IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 27, 2021, accepted April 21, 2021, date of publication April 30, 2021, date of current version May 24, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3076897

LogM: Log Analysis for Multiple Components of Hadoop Platform

YUXIA XIE, KAI YANG, AND PAN LUO
Department of Computer Science, Tongji University, Shanghai 201800, China

Corresponding author: Kai Yang (kaiyang@tongji.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61771013, in part by the Fundamental Research Funds for the Central Universities of China, and in part by the Fundamental Research Funds of Shanghai Jiading District.

ABSTRACT The Hadoop platform provides a powerful software framework for distributed storage and processing of massive amounts of data. It is at the heart of big data processing and has found numerous applications in diverse areas, ranging from environmental monitoring to security analysis. To facilitate the storage and processing of big data, a Hadoop platform typically runs on a cluster of servers and may scale up to process big data over thousands of hardware nodes. However, the growing scale and complexity of the Hadoop platform also make it increasingly challenging to manage and operate. In this paper, we present a framework called *LogM* that leverages not only the deep learning model, but also the knowledge graph technology for failure prediction and analysis of the Hadoop cluster. In particular, we first develop a CAB net (Convolutional Neural Network (CNN) with attention-based Bi-directional Long Short Term Memory (Bi-LSTM)) architecture to effectively learn the temporal dynamics from the sequential log data, which allows us to predict system failures. We then adopt a knowledge graph approach for failure analysis and diagnosis. Extensive experiments have been carried out to assess the performance of the proposed approach. It is seen that *LogM* is highly effective in predicting and diagnosing system failures.

INDEX TERMS Log analysis, Hadoop, anomaly detection, failure diagnosis.

I. INTRODUCTION

Large-scale data processing and analysis is becoming ubiquitous in a variety of application areas, ranging from environmental monitoring, e-commerce to network security [1]. Hadoop has recently emerged as a new technological trend for the distributed processing of massive amount of data. The Hadoop platform is deemed as the big data processing engine and the core enabling technology for big data analytics, and has been adopted by several top-tier companies.

Hadoop is a distributed system infrastructure developed by the *Apache*^{TM} Foundation and maintained by many open-source contributors. It is composed of a collection of components, and each of them is responsible for a different task [2]. Two important components are Hadoop Distributed File System (HDFS) [3] and MapReduce [4], which work together to offer high-quality storage and computing services to commercial users. HDFS is a file management system that provides file storage and management services for the other components. HDFS is fully distributed and is more scalable and fault-tolerant than the conventional file management system. MapReduce lies at the heart of Hadoop. It provides a programming paradigm for distributed computing over hundreds or even thousands of servers in a Hadoop cluster. Apart from HDFS and MapReduce, other components can also be installed on top of the base Hadoop platform, including YARN [5], Spark [6], HBase [7] and ZooKeeper [8]. These components collaborate together to constitute the base platform of Hadoop in practical applications. As a newly added framework in Hadoop 2.0, YARN is responsible for the management of cluster resources and job scheduling. As a scalable and distributed database, HBase provides convenience for users to store a large amount of data. Spark is designed to well support users in carrying out complex computations.

In terms of design principle, the Hadoop platform deployed on a large number of hardware nodes that can provide computing and storage services locally. In practice, the stability of the Hadoop platform depends heavily on the collaboration of various components. However, these components typically involve many geographically distributed computing devices,

The associate editor coordinating the review of this manuscript and approving it for publication was Jan Chorowski.

which are prone to failures due to limited capacity. Consequently, a single failure from one component may propagate to other parts of the Hadoop platform and eventually lead to a global outage of the entire system [9]. Meanwhile, with the ever-increasing scale and complexity, the Hadoop platform has became more and more difficult to manage and operate, which significantly increase the operation and maintenance cost.

System logs are mainly designed to record the system status and significant events at various critical points to ease debugging [10]. They are informative and exist practically in every computer system, which are valuable for tracking and investigating the system status. Engineers can examine system logs to understand the status of systems, detect system performance degradation and perform root cause analysis. Therefore, logs naturally become one of the most valuable data resources for system operation and play a critical role in the maintenance of the Hadoop platform.

However, there exist a number of challenges against efficient processing and analysis of system logs, especially for logs from multiple components of the Hadoop platform. First, the log data is unstructured and comes with a variety of formats, including text, date, and number. The heterogeneity of the Hadoop system logs makes it difficult to understand and analyze. Besides, a Hadoop platform consists of several hardware nodes and may generate numerous logs every second, thus requires real-time processing. There are correlations among logs of different components, which brings difficulties in analysis. At the same time, system failures involving multiple components are usually more complicated than those with single components. Moreover, the analysis of root cause for the system failure requires extensive domain knowledge and labelling efforts from a subject matter expert, which is often time-consuming and costly.

To cope with the above challenges, we develop a framework called *LogM* for failure prediction and diagnosis on unstructured log streams, which combines deep learning technologies with the knowledge graph. Compared with the conventional frameworks, *LogM* has the following characteristics: 1) *LogM* timely predicts and diagnoses the system failures, which allows operators to take immediate actions to avoid system outage. 2) *LogM* not only analyzes the failure associated with a single component, but also analyzes the failure involving multiple components, which is more in line with the real-world scenario. 3) *LogM* employs the CNN to extract the semantic information of the logs and leverages the attention-based Bi-LSTM to extract the temporal dynamics from the sequential log data. By using the end-to-end model, *LogM* can automatically learn from the historic system logs and forecast whether a failure will occur afterwards. 4) *LogM* adopts the knowledge graph to characterize the relationships among different anomalous logs, thereby effectively assisting operators to diagnose system failures.

The main contributions of this work are summarized as follows.

- As far as we are aware of, *LogM* is the first framework that can predict and diagnose system failures of Hadoop via analyzing logs from *multiple* components.
- We propose an end-to-end CNN combined with the attention-based Bi-LSTM model (CAB net) which can effectively *predict* system failures.
- We adopt an anomaly *knowledge graph* to characterize the relationships among multiple anomalous logs. Based on this knowledge graph, both supervised and unsupervised learning approaches have been developed for failure analysis.

We have evaluated the *LogM* framework using the logs collected from a Hadoop platform, including six different components, with more than fifty million system logs. The experimental results show that the proposed framework can effectively predict and diagnose the failures for the Hadoop platform.

The rest of the paper is organized as follows. Section II provides an overview of the related work. The details of *LogM* are presented in Section III, including the CAB net model for failure prediction and the failure diagnosis based on the knowledge graph. Section IV demonstrates the performance of *LogM* on a real-world dataset from the Hadoop platform. Section V concludes our work.

II. RELATED WORK

As a valuable resource in system maintenance, system logs can be used for effective anomaly detection and problem diagnosis. However, due to diverse formats, the analysis of system logs is challenging. To effectively tackle these challenges, many efforts have been taken recently to explore the methods for anomaly detection on unstructured logs, including supervised and unsupervised anomaly detection. Supervised approaches such as Logistic Regression, Decision Tree and Support Vector Machine (SVM) can be used to detect anomalies. Unsupervised approaches like clustering, Principal Component Analysis (PCA) and Invariants Mining (IM) can also be used to perform anomaly detection.

