with the third observation mentioned in Section 3.1. For example, both L_3 and L_4 in Table 1 indicate the same event – an interface of the switch changes its state to up. If this type of events occur frequently in a period of time (this often happens because of interface flap), the device will generate lots of logs like L_3 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. For instance, the word "Ten-GigabitEthernet" appears twice in L_6 in Table 1. It means that this event involves two line card, which should be paid more attention by operators.

ILF measures how many different locations a word appearing 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

$$\text{ILF}_w = \log(\frac{\mathcal{L}}{\sum_{k=1}^{\mathcal{L}} w \Diamond l_k}) \tag{1}$$

where $w \Diamond l_k$ is

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

As listed in Table 1, there are five logs containing the word "to". It is the fifth word in L_1 to L_4 ("to"$\Diamond l_5 = 1$), and the seventh word in L_5 and L_6 ("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 detection/classification. Therefore, the weight of the word w in a log message is

$$\text{TF-ILF}_w = \text{TF}_w * \text{ILF}_w \tag{3}$$

where TF_w is the frequency of word w in the log message.

3.3.2 Why TF-ILF Works?

We here explain why TF-ILF performs better than TF-IDF. Intuitively, for a given word, *IDF* cares about the number of different *logs* it appearing in, while *ILF* counts the number of different *locations* where it appears. In the training procedure, for an "important" text word (see Section 3.2 for definition), its IDF value will become smaller when a new log containing it is added, while its ILF value will likely remain unchanged.

Figure 5 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 $\text{IDF} = \log(\frac{3}{1})$, and the ILF value is $\text{IDF} = \log(\frac{7}{1})$. 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 $\text{IDF} = \log(\frac{4}{2})$ 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 detection/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 assign 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 4.4. It demonstrates that, in real-world logs, the words and structures of logs, rather than abbreviations or semantics, are important to the log based anomaly detection and classification.

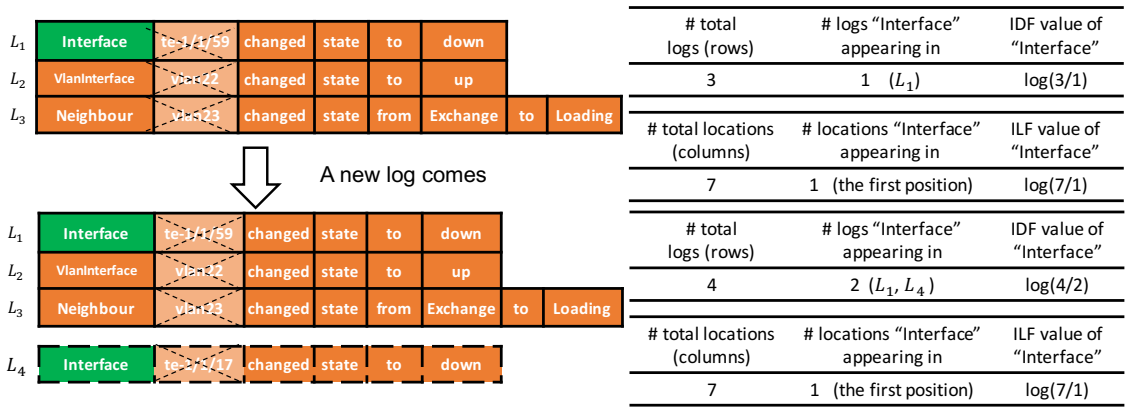


Fig. 5: An example of adding a new log in offline learning

3.4 Binary and Multiclass Classification

In the offline learning procedure, after obtaining the feature vectors, we apply the PU learning model [18] 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 apply SVM to train a multiclass classifier to classify anomalous logs into different categories in an interpretable manner. Similarly, in the online classification procedure, we use the PU learned binary classifier to determine whether a real-time log is anomalous. If yes, we then apply the trained multiclass classifier to decide its category.

To the best of our knowledge, this is the first work to utilize the PU learning model in log based anomaly detection. In this section, we first introduce how PU learning is used in our scenario, after which we depict the binary and multiclass classifiers.

