2020 International Conference on Artificial Intelligence and Computer Engineering (ICAICE) | 978-1-7281-9146-1/20/$31.00 ©2020 IEEE | DOI: 10.1109/ICAICE51518.2020.00073

LogSpy: System Log Anomaly Detection for Distributed Systems

Haoming Li
School of Computer Science
Beijing University of Posts and Telecommunications
Beijing, China
lhm@bupt.edu.cn

Yuguo Li
International School
Beijing University of Posts and Telecommunication
Beijing, China
2018213162@bupt.edu.cn

***Abstract*—Log analysis is an important part of distributed system management. System log records the running status of the system and contains a lot of important and valuable information. This paper proposes an anomaly detection method, LogSpy, for distributed systems. It uses the combination of natural language processing technology and clustering algorithm for log template mining and feature extraction. In anomaly detection, it is found that there are a large number of remote calls in the distributed systems and traditional CNN has certain limitations on this small amount of negative sample data. LogSpy introduces the attention mechanism in detection algorithm and optimizes the detection window and computational complexity. Experiments conducted on the OpenStack test platform show that LogSpy can perform excellent anomaly detection on distributed systems compared to traditional anomaly detection methods.**

Keywords-AIOps; anomaly detection; attention mechanism; CNN; distributed systems

I. INTRODUCTION

Distributed systems usually run on thousands of computing routines and consist of numerous operating components. The systems have powerful functions and good scalability, which bring great convenience to business and enterprise services. However, they bring great challenges to operation and maintenance personnel at the same time.

In the past, when a single system failed, the operation and maintenance personnel could troubleshoot by viewing the log information, relying on their own professional technology, sufficient experience and understanding of the source code. But when the era of distributed systems comes, the huge scale, numerous components, and complex logic make it difficult for operation and maintenance personnel to find problems in a short time [1]. No matter they detect anomalies of logs by manually searching for keywords such as error and critical or writing rules or any other method else, they can't find the problem quickly and comprehensively. However, if each component and each module is monitored, the overall cost of the system will be greatly affected, and the performance will be greatly reduced. Also, the monitoring will need to be updated as if the components are upgraded, which makes the system more complex.

Therefore, there is an urgent need for a more convenient and intelligent anomaly detection method.

II. RELATED WORK

A large amount of log data is generated during the operation of a distributed system, which reflects the whole situation of the system, including information such as system operation status, activities, and interface calls. The log is mainly composed of various freely combined texts. The logs of different systems will have different formats and contents, and even the logs of different components in the same system will not be the same. This diversification of semi-structured data allows log-based Analysis becomes extremely challenging.

Over the years, researchers in various countries have used methods in different fields in log anomaly detection, and have achieved many research results [2]. Fu et al. [3] used Finite State Automation (FSA) to detect abnormalities by evaluating the order of existing logs. XU et al. [4] used Principal Component Analysis (PCA) and AST Abstract Syntax Tree (AST) methods to process the parsed log feature set, which reduced the complexity of the feature set to be analyzed and obtained effective the results of anomaly detection. Yu et al. [5] proposed an anomaly detection system, CloudSeer, which obtains execution exceptions by checking error information in the interleaved log sequence. Lin et al. [6] used the similarity between logs to cluster system logs. Yen et al. [7] used unsupervised clustering of data characteristics to identify potential security threats to logs and manually labeled the data. Lu er al. [8] used Convolutional Neural Network (CNN) to detect anomalies in large-scale system logs and extracted the internal correlations between logs based on convolution kernels of different sizes. Kimura et al. [9] extract typical features from log sequences based on expert experience and use traditional machine learning methods to learn log abnormal patterns. So that the purpose of detecting abnormal logs is achieved. At the same time, optimizing the amount of anomaly detection calculation is also an important direction. For example, Priolog [10] compresses logs to reduce interference and reduces the amount of calculation, while other algorithms strengthen important logs for anomaly detection [11].