Both supervised and unsupervised anomaly detection on unstructured logs have attracted a lot of attention over the past decade and there exists a large body of related work. For example, in [11] SVM has been employed for anomaly detection via utilizing event logs. A comparison of the state-of-the-art anomaly detection methods for logs, including Logistic Regression, Decision Tree and SVM are presented in [12]. FIU Log Analysis Platform (FLAP) [13] is an integrated system, which aims to facilitate the analysis for system event logs. The platform implements anomaly detection by treating anomaly detection as a classification problem and provides users with interfaces of the existing classification methods, such as SVM, Ridge Regression, Lasso Regression, etc. Adele [14] is a data-driven anomaly detection method for logs, which learns from the historic log data to extract the characteristics of abnormal behaviors and uncover the anomaly.

Apart from supervised approaches, considerable efforts have been taken toward anomaly detection on logs via unsupervised learning approaches. HLAer [15] is a system designed to analyze heterogeneous logs, which can provide common log operations, including log indexing, log pattern recognition, log query and log anomaly detection. Clustering is used to perform anomaly detection in HLAer. Although HLAer is unsupervised and robust to heterogeneous logs, it is not efficient and scalable due to large memory requirements and communication overhead in parallel implementation. As a remedy, LogLens is proposed in [16], which is a real-time log analysis system and can automatically detect anomalies with less human intervention. The anomaly detection model is divided into two parts. One is to detect anomalies through clustering, and the other is to use the Finite State Automata (FSA) to detect anomalies. Compared with HLAer, LogLens can be easily deployed on Spark and is more efficient and scalable. LogCluster [17] is a clustering-based problem identification method, which first establishes a knowledge base by clustering log messages to obtain representative log sequences, and then compares the extracted log sequences with the knowledge base during problem identification. In [18], Xu *et al.* propose a novel two-stage online log anomaly detection method, which can effectively combine frequent pattern mining with the PCA method to detect system runtime problems. Lou *et al.* [19] analyze the system logs with IM by mining the invariants (i.e. linear relationships) and regard the log sequences that violate the invariants as anomalies.

With the ever-increasing demand for intelligent operation and maintenance of complex Information Technology (IT) systems, there has emerged a growing interest in applying deep learning technologies to log anomaly detection. Zhang *et al.* [20] propose to use the Long-Short Term Memory (LSTM) network to model the sequential log data by capturing the dependencies among log sequences. In addition to the log sequences, DeepLog [21] also uses the log variable vectors to train the LSTM network and is more efficient than [20]. LogAnomaly [22] still uses the LSTM network to detect anomalies based on log sequences, which includes a novel and efficient log vectorization method called template2vec, and can effectively transform log templates to vectors. Due to the improvement of the log vectorization, LogAnomaly performs better than DeepLog for online detection. LogFlow [23] utilizes an attention-based LSTM to identify correlations between log entries and perform anomaly detection for distributed infrastructures. Also using LSTM network, LogGAN [24] combines it with a Generative Adversarial Network (GAN) to perform log anomaly detection. LogGAN consists of a generator and a discriminator, in which the generator learns the distribution of real data, so as to generate the synthetic samples, and the discriminator attempts to distinguish the fake samples from real and synthetic data. Yang *et al.* [25] employ a stacked LSTM with self-attention mechanism to effectively extract the hidden patterns of the log template sequences and perform anomaly detection for logs.

In addition to leveraging Recurrent Neural Network (RNN) as well as its variant like LSTM to model log sequences, some approaches adopt graphs to model logs. Log2Vec is proposed in [26] as a graph embedding based method for threat detection. Specifically, they construct a heterogeneous graph with heuristic rules and embed the log entries by graph embedding approach. Then, they cluster the logs based on the embedding vectors and consider clusters whose size smaller than a threshold to be malicious. However, some researchers find that existing methods have not performed well in practice, because they assume that the log data is stable over time, and the set of distinct log events is known. Nevertheless, the empirical study shows that log data often contains previously unseen log events or log sequences in practice. Therefore, some researchers focus on identifying and handling unstable log events and sequences caused by the evolution of logging statements and processing noise. LogRobust [27] eliminates the impact of instability by converting log event sequences into semantic vectors and takes advantage of the attention-based Bi-LSTM network to perform anomaly detection of sequential log data.

Due to the explosive growth of logs, researchers pay more attention to automated log analysis. For instance, He *et al.* [28] study the related work of automated log analysis and emphasize the importance of automated log analysis. The complexity of IT systems grows rapidly, which makes the failure diagnosis even challenging. As mentioned in the paper [29], failure diagnosis takes over 100 billion dollars and developers spend more than half of their time on debugging. To reduce the human intervention, researchers have made a lot of efforts to come up with solutions for automatic root cause analysis. In [30], the system abnormal task is detected and diagnosed via transforming the root cause analysis problem into a multi-classification problem by feature engineering. However, the analysis is not accurate and relies on feature engineering. As a remedy, DISTALYZER [31] automatically performs failure diagnosis for a distributed system using logs without feature engineering. Still focus on distributed systems, Shang *et al.* [32] perform failure diagnosis for Hadoop system by analyzing the logs. In detail, they manually inject failures into the system and make a comparison of different log sequences, thus diagnosing big data analytics applications in Hadoop system.

Some statistical methods can also diagnose failures. For instance, FDiag is a diagnostics tool proposed in [33], which leverages statistical correlation analysis to deal with the parsed logs and find a possible root cause for a failure. But FDiag is only capable of diagnosing the known failures. As a remedy, Chuah *et al.* [34] further improve their approach, making it possible to diagnose the unknown failures. Specifically, they use PCA-based and Independent Component Analysis (ICA) based correlation analysis approaches to extract the features of failures automatically. Different from the previous approaches, [35] analyzes the logs to extract the causality of the system event. Yuan *et al.* [36] make use of word2vec model to vectorize historical failures to built

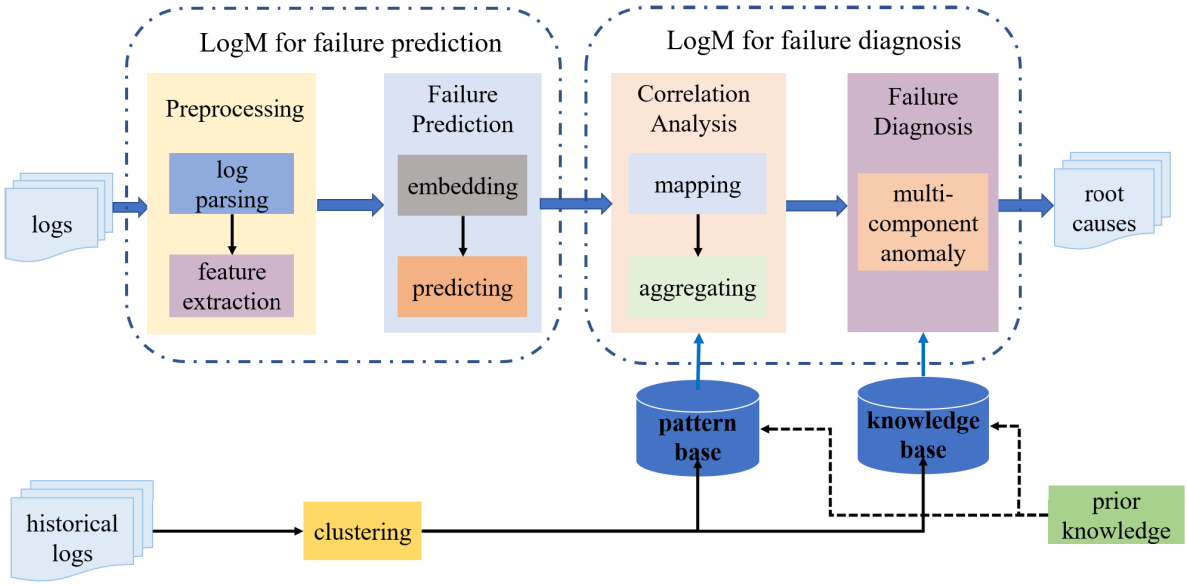


FIGURE 1. An overview flowchart of the *LogM*. One module in the framework performs failure prediction and the other performs failure diagnosis.

different classifiers, and when a failure occurs, the corresponding classifier is employed to find the root cause. However, such studies only focus on dealing with log data generated by the system. Log3C [37] proposes to use both log sequence data and system KPI (Key Performance Indicator) data. Through iterative sampling and cascading clustering, Log3C can identify the system problems timely and accurately without much user intervention.