3.4.1 PU Learning Based Binary Classification

Under the assumption that the labeled examples are selected randomly from the positive examples, PU learning is trained on positive and unlabeled examples, and predicts probabilities of being positive, which differ by only a constant factor from the true conditional probabilities [18], [27]. From the third observation mentioned in Section 3.1, we can see that our scenario exactly obeys the above assumption. Therefore, PU learning can be used in anomalous log detection.

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

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

Since the anomalous logs are labeled randomly, 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 [18], a simple yet effective estimator of c is

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

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 5 for more information). Therefore, we apply random forest (RF), which has been demonstrated to perform well in the sample imbalance scenario [4], to train the binary classifier. RF is a well-known powerful ensemble learning algorithm for binary classification by constructing a multitude of decision trees at training procedure and outputting the class based on the mode of the classes of individual trees [28]. It learns patterns and makes decisions based on voting, and thus the imbalance problem of samples impacts little on RF. Note that applying RF to train the binary classifier is not the main contribution of this work.

3.4.2 Multiclass Classification

Although sometimes the message type (see Table 1 for definition) can be used to identify the anomaly category, it cannot identify all anomaly categories. For example, the anomalous logs of the message type "SIF" can belong to the anomaly category "INTERFACE_DOWN", "PORT_DOWN", or "PROTOCOL_DOWN". In addition, not every log contains the message type field. Consequently, we should use the detailed message field (see Table 1 for definition) of a log message to identify its anomaly category. Moreover, the category of some anomalous logs can be obtained using the rule based methods, which are heavily relying on operators' domain knowledge. For example, if a log message contains both "Interface" and "down" , or "Vlan-interface" and "down", it is likely to belong to the anomaly category "INTERFACE_DOWN". However, the large number of anomalous log messages generated every day lead to that, obviously, operators cannot manually obtain all these

rules. As a result, classifying anomalous logs based on only operators' manual rules is infeasible. *Therefore, in this work, we aim to automatically classify anomalous logs based on the detailed message field of log messages.*

In the online classification procedure, when the above binary classifier determines a realtime log to be anomalous, the multiclass classifier will then decide which anomaly category this log belongs to. As shown in Figure 4, the multiclass classifier is trained using the labeled anomalous logs that have been well investigated and classified by operators, and their TF-ILF vectors (see Section 3.3).

To obtain an interpretable model, we apply SVM as the multiclass classifier. In addition to which category an anomalous log belongs to, operators are also eager to understand why it happens. Therefore, an interpretable multiclassifier is necessary. For a trained SVM model, we can get the coefficient assigned to each feature (word). This way, the top n important words can be obtained by sorting these coefficients. Operators can further understand every category by its top n important words.

Note that applying SVM to classify anomalies is not the main contribution of this work. To the best of our knowledge, this is the first work to investigate the problem of log based anomaly classification for network devices.

4 Evaluation

In this work, we try to *automatically and accurately* detect and classify anomalous logs for *diverse types* of network devices in cloud using *partial labels*. For the first time, PU learning is used for log based anomaly detection. In addition, we propose a novel, simple yet effective method, TF-ILF, to weight the words of logs in feature construction.

We have conducted extensive experiments to evaluate LogClass's performance, using the real-world data described in Section 4.1. Specifically, to demonstrate the performance of the PU learning based binary classifier (for anomaly detection), we calculate its accuracy as shown in Section 4.2. After that, in Section 4.3 we evaluate how LogClass performs in classifying anomalous logs by comparing it with Labeled-LDA (L-LDA) [19]. Finally, in order to show TF-ILF's performance, in Section 4.4 we use LogClass without weighting and that with TF-IDF (for weighting words in feature construction) as baselines.

We conduct all the experiments on a Linux server with Intel Xeon 2.40 GHz CPU and 64G memory. In addition, we implement LogClass with Python 2.7.

4.1 Data sets

In a typical datacenter, switches dominate in all of network devices (including routers, VPNs, firewalls, *etc.*) in terms of the total number, the number of failures, and failure duration [1]. In addition, switches and routers share common log structures as listed in Table 1. Consequently, in this work we use switch logs to evaluate LogClass's performance.

