RESEARCH ON NETWORK CLOUD EQUIPMENT ANOMALY AND ROOT CAUSE ANALYSIS

Dandan Zou[1], Jianbing Ding[1], Xidong Wang[1], Xiaozhou Ye[1], Ye Ouyang[1]
[1]AsiaInfo Technologies(China), Inc. Beijing, China

NOTE: Corresponding author: Jianbing Ding, dingjb@asiainfo.com

Abstract *– With the development of 5G communication technology, a cloud computing system has become a trend. However, with the expansion of the scale of deployment and the increase in framework complexity, ensuring the security and stability of cloud-based systems has become a serious challenge. In a real business environment, existing algorithms are powerless in the face of current problems, such as complex types of abnormal logs, inaccurate time information, and the lack of key information. This paper proposes network cloud equipment anomaly detection and a root cause analysis scheme based on large-scale logs in distributed cluster systems. The scheme uses unsupervised integrated learning, keyword search, and root cause generalization to analyze logs, accurately find anomalies and locate root causes. The F1 score in log anomaly detection is 0.962, and the accuracy in root cause location of anomalies is 0.849. In the ITU AI/ML in 5G Challenge 2021, the solution got the highest final score 93.904 in the China Mobile problem statement of network cloud equipment anomaly and root cause analysis. Furthermore, the scheme has been deployed on China Mobile's 5G network management system and achieves the detection and location of anomalies under intelligent operation.*

Keywords – Artificial intelligence for IT operations, ensemble learning, keyword search, log anomaly detection, network cloud equipment, root cause analysis

1. INTRODUCTION

With the development of network function virtualization (NFV), network operators have accelerated the transformation of communication networks into the cloud. The network cloud equipment not only promotes cost reduction and efficiency with the scale effect of existing equipment but also accelerates business development and realizes resource sharing. The clouding of the network has led to a dramatic increase in the number of devices managed by operations and maintenance staff. In the Artificial Intelligence for IT Operations (AIOps) of modern large-scale distributed systems, timely and efficient detection of anomalous behavior in the system is crucial, so a robust system anomaly detection strategy is needed. Millions of logs that store information such as system operation event flow, hardware information, and key performance indicators are generated every day during system operation. The reasonable usage of artificial intelligence algorithms for log anomaly detection can identify anomalies and greatly reduce the operation and maintenance pressure on complex system anomaly detection.

In recent years, log anomaly detection has been a hot topic in AIOps [1]. Some researchers extracted log sequence information, manually defined the correct sequences, and created rule sets [2]. The above methods are based on manual statistics and analysis, which are powerless for handling huge amounts of data. Then, some machine learning algorithms are applied to log anomaly detection, such as isolation forest [3], Principal Component Analysis (PCA) [4], invariants mining [5], log cluster [6], etc. The effectiveness of the supervised-based approach depends on the quality and quantity of the labeled data. Earlier, logistic regression and support vector machine are used to classify anomalies [7, 8]. From a temporal perspective, the DeepLog framework combined with Long Short-Term Memory (LSTM) is proposed [9]. Furthermore, the Template2Vec approach can extract semantic information hidden in log templates [10]. However, the acquisition of manually labeled tags is difficult. There is a preference for using unsupervised algorithms in scenarios with huge amounts of data.

The logs provided by China Mobile included system operation information and performance indicators. Typically, there are two requirements for log anomaly detection: First, log anomaly detection, which can detect whether there is an anomaly in the daily system logs. Second, log root cause analysis, which can pinpoint the specific location in the log where the anomaly was generated. Most of the methods in the existing papers are limited by the following problems that make them difficult to put into production:

- The types of anomalies in practical application scenarios are complex. The anomalies in the public data sets (HDFS, BGL, etc.) usually have only dozens of types. In real logs, the types of anomalies from different components and processes may reach hundreds or thousands which leads to high difficulty in log anomaly detection.
- The time information is not accurate. There are time errors, delays, and other inaccurate time information in the logs of different machines. Therefore, the time in the log is not reliable.
- Information is missing or invalid. When component information is missing, the logs cannot extract the complete sequence of events. In addition, INFO messages may be the root cause of errors in the eyes of business people, while DEBUG messages are normal instead.