However, all the works mentioned above do not take into account the case of heterogeneous logs from different components and thus cannot solve the anomaly prediction and failure diagnosis problem of multiple components within a system. As a remedy, we employ a CNN to extract semantic information from the system logs and then leverage an attention-based Bi-LSTM to learn the temporal dynamics from the sequential log data to improve the performance of failure prediction. Furthermore, we adopt a knowledge graph to aid failure diagnosis and implement both Siamese neural network and the clustering algorithm for failure diagnosis.

III. LOG ANALYSIS FOR MULTIPLE COMPONENTS OF HADOOP PLATFORM

This paper proposes a general-purpose framework called *LogM* to tackle the problem of log-based failure prediction and diagnosis. Our framework mainly consists of two modules, namely *LogM* for failure prediction and *LogM* for failure diagnosis. The first module focuses on preprocessing the high-dimensional and unstructured logs and realizes the early detection of the system failures using logs from multiple components. The second module carries out the anomaly root cause analysis via analyzing the logs from multiple components. Figure 1 depicts the overall flowchart of *LogM*. For clarity, we summarize the notations in the following Table 1.

TABLE 1. Symbols and description of variables.

Symbol	Description
l	log entries $l = \{l^{(1)}, l^{(2)}, ..., l^{(m)}\}$
p	log template base $p = \{p^{(1)}, p^{(2)}, ..., p^{(m)}\}$
$l^{(i)}$	i^{th} log sequence
$p^{(i)}$	i^{th} log template sequence
t_ω, Δt_s	time window and time step
$l_{(a)}, l_{(n)}$	anomalous and normal log groups
$p_{(a)}, p_{(n)}$	templates in the anomalous and normal log groups
$\beta[i]$	anomalous probability of the i^{th} log sequence
α	a threshold to determine whether a log sequence is anomalous
g	log group $g = \{g_1, g_2, ..., g_m\}$
$\underline{\boldsymbol{v_i}}$	i^{th} log vector in the log group g
$\underline{\boldsymbol{k_i}}$	j^{th} log vector from the anomaly knowledge base
s_{ij}	cosine similarity of $\underline{\boldsymbol{v_i}}$ and $\underline{\boldsymbol{k_j}}$

A. LogM FOR FAILURE PREDICTION

1) PREPROCESSING

Log preprocessing has three purposes. First, it filters out a large amount of redundant information. Second, it parses the high-dimensional and unstructured log entries into structured data. Finally, it builds a template base using a one-pass clustering algorithm we have developed.

As a matter of fact, some log entries are useless and redundant, such as the logs that periodically report the system running status. They usually do not contain much useful information for failure prediction and diagnosis, and will affect root cause analysis. Therefore, these logs need to be removed during the preprocessing stage. Besides, the raw logs are high-dimensional and unstructured, we need to parse it into structured data to ease the analysis.

Due to the lack of knowledge about log formats, usage and sources, an important step in analyzing heterogeneous logs is to unveil the internal structure of log data. To this end, a clustering algorithm called one-pass clustering is applied on heterogeneous logs to extract the semantics. We employ

the distance metric shown in (1) to measure the similarity between two logs, where l_i denotes the i^{th} log, p_j denotes the j^{th} log in the template, and $C\_sim(l_i, p_j)$ measures the similarity between two log entries.

$$C\_sim(l_i, p_j) = \frac{l_i \cdot p_j}{||l_i|| \times ||p_j||} = \frac{\sum_{k=1}^{n} l_{ik} \times p_{jk}}{\sqrt{\sum_{k=1}^{n} l_{ik}^2} \times \sqrt{\sum_{k=1}^{n} p_{jk}^2}} \tag{1}$$

One-pass Clustering: The key to the one-pass clustering algorithm is the calculation of pair-wise log similarity. The similarity metric measures the minimum similarity among logs to be analyzed and the existing clusters. Specifically, the algorithm starts from the first log and processes the log messages one by one. For each log, we calculate the similarity C_sim between it and the existing clusters. We find out all the existing clusters whose similarities exceed a pre-defined threshold and take the cluster with the largest similarity as the nearest cluster. If no similarity exceeds the pre-defined threshold, we create a new cluster by taking the corresponding log as the representative log.

In the process of one-pass clustering, we determine the pre-defined threshold automatically. In detail, we first calculate the cosine distance between each two samples. Then, we use K-means algorithm to cluster all these cosine distances into two clusters. Next, we calculate the mean of these two clusters. For the cluster with a smaller mean we find its maximum value, denoted as t_1. Likewise, for the cluster with a larger mean we find its minimum value, denoted as t_2. Finally, we choose a threshold from t_1 and t_2. In the experiment, we set t_1 as the pre-defined threshold.

Our proposed algorithm only needs to scan all logs once and is computationally efficient. For the OPTICS-based algorithm [38], it needs to calculate the distance between each input data vector and the reachable object for a number of times, which is computationally intensive. Its computational complexity is $O(n^2)$, where n is the number of samples. Instead, the proposed one-pass clustering algorithm only needs to calculate the distance between the sample and all existing clusters. The computational complexity is $O(kn)$, where k is the number of clusters, which is much lower than $O(n^2)$. Thus our algorithm has lower time and memory requirements than OPTICS-based algorithm and is thus particularly suitable for real-time data processing of massive logs.

2) FAILURE PREDICTION

When a failure occurs, the components in the Hadoop platform will generate a large amount of real-time log stream data, which makes it possible to predict the failure. In this paper, an end-to-end failure prediction model is adopted, which aims to predict the failure and report to the operators in advance, thus effectively reducing the number of users affected by failures.

CAB net implicitly captures the non-linear and high-dimensional dependencies among logs from the training data containing normal system execution workflows.

Figure 2 illustrates the structure of CAB net. More details about failure prediction using CAB net are given in Algorithm 1, where $\underline{s}$ denotes the failure prediction result, s_i denotes the i^{th} log sequence in the test set, $\beta[i]$ denotes the anomaly probability of the i^{th} log sequence, and $s[i]$ is a 0-1 indicator variable indicating whether the i^{th} log sequence is anomalous.

For the log set l, we use the following steps to train the failure prediction model.

- First, we utilize a sliding time window with the setting of t_w and Δt_s to divide the logs into different groups, i.e., $l_{(a)}$ and $l_{(n)}$. The t_w denotes the size of the time window,
  Δt_s represents the time step.
- Next, we perform the template matching for each log within the group, after which we can get the log template group denoted as $p_{(a)}$ and $p_{(n)}$ with the help of the template base obtained in the preprocessing stage.
- Finally, we train the CAB net with these log template groups for system failure prediction.

In the experiment, a failure is defined as a log group containing error messages, which is denoted as $l_{(a)}$, and a normal log group is denoted as $l_{(n)}$. This CAB net leverages a CNN with attention-based Bi-LSTM to model the system logs, which can automatically extract the semantic information of logs and capture the temporal dynamics of sequential log data.