In recent years, with the further development of distributed systems, more anomaly detection has focused on the mutual invocation of logs. Judge whether there is an abnormality through the call changes between each node. However, the problem of the anomaly detection is that it does not consider the long-distance call in the global situation, which is particularly prone to false positives and false negatives. At the same time,

978-1-7281-9146-1/20/$31.00 ©2020 IEEE
DOI 10.1109/ICAICE51518.2020.00073

since deep learning models based on CNN often require a lot of training data, the method of anomaly detection requires a lot of abnormal annotation data. For log data sets with only a few annotations, the accuracy of the method will be significantly reduced. Therefore, this paper proposes a log anomaly detection method, LogSpy.

III. LogSpy Framwork

The distributed system architecture is complex because hundreds of programs call each other and the components which seemingly unrelated often have hidden connections. In view of this situation, LosSpy pays attention to the call chain frequency, success rate, response time and other parameters of logs. Considering that distributed system has a large amount of nodes and data sparsity, it extracts features based on CNN and pulls in attention mechanism to give more attention to important anomalies.

A. Template Extraction

Distributed systems will generate plenty of complex log structures, while are generated through splicing, placeholders and dynamic keywords. Before analyzing this kind of unstructured text, it needs to be parsed into structured data. System log data is a kind of unstructured text data which can be obtained directly from log files. Log data needs to be parsed into structured data before being analyzed and each log instance is composed of fixed parameters and dynamical variables. A log template usually refers to a fixed part of a log message which can summarize the meaning expressed by the log and similar log messages can be represented by the same template. For example, a log in the OpenStack cluster is: "Create instance tempest-INSTANCE_SAMPLE-1530127935 and boot it.", which can be regarded as composed of the constant "Create instance * and boot it" and the variable "tempest-INSTANCE_SAMPLE-1530127935". “Create instance * and boot it" can be expressed as a template for all similar log messages.

There have been many methods for template extraction so far and large parts of them have high extraction accuracy [13-15]. LogSpy uses FT-Tree [16] for template extraction. FT-tree is an extended prefix tree structure to represent the system log message template. The subtype of the detailed information field in the system log message is usually the longest combination of frequently occurring words. FT-tree uses this idea to identify the longest combination of frequently occurring words from the system log message.

B. Template Mining

After the template extraction is completed, a feature vector for anomaly detection needs to be constructed based on the words in the template, which means converting the log into parameter data, clustering the template vector and then representing all the vectors in this cluster by using the cluster center vector.

LogSpy uses the skip-gram model of the Word2Vec algorithm for aggregation and matching.The Skip-gram model is based on the Bayesian formula. Knowing the current word w_i, it predicts the words in the context C(w_i). The objective function is

$$p(C(w_i)|w_i) = \prod_j p(w_j|w_i) \tag{1}$$

$p(w_j|w_i)$ means the probability that w_j appears in the context of w_i.

After completing the conversion of word vectors, LogSpy uses the clustering algorithm AGNES for the word vectors to distinguish various logs and form different log formats. We define the word vector as $w_i(f_{1i}, f_{2i}, f_{3i}, \dots, f_{\lambda i})$, then the Euclidean distance formula between word embedding w_i and w_j is

$$d_{ij} = \sqrt{\sum_{k=1}^{\lambda} (f_{ik} - f_{jk})^2} \tag{2}$$

We cluster all word vectors through Euclidean distance. Considering that the total number of keywords in English is not large, we selects the AGNES hierarchical clustering algorithm, also known as agglomeration hierarchical clustering. There are three ways to measure the distance between class clusters, Single chain, Average chain. Considering that the single chain and the full chain are susceptible to noise interference and the log is very complex as semi-structured data, this paper chooses the average distance between class clusters as the criterion for class cluster merging. We define the average distance between class clusters as

$$d_{avg}(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{q_j \in C_1} \sum_{q_j \in C_2} d_{jk} \tag{3}$$

C. Anomaly Detection

Firstly, We analyze the call relationship between the log templates generated by each service component in the distributed system [17] and pay attention to the call success rate, call frequency, delay time and so on. Then feature pictures which can enter the convolutional neural network are formed