At first, considering the huge number of *all* switch logs (millions per day) in the studied top global search engine, we randomly selected 6574 switches (of *58 types*) deployed in *10+ datacenters*. Then, we collect *all* the switch logs of the above switches over *a 2-week period*. Table 5 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 2. 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 2.3 for more detail). However, as Table 5 shows, a large number of logs remain unlabeled with these regular expressions, which brings great challenges to LogClass.

4.2 Evaluation of The PU Learning Based Anomalous Log Detection

To address the challenge imposed by device-agnostic vocabulary, we introduce PU learning to train a binary classifier using *labeled (anomalous) logs and unlabeled (anomalous and healthy) logs*, in order to classify logs into *anomalous and healthy logs*. We here demonstrate how this binary classifier performs in terms of accuracy, and how LogClass performs in addressing the challenges of device-agnostic vocabulary and partial labels.

The dataset described in Table 5 is the training set (called the training set $\mathbb{T}$ henceforth). Considering the huge number $(16,702,547)$ of unlabeled logs, manually classifying them into healthy and anomalous logs to evaluate LogClass's performance is infeasible. Therefore, we randomly pick 0.1% of these unlabeled logs (*i.e.,* $16,703$ logs), and let experienced operators manually classify the sampled unlabeled logs into healthy and anomalous logs. Each anomalous log is then manually classified into a specific anomaly category as the groundtruth in our evaluation. Among these $16,703$ 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). Therefore, the $1,758$ sampled labeled and $16,703$ sampled unlabeled logs constitute the testing set (called the testing set $\mathbb{E}$ henceforth).

In the testing set, let the set of labeled anomalous logs be L, the set of unlabeled logs be U. Apparently, U consists of the subset of unlabeled healthy logs H, and the subset 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).

A classification method's capability is usually assessed by three metrics that have intuitive interpretations, *i.e.*, precision, recall, F1 score. Thus, we use these metrics 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. So 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

TABLE 5: The detailed information of the dataset

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

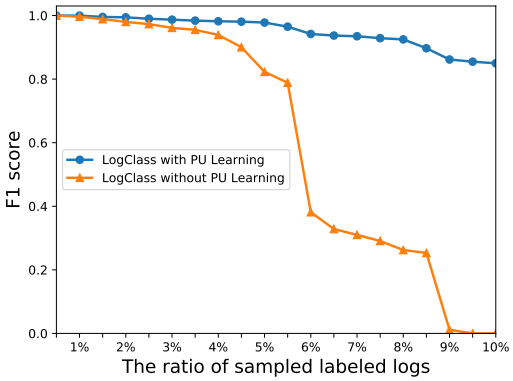


Fig. 6: Comparison between LogClass with/without PU learning as the ratio of sampled labeled logs increases

F1 score as follows: precision $= \frac{TP}{TP+FP}$, recall $= \frac{TP}{TP+FN}$, F1 score $= \frac{2\times\text{precision}\times\text{recall}}{\text{precision}+\text{recall}}$.

The second row of Table 6 (experiment 1) lists the experiment results in terms of precision, recall and F1 score. The PU learning based binary classifier achieves very-close-to-one precision and recall, demonstrating its superior performance in anomalous log detection.

To demonstrate how LogClass performs in addressing the challenge of device-agnostic vocabulary, we conduct experiment 2. We randomly select 53 types of switches and use their logs in the training set $\mathbb{T}$ to train LogClass. 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 6 lists the precision, recall and F1 score of this experiment. The LogClass trained using the logs of 53 types of switches achieves a very-close-to-one F1 score in detecting anomalous logs for the other five types of switches. This demonstrates LogClass's good performance for the device-agnostic vocabulary of diverse types of switches.

LogClass introduces 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 a ratio of (say 10%) 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 the testing set $\mathbb{E}$. In addition, we use LogClass without PU learning (namely, using only RF to train the binary classifier) as the baseline method.