Algorithm 1 End-to-End Failure Prediction

Input: Log groups $l_{(a)}$, $l_{(n)}$
Output: Sequence of failure prediction $\underline{s}$

```
// Train the failure prediction model
use l_(a),l_(n) to train the CAB net (CAB(∗))
// Initialize the anomaly threshold
α ← Initialize parameters
for each log sequence s_i in the test set do
  // Obtain the anomaly probability of
  the i^th log sequence
  β[i] = CAB(s_i)
  // Predict whether the log sequence
  is anomalous
  if β[i] > α then
    s_i is an anomaly, s[i] = "Anomalous"
  else
    s_i is not an anomaly, s[i] = "Normal"
  end if
end for
```

B. LogM FOR FAILURE DIAGNOSIS

1) CORRELATION ANALYSIS

In a large-scale system, logs are usually generated by many geographically distributed components and then uploaded to

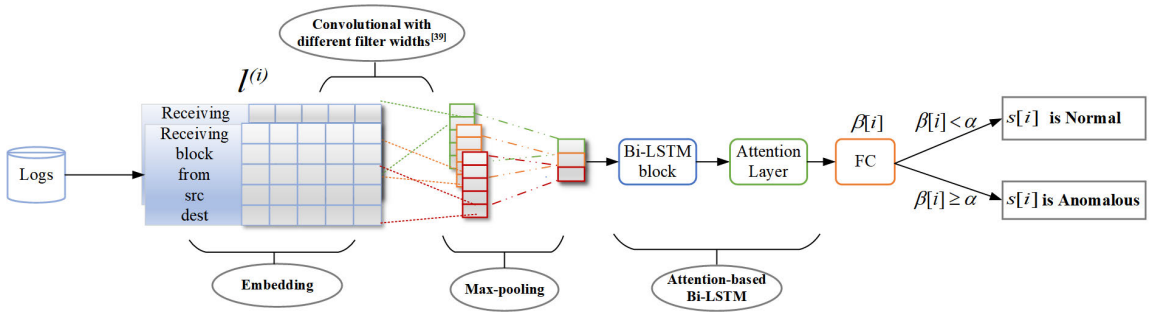


FIGURE 2. A detailed view of CAB net for failure prediction on logs [39]. $I^{(i)}$ is the i^{th} log sequence, $\beta[i]$ is the anomalous probability of the log sequence(best viewed in color).

a centralized platform for analysis. However, due to network disconnection or limited system throughput, it will lead to a series of problems, such as missing logs, duplicate logs or disordered logs. This will bring us lots of difficulties in analyzing system failures. Therefore, we aggregate the correlated logs into a group to help users diagnose the failure more accurately once a failure is identified. To properly aggregate the logs from different components involved in the same failure, we use the temporal feature and keywords feature at the same time, as detailed in the following *mapping* and *aggregation* paragraphs. After aggregating the logs into candidate groups, we use the metrics of *confidence* and *lift* [40] to filter out the log groups that are not strongly correlated.

Mapping: To perform efficient correlation analysis, we first map the log vector within the template to the category and use the corresponding tag to represent the template. Besides, we adopt the one-pass clustering algorithm to build a category library using anomaly logs, each of which represents a type of anomaly log template within a component. Specifically, we assign different tags to different components, such as Hive-A, HBase-B, HDFS-C, etc., and use different tags to represent different template categories within the component, such as Class 1, Class 2, Class 3, etc. In doing so, we can get A-2 and C-1, etc. as shown in the label column in Table 2.

Aggregation: After mapping each log to the corresponding template, we use timestamps, keywords and labels within the template to generate a tuple, as shown in Table 2. Among them, the timestamp comes directly from the system logs; the keywords mainly consist of the identifiers and variables, where identifiers are used to identify objects utilized by the program; and the variables are used to enumerate a set of possible states of the identifiers. Logs with common identifiers are likely to come from the same event and have a strong correlation. Besides, logs with common variables over a period of time possibly come from the same event [18]. Therefore, for logs from the same component, we aggregate the logs with common identifiers or variables into one group within the time interval $[t_{start}, t_{end}]$. As shown in Figure 3, all logs of component A with the identifier blk_1 will be aggregated into a group to generate a variable set $\Gamma_{var} = \{var_1, var_2, var_3\}$. Next, we look into the logs of component B within a time interval $[t_{start} - \Delta t, t_{end} + \Delta t]$, where Δt is a parameter used to adjust the size of the time window. Then, other logs

TABLE 2. Example of log entries from different components.

Label	Timestamp	Key words
A-1	2018-05-29 13:40:06	(identifier, variable)
A-2	2018-05-29 14:03:44	(identifier, variable)
B-1	2018-05-29 17:12:32	(identifier, variable)
C-1	2018-05-29 17:12:00	(identifier, variable)
C-2	2018-05-29 17:05:18	(identifier, variable)

with a variable in Γ_{var} or with the identifier blk_1 are also aggregated into the same group. In doing so, logs with a strong correlation but belonging to different components are aggregated into a group to aid failure diagnosis.

2) FAILURE DIAGNOSIS

Since it usually takes a while to diagnose and repair a failure, the system service may get interrupted for several hours, resulting in significant system loss. To address this problem, we aim to predict system failures and offer the operators reasonable solutions in advance. Once the failure is predicted, we diagnose it without waiting to collect all the logs until the failure occurs. Compared with traditional methods that analyze failures after they have occurred, our approach uses relatively fewer logs and is much more difficult.

Accurate and efficient failure diagnosis can effectively help operators to locate failures and speed up the diagnosis of system problems. In a Hadoop platform, there exist dependencies among various components, and the failure caused by one component can also make the other related components report an error. The purpose of the failure diagnosis is to analyze the root cause using the logs from multiple components. In this paper, we offer two approaches. One is an unsupervised learning approach, and the other is a supervised learning approach. Both approaches have pros and cons. In the following, we will describe and compare them in detail.

Unsupervised failure diagnosis: The key of the unsupervised failure diagnosis approach is to measure the pair-wise similarity of the logs from anomaly log group and the knowledge base.

To calculate the similarities among different logs, we map multiple correlated logs into vectors. For the anomaly knowledge base, which will be introduced at the end of Section III, the d logs i.e., $l_1, l_2, \ldots, l_d$ are converted into d n-dimensional vectors i.e., $\underline{\boldsymbol{k}}_{\boldsymbol{1}}, \underline{\boldsymbol{k}}_{\boldsymbol{2}}, \ldots, \underline{\boldsymbol{k}}_{\boldsymbol{d}}$. For the anomaly

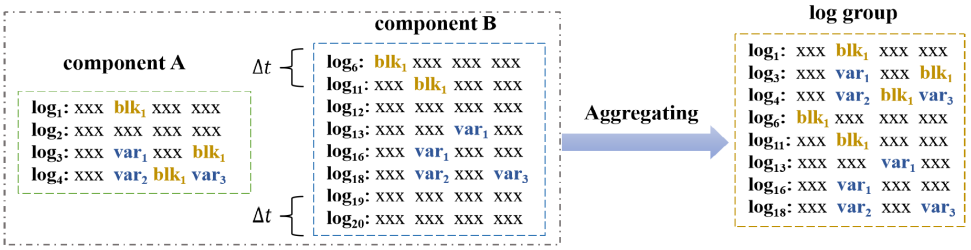


FIGURE 3. An example of correlation analysis for multi-component logs (best viewed in color).

log group g, the m logs i.e., $g_1, g_2, \ldots, g_m$ are converted into m n-dimensional vectors i.e., $\underline{v_1}, \underline{v_2}, \ldots, \underline{v_m}$, where n is the size of the dictionary generated by the *bag-of-words* model [41] and each dimension in the vector is a value of *TF-IDF* corresponding to the word item within the log. The *TF-IDF* value is calculated by (2), where t denotes a word item in the log l, $f_{t,l}$ denotes the frequency of word item t appears in a log, n denotes the total number of logs, and n_t denotes the number of logs with the word item t.