In this paper, we propose a log anomaly and root cause analysis framework based on an OpenStack-based network cloud system. The framework uses unsupervised ensemble learning algorithms to discriminate anomalies and achieve log anomaly detection. Construct a root cause database based on keywords to find root causes and realize root cause analysis. In addition, the scheme uses root cause generalization to optimize the discrimination and search of root causes. This solution can detect anomalies in time, ensure the stability of the system, and avoid production failures in actual business scenarios.

2. FRAMEWORK

Cloud computing systems under distributed clusters have an urgent need for efficient maintenance. The security information, hardware and load conditions of the system need to be dynamically monitored and maintained in real time. Our solution is divided into three main processes as follows:

- Data preprocessing: Distributed clusters collect and distribute the log from network cloud equipment. Logs from various component services are mixed. Separate processing by time and content is required. Hardware information for monitoring also needs to be parsed. Many similar logs expressing the same meaning require data fusion. Data needs to be processed to facilitate subsequent stages.
- Log anomaly detection: The preprocessed log text can usually be split into different log blocks based on specific chunking methods, such as relying on timestamps, process ID, and component types. After extracting the log templates, a library of templates with corresponding numbers is generated. Each log can be matched to the corresponding template and ID. Unsupervised classification models with different focuses are used to learn the data and integrate the results for learning. Abnormal log sequences can be identified more accurately, and anomaly detection can be achieved.
- Log root cause analysis: The distinguished sequence of abnormal logs can be used by the operation and maintenance staff to find the root cause of the logs. The workload is heavy under a large-scale cluster with high throughput and low latency. Root cause analysis algorithms can be added to automatically locate anomalies and detect the time and log nodes of anomalies. Root cause keywords form a root cause library that can be used to retrieve root causes. Root cause generalization optimizes the root cause database. Further manual safeguarding and optimization of the root cause base are required for higher-quality targeting.

It is a general trend to introduce artificial intelligence technology into the field of log analysis and processing. Based on the above process, Fig. 1 shows the flow chart of network cloud equipment anomaly and root cause analysis. The following sections describe log anomaly detection and log root cause analysis in detail.

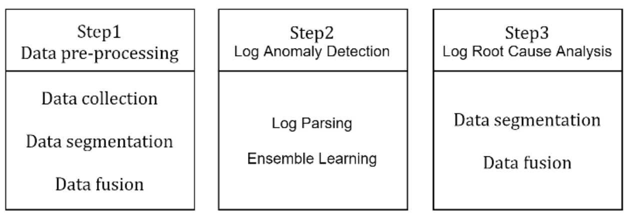

Fig. 1 – The flow chart of network cloud equipment anomaly and root cause analysis

3. LOG ANOMALY DETECTION

The main function of the log anomaly detection module is to classify log data, determine whether it is an abnormal log, and provide the corresponding abnormal discrimination results for the subsequent steps. Here we divide it into two parts: log parsing and unsupervised ensemble learning. Fig. 2 shows the flow chart of log anomaly detection.

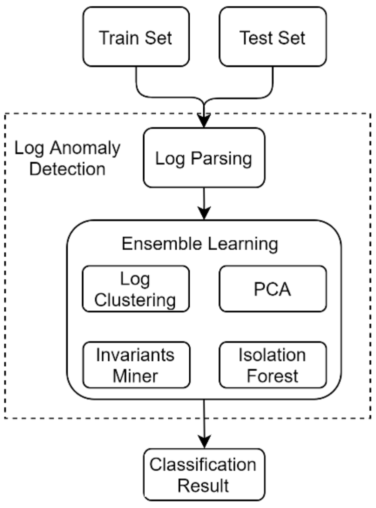

Fig. 2 – The flow chart of log anomaly detection

3.1 Log parsing

Log parsing extracts the log template and obtains the log template ID. The mainstream log parsing algorithms are Drain [11], Spell [12], IPLoM [13], etc. Drain uses a tree structure to store log messages and efficiently extract generic templates. In this paper, the extraction of logs is implemented using the Drain algorithm. The processing object is log data based on the OpenStack framework [14].