Figure 6 compares the accuracy (in terms of F1 score) of LogClass with and without PU learning as the ratio of sampled labeled logs increases. The accuracy of LogClass with PU learning remains stable as the ratio increases. That is because the PU leaning of LogClass 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 quantitatively demonstrate how LogClass performs in addressing partial labels, *i.e.*, detecting anomalous logs from the unlabeled dataset, we randomly sampled $1,000,000$ unlabeled logs from $\mathbb{T}$ (because manually investigating the anomalous logs in the unlabeled logs of the entire dataset $\mathbb{E}$ will consume operators too much time). We then leverage LogClass to detect anomalous logs from the above unlabeled dataset. 710 anomalous logs are detected by LogClass, all of which are manually investigated and confirmed by operators. Table 7 shows a case of anomalous log detection by LogClass. $L1$ is an unlabeled log in the above 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.

4.3 Evaluation of Anomalous Log Classification

After detecting anomalous logs, LogClass then classifies these logs into different categories using SVM in an interpretable manner. We here show LogClass's performance in anomalous log classification. To the best of our knowledge, in the literature there is no previous work on anomalous log classification using partial labels. Therefore, we apply L-LDA [19], an efficient text classification method in natural language processing, as the baseline method. This method is discussed in detail in Section 5. We use a popular open-source implementation of L-LDA, MALLET [29], in our evaluation experiment. Note that *the parameters of L-LDA are set best for Macro-F1*. In addition, we train LogClass and L-LDA using the training set $\mathbb{T}$, and test them using the testing set $\mathbb{E}$.

A multiclass classification method's capability is usually assessed by Micro-F1 and Macro-F1 [30]. 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 are 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

TABLE 6: The evaluation results of anomalous log detection

Experiment 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%

TABLE 7: An actual case of anomalous log detection 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)

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

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

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

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

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.

Table 8 lists the comparison results between LogClass and L-LDA. LogClass achieves better performance than L-LDA in terms of both Macro-F1 and Micro-F1. Specifically, LogClass improves 9.89% over L-LDA in Macro-F1. That is because L-LDA is not as robust as LogClass to all anomaly categories. In addition, the training time on training set $\mathbb{T}$ (containing $1,758,458$ labeled logs and $16,702,547$ unlabeled logs), and the classification time on the testing set $\mathbb{E}$ (containing $1,758$ labeled logs and $16,703$ unlabeled logs), of LogClass and L-LDA, are also listed in Table 8, respectively. Compared to L-LDA, LogClass respectively improves the training and classification efficiency by 19.51 and 5.25 times. According to investigations, operators are quite satisfactory with the computational efficiency of LogClass in online anomalous log classification.

As described in Section 3.4.2, LogClass extracts the top-n important words for each anomaly category, in oder to help operators better understand every anomaly category. In Table 9, we list the top 5 important words of the anomaly category "INTERFACE_DOWN" (see Table 2 for more details). Based on an on-site interview, we find that the top-n important words really help operators further comprehend what every anomaly category stands for.

4.4 Evaluation of TF-ILF

As described in Section 3.3.1, 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-ILF – it preserves "term frequency (TF)" and improves from "document frequency" to "location frequency". To show the performance of TF-ILF, we here 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, and that of LogClass without word weighting. Note that *the parameters of the later two methods are set best for Macro-F1*. Similarly, we train the above three methods using the training set $\mathbb{T}$, and test them using the testing set $\mathbb{E}$.

Table 10 shows the comparison results of the above three methods. We observe that the Macro-F1 of LogClass without weighting is much smaller than that of LogClass with word weighting (using TF-IDF or TF-ILF). This proves that assigning a weight to each word in feature construction can greatly improve the accuracy of anomalous log classification. 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 accuracy from TF-IDF's 93.32% to a very-close-to-one value. 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.

5 Related Work

Network device failures in datacenter networks has drawn a lot of attention recently [31], [32], [33], [34], [35], [36], [37], [38], [39]. Gill *et al.* presented a detailed analysis of failures in a large datacenter network, and demonstrated the number, downtime, and root causes of failures for different types of network devices including routers, switches, middle boxes, *etc* [31]. Potharaju *et al.* analyzed failure events over three years across multiple datacenters, and revealed the percentage of different root causes that contribute to Intra-datacenter and Inter-datacenter network failure events, and showed that network device failures lead to a large portion of network failure events [37]. In [32], Potharaju *et al.*