By calculating the pair-wise similarity between each log in g and each log from the knowledge base, we can obtain a d-dimensional similarity vector. As shown in (3), Sim_i indicates the similarity vector of the i^{th} log in g. s_{ij} can be calculated by (4), where $cos(\underline{v_i}, \underline{k_j})$ represents the Cosine similarity of vector $\underline{v_i}$ and $\underline{k_j}$. Specifically, $\underline{v_i}$ is the i^{th} log vector within g, and $\underline{k_j}$ is the j^{th} log vector from the knowledge base. To effectively select the root cause of each anomaly log group, we analyze each Sim_i from the similarity list.

$$TF\text{-}IDF_{t,l} = f_{t,l} * log\frac{n}{n_t} \tag{2}$$

$$Sim_i = [s_{i1}, s_{i2}, s_{i3}, \ldots, s_{id}] \tag{3}$$

$$s_{ij} = cos(\underline{v_i}, \underline{k_j}) \tag{4}$$

$$p(s_{ij}) = \frac{s_{ij}}{\sum_{j=1}^{d} s_{ij}} \tag{5}$$

Analysis strategy for similarity list: In order to analyze the similarity list, we first build a multi-layer weighted undirected graph using the similarity list and anomaly knowledge graph. Figure 4 illustrates our multi-layer weighted undirected graph. The multi-layer weighted undirected graph mainly includes three layers, namely log layer, knowledge layer, and root cause layer, where *log* represents a log to be analyzed, $\underline{k_i}$ indicates the log in the knowledge base, and rc_i represents the corresponding root cause. As shown in Figure 4, the weight $p(\cdot)$ between two layers indicates the probability of the *log* corresponding to the $\underline{k_i}$. We calculate the weight between the log layer and the knowledge layer according to (5), where s_{ij} comes from the similarity list.

Based on the multi-layer weighted undirected graph, we propose two methods to infer the candidate root causes. For simplicity, the proposed methods are called unsupervised method 1 and unsupervised method 2 hereafter, respectively. The unsupervised method 1 aims to select the top-n logs from the knowledge base that are most similar to each anomaly

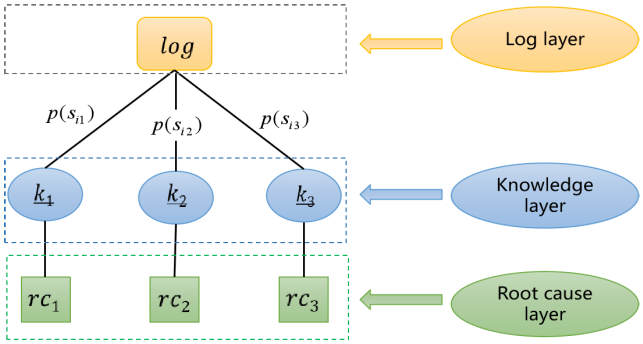


FIGURE 4. The illustration of the multi-layer weighted undirected graph. $p(\cdot)$ is the weight between log layer and knowledge layer(best viewed in color).

log group. We calculate the number of occurrences for each corresponding root cause, from which the most frequent one is regarded as the candidate root cause. Upon finding the most frequent cause, the unsupervised method 2 suggests to further sum up the $p(\cdot)$ for the n logs according to their root causes and take the root cause with the largest sum of $p(\cdot)$ as the candidate root cause.

Supervised failure diagnosis: Essentially, each anomaly log group is an event caused by a failure, and the ultimate goal of supervised failure diagnosis is to establish a reasonable mapping between the root cause and the event. To achieve efficient failure diagnosis, we use a Siamese LSTM network [42] to compare the log sequences. The structure diagram of the Siamese LSTM network is shown in Figure 5.

Unlike the other neural networks, the Siamese network requires paired data at the input layer. Specifically, log sequences from the anomaly group and the knowledge base are simultaneously input to the Siamese LSTM network in order. As shown in Figure 5, l_i represents the i^{th} log from the knowledge base, g_i represents the i^{th} log in the anomaly group g, $\underline{w_i}$ is the corresponding word vector obtained through the trained *word2vec* model [43]. The distance between two inputs is calculated to determine whether they are caused by the same failure. To measure the similarity, we introduce a metric called *scaled_sim*. The specific formula is shown in 6, where $\underline{m_1}, \underline{m_2}$ represent the output of the last hidden neuron, respectively. The two LSTM sub-networks in the Siamese network share the same weights, which can effectively reduce the number of training parameters and improve the computing

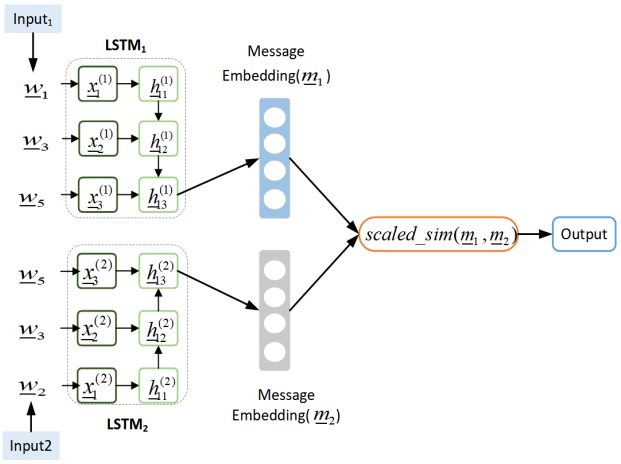


FIGURE 5. The structure of Siamese LSTM network. The output is the *scaled_sim* between two log sequences.

and storage performance.

$$scaled\_sim(\underline{\boldsymbol{m}}_1, \underline{\boldsymbol{m}}_2) = e^{-(||\underline{\boldsymbol{m}}_1 - \underline{\boldsymbol{m}}_2||_1)} \tag{6}$$

$$||\underline{\boldsymbol{m}}_1 - \underline{\boldsymbol{m}}_2||_1 = \sum_{i=1}^{n} |\underline{\boldsymbol{m}}_1 - \underline{\boldsymbol{m}}_2| \tag{7}$$

More details about the supervised failure diagnosis using Siamese LSTM network are given in Algorithm 2. First, we train a *word2vec* model using the historic system logs to obtain the corresponding word vectors. Then, the paired word vectors are sequentially input into the Siamese LSTM network to calculate the similarity through the *scaled_sim* shown in (6) and (7). Finally, we analyze the similarity to obtain the root causes. The *word2vec* model we use in this paper is mainly based on the continuous bag-of-words(CBOW) model, which tries to predict the current word based on the context [41]. Equation (8) shows the objective function of the CBOW model, where ω_t denotes the current word, c_t denotes the context,

$$\frac{1}{T}\sum_{t=1}^{T} logP(\omega_t|c_t) \tag{8}$$

For the two failure diagnosis approaches described above, both of them aim to measure the pair-wise similarity between logs. The unsupervised failure diagnosis approach is relatively simple and efficient, while slightly inferior to the supervised approach in diagnosis performance. As for the supervised failure diagnosis approach, it performs better in diagnosis, while it is often time-consuming and costly due to the necessity of labelling the training data.