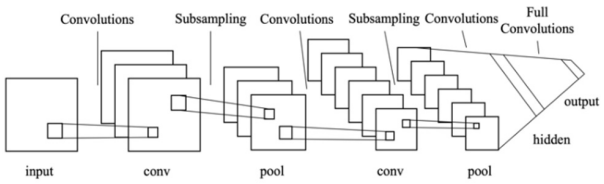

Figure 1. CNN architecture

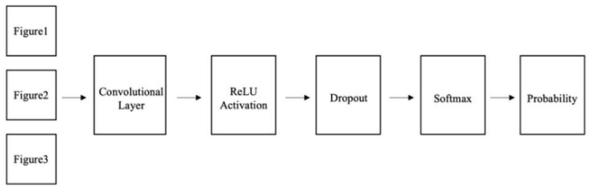

Figure 2. Network architecture in LogSpy

based on the call relationships [18]. The specific architecture is shown in the Fig. 1.

With the development of neural networks in recent years, image recognition based on CNN convolutional neural networks can better identify anomalies. The first is the feature extraction layer: the input of each neuron is connected to the local receptive field of the previous layer and extract the local features. Once the local feature is extracted, its positional relationship with other features is also determined; the second is the feature mapping layer: each computing layer of the network is composed of multiple feature maps. Each feature map is a plane and the weights of all neurons above are equal.

The CNN network structure used by this algorithm is shown in the Fig. 2.The input data is the characteristics of template call, including call frequency, call delay and call success rate. After the three pictures enter the convolutional network, each output is connected to Dropout for generalization and use Softmax to classify and discriminate at the end of the network structure. In this way a complete end-to-end structure is realized.

The original CNN network can extract state information in a certain degree. However, unlike the Traditional cognitive of local image perception, the relationship between two data that have a distance is also important in anomaly detection. For example, in an ordinary Remote Procedure Call (RPC) access, the thread A accesses the interface B and lets it query a certain data m in the service C on its behalf. However, service C occasionally becomes unavailable which makes interface B cannot obtain data m so that an error is returned to thread A. Thread A calls interface D after receiving the message and asks it to query the backup data m' in service E on its behalf. Therefore, in this example, the visit number of $D \rightarrow E$ will increase. From above, attention should be paid to the call of $B \rightarrow C$. The two seemingly unrelated remote calls, $D \rightarrow E$ and $B \rightarrow C$, are actually inter-related. However, remote calls do not have a specific rule to connect them closely in ordinary CNN convolutional neural networks.

Considering the above problems, this paper introduces the attention mechanism MLP-ATT-CNN, which can optimize the weight of input data [19]. before the input layer of the ordinary CNN network, add a special coefficient matrix $W \in R^{K \cdot K}$. The coefficient matrix W comes from a special MLP network whose input layer has a total of $K \times K$ dimensions. Its input call frequency matrix is $In_{att} \in R^{K \cdot K}$ and its output layer is also a matrix of the same dimension $Out_{att} \in R^{K \cdot K}$. Then put $In_{convl} = figure_2 \cdot W$ into the CNN convolutional neural network, where $figure_2$ is the frequency matrix of the call. The specific CNN structure is shown in the Fig. 3.

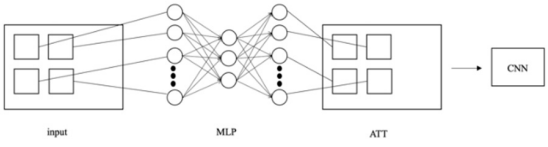

Figure 3. MLP-ATT-CNN stucture

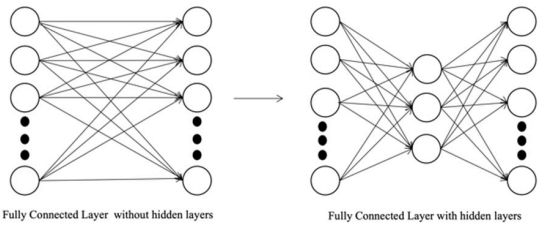

Figure 4. Fully connected layer

Considering the total number of weight W in the MLP network that need to be updated is $sum_w = K^2 \cdot K^2$, which is a large amount of calculation, $O(K^4)$. We optimizes the weights: Considering that the entire $K \times K$ call chain matrix space is relatively sparse and different call relationships can be coupled. In this paper, a hidden layer is added in the middle of MLP in Fig. 4. The node data of the hidden layer is a constant. Set the constant to m, then the total number of weights that need to be updated in the MLP network is $sum_w = K^2 \times m \times 2$. In this way, the number of weights and the number of call chain matrix spaces are at the same level, which greatly reduces the burden of updating the weight of the MLP network.