TABLE 8: Comparison of the Macro-F1, Micro-F1, training time and classification time between LogClass and L-LDA

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

TABLE 9: The top 5 important words of the category "INTERFACE_DOWN"

Examples of 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"

TABLE 10: Comparison of LogClass with TF-ILF, LogClass with TF-IDF and LogClass without word weighting

Methods	Macro-F1	Micro-F1
LogClass with TF-ILF	99.57%	99.96%
LogClass with TF-IDF	93.32%	99.74%
LogClass without weighting	45.86%	99.15%

demonstrated that router failures and switch failures caused the bulk of high-severity incidents that impact customer and business significantly, which is in consistency with the study in [37], [34], [40]. The above works motivate us to detect network device anomalies and take corrective measures to prevent failures from impacting services.

Recently, many researches have paid attention to detecting anomalies using KPI streams. They leveraged traditional statistical methods [41], [42], [43], supervised learning methods [5], unsupervised learning methods [44], or semi-supervised learning methods [45] to detect the anomalous patterns in KPI streams. However, the approaches which are based on KPI stream analysis only determines whether an anomaly occurs, and they cannot locate the root cause of an anomaly. Therefore, in this work we analyze log messages which contain rich information, rather than KPI streams, to detect and classify anomalies of network devices.

Using log files for failure detection, diagnosis and prediction has been widely applied in networks [46], [47], [13], [14], [9] and computers [48], [49], [50], [51], [52], [53], [24]. Specifically, Qiu *et al.* proposed SyslogDigest to understand the events happened on network devices [13]. They first parsed unstructured log messages by matching them to message templates, and in turn generated high-level network events based on these templates. However, this model is specific for a particular network device model, and thus cannot be used in our (device-agnostic vocabulary) scenario. In addition, Kimura *et al.* presented a statistical template extraction method to automatically extract templates from unstructured log messages, and a log tensor factorization method to capture the spatial-temporal patterns of log messages [9]. After that, Kimura *et al.* constructed log feature vectors to characterize the generation patterns of log messages, and leveraged a supervised learning method to learn failure patterns [14]. However, both of the above two methods are detecting anomalies based on a chunk of log messages, rather than a single one, and they cannot classify anomalous log messages. Typically, network vendors do not release the code to their clients, *i.e.*, the operators of datacenters. Therefore, the approaches in [50], [48], which heavily relied on the code that generate the logs, cannot be used in our scenario.

In natural language processing, the *topic model* is a popular method for text classification. LDA, which identifies latent topic information in document collections, is a typical machine learning based method of topic model [54]. 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 [19]. 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. However, it is not suitable to short text and thus cannot be used in our (device log) scenario.

6 Conclusion

Network devices logs describe a vast range of events, which are extremely valuable in network management (say, anomaly detection, troubleshooting) for datacenter networks. However, it is quite difficult to apply device logs in anomaly detection and classification because of device-agnostic vocabulary, partial labels and lack of accurate word weighting methods. We propose LogClass, a framework to automatically and accurately detect and classify anomalous logs for diverse types of network devices in datacenter networks using partial labels. LogClass applies a novel, simple yet effective method, TF-ILF, to weight the words of logs in feature construction using the bag-of-words model. To address the challenge posed by partial labels, LogClass introduces PU learning to detect anomalous logs. Extensive experiments using real-world data demonstrate that LogClass achieves superior performance in anomalous log detection and classification in terms of accuracy. In addition, the novel TF-ILF method shows its better performance compared with TF-IDF.

REFERENCES

[1] Phillipa Gill et al. Understanding network failures in data centers: measurement, analysis, and implications. In *ACM SIGCOMM Computer Communication Review*, volume 41, pages 350–361. ACM, 2011.

[2] Transfer switch failure causes outage at colo4 data center. http://www.datacenterdynamics.com/content-tracks/power-cooling/transfer-switch-failure-causes-outage-at-colo4-data-center/32548.fullarticle, Augest 2011.

[3] Data center failure downs virginia state computer network. http://www.datacenterdynamics.com/content-tracks/security-risk/data-center-failure-downs-virginia-state-computer-network/96247.article, May 2015.

