Hindawi
Scientific Programming
Volume 2021, Article ID 6617882, 8 pages
https://doi.org/10.1155/2021/6617882

Research Article

A System Fault Diagnosis Method with a Reclustering Algorithm

Zhe Yang,[1] Shi Ying,[1] Bingming Wang,[1] Yiyao Li,[2] Bo Dong,[1] Jiangyi Geng,[1] and Ting Zhang[1]

[1]*School of Computer Science, Wuhan University, Wuhan, China*
[2]*School of Software Engineering, Tongji University, Shanghai, China*

Correspondence should be addressed to Shi Ying; yingshi@whu.edu.cn

Received 19 December 2020; Revised 28 January 2021; Accepted 27 February 2021; Published 9 March 2021

Academic Editor: Pengwei Wang

The log analysis-based system fault diagnosis method can help engineers analyze the fault events generated by the system. The *K*-means algorithm can perform log analysis well and does not require a lot of prior knowledge, but the *K*-means-based system fault diagnosis method needs to be improved in both efficiency and accuracy. To solve this problem, we propose a system fault diagnosis method based on a reclustering algorithm. First, we propose a log vectorization method based on the PV-DM language model to obtain low-dimensional log vectors which can provide effective data support for the subsequent fault diagnosis; then, we improve the *K*-means algorithm and make the effect of *K*-means algorithm based log clustering; finally, we propose a reclustering method based on keywords' extraction to improve the accuracy of fault diagnosis. We use system log data generated by two supercomputers to verify our method. The experimental results show that compared with the traditional *K*-means method, our method can improve the accuracy of fault diagnosis while ensuring the efficiency of fault diagnosis.

1. Introduction

It is inevitable that the system fails during operation [1], and it is necessary to quickly detect and diagnose system failures. Since the system log records relevant information about hardware, software, and system problems during system operation [2], the system fault diagnosis method based on log analysis is a common and effective method.

When the existing system fault diagnosis methods vectorize system logs, most of them dig out the text features between logs (such as text similarity), mainly including the following: Lin et al. [3] use the TF-IDF method to vectorize logs and use the agglomerative hierarchical clustering method for system anomaly detection; Fu et al. [4] use log types and the finite state machine algorithm to generate a workflow model for anomaly detection; LouJG and Yang [5] propose a parser based on the log variable mining method, and the variable path in the message constructs a linear log feature sequence for fault diagnosis; Xu et al. [6] combine source code analysis with the log information retrieval to build composite features of the log and use machine learning methods to analyze the features and perform fault diagnosis. Shang et al. [7] extract Hadoop logs and parse them into log sequences and compare the execution sequences of two different deployments for fault identification. Chen et al. [8] use the machine learning algorithm and data mining methods to generate decision trees for fault diagnosis by recording the runtime attributes of the Internet requests. Reidemeister et al. [9] establish a log model by mining frequent log sequences and use a decision tree classifier to learn and classify system failures. Yuan et al. [10] use statistical learning methods to classify system call sequences based on log data which records system behavior information and use fault injection to evaluate the method. The above studies are all fault diagnosis methods based on log text feature mining.

Since the log statements that did not produce a failure (that is, normal logs) usually have a high text similarity and the log statements that have a failure (abnormal logs) have a low text similarity between the log statements which have not produced a failure, we can identify and diagnose system faults by mining the text features or other features between

the abnormal logs and normal logs, and clustering these features [11]. However, the implementation of this method faces the following challenges:

(1) The challenge of obtaining low-dimensional log vectors: due to the complexity of the cluster system, there are usually a large number of parallel services in the system. If the traditional text vectorization methods are used, such as the TF-IDF method, to vectorize a large number of different types of logs generated by the service, the dimension of the log vector is equal to the number of word types that appear in the logs, which may cause feature explosion problems.

(2) The challenge of distinguishing log data with similar text but different semantics: there may be a certain textual similarity between abnormal logs and normal logs, that is, logs with similar text may have a large semantic gap. As shown in Table 1, log messages refer to the content description part of the log data (Section 2.1). Log message 1 and log message 2 have a strong text similarity, but log message 1 indicates that the system has no failure, and log message 2 indicates that the system has a failure. Traditional log text mining methods cannot identify these kinds of characteristics well.