Anomaly knowledge graph: For system failure diagnosis, we need a knowledge base containing various root causes and abnormal events of the Hadoop platform. In this paper, we propose to use the knowledge graph to store the root causes and abnormal events.

We propose a knowledge graph called *anomaly knowledge graph*, as shown in Figure 6. In this anomaly knowledge graph, we represent the abnormal events with green nodes and

Algorithm 2 Failure Diagnosis With Siamese LSTM

Input: $l = \{l_1, l_2, \ldots, l_d\}, g = \{g_1, g_2, \ldots, g_m\}$
Output: Similarity value $scaled\_sim(i, j)$

```
Train the word2vec model using historic logs
// Get the corresponding word vectors
for each log g_i in g do
    word2vec(g_i) = {w_1, ..., w_k}
    for each log l_j in l do
        word2vec(l_j) = {w_1, ..., w_n}
        // Calculate the similarity
        sequentially input the word2vec(g_i) and word2vec(l_j)
        to the Siamese LSTM network
        calculate the scaled_sim(i, j)
    end for
end for
```

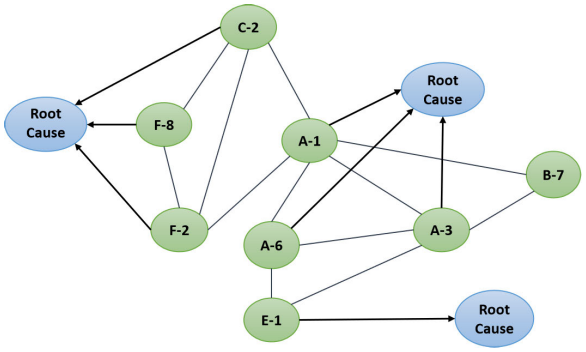


FIGURE 6. An example of the anomaly knowledge graph. The green node indicates the abnormal event and the blue node denotes root cause (best viewed in color).

the root causes with blue nodes, where correlated green nodes are connected with undirected edges. Blue and green nodes are linked by directed edges, which indicates that an abnormal event is caused by the root cause. Besides, our proposed knowledge graph is more concise and easier to implement with little prior knowledge. With the help of the anomaly knowledge graph, we can capture the relationships among different anomalous logs and their corresponding root causes.

IV. PERFORMANCE EVALUATION

To evaluate the performance of our proposed *LogM* framework, we analyze the logs generated by a Hadoop platform in the production environment.

A. DATASET DESCRIPTION

The data we use are logs collected from a cluster of four nodes in a commercial Hadoop platform. Totally more than fifty million logs are collected in one month, covering six different components of the Hadoop system, namely Hive, HBase, YARN, HDFS, MapReduce and Zookeeper. The real-world dataset contains four levels of logs, namely INFO, WARN, ERROR, and FATAL. Table 3 shows the statistics of the log dataset we use, where the *log messages* column shows the specific number of logs of the corresponding component.

TABLE 3. Dataset used for evaluation.

System	Module	Log Messages
Hadoop	Hive	1,823,405
	HBase	19,720,672
	Yarn	23,178,246
	HDFS	12,177,270
	MapReduce	593,473
	Zookeeper	632,795

B. EXPERIMENT RESULTS

1) PERFORMANCE COMPARISON OF LOG CLUSTERING ALGORITHMS

To verify the effectiveness of the one-pass clustering algorithm, we compare it with the OPTICS-based clustering algorithm on the same dataset. Figure 7 depicts the difference in clustering speed between two algorithms as the number of logs increases. From Figure 7, it is seen that one-pass clustering algorithm always requires less processing time than the OPTICS-based algorithm.

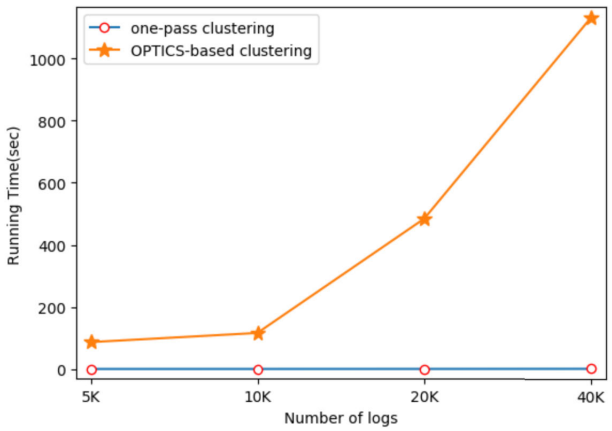


FIGURE 7. Time complexity comparison between two clustering algorithms (best viewed in color).

2) PERFORMANCE COMPARISON OF FAILURE PREDICTION ALGORITHMS

To validate the effectiveness of *LogM* in failure prediction, we use the Precision-Recall curve (PR curve) as the evaluation metric. Equation (9) and (10) calculate the evaluation metrics corresponding to the x-axis and y-axis in the PR curve, respectively. Specifically, true positive (TP) indicates the number of anomalous log sequences that are correctly predicted, false positive (FP) indicates the number of anomalous log sequences that are incorrectly predicted, and false negative (FN) indicates the number of anomalous log sequences that are never detected. We can get multiple sets of Precision and Recall to draw a PR curve with different threshold settings. The CAB net model is built with Keras toolbox [44] and trained on NVIDIA TITAN X GPUs. We choose the cross-entropy as the loss function and use Adam [45] for model training.

$$Precision = \frac{TP}{TP + FP} \tag{9}$$

$$Recall = \frac{TP}{TP + FN} \tag{10}$$

We compare the CAB net with the attention-based LSTM network, attention-based Bi-LSTM network, and CNN with LSTM (C_LSTM) on the dataset collected from a Hadoop platform. Figure 8 shows the performance comparison of failure prediction on the real-world dataset. According to the PR curves, it is seen that our proposed CAB net model outperforms all the other three networks, namely the attention-based LSTM network, attention-based Bi-LSTM network, and C_LSTM.

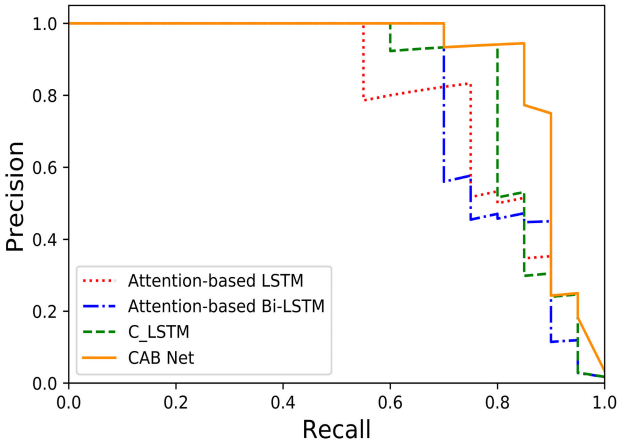


FIGURE 8. The performance comparison of four different failure prediction models on the real-world dataset, where the solid yellow line represents our CAB net (best viewed in color).

3) PERFORMANCE COMPARISON OF FAILURE DIAGNOSIS APPROACHES

The test dataset used to evaluate the performance of these failure diagnosis approaches contains both anomaly log groups and their root causes. We use *accuracy* to measure the performance of different failure diagnosis approaches, which is defined as the ratio of the number of anomaly log groups correctly analyzed, denoted as N_{right}, and the number of all anomaly log groups be analyzed, denoted as N. According to the definition, a larger *accuracy* indicates a better performance of the proposed failure diagnosis approach.