[4] Shenglin Zhang, Ying Liu, Weibin Meng, Zhiling Luo, Jiahao Bu, Sen Yang, Peixian Liang, Dan Pei, Jun Xu, Yuzhi Zhang, et al. Prefix: Switch failure prediction in datacenter networks. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 2(1):2, 2018.

[5] Dapeng Liu, Youjian Zhao, Haowen Xu, Yongqian Sun, Dan Pei, Jiao Luo, Xiaowei Jing, and Mei Feng. Opprentice:towards practical and automatic anomaly detection through machine learning. In *Internet Measurement Conference*, pages 211–224, 2015.

[6] Zhe Fu, Shijie Zhou, and Jun Li. bitfa: A novel data structure for fast and update-friendly regular expression matching. In *the SIGCOMM Posters and Demos*, pages 130–132, 2017.

[7] Domenico Ficara et al. An improved dfa for fast regular expression matching. *ACM SIGCOMM Computer Communication Review*, 38(5):29–40, 2008.

[8] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *ACM Sigsac Conference*, pages 1285–1298, 2017.

[9] Tatsuaki Kimura et al. Spatio-temporal factorization of log data for understanding network events. In *INFOCOM, 2014 Proceedings IEEE*, pages 610–618. IEEE, 2014.

[10] Shenglin Zhang, Ying Liu, Dan Pei, Yu Chen, Xianping Qu, Shimin Tao, and Zhi Zang. Rapid and robust impact assessment of software changes in large internet-based services. In *CONEXT*, Heidelberg, Germany, December 2015.

[11] Shenglin Zhang, Ying Liu, Dan Pei, Yu Chen, Xianping Qu, Shimin Tao, Zhi Zang, Xiaowei Jing, and Mei Feng. Funnel: Assessing software changes in web-based services. *IEEE Transactions on Services Computing*, 2016.

[12] Shenglin Zhang, Weibin Meng, et al. Syslog processing for switch failure diagnosis and prediction in datacenter networks. In *Quality of Service (IWQoS), 2017 IEEE/ACM 25th International Symposium on*, pages 1–10. IEEE, 2017.

[13] Tongqing Qiu, Zihui Ge, Dan Pei, et al. What happened in my network: mining network events from router syslogs. pages 472–484, 2010.

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

[15] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008.

[16] Pascal Soucy and Guy W Mineau. Beyond tfidf weighting for text categorization in the vector space model. In *International Joint Conference on Artificial Intelligence*, pages 1130–1135, 2005.

[17] Yin Zhang, Rong Jin, and Zhi-Hua Zhou. Understanding bag-of-words model: a statistical framework. *International Journal of Machine Learning and Cybernetics*, 1(1-4):43–52, 2010.

[18] Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *ACM SIGKDD*. ACM, 2008.

[19] Daniel Ramage, David Hall, Ramesh Nallapati, and Christopher D Manning. Labeled lda: A supervised topic model for credit attribution in multi-labeled corpora. In *EMNLP 2009*.

[20] Mohammad Al-Fares et al. A scalable, commodity data center network architecture. 38(4):63–74, 2008.

[21] Albert Greenberg et al. Vl2: a scalable and flexible data center network. 39(4):51–62, 2009.

[22] Chuanxiong Guo, Lihua Yuan, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *ACM SIGCOMM CCR*. ACM, 2015.

[23] Rahul Potharaju and Navendu Jain. Demystifying the dark side of the middle:a field study of middlebox failures in datacenters. In *Conference on Internet Measurement Conference*, pages 9–22, 2013.

[24] Ke Zhang, Jianwu Xu, Martin Renqiang Min, Guofei Jiang, Konstantinos Pelechrinis, and Hui Zhang. Automated it system failure prediction: A deep learning approach. In *BigData*, pages 1291–1300, 2016.

[25] Pinjia He, Jieming Zhu, Shilin He, Jian Li, and Michael R. Lyu. An evaluation study on log parsing and its use in log mining. In *Ieee/ifip International Conference on Dependable Systems and Networks*, pages 654–661, 2016.

[26] Aixin Sun. Short text classification using very few words. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1145–1146. ACM, 2012.