Table 1 – Example of train set and test set

Data set	Quantity	Example
Train	264392	1 2021-05-21 14:37:51.029 28838 INFO glance.comon.wsgi [-] Starting 160 workers
Test	224800	1 Start building networks async hronously for instance. build_ resources /usr/lib/python2.7/s itepackages/nova/compute/ma nager.py:2159

The data used in this solution is based on the competition held by China Mobile: network cloud equipment anomaly and root cause analysis [15]. The data set contains the train set and the test set. It was confirmed by the operational experts that the train set was operating normally with no abnormal logs. It contains a total of more than 220,000 normal logs in a complete 3-day period. The test set is 1124 slices of log streams, each log stream contains 200 logs, where the log labels are unknown. That is, there may be anomalies or normal. Table 1 shows the example of the train set and test set.

The train set contains the log ID, time, process ID, log level, log components, content, etc. The test set contains only the log content. Therefore, it is not possible to use log information that is present in the training set but missing in the test set. By comparing the content data in the training set and the test set, we found that there is a large difference in content between the two. So, it limits the use of the train set. To reduce the number of correct logs in the test set, we eliminated the logs in test data which is the same as the logs in the training set. The test set lacks information and differs greatly from the training set. Here, the algorithm mainly processes and analyzes the test set. The template extraction of the body content part of the log using the Drain algorithm, the extraction result contains the template ID, the template content, and the list of extracted relevant parameters, the specific results are shown in Table 2.

Table 2 – Examples of templates drawn by the Drain

Content	Templa te ID	Template	Parameter List
Starting 160 workers	1	Starting <*> workers	['160']
(28848) wsgi starting up on http://0.0.0.0:9 191/	2	(<*>) wsgi starting up on http:// <*>/	['28848', '0.0.0.0:9191']
……	……	……	……

The input of an unsupervised model usually needs numerical representation, so the log data in text format needs to be transformed into a sequence. Template ID can be obtained after template extraction. The corresponding 1124 test sets can be represented by the template ID sequence. Some researchers treat it as time series. However, the time information cannot be directly processed as time-series data in this paper.

Through the above method, the log sequence matrix composed of template ID sequence can be obtained. The word frequency-inverse file frequency (TF-IDF) algorithm can be used to convert the log sequence matrix into a new conversion matrix to further evaluate the importance of a log statement to the whole log. Then, the standardized adjustment matrix distribution is used. The obtained matrix can be input into the unsupervised model to classify the anomaly of the log sequence in the test set.

3.2 Unsupervised ensemble learning

The goal of this part is to divide the test data into blocks and classify whether each log block is abnormal. Providing sufficient annotation data requires high capital and a large amount of time cost. Unsupervised log anomaly detection algorithms does not need much manual annotation data. Therefore, they are more economical and practical.

In this paper, four machine learning classification algorithms with different emphases in principle are used for modeling and classification tasks, including log clustering [6], Principal Component Analysis (PCA) [4], invariants mining [5], and isolation forest [3]. Four classifiers are used as meta-classifiers for ensemble learning. After adjusting the model super parameters, ensemble learning combines the prediction results output by the four models with the same weight to obtain whether the final prediction results are abnormal. The basic principles of the four classifiers are briefly introduced below:

- Log clustering is divided into two stages: knowledge base initialization stage and online training stage. Firstly, the knowledge base initialization stage usually includes three parts: log vectorization, log clustering, and log representative vector extraction. Log sequences are vectorized using algorithms such as TF-IDF and standardization. The vectorized log sequence is agglomerated and hierarchically clustered to generate multiple normal and abnormal classes. In the online training stage, the log sequence vectors are added to the knowledge base. If the Euclidean distance between it and the existing representative vector is less than the threshold, add the set and give the same label. Otherwise, create a new set and set a new label.
- PCA is a statistical method with dimensionality reduction. In log anomaly detection, the basic idea is to project high-dimensional data into a new coordinate system composed of k principal components, where k is less than the original dimension. Manually set the threshold of the principal component after dimensionality reduction. If it is greater than the threshold, it is judged as an anomaly, otherwise, it is judged as a normal log.
- The main idea of invariant mining is to mine and reasonably use the linear relationship between the processes of the program. When events do not occur in pairs, the linear relationship is violated. Firstly, the invariant space can be estimated by singular value decomposition. Secondly, the violence search algorithm is used to find the invariants. Finally, a threshold is defined to verify each candidate object. If the threshold is exceeded, it is determined as an anomaly.
- The idea of isolation forest is to find and delimit the isolated outliers with sparse distribution and far away from the group as anomalies. The algorithm believes that the smaller the distance from the root node to the leaf node, the easier it is to separate, and the higher the probability is the outlier of the anomaly. Anomaly detection includes training and testing stages. In the training stage, isolated trees are established, and isolation forests are formed. Different trees act as experts in different anomaly recognition. In the test phase, the anomaly score of the test set is calculated to detect the anomaly.