D. Optimize the window

In practical anomaly detection operations, anomaly detection systems often use the way of real-time online calculations. In order to extract the characteristics of log data, first of all, this paper needs to divide the log data into different windows, each window is used as the input log data sequence, and the division of windows occupies an important position in real-time calculation [20]. This article will discuss three window division methods, rolling window, sliding window and session window, and optimize the window algorithm according to the characteristics of the log.

1) Rolling window

Rolling windows are windows based on the time data of log. Each window has the same size and does not overlap with each other. Suppose the window size is Δ_t, the starting time is t_0, then the time window of the j-th group is:

$$S_j = \{t \mid t_0 + \Delta_t \cdot (j-1) \leq t < t_0 + \Delta_t \cdot j\} \quad (4)$$

2) Slide window

Sliding window is different from a rolling window for the front and back windows will overlap. It contains two attribute values: window size and step length. Suppose the window size is Δ_t, the starting time is t_0, the step size is s, then the time window of the j-th group is:

$$S_j = \{t \mid t_0 + s \cdot j \leq t < t_0 + s \cdot j + \Delta_t, s < \Delta_t\} \quad (5)$$

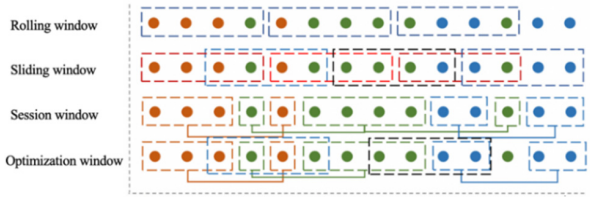

Figure 5. four different types of windows

3) Session window

Different from the above two types of windows, the session windows are not grouped by time data but the identifier of the session. For example, if the time interval between two pieces of logs is long, it can be separated or grouped by transaction ID or session ID.

This paper will discuss the advantages and disadvantages of these three types of windows. As shown in the Fig. 5, the rolling window groups the log data by time accurately and the calculation is simple as well. However, the shortcomings are also obvious: On the one hand, it roughly divides the log group through time, which may cause the log data belonging to the same transaction ID to be divided into two groups; on the other hand, in order to detect a complete abnormal pattern and pattern generalization, the window division is usually relatively large. Such as 30 minutes, which will cause the maximum delay in real-time detection to reach 30 minutes. It will greatly affect the real-time performance and separate some abnormal patterns, making some abnormalities escape the detection.

Based on the above analysis, the sliding window greatly reduces the shortcomings of the rolling window. Adjusting the step size can reduce both the delay of anomaly detection and the possibility of the abnormal pattern being split. However, the sliding window is still grouped by time. This way may separate the logs of the same transaction, which will lose part of the original rule characteristics based on the log call chain. Therefore, as shown in the Fig. 5, this paper will introduce some features of the session window. The logs of the same transaction that are divided at both ends of the timestamp can be put into the same sliding window by combining the time division.