(3) The challenge of low accuracy of fault diagnosis methods based on clustering algorithm: frequent logs with similar system failures (log statements after preprocessing are called logs) are not necessarily strongly correlated with the types of failures, that is, logs clustered into one type are not necessarily all of the same type of failure, or all even must not indicate failure events.

Aiming at the above problems, this paper proposes a system fault diagnosis method based on the clustering algorithm. Firstly, we use PV-DM language model to generate low-dimensional log vectors; then, we improve the traditional K-means algorithm and propose a reclustering method based on keyword extraction to deal with the previous clustering results to improve the accuracy of fault diagnosis. The main contributions of this method are as follows:

(1) Aiming at the high-dimensional characteristics of traditional log vectors, we use the PV-DM language model to generate low-dimensional log vectors to provide effective data support for fault diagnosis

(2) Aiming at the problem that the log clustering effect based on the traditional K-means algorithm is poor, we improve the K-means from two aspects of k-value selection and center point merging to improve the clustering effect of traditional K-means

(3) Aiming at the problem of low accuracy of log fault diagnosis based on K-means algorithm, we propose a reclustering method based on keyword extraction to improve the accuracy of system fault diagnosis with log analysis

Table 1: The example of similar logs with different semantics.

Log lines	Description of log messages
Log message1	Instruction cache parity error corrected
Log message2	Instruction cache parity error uncorrected

2. Our Method

In this paper, a fault diagnosis method based on reclustering algorithm is proposed, which mainly includes three parts: log vectorization, improvement of K-means, and fault diagnosis method based on reclustering.

The acquisition of log vectorization mainly describes how we process the original log data and identify the log event. After preprocessing, the log data is trained by the PV-DM language model to get the semantic feature expression vector of logs, which completes the log vectorization. Then, we improve the traditional K-means from the selection of the K value and the merging of center points and use the improved k-means algorithm to cluster the logs. Finally, we combine the clustering results with the TF-IDF method to extract the keywords of each cluster and recluster the log data according to the keywords. The log fault diagnosis is completed according to the results of the reclustering.

2.1. Log Vectorization

2.1.1. The Structure of Log Data. The system logs record the relevant information of hardware, software, and system problems during the system operation, such as network access request, file read-write operation, and service stop and start [12]. Although the log information is complex and changeable, it usually has the following parts:

(1) Time stamp: time stamp indicates when the log message was generated

(2) Event ID: event ID indicates the record sequence number of a service or operation event during system operation

(3) Location: location indicates the location or node where the log message was generated

(4) Content description: content description is the description of specific events recorded in the log

With the execution of the system, the time stamp, event ID, and other contents will change. We call the information in the log message that changes with the system execution process as the message variable, and the unchanged part of the log message is called the message constant or log event template. The purpose of preprocessing of the log data is to extract the log event template from the original logs, which mainly includes two parts:

(1) Replacement of Message Variables. Because our method does not need the variable parts in log data, the message variable in the original log message is replaced with wildcard. We use "[]" to replace variables (such as time stamp, memory address, and file path) in log messages.

(2) The Identification of the Log Event. After the part of the message variable in the log is replaced by the wildcard, the log messages describing the same log event will have strong similarity. Therefore, we extract the log event template and the log message line number in the log message to form the identification of the log event.

2.1.2. Log Vectorization Based on the PV-DM. The Distributed Memory model of Paragraph Vectors (PV-DM) model is a neural network language model for training paragraph vectors [13, 14]. It takes three-layer (input layer, mapping layer, and output layer) neural network as its framework. The structural framework is shown in Figure 1. The model uses a simple three-layer neural network; in the input layer, each paragraph is mapped as a vector, as the column vector of matrix D, and each word is mapped as a vector, as the column vector of matrix W. The output layer is the word vector corresponding to the feature word. The goal of the model is to find the probability of the word that should appear in the current context window. In the training process of a document, the document ID remains unchanged, sharing the same paragraph vector, so the model uses the semantics of the whole sentence when predicting the probability of words. Therefore, the semantic similarity between words or sentences can be obtained by calculating the distance between sentence vectors.