Finally, we vote on the prediction results of the four unsupervised models:

$$pre = \frac{1}{4}\sum_{i=1}^{4} k_i * s_i \tag{1}$$

Where k_i is the weight of each model, s_i is the prediction result of each model, and i is the number of the model.

Log anomaly detection is a binary classification task. To measure the accuracy and coverage of the binary classification model, we select Precision, Recall, and F1 as the evaluating indicator. F1 score is the harmonic average of Precision and Recall.

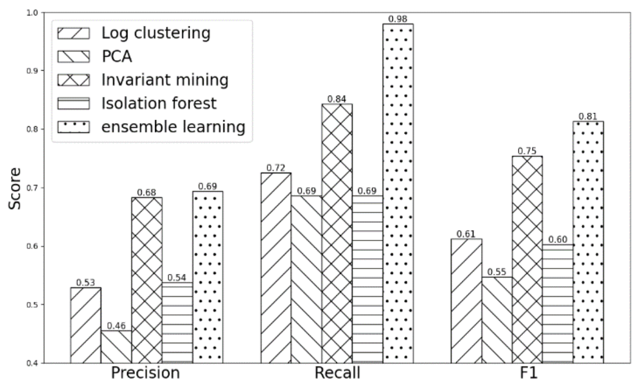

Fig. 3 – Precision, Recall, and F1 scores of different models

Fig. 3 shows the Precision, Recall, and F1 scores of different models in the same test set. Among the four unsupervised algorithms, the invariant mining algorithm has the best effect, and the F1 score can reach 0.75. After ensemble learning of the four results with the same weight of each result, the scores of ensemble learning are better than scores of a single algorithm, and the F1 score can reach the best score of 0.81. This is because the algorithm strategies of the four meta-classifiers are different. There are essential differences in principle. The meta-learner has diversity and accuracy. "Good but different" finally ensures that the ensemble result is better than a single classifier.

The log classification results can show whether there are anomalies in the log flow, but it is still impossible to know which line has an error. Viewing the complete log flow and determining the root cause is still time-consuming operation and maintenance work. Further automated root cause analysis and secondary optimization of classification results are needed.

4. LOG ROOT CAUSE ANALYSIS

Fig. 4 shows the flow chart of log root cause analysis. Start the log root cause analysis module if the prediction result is abnormal. Firstly, we need to manually define the common error keywords, and search the key root cause statements containing these error keywords through algorithms such as regular matching to form a key root cause database. Then, to generalize the root cause and expand the root cause database, similarity calculations based on edit distance are used to search similar root cause statements and improve the key root cause database. Finally, the module is scalable. The root cause database can be verified through expert experience injection, and more appropriate root cause analysis logic can be customized to ensure the final effect.