Suppose the window size is Δ_t, the starting time is t_0, the step size is s, the mixing tolerance time length is r. The overall data is calculated by 3 steps: 1) The data in the range $(t_0 + s \cdot j + r) \le t < (t_0 + s \cdot j + \Delta_t - r)$ is accepted unconditionally and added to the Set; 2) Judge the data within the time range of $(t_0 + s \cdot j) \le t < (t_0 + s \cdot j + r)$ and $(t_0 + s \cdot j + \Delta_t - r) \le t < (t_0 + s \cdot j + \Delta_t)$. If any data of the transaction to which the log belongs does not appear neither in the Set nor in the time range $(t_0 + s \cdot j - r) \le t < (t_0 + s \cdot j)$and $(t_0 + s \cdot j + \Delta_t) \le t < (t_0 + s \cdot j + \Delta_t + r)$, then discard the data and add the rest of the data to the Set; 3) After the first two steps are executed, Judge the data in the time range $(t_0 + s \cdot j - r) \le t < (t_0 + s \cdot j)$ or $(t_0 + s \cdot j + \Delta_t) \le t < (t_0 + s \cdot j + \Delta_t + r)$, if any data in the transaction to which the log belongs appears in the Set, it is added to the set.

So far, the log data grouping of an optimized window is completed.

IV. Experiment and Analysis

A. Introduction to the data set

The experimental data comes from log data generated by the OpenStack cluster built with four host computers. The log data source components include textual data generated by Nova, Cinder, Swift, Neutron, Glance, and Keystone. All data is based on normal virtual machine creation, restart, and deletion, block storage creation, mounting, network connection, transmission, image generation, start up, deletion, and object storage creation, upload, download and so on.

B. Evaluation criteria

Anomaly detection refers to discovering some characteristics different from normal operation in log data. Normal operation means that the distributed system can provide users with correct service functions, including service functions can be used, and various services can be obtained in time. And abnormality refers to the influence of various functions and services provided externally due to incorrect operation activities. The probability of this anomaly occurring is generally smaller than normal operation. Anomaly detection in this paper refers to the discovery of abnormal behavior of the system through the log data of the system. Given a set of system log data and the corresponding data of the system state, we need a method to judge which state the log data output from the system belongs to, so as to determine whether the system is abnormal.

Let D be a set of log messages, $D = \{X_1, X_2, \dots, X_i, \dots, X_n\}$, where X_i is i-th log message, $i = 1,2,3, \dots, n$. Log message X_i consists of a sequence of words, such as $X_i = W_{i1}W_{i2} \dots W_{in}$。

Suppose S is the actual state of the distributed system and the problem input is D. This paper defines algorithm f, that is, the expected system state calculated by the algorithm is $S' = f(D)$. The algorithm f should meet the minimum :

$$Diff = \sum (S - S') \tag{6}$$

1) Log parsing

Let $X_1, X_2, \dots, X_m$ be the log data, $X_1, X_2, \dots, X_n$ be the log templates, and $P_{t1}, P_{t2}, \dots, P_{tn}$ are the parsing methods corresponding to the log templates. The parsing method P_t should convert the log to the successful parsing state (*y*), timestamp (*d*), printing type (*c*), parsing template number (*p*), state (*s*), transaction number (*t*):

$$y, d, c, p, s, t = P_t(X) \tag{7}$$

Log parsing result using parsing methods is:

$$y = \begin{cases} 0, & true \\ 1, & false \end{cases} \tag{8}$$

Where true represents that the X is parsed successfully, in contrary, false represents that the X is parsed unsuccessfully. When parsing logs, the parsing success rate is:

$$\sigma = \frac{\sum y}{sum(X)} \tag{9}$$

where $sum(X)$ is the total number of logs.

The evaluation index of this experiment is the log matching accuracy rate. The specific judgment rule is whether the algorithm can be successfully matched and parsed after the algorithm matches the corresponding rule. Then verify the parsed parameters. The parsing is successful only if the parsing is successful and the parsing parameters meet the format rules.

2) Anomaly detection

In the evaluation criteria, anomaly detection can be transformed into a two-category problem. The commonly used

evaluation indicators are $Precision$、$Recall$ and $F_1\ Score$. The calculation formula is:

$$Precision = \frac{TP}{TP + FP} \tag{10}$$

$$Recall = \frac{TP}{TP + FN} \tag{11}$$

$$F_1\ Score = \frac{2 \times Recall \times Precision}{Recall + Precision} \tag{12}$$

where TP (true positive) represents the number of anomaly log sequences that are successfully detected, FP (false positive) represents the number of normal log sequences judged as abnormal by the anomaly detection model, FN (false negative) represents the number of abnormal log sequences judged by the anomaly detection model.