By studying the characteristics of the log data, after determining the size of the context window, each log and word in the window is initialized as a random k-dimensional vector as input. According to equation (1), the input of the hidden layer is calculated, where C is the number of context words, x is the input vector, and W is the weight matrix:

$$h = \frac{1}{C} W \cdot \left(\sum_{i=1}^{C} x_i \right). \tag{1}$$

Then, the input of each node in the output layer is calculated according to equation (2), where $v_{wj}^{\prime T}$ is the jth column of the hidden layer output matrix $W\prime$:

$$u_j = v_{wj}^{\prime T} \cdot h. \tag{2}$$

Finally, we calculate the output of the output layer according to equation (3), in which $y_{c,j}$ represents the output value of the jth neuron node on the neuron combination of the cth context word in the output layer, that is, the probability:

$$y_{c,j} = p\left(w_{y,j} | w_1, \ldots, w_j\right) = \frac{e^{u_j}}{\sum_{j=1}^{V} e^{u_j}}. \tag{3}$$

In the process of learning the weight matrix W and $W\prime$, we assign a random value to the weight to initialize the weight matrix. Then, we train the samples in order, and calculate the gradient of the error between the output value and the true value one by one to adjust the weight matrix in the gradient direction.

The weight update equation is shown in equation (4), in which e_j is the prediction error of the output layer and η is the learning rate of the gradient descent algorithm. The vector update equation is shown in equation (5). V_w and $V_w^{\prime}$ are two representations of the word w, and V_w is a row of matrix W, which is the weight matrix from the input layer to the hidden layer. $V_w^{\prime}$ is a column of matrix $W\prime$, which is the weight value matrix from the hidden layer to the output layer:

$$w_{ij}^{\prime} \coloneqq w_{ij}^{\prime} - \eta \cdot e_j \cdot h, \tag{4}$$

$$V_{wj}^{\prime} \coloneqq V_{wj}^{\prime} - \eta \cdot e_j \cdot h. \tag{5}$$

After the training process, all log vectors are obtained.

2.2. The Improvement of K-Means. After vectorizing the log data, we use the K-means algorithm to cluster the log vectors.

The general steps of the K-means clustering algorithm are as follows. First, K data points should be randomly selected as the initial cluster centers. Then, we calculate the distance between each data point and the cluster centers and assign each data point to the cluster center closest to it. The cluster centers and the data points assigned to them represent a cluster. Once all the data are allocated, the center point of each cluster will be recalculated and updated based on the existing data samples in the cluster. This process will be repeated until a certain termination condition is met [15].

In the K-means algorithm, the selection of the k value and the initial center point sample will have a certain impact on the clustering results, so we made two improvements to the K-means algorithm.

2.2.1. Selection of k Value. Since the number of clusters will have a greater impact on the clustering effect, when initializing the K value, that is, specifying the number of clusters, we does not directly select a fixed K value manually, but according to the clustering cost function, as shown in equation (6). $D(x_i)$ represents the distance between the ith data point with its nearest cluster center, and is the total number of data points. This equation represents the sum of squared distances of data points under different K values. If the number of clusters is reasonable, the data points in the cluster will be more compact, and $S(K)$ will be smaller. Conversely, if the number of clusters is unreasonable, the data points within the cluster will be more scattered, and $S(K)$ will be larger. Therefore, when the descending speed of $S(K)$ is less than a certain threshold, the corresponding K value is a suitable result:

$$S(K) = \sum_{i=1}^{N} D^2\left(x_i\right). \tag{6}$$

2.2.2. The Selection of the Initial Center Point. Because the initial clustering centers are randomly selected, some data points which belong to the same cluster may be divided into different clusters. Therefore, if the distance between the center points of two clusters is less than a certain threshold,

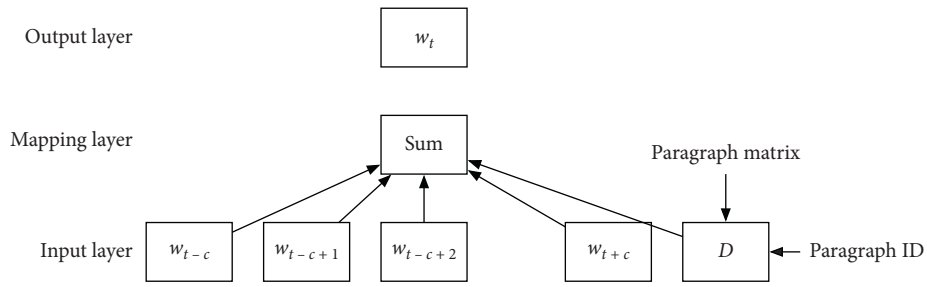

Figure 1: The structural framework of the PV-DM model.

we will merge the two clusters, that is, the number of the final generated clusters is not necessarily K.