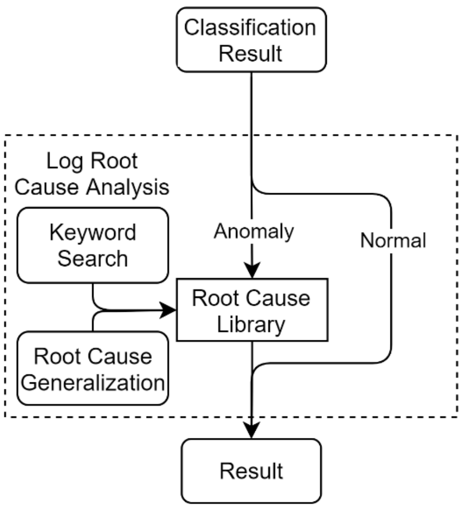

Fig. 4 – The flow chart of log root cause analysis

4.1 Keyword search

Although the strategy of searching root causes based on keywords is relatively simple, it is still an important way of judgment and cannot be replaced. Because it is the most direct method to identify faults and has been widely used and accumulated in the past few decades. The steps of the keyword search are as follows: All the templates extracted from the logs form a complete template library. Finding the templates that contain keywords and composing the root cause library. Combining the experience in the past, setting specific and important keywords can have a higher root cause positioning accuracy, such as "Errno", "Fail", "Exp", etc. Last, if the template extracted from the online log is in the root cause library, it can be directly judged as an anomaly. Table 3 lists several common errors that include keywords.

Table 3 – Examples of log templates containing keywords

Keywords	Log Template
Errno	AMQP server on controller:<*> is unreachable: [Errno 104] Connection reset by peer. Trying again in <*> seconds.
	AMQP server on controller: <*> is unreachable: [Errno 111] ECONNREFUSED. Trying again in <*> seconds.
Failed	Failed to compute_task_migrate_server: No valid host was found. There are not enough hosts available.
	Failed to bind port <*> on host <*> for vnic_type normal using segments
Exp	HTTP exp thrown: Instance <*> could not be found.
	DBAPIevent exp wrapped from (pymysql.err.Internalevent) (1927, u'Connection was killed') [SQL: u'SELECT 1']

4.2 Root cause generalization

Keywords are usually defined by business experts, but it is unrealistic to exhaust all keywords. In the scenario where keywords are not fully defined, how to automatically expand the root cause library is an important issue. We use root cause generalization to search for templates that are similar to templates which already exist in the root cause library and add them to the root cause library. Setting appropriate thresholds can ensure the accuracy of similar root causes.

Here we use the Levenshtein Distance: For two strings, the minimum number of edits required to convert from the first to the second is called the edit distance. The conversion method includes character replacement, insertion, and deletion. The smaller the edit distance, the greater the similarity between the two strings.

To be more intuitive, Table 4 gives an example of root cause generalization. It shows one of the reset connection errors of the AMQP service. The output text contains the keyword "[Errno 104]". Through root cause generalization, two new error templates that belong to AMQP services can be obtained called timeout error and heartbeat missing error. The keyword "Errno" does not appear in these two errors, so it is difficult to filter directly into the root cause library if root cause generalization is not used.

Table 4 – Examples of root cause generalization in AMQP service

Root Cause Template	
AMQP server on controller:<*> is unreachable: **[Errno 104] Connection reset by peer.** Trying again in <*> seconds.	
Similar templates	**Similarity**
AMQP server on controller:<*> is unreachable: **timed out.** Trying again in <*> seconds.	0.81
AMQP server on controller:<*> is unreachable: **Too many heartbeats missed.** Trying again in <*> seconds.	0.79

In this example, we can sort out all the error categories of AMQP services manually, but this method is inefficient and costly. Extending to other service categories, there are the same root causes that cannot be exhausted and are difficult to directly match. The root cause generalization algorithm can be used to efficiently search for similar root cause templates, enrich the root cause library, and improve the efficiency and accuracy of positioning. In addition, the second root cause generalization can be used to further expand the root cause library.

Of course, this method will introduce non-root cause templates as the number of times increases, and unreasonable thresholds. Therefore, business experts are required to review the templates newly added to the root cause library to ensure the rationality and accuracy of the root cause library.

4.3 Evaluation of results

In addition to the accuracy of log anomaly detection, the accuracy of root cause analysis and the time-consuming algorithm need to be considered. The root cause analysis timeliness score is used to evaluate the accuracy and timeliness of root cause analysis:

$$InTimeScore = Precision * \frac{InTimeCnt}{TP} \qquad (2)$$

Where InTimeCnt is the number of logs that identify the location of the anomaly in time, and TP is the number of anomaly sequences accurately identified. FinalScore can be divided into the following two parts:

$$FinalScore = F1 * 0.8 + InTimeScore * 0.2 \quad (3)$$

In terms of time performance, the inference time score is used to evaluate the efficiency of the algorithm:

$$PerformanceScore = \frac{\sum_1^N SinglePerformanceScore_N}{N} \quad (4)$$

Where SinglePerformanceScore is the time score of single inference. PerformanceScore is the average score. The total number of inferences is N.