C. Experimental content

1) Log parsing

In this experiment, we set the keywords with the word frequency in the top 50%. Perform 3 iterations of keyword selection. If the position changes more than 2 times, the keyword will be excluded. The dimension of the generated word vector is 16. The control group is set to 01 word bag method and text edit distance matching method. If the text edit distance is greater than 3, it is regarded as a different log template.

From the results in Table I, the algorithm proposed in this paper is better than 01 bag of words both in terms of the number of successful analysis and accuracy. Compared with the text edit distance, the number of successful analysis is lower than the control group, but its success rate is higher than that of control group after parameter verification, which verifies the effectiveness of our algorithm.

2) Anomaly detection

Input and output: For input data, input three 71×71 data graphs, including the success rate of the call, the number of calls, and the call time; For the output data, output a Tensor of $Node \times 2$ dimensions.

From the perspective of traditional machine learning algorithms and deep learning algorithms, we compare GBDT, Node2Vec and CNN algorithms.

TABLE I. RESULT OF LOG

	Number of logs	Number of successful parse	Number of correct parameter verification	Accuracy
01 bag of words	5,795,083	4,823,834	4,514,369	77.9%
Text edit distance		5,235,228	4,798,328	82.8%
LogSpy		5,183,113	5,093,877	87.9%

TABLE II. RESULT OF ANOMALY DETECTION

	F1	Precision	Recall
GBDT	0.91514	0.94162	0.89011
Node2Vec	0.48566	0.47102	0.50123
CNN	0.95286	0.94721	0.95857

TABLE III. RESULT OF ANOMALY DETECTION

	F1	Precision	Recall
CNN	0.95286	0.94721	0.95857
FC-ATT-CNN	0.95689	0.95182	0.96201
MLP-ATT-CNN	0.96336	0.95947	0.96729

Network structure: The experimental network is 4 convolutional layer, with 4 pooling layer placed in the middle and at the end. The activation function is ReLU. The loss function is evaluated using Softmax cross entropy and the optimizer is AdamOptimizer.

In Table II, although the GBDT algorithm inputs a large number of dimensions, it still performs well. However, the performance of the data transformed by the flowchart through Node2Vec [21] is not good, Considering that it cannot distinguish different template feature maps. The experiment proposes that the CNN algorithm is better than the control algorithm to verify the effectiveness of this algorithm.

In order to verify that the CNN convolutional neural network based on the attention mechanism is better than the ordinary CNN network, this paper adds an experiment of the MLP-ATT-CNN algorithm for extracting data relations from a longer distance on the basis of the ordinary CNN network algorithm. At the same time, compare the fully connected neural network without hidden layers as the weight trainer of the MLP-ATT-CNN algorithm and the MLP network containing hidden layers as the weight trainer of the MLP-ATT-CNN algorithm and evaluate the result and performance.

Network structure: The experimental CNN network is 4 layer convolutional layer, with 4 pooling layers in the middle and last, and the activation function is ReLU. The fully connected part of the fully connected network model FC- ATT-CNN uses two layers: one input layer node 71×71, the other one output layer node 71×71; And input layer node 71×71 in the MLP part of the MLP-ATT-CNN. The hidden layer node is 8 and the output layer has a dimension of 71*71. The loss function is evaluated by Softmax cross entropy and the optimizer is AdamOptimizer.

From the Table III, the performance of MLP-ATT-CNN is better than that of ordinary CNN network and FC-ATT-CNN algorithm, which verifies the effectiveness of our algorithm.

V. CONCLUSION

System log contains a wealth of system operation information, which provides important support for detecting abnormalities in modern large-scale systems. This paper proposes a method, Logspy, to detect system abnormalities from distributed system system log. Logspy first uses FT-Tree to process unstructured log data and completes template extraction. Then the method of natural language processing is used for reference and the clustering method is combined on the basis of the skip-gram model of the Word2Vec algorithm. So that the log template mining is performed without understanding the source code and the log template relationship characteristics can be extracted. In view of the unique characteristics of the distributed system, the MLP-ATT-CNN algorithm based on the MLP network to extract the long-distance relationship is introduced in

the traditional CNN method. And then the detection window and computational complexity are optimized. Experiments on OpenStack clusters have verified the effectiveness of the method in this paper.