Algorithm 1 gives the pseudocode of the improved K-means clustering algorithm. In the fourth line of the algorithm, if the clustering cost is less than the threshold, the number of clusters is set as the current K value. In line 11, when an iteration is completed if the center distance of two clusters is less than the threshold, the two clusters will be merged.

2.3. Log Fault Diagnosis Based on Reclustering. When clustering logs, the distance between some log vectors is at a boundary value, so some logs cannot be effectively clustered. In addition, frequent log sequences related to fault events are not necessarily strongly related to fault event types, that is, the same frequent event sequence may be related to multiple fault events or nonfault events. Therefore, we need to recluster the initial clustering results to make the log data in the same cluster as similar as possible. The fault diagnosis method based on the reclustering method mainly includes the following three steps.

2.3.1. Building a Corpus. After obtaining the log clustering results, we calculate the clusters with the log vector's average distance, and then, we look for the original log messages according to each log event vector identifier (described in Section 2.1). In all clusters, the original log messages are obtained, and all the words in these log messages form the corpus of this cluster.

2.3.2. Reclustering Based on Keyword Extraction. Since a robust system is in a normal state in most cases, the number of log messages that produce failures usually accounts for a relatively low percentage of the entire log file. Therefore, words that appear frequently may not be important, so we use the TF-IDF method to extract keywords from the log data.

According to equation (7), we calculate the IDF value of each word in the log message, namely, IDF(x), where N represents the total number of words in the corpus and $N(x)$ represents the total number of texts containing the word x in the corpus:

$$\mathrm{IDF}(x) = \log \frac{N}{N(x)}. \tag{7}$$

After obtaining IDF(x), we calculate TF-IDF(x):

$$\mathrm{TF-IDF}(x) = \mathrm{TF}(x) \cdot \mathrm{IDF}(x). \tag{8}$$

We sort the TF-IDF value of each word in the log message. The word with the largest TF-IDF value in each log message is selected as the key word for the current log message, and we store it in keyword list $w[i]$ (i is the sequence number in the cluster).

We select the top K (K value is the same as the number of clusters) keywords from the keyword list of each cluster as the cluster's topic word set. We check the keywords of each log and recluster the logs with the same keywords into the same cluster.

2.3.3. Fault Diagnosis. After reclustering the log data, we recalculate the centroids of all clusters. Since the calculated centroid may not correspond to the sample points in the cluster, we obtain the vector closest to the centroid as the center point of this cluster, which is also the feature log of this cluster.

The feature log data of each cluster is checked, and if the feature log indicates a system failure, we mark the sample in the corresponding cluster as a failure to complete the fault diagnosis.

3. Experiment

3.1. Log Data. We use system log data generated by two supercomputers, BlueGene/L and Thunderbird, to verify our proposed method. BlueGene/L is a supercomputer developed by IBM, and it ranked first in the world's TOP500 supercomputer rankings. Thunderbird is located in the Sandia National Laboratory in the United States. The system is manufactured by Dell and ranked sixth in the Top500 supercomputing competition in November 2006. The log data used in this article comes from the public log dataset, which can be found in https://www.usenix.org/cfdr-data.

In BlueGene/L, logs are managed through the Machine Management Control System (MMCS). The computer chip first stores the log message locally, and when it is polled, it sends the log message to the DB2 database [16]. In Thunderbird, logs are generated by syslog-ng on each local machine, stored in /var/log/ and then sent to the log server [17]. More detail information about the two log data is shown in Table 2.