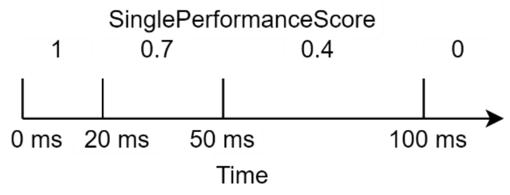

Fig. 5 – The SinglePerformanceScores in different time intervals

The model training time is required to be less than 20 minutes, and the single inference time is less than 100 ms. The SinglePerformanceScores in different time intervals are shown in Fig. 5.

Table 5 – Comparison of scores in different methods

Algorithm	Log Anomaly Detection	Log Root Cause Analysis	Deep Log
Precision	0.694	0.943	0.047
Recall	0.980	0.980	1
F1	0.813	0.962	0.090
InTimeScore	–	0.849	0.043
FinalScore	–	0.939	0.081
PerformanceScore	–	1	1

Table 5 shows the scores in different methods. The following conclusions can be drawn from the table:

- Log root cause analysis can optimize anomaly detection results. The abnormal result obtained by log anomaly detection is not necessarily accurate. If the location of the root cause is obtained from the log root cause analysis, the root cause line will be directly output. Otherwise, it can be re-determined as normal. Log root cause analysis has greatly improved the accuracy and F1 score.
- The conventional anomaly detection algorithm is not universal. The DeepLog scheme using LSTM has very poor results on this data set. This is because the types of abnormal logs in actual application scenarios are complex. DeepLog judges the sequence that has not occurred before as the anomaly and over-judgments the abnormal log. The public data sets (e.g., HDFS) usually have only dozens of types after data cleaning. There are no more than ten types of abnormal logs. Hundreds of logs are difficult to distinguish, and the existing traditional algorithms are powerless.
- The scheme is accurate and feasible. The first national AI innovation and application competition [15] provides the data set used in this paper. The F1 score in log anomaly detection is 0.962, and the accuracy in root cause location of anomaly is 0.849. In the ITU AI/ML in 5G Challenge 2021, the solution got the highest final score 93.904 in the China Mobile problem statement of network cloud equipment anomaly and root cause analysis.
- The main differences between the second-ranked scheme and our scheme in the competition are as follows: It uses a pre-trained sentiment classification model to classify the sentiment of the logs one by one, and the logs classified as negative sentiment are considered abnormal. Due to the lack of labels, the sentiment classification model was not optimized using log data. We consider the effect of the model to be instability. In comparison, our strategy is more cautious. Considering many unknown new logs in the test set (never in the train set), we only rely on keywords to judge logs that explicitly contain errors and their similar logs as anomalies.

5. CONCLUSION

Network cloud equipment anomaly and root cause analysis are the basic AI functions for network digital transformation and the construction of autonomous driving networks. ITU has specified some network intelligence use cases of log anomaly detection. This paper proposes a solution for network cloud equipment anomaly detection and root cause analysis, which solves the problems of complex abnormal log types, inaccurate time information, and missing or invalid information in actual business scenarios. This solution does not require large amounts of manual log annotation and can construct an efficient anomaly detection strategy to process millions of logs in a distributed system with complex components. Unsupervised model ensemble learning is used to achieve anomaly detection and classification tasks. A root cause library matching search is used to achieve root cause analysis. The scheme uses log data generated by network cloud equipment under a distributed network system, ensuring that the anomaly detection and root cause analysis models generated by this training have good practicability. This solution greatly reduces operation and maintenance costs, improves efficiency and system stability.

In addition to the scenarios of network cloud equipment, this solution can also be applied to more scenarios, such as single-source and multi-source log root cause positioning of the call chain, anomaly detection of system-level logs and application logs, and multi-dimensional data fusion root cause location-based on alarms, indicators, and logs. The scheme has also been deployed on China Mobile's 5G network management system. This article gives an overall solution of how to perform anomaly detection based on log data. We still need to think about how to further improve the accuracy rate and meet the management and distribution of higher throughput and higher security in the future.