REFERENCES

[1] Dang Y, Lin Q, and Huang P, "Real-world challenges and research innovations," 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion). IEEE, 2019, pp. 4-5.

[2] He S, Zhu J, He P, et al. "Experience report: system log analysis for anomaly detection," 2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE). IEEE, 2016, pp. 207-218.

[3] Fu Q, Lou J G, Wang Y, et al. "Execution anomaly detection in distributed systems through unstructured log analysis," 2009 ninth IEEE international conference on data mining. IEEE, 2009, pp. 149-158.

[4] XU W, HUANG L, FOX A, et al. "Detecting large-scale system problems by mining console logs," ACM SIGOPS 22nd symposium on Operating systems principles, 2009, pp. 117-132.

[5] YU X, JOSHI P, XU J, et al. CloudSeer, "Workflow monitoring of cloud infrastructures via interleaved logs," ACM Sigarch Computer Architecture News, 2016, 44(2), pp.489-502.

[6] Lin Q, Zhang H, Lou J G, et al. "Log clustering-based problem identification for online service systems," 2016 IEEE/ACM 38th International Conference on Software Engineering Companion (ICSE-C). IEEE, 2016, pp. 102-111.

[7] Yen T F, Oprea A, Onarlioglu K, et al. "Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks," Proceedings of the 29th Annual Computer Security Applications Conference, 2013, pp. 199-208.

[8] Lu S, Wei X, Li Y, et al. "Detecting anomaly in big data system logs using convolutional neural network," 2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/ PiCom/ DataCom/ CyberSciTech). IEEE, 2018, pp. 151-158.

[9] Kimura T, Watanabe A, Toyono T, et al. "Proactive failure detection learning generation patterns of large-scale network logs," 2015 11th International Conference on Network and Service Management (CNSM). IEEE, 2015.

[10] Tak B, Park S, Kudva P. Priolog, "Mining important logs via temporal analysis and prioritization," Sustainability, 2019, 11(22): 6306.

[11] Wang X, Wang D, Zhang Y, et al. "Unsupervised learning for log data analysis based on behavior and attribute features," Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science. 2019: 510-518.

[12] Jia T, Li Y, Zhang C, et al. "Machine deserves better logging: a log enhancement approach for automatic fault diagnosis," 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW). IEEE, 2018, pp. 106-111.

[13] Du M, Li F. Spell, "Streaming parsing of system event logs," 2016 IEEE 16th International Conference on Data Mining (ICDM). IEEE, 2016, pp. 859-864.

[14] He P, Zhu J, Zheng Z, et al. "Drain: An online log parsing approach with fixed depth tree," 2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017, pp. 33-40.

[15] Hamooni H, Debnath B, Xu J, et al. "Logmine: Fast pattern recognition for log analytics," Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. 2016, pp. 1573-1582.

[16] Zhang S, Meng W, Bu J, et al. "Syslog processing for switch failure diagnosis and prediction in datacenter networks," 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). ACM, 2017.

[17] Nandi A, Mandal A, Atreja S, et al. "Anomaly detection using program control flow graph mining from execution logs," Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2016, pp. 215-224.

[18] Jia T, Chen P, Yang L, et al. "An approach for anomaly diagnosis based on hybrid graph model with logs for distributed services," 2017 IEEE International Conference on Web Services (ICWS). IEEE, 2017, pp. 25-32.

[19] Xiao J, Ye H, He X, et al. "Attentional factorization machines: Learning the weight of feature interactions via attention networks," arXiv preprint arXiv:1708.04617, 2017.

[20] Carbone P, Katsifodimos A, Ewen S, et al. "Apache flink: Stream and batch processing in a single engine," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2015, 36(4).

[21] Grover A, Leskovec J, "Node2vec: Scalable feature learning for networks," Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining, 2016, pp. 855-864.