Require: the initial training set $C = \{c_1, c_2, \ldots, c_n\}$, c_i is the ith log message in C; maximum number of clusters *max_k*; the threshold of the sum of squares of distance between data point and cluster center *min_dist*; the distance threshold between two cluster centers *center_list*.
Ensure: log vector matrix D
(1) **function** Improvement K-means $(C, max\_k, min\_dist, center\_list)$
(2) **for** $i \leftarrow max\_k$ **do**
(3) **if** $sum(pow(c\ d\_dist[i], 2)) - sum(pow(c\ d\_dist[i+1], 2)) \leq min\_dist$ **then**
(4) set $K = i$
(5) **end if**
(6) **end for**
(7) $centers = random(C)$
(8) **While** centers **do**
(9) **for** $k \leftarrow K$ **do**
(10) divide c_i int the closest cluster cluster[k]
(11) **end for**
(12) **end for**
(13) **if** $dist[center[k], center[k+1]] \leq center\_dist$ **then**
(14) merge $center[k]$ and $center[k+1]$
(15) **end if**
(16) **end while**
(17) **return D**
(18) **end function**

Algorithm 1: The improvement of K-means.

Table 2: Log data.

Systems	Size	Log lines	Number of anomalies	System type
BGL	1.207G	4747963	949024	Supercomputer
Thunderbird	27.367G	211212192	43,087,287	Supercomputer

3.2. Experimental Results and Analysis. The experimental machine used in this paper is configured with Intel® CoreTM i7-4790 CPU @ 3.60 GHz, 8.0 GB RAM. Algorithm evaluation indexes are accuracy, recall, and detection time.

Experiment 1. In the system log data of Bluegene/*L*, we selected the log data with a time span of 7 months. We randomly select different amounts of log data, input them into the PV-DM language model described in Section 2.2.2 for training, and then use K-means clustering algorithm clusters the log vectors generated by the model. Finally, we recluster the initial clustering results and extract the characteristic log events. Samples in the cluster are marked according to the types of the characteristic log events to complete the log fault diagnosis.

Figure 2 shows the accuracy, recall, and time consumption of fault detection when the number of log records used in training is 200000, 400000, 600000, 800000, and 1000000. As can be seen from the figure, with the increase of the number of logs used in training, the accuracy rates are 94.1%, 96.2%, 97.3%, 98.7%, and 99.0%, the recall rates are 88.5%, 89.7%, 90.2%, 90.9%, and 91.1%, and the diagnosis time is 5.3 minutes, 8.1 minutes, 12.5 minutes, 23.6 minutes, and 31.2 minutes, respectively.

Experiment 2. In Thunderbird, the log data with a time span of 6 months was selected. The preprocessing operation of log data is the same as experiment 1, and the detection results are shown in Figure 3. The number of log records used for training is 200000, 400000, 600000, 800000, and 1000000 lines, respectively. As shown in Figure 3, with the increase of the number of logs used in the training set, the accuracy rates are 89.8%, 93.7%, 96.3%, 97.2%, and 97.9%, the recall rates are 87.4%, 88.6%, 89.7%, 90.2%, and 90.3%, respectively, and the detection time is 6.4 minutes, 9.2 minutes, 15.7 minutes, 24.6 minutes, and 33.0 minutes, respectively.

Comparing the experimental results of the two system logs, we can find that, with the increase of the number of training logs, the accuracy and recall rate are gradually increasing. Because the PV-DM language model predicts the current words through the context, the context information will have a certain impact on the effect of the model. When the log data is small, the vocabulary size is small, and the difference of context information is small, so the accuracy and recall rate are low. When the number of logs increases, the size of vocabulary becomes larger, and the difference of context information becomes larger, then the accuracy and recall rate will be improved. When the number of log records reaches 1 million lines, the parameters of the model have no change after many iterations, so the accuracy and recall rate tend to be a fixed value. Although our algorithm performs better on Thunderbird dataset than Bluegene/L, both of them can achieve detection accuracy of more than 97% and recall rate of more than 90%, and the consumption of time is basically the same.