[27] Gill Ward, Trevor Hastie, Simon Barry, Jane Elith, and John R Leathwick. Presence-only data and the em algorithm. *Biometrics*, 65(2):554–563, 2009.

[28] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC medical informatics and decision making*, 11(1):51, 2011.

[29] Mallet. https://github.com/mimno/Mallet.

[30] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5(2):361–397, 2004.

[31] Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: Measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 Conference*, SIGCOMM '11, pages 350–361, 2011.

[32] Rahul Potharaju and Navendu Jain. Demystifying the dark side of the middle: A field study of middlebox failures in datacenters. In *Proceedings of the 2013 Conference on Internet Measurement Conference*, IMC '13, pages 9–22, 2013.

[33] Rahul Potharaju and Navendu Jain. When the network crumbles: An empirical study of cloud network failures and their impact on services. In *Proceedings of the 4th Annual Symposium on Cloud Computing*, SOCC '13, pages 15:1–15:17, 2013.

[34] Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, Anand Kanagala, Jeff Provost, Jason Simmons, Eiichi Tanda, Jim Wanderer, Urs Hölzle, Stephen Stuart, and Amin Vahdat. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 183–197, 2015.

[35] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 139–152, 2015.

[36] Justine Sherry, Peter Xiang Gao, Soumya Basu, Aurojit Panda, Arvind Krishnamurthy, Christian Maciocco, Maziar Manesh, João Martins, Sylvia Ratnasamy, Luigi Rizzo, et al. Rollback-recovery for middleboxes. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '15, pages 227–240, 2015.

[37] Rahul Potharaju and Navendu Jain. An empirical analysis of intra- and inter-datacenter network failures for geo-distributed services. In *ACM SIGMETRICS Performance Evaluation Review*, volume 41, pages 335–336. ACM, 2013.

[38] Xin Wu, Daniel Turner, Chao-Chih Chen, David A Maltz, Xiaowei Yang, Lihua Yuan, and Ming Zhang. Netpilot: automating datacenter network failure mitigation. In *Proceedings of the 2012 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '12, pages 419–430, 2012.

[39] Justine Sherry, Shaddi Hasan, Colin Scott, Arvind Krishnamurthy, Sylvia Ratnasamy, and Vyas Sekar. Making middleboxes someone else's problem: network processing as a cloud service. In *Proceedings of the 2012 ACM Conference on Special Interest Group on Data Communication*, SIGCOMM '12, pages 13–24, 2012.

[40] Guosai Wang, Lifei Zhang, and Wei Xu. What can we learn from four years of data center hardware failures? In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 25–36. IEEE, 2017.

[41] Ajay Mahimkar, Zihui Ge, Jia Wang, Jennifer Yates, Yin Zhang, Joanne Emmons, Brian Huntley, and Mark Stockert. Rapid detection of maintenance induced changes in service performance.

In *Proceedings of the Seventh COnference on emerging Networking EXperiments and Technologies*, page 13. ACM, 2011.

[42] Ajay Anil Mahimkar, Han Hee Song, Zihui Ge, Aman Shaikh, Jia Wang, Jennifer Yates, Yin Zhang, and Joanne Emmons. Detecting the performance impact of upgrades in large operational networks. *ACM SIGCOMM Computer Communication Review*, 41(4):303–314, 2011.

[43] He Yan, Ashley Flavel, Zihui Ge, Alexandre Gerber, Dan Massey, Christos Papadopoulos, Hiren Shah, and Jennifer Yates. Argus: End-to-end service anomaly detection and localization from an isp's point of view. In *INFOCOM, 2012 Proceedings IEEE*, pages 2756–2760. IEEE, 2012.

[44] Haowen Xu, Wenxiao Chen, Nengwen Zhao, Zeyan Li, Jiahao Bu, Zhihan Li, Ying Liu, Youjian Zhao, Dan Pei, Yang Feng, et al. Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 187–196. International World Wide Web Conferences Steering Committee, 2018.

[45] Jiahao Bu, Ying Liu, Shenglin Zhang, Weibin Meng, Qitong Liu, Xiaotian Zhu, and Dan Pei. Rapid deployment of anomaly detection models for large number of emerging kpi streams. In *2018 IEEE 36th International Performance Computing and Communications Conference (IPCCC). IEEE*, pages 1–8, 2018.