We carry out detailed experiments to compare different failure diagnosis approaches. In order to improve the reliability of failure diagnosis, we adjust the n value in *top n*. Figure 9 depicts the performance comparison of different unsupervised diagnosis approaches, which shows that as n increases, the performance of unsupervised method 2 tends to become slightly better than that of unsupervised method 1.

The performance comparison of the supervised and unsupervised diagnosis approaches is given in Figure 10. To facilitate the analysis, we compare the best *accuracy* of the unsupervised failure diagnosis approach with that of the

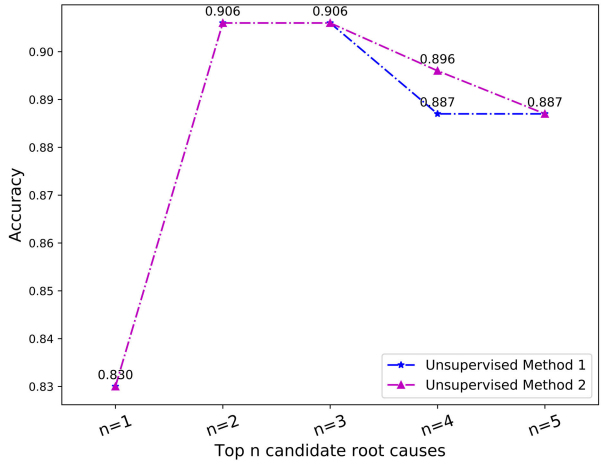


FIGURE 9. **Performance comparison between unsupervised failure diagnosis approaches as *n* varies (best viewed in color).**

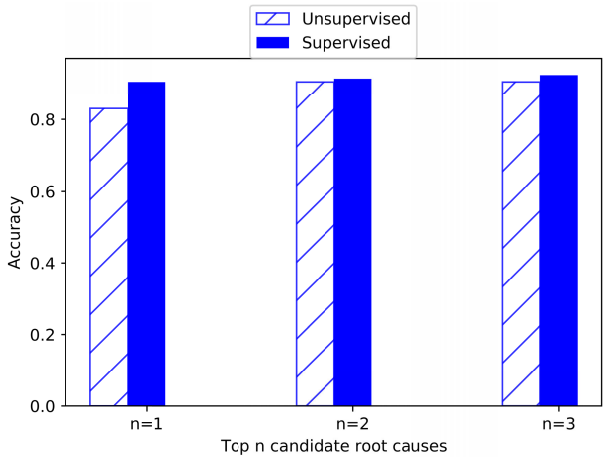


FIGURE 10. **The comparison of supervised and unsupervised failure diagnosis algorithms with different *n* values (best viewed in color).**

supervised failure diagnosis approach. To better assist the operators in finding the root cause of the failure, the value of n cannot be too large, so we set n to be 1, 2, and 3 in the experiment. As shown in Figure 10, the performances of both approaches increase with n, while the supervised failure diagnosis approach is slightly better than the unsupervised one. Specifically, when $n = 3$, the *accuracy* of the unsupervised failure diagnosis approach can reach **0.906**, and the *accuracy* of the supervised approach can reach **0.925**.

V. CONCLUSION AND FUTURE WORK

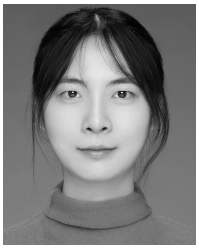
In this paper, we propose a framework called *LogM* for failure prediction and diagnosis on unstructured logs. Specifically, it parses high-dimensional unstructured logs into structured data to efficiently process large amounts of log data. Then, *LogM* employs an end-to-end CAB net model to extract the semantic information and temporal dynamics of the sequential log data for failure prediction. After that, we adopt correlation analysis to aggregate the anomalous logs to effectively aid the failure diagnosis. With the help of the anomaly knowledge graph, we finally carry out both the unsupervised and supervised failure diagnosis approaches.

In general, it is significant but challenging to analyze logs for multiple components of Hadoop platform. We design an CAB net model that can effectively predict system failures, thus enables the engineers to perform preventative actions and improve service availability. Besides, the correlation analysis approach and the anomaly knowledge graph efficiently aid the failure diagnosis in real-world scenarios. Our framework has been trained and tested on a real-world dataset and the experimental results show that the proposed framework can effectively predict and diagnose the failure of the Hadoop platform. We also plan to make the real-world test dataset publicly available for the research community.

REFERENCES

[1] J. Fan, F. Han, and H. Liu, "Challenges of big data analysis," *Nat. Sci. Rev.*, vol. 1, no. 2, pp. 293–314, 2014.
[2] T. White, *Hadoop: Definitive Guide*. Newton, MA, USA: O'Reilly Media, 2012.
[3] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, "The Hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol.*, May 2010, pp. 1–10.
[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.
[5] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O'Malley, S. Radia, B. Reed, and E. Baldeschwieler, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, Oct. 2013, pp. 1–16.
[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *Proc. 2nd USENIX Conf. Hot Topics Cloud Comput.*, 2010, pp. 95–102.
[7] L. George, *HBase: The Definitive Guide*. Newton, MA, USA: O'Reilly Media, Inc., 2011.
[8] P. Hunt, M. Konar, F. P. Junqueira, and B. Reed, "ZooKeeper: Wait-free coordination for Internet-scale systems," in *Proc. USENIX Annu. Tech. Conf.*, 2010, vol. 8, no. 9, pp. 1–14.
[9] D. Oppenheimer, A. Ganapathi, and D. A. Patterson, "Why do Internet services fail, and what can be done about it," in *Proc. 4th USENIX Symp. Internet Technol. Syst.*, 2003, pp. 1–15.
[10] A. Oliner, A. Ganapathi, and W. Xu, "Advances and challenges in log analysis," *Commun. ACM*, vol. 55, no. 2, pp. 55–61, Feb. 2012.
[11] Y. Zhang and A. Sivasubramaniam, "Failure prediction in IBM BlueGene/L event logs," in *Proc. IEEE Int. Symp. Parallel Distrib. Process.*, Apr. 2008, pp. 583–588.
[12] S. He, J. Zhu, P. He, and M. R. Lyu, "Experience report: System log analysis for anomaly detection," in *Proc. IEEE 27th Int. Symp. Softw. Rel. Eng. (ISSRE)*, Oct. 2016, pp. 207–218.
[13] T. Li, Y. Jiang, C. Zeng, B. Xia, Z. Liu, W. Zhou, X. Zhu, W. Wang, L. Zhang, J. Wu, L. Xue, and D. Bao, "FLAP: An end-to-end event log analysis platform for system management," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2017, pp. 1547–1556.
[14] S. Khatuya, N. Ganguly, J. Basak, M. Bharde, and B. Mitra, "ADELE: Anomaly detection from event log empiricism," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2018, pp. 2114–2122.
[15] X. Ning, G. Jiang, H. Chen, and K. Yoshihira, "HLAer: A system for heterogeneous log analysis," in *Proc. SDM Workshop Heterogeneous Learn.*, 2014, pp. 1–22.
[16] B. Debnath, M. Solaimani, M. A. G. Gulzar, N. Arora, C. Lumezanu, J. Xu, B. Zong, H. Zhang, G. Jiang, and L. Khan, "LogLens: A real-time log analysis system," in *Proc. IEEE 38th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2018, pp. 1052–1062.
[17] Q. Lin, H. Zhang, J.-G. Lou, Y. Zhang, and X. Chen, "Log clustering based problem identification for online service systems," in *Proc. 38th Int. Conf. Softw. Eng. Companion*, May 2016, pp. 102–111.