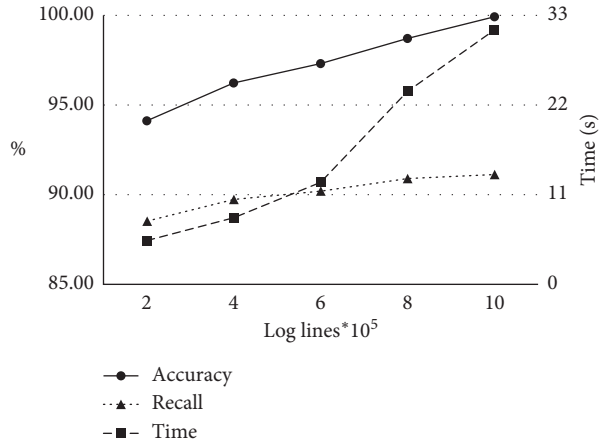

Figure 2: The experimental result of fault diagnosis on BlueGene/L.

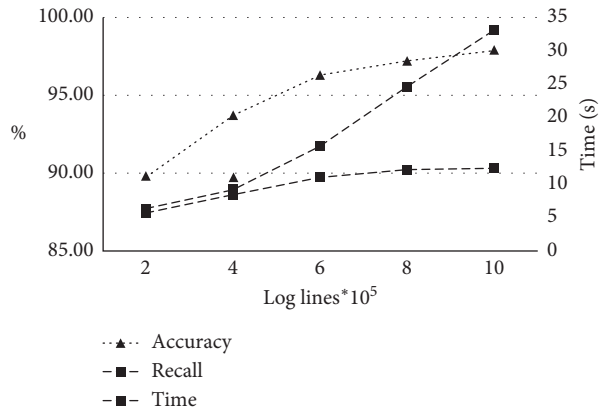

Figure 3: The experimental result of fault diagnosis on Thunderbird.

Experiment 3. System fault diagnosis algorithm based on text feature mining usually uses the TF-IDF method for vectorization and then uses the clustering method to cluster log vectors for fault diagnosis. The algorithm proposed in this paper mines log semantic information for log vectorization, so we compare it with the TF-IDF-based system fault diagnosis algorithm. The vectorization method described in Section 2.1 of this paper is changed to the TF-IDF method, and then, fault diagnosis is carried out. The experimental results are shown in Figure 4.

Figure 4 shows the comparison results of the accuracy between the two algorithms when the number of log records is 200,000, 400,000, 600,000, 800,000, and 1 million rows. It can be seen that the accuracy of the algorithm of our method is better than the fault diagnosis method based on TF-IDF, in the case of different amounts of log data.

Figure 5 shows the comparison results of recall rates of the two algorithms when the number of log records is 200,000, 400,000, 600,000, 800,000, and 1 million rows. It can be seen that the recall rate of our method is better than the fault diagnosis algorithm based on TF-IDF, in the case of different amounts of log data.

Figure 6 shows the comparison results of the detection time of the two algorithms when the number of log records is

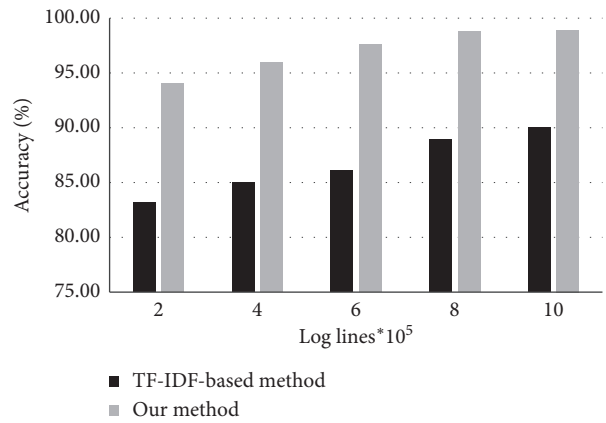

Figure 4: The results of accuracy between the fault diagnosis method based on TF-IDF and our method.

200,000, 400,000, 600,000, 800,000 and 1 million rows. It can be seen that the time consumed by the algorithm with our method is less than that of the fault diagnosis algorithm with TF-IDF under different amounts of log data. Moreover, compared with traditional log vectorization methods, the language model used in this paper can directly vectorize log

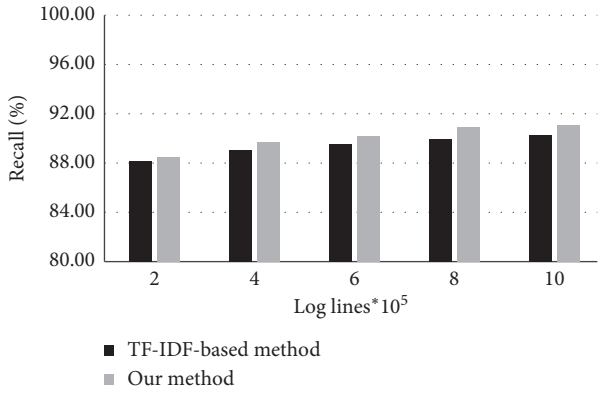

Figure 5: The results of the recall rate between the fault diagnosis methods based on TF-IDF and our method.