[46] Felix Salfner and Steffen Tschirpke. Error log processing for accurate failure prediction. In *Proceedings of the First USENIX Conference on Analysis of System Logs*, WASL'08, 2008.

[47] Felix Salfner and Miroslaw Malek. Using hidden semi-markov models for effective online failure prediction. In *Reliable Distributed Systems, 2007. SRDS 2007. 26th IEEE International Symposium on*, pages 161–174. IEEE, 2007.

[48] Wei Xu, Ling Huang, Armando Fox, and Patterson. Detecting large-scale system problems by mining console logs. In *SOSP*, 2009.

[49] Errin W Fulp, Glenn A Fink, and Jereme N Haack. Predicting computer system failures using support vector machines. *WASL*, 8:5–5, 2008.

[50] Wei Xu, Ling Huang, Armando Fox, David A Patterson, and Michael I Jordan. Mining console logs for large-scale system problem detection. *SysML*, 8:4–4, 2008.

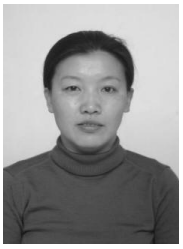
[51] Z. Zheng, Z. Lan, B. H. Park, and A. Geist. System log preprocessing to improve failure prediction. In *Dependable Systems Networks, 2009. DSN '09. IEEE/IFIP International Conference on*, pages 572–577, June 2009.

[52] Yinglung Liang, Yanyong Zhang, Morris Jette, Anand Sivasubramaniam, and Ramendra Sahoo. Bluegene/l failure analysis and prediction models. In *Dependable Systems and Networks, 2006. DSN 2006. International Conference on*, pages 425–434. IEEE, 2006.

[53] Ilenia Fronza, Alberto Sillitti, Giancarlo Succi, Mikko Terho, and Jelena Vlasenko. Failure prediction based on log files using random indexing and support vector machines. *Journal of Systems and Software*, 86(1):2–11, 2013.

[54] Liangjie Hong and Brian D. Davison. Empirical study of topic modeling in twitter. In *Proceedings of the First Workshop on Social Media Analytics*, SOMA '10, pages 80–88, New York, NY, USA, 2010. ACM.

Weibin Meng received the B.S. degree in Software Engineering from Jilin University, Changchun, China, in 2016. He is currently a Ph.D. student in the Department of Computer Science and Technology and the Institute for Network Sciences and Cyberspace at Tsinghua University, Beijing, China. His research interests include anomaly detection, syslog analysis, and failure prediction in datacenter networks.



Ying Liu received the B.S. degree in information engineering, M.S. degree in computer science and Ph.D. degree in applied mathematics from Xidian University in 1995, 1998 and 2001, respectively. She made postdoctoral research in the Department of Computer Science and Technology, Tsinghua University during 2001-2003. She is currently an associate professor in the Institute for Network Sciences and Cyberspace, Tsinghua university. Her research interests include multicast routing, network architecture and router design and implementation. She is an IEEE member.

Shenglin Zhang received B.S. in network engineering from the School of Computer Science and Technology, Xidian University, Xi'an, China, in 2012 and Ph.D. 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. He is an IEEE Member.

Dan Pei 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 (UCLA) in 2005. He is currently an associate professor in the Department of Computer Science and Technology, Tsinghua University. His research interests include network and service management in general. He is an IEEE senior member and an ACM senior member.

Yuzhi Zhang received the B.S. and M.S. degree 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 dean of the College of Software, Nankai University, and is also a distinguished professor. His research interests include deep learning and other aspects in artificial intelligence.

Hui Dong received the B.E. and M.E. degree in Computer Science and Technology from Harbin University of Science and Technology, Harbin, China, in 2011 and 2014, respectively. He is currently a senior engineer with Baidu, inc. His research interests include anomaly detection, syslog analysis, and failure prediction in huge datacenter networks.

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.

Xulong Luo received the B.S degree from Computer Science and Technology, China University of Geosciences, Beijing, China, in 2005. he is currently a senior engineer with Baidu, Inc.

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 of China Construction Bank.