[18] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, "Detecting large-scale system problems by mining console logs," in *Proc. ACM 22nd Symp. Oper. Syst. Princ. (SOSP)*, 2009, pp. 117–132.
[19] J.-G. Lou, "Mining invariants from console logs for system problem detection," in *Proc. USENIX Annu. Tech. Conf.*, 2010, pp. 24–38.
[20] K. Zhang, J. Xu, M. R. Min, G. Jiang, K. Pelechrinis, and H. Zhang, "Automated IT system failure prediction: A deep learning approach," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2016, pp. 1291–1300.
[21] M. Du, F. Li, G. Zheng, and V. Srikumar, "DeepLog: Anomaly detection and diagnosis from system logs through deep learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1285–1298.
[22] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, and R. Zhou, "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, Aug. 2019, pp. 4739–4745.
[23] M. Platini, T. Ropars, B. Pelletier, and N. De Palma, "LogFlow: Simplified log analysis for large scale systems," in *Proc. Int. Conf. Distrib. Comput. Netw.*, Jan. 2021, p. 116.
[24] B. Xia, Y. Bai, J. Yin, Y. Li, and J. Xu, "LogGAN: A log-level generative adversarial network for anomaly detection using permutation event modeling," *Inf. Syst. Frontiers*, vol. 7, pp. 1–14, Jun. 2020.
[25] R. Yang, D. Qu, Y. Gao, Y. Qian, and Y. Tang, "NLSALog: An anomaly detection framework for log sequence in security management," *IEEE Access*, vol. 7, pp. 181152–181164, 2019.
[26] F. Liu, Y. Wen, D. Zhang, X. Jiang, X. Xing, and D. Meng, "Log2vec: A heterogeneous graph embedding based approach for detecting cyber threats within enterprise," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 1777–1794.
[27] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, J. Chen, X. He, R. Yao, J.-G. Lou, M. Chintalapati, F. Shen, and D. Zhang, "Robust log-based anomaly detection on unstable log data," in *Proc. 27th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, Aug. 2019, pp. 807–817.
[28] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," 2020, *arXiv:2009.07237*. [Online]. Available: http://arxiv.org/abs/2009.07237
[29] Y. Zhang, K. Rodrigues, Y. Luo, M. Stumm, and D. Yuan, "The inflection point hypothesis: A principled debugging approach for locating the root cause of a failure," in *Proc. 27th ACM Symp. Oper. Syst. Princ.*, Oct. 2019, pp. 131–146.
[30] S. Lu, B. Rao, X. Wei, B. Tak, L. Wang, and L. Wang, "Log-based abnormal task detection and root cause analysis for spark," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, Jun. 2017, pp. 389–396.
[31] K. Nagaraj, C. Killian, and J. Neville, "Structured comparative analysis of systems logs to diagnose performance problems," in *Proc. 9th USENIX Conf. Networked Syst. Design Implement.*, 2012, pp. 26–40.
[32] W. Shang, Z. M. Jiang, H. Hemmati, B. Adams, A. E. Hassan, and P. Martin, "Assisting developers of big data analytics applications when deploying on Hadoop clouds," in *Proc. 35th Int. Conf. Softw. Eng. (ICSE)*, May 2013, pp. 402–411.
[33] E. Chuah, S.-H. Kuo, P. Hiew, W.-C. Tjhi, G. Lee, J. Hammond, M. T. Michalewicz, T. Hung, and J. C. Browne, "Diagnosing the root-causes of failures from cluster log files," in *Proc. Int. Conf. High Perform. Comput.*, Dec. 2010, pp. 1–10.
[34] E. Chuah, A. Jhumka, J. C. Browne, B. Barth, and S. Narasimhamurthy, "Insights into the diagnosis of system failures from cluster message logs," in *Proc. 11th Eur. Dependable Comput. Conf. (EDCC)*, Sep. 2015, pp. 225–232.
[35] S. Kobayashi, K. Otomo, K. Fukuda, and H. Esaki, "Mining causality of network events in log data," *IEEE Trans. Netw. Service Manage.*, vol. 15, no. 1, pp. 53–67, Mar. 2018.
[36] Y. Yuan, W. Shi, B. Liang, and B. Qin, "An approach to cloud execution failure diagnosis based on exception logs in OpenStack," in *Proc. IEEE 12th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2019, pp. 124–131.
[37] S. He, Q. Lin, J. G. Lou, H. Zhang, and M. R. Lyu, "Identifying impactful service system problems via log analysis," in *Proc. 26th ACM Joint Meeting Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng.*, 2018, pp. 60–70.
[38] M. Ankerst, M. M. Breunig, H. P. Kriegel, and J. Sander, "OPTICS: Ordering points to identify the clustering structure," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1999, pp. 49–60.
[39] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1746–1751.
[40] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2000, pp. 1–12.
[41] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: http://arxiv.org/abs/1301.3781
[42] J. Mueller and A. Thyagarajan, "Siamese recurrent architectures for learning sentence similarity," in *Proc. 30th AAAI Conf. Artif. Intell.*, 2016, pp. 2786–2792.
[43] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 1188–1196.
[44] *Keras: The Python Deep Learning Library*. [Online]. Available: https://keras.io
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent.*, 2014, pp. 1–15.



YUXIA XIE was born in Sichuan, China, in 1995. She received the B.S. degree from Qingdao University, Qingdao, China, in 2017. She is currently pursuing the Ph.D. degree in computer science with the Department of Computer Science and Technology, Tongji University, Shanghai, China. Her major research interests include big data analytics, machine learning, and AIOps.

KAI YANG received the B.Eng. degree from Southeast University, Nanjing, China, the M.S. degree from the National University of Singapore, and the Ph.D. degree from Columbia University, NY, USA. He was a Technical Staff Member with Bell Laboratories, NJ, USA; a Senior Data Scientist with Huawei Technologies, Plano, TX, USA; and a Research Associate with NEC Laboratories America, Princeton, NJ, USA. Since 2011, he has been an Adjunct Faculty Member with Columbia University. He is currently a Distinguished Professor with Tongji University, Shanghai, China. The products he has developed have been deployed by Tier-1 operators and served billions of users worldwide. He has been published extensively in leading IEEE journals and conferences and holds over 20 patents. His current research interests include big data analytics, machine learning, wireless communications, and signal processing. He was a recipient of the Eliahu Jury Award from Columbia University, the Bell Laboratories Teamwork Award, the Huawei Technology Breakthrough Award, and the Huawei Future Star Award. From 2012 to 2014, he was the Vice-Chair of the IEEE ComSoc Multimedia Communications Technical Committee. In 2017, he founded and served as the Chair for the IEEE TCCN Special Interest Group on AI Embedded Cognitive Networks. He served as the Demo/Poster Co-Chair for IEEE INFOCOM, the Symposium Co-Chair for IEEE GLOBECOM, and the Workshop Co-Chair for IEEE ICME. He also serves as an Editor for the IEEE Internet of Things Journal and IEEE Communications Surveys and Tutorials. He also serves as a Guest Editor for the IEEE Journal on Selected Areas in Communications.

PAN LUO was born in Sichuan, China, in 1995. He received the bachelor's degree from Tongji University, Shanghai, China, in 2018, where he is currently pursuing the master's degree in computer science with the Department of Computer Science and Technology. His research interests include big data analytics, data mining, and machine learning.