REFERENCES

[1] Prewett J E. “Analyzing cluster log files using logsurfer”. In: Proceedings of the 4th Annual Conference on Linux Clusters, Citeseer, 2003.

[2] Rouillard J P. “Real-time Log File Analysis Using the Simple Event Correlator (SEC)”. LISA, 2004, 4: pp. 133-150.

[3] Liu F T, Ting K M, and Zhou Z H. “Isolation forest”. In: 2008 eighth IEEE international conference on data mining, IEEE, 2008, pp. 413-422.

[4] W. Xu, L. Huang, A. Fox, D. Patterson, and M.I. Jordon. “Detecting large-scale system problems by mining console logs”. In: SOSP’09: Proc. of the ACM Symposium on Operating Systems Principles, 2009, pp. 117-132.

[5] Lou J G, Fu Q, Yang S, Y, Xu, Y., and Li, J. “Mining Invariants from Console Logs for System Problem Detection”. In: USENIX Annual Technical Conference, 2010, pp. 1-14.

[6] Q. Lin, H. Zhang, J. -G. Lou, Y. Zhang and X. Chen, “Log Clustering Based Problem Identification for Online Service Systems”. In: 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C), 2016, pp. 102-111.

[7] Bodik P., Goldszmidt M., Fox A., Woodard D. B., and Andersen H. “Fingerprinting the datacenter: automated classification of performance crises”. In: Proceedings of the 5th European conference on Computer systems, 2010, pp. 111-124.

[8] Liang Y, Zhang Y, Xiong H, and Sahoo R. “Failure prediction in ibm bluegene/l event logs”. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), IEEE, 2007, pp. 583-588.

[9] Du M, Li F, Zheng G, and Srikumar V. “Deeplog: Anomaly detection and diagnosis from system logs through deep learning”. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285-1298.

[10] Meng W, Liu Y, Zhu Y, et al. “LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs”. IJCAI, 2019, Vol.19, No. 7, pp. 4739-4745.

[11] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, “Drain: An Online Log Parsing Approach with Fixed Depth Tree”. In: 2017 IEEE International Conference on Web Services (ICWS), 2017, pp. 33-40.

[12] M. Du and F. Li, “Spell: Streaming Parsing of System Event Logs”. In: 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016, pp. 859-864.

[13] Makanju, A. A., Zincir-Heywood, A. N., and Milios, E. E. “Clustering event logs using iterative partitioning”. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '09), 2009, pp. 1255-1264.

[14] Sefraoui O, Aissaoui M, and Eleuldj M. “OpenStack: toward an open-source solution for cloud computing”. In: International Journal of Computer Applications, 2012, 55(3): pp. 38-42.

[15] Anomaly and root cause analysis of network cloud equipment. Available: http://36.133.53.121:1080/index/content_page?pageType=10#introduce

AUTHORS

Dandan Zou M.S. in optical engineering, School of Optical and Electronic Information, Huazhong University of Science and Technology, Wuhan, China. He is a data scientist of Telco AI Lab, AsiaInfo Technologies, with research interests including machine learning, data mining, root cause analysis, etc.

Jianbing Ding M.S. in software engineering, School of Computer Science, Wuhan University, Wuhan, China. He is a data scientist of Telco AI Lab, AsiaInfo Technologies, with research interests including machine learning, data mining, causal analysis, etc.

Xidong Wang M.S. in communication and information systems, School of Beijing University of Posts and Telecommunications. Project Manager at Telco AI Lab, AsiaInfo Technologies. His research interests include wireless techniques, 5G communications, machine learning and its applications.

Xiaozhou Ye Ph. D., Director of Telco AI Lab, Asiainfo Technology (China) Co., LTD. His research interest is communication artificial intelligence and future network technology.

Ye Ouyang Ph.D., CTO & Senior Vice President of AsiaInfo Technologies. Dr. Ouyang has distinguished experience in R&D and management in the telecommunications industry. Prior to AsiaInfo, Dr. Ouyang has been a Verizon Fellow and a senior manager in Verizon. His research is in the interdisciplinary area of wireless communications, data science, and AI. Dr. Ouyang authored more than 30 academic papers, 40 patents, 10 international standards, and 8 books.