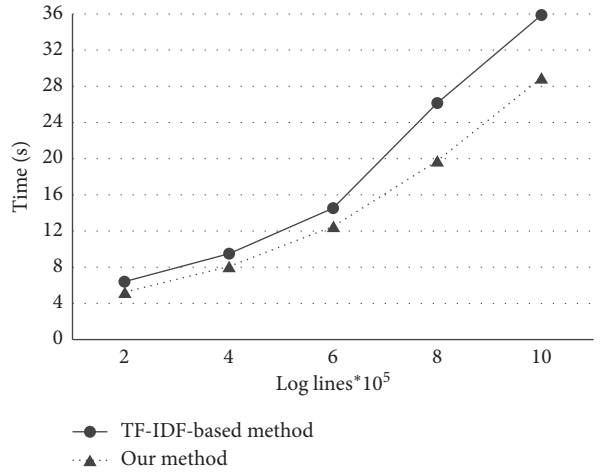

Figure 6: The results of time between the fault diagnosis method based on TF-IDF and our method.

events after training, so when processing new log input, the detection time of the algorithm proposed in this paper will be significantly reduced.

4. Conclusions

Due to the large-scale of log data and that many log data cannot provide sufficient prior knowledge, the *K*-means algorithm can analyze log data well without much prior knowledge. However, log fault diagnosis based on traditional *K*-means algorithm needs to be improved in both efficiency and accuracy. Therefore, we propose a system fault diagnosis method based on reclustering algorithm. This method first uses the PV-DM language model to train the input log data to obtain a lower-dimensional log vector; then, we improve the *K*-means algorithm from two aspects: the selection of k value and the merging of the initial center points, and use the improved *K*-means to cluster the log vectors; finally, we propose a reclustering method with the keyword extraction technology to reorganize the previous clustering results and complete the log fault diagnosis based on the new clustering results after the reorganization. We use the system log data generated by the BlueGene/L and Thunderbird supercomputers to verify our method. The experimental results show that our method can not only accurately mine log semantic information and perform fault diagnosis but also balance the accuracy and time of fault diagnosis. However, how to adapt the semantic similarity threshold according to the log characteristics in the clustering stage is the main content of the next research.

Data Availability

The implementation and datasets used in this paper are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was supported in part by the grants of National Natural Science Foundation of China (62072342 and 61672392).

References

[1] Y. Liang, Y. Zhang, A. Sivasubramaniam et al., "Filtering failure logs for a bluegene/L prototype," in *Proceedings of the International Conference on Dependable Systems and Networks*, pp. 476–485, Washington, DC, USA, July 2005.

[2] N. R. Adiga, G. Almasi, G. S. Almasi et al., "An overview of the BlueGene/L supercomputer," in *Proceedings of the Supercomputing ACM/IEEE 2002 Conference*, p. 60, Washington, DC, USA, March 2002.

[3] Q. Lin, H. Zhang, J. G. Lou et al., "Log clustering based problem identification for online service systems," in *Proceedings of the International Conference on Software Engineering Companion. ACM*, pp. 102–111, NewYork, NY, USA, June 2016.

[4] Q. Fu, J. G. Lou, Y. Wang et al., "Execution anomaly detection in distributed systems through unstructured log analysis," in *Proceedings of the IEEE International Conference on Data Mining*, pp. 149–158, Washington, DC, USA, December 2009.

[5] J. G. Lou, Q. Fu, S. Yang, and Y. Xu, "Mining invariants from console logs for system problem detection," in *Proceedings of the Usenix Atc*, pp. 231–244, Boston, MA, USA, July 2010.

[6] W. Xu, L. Huang, A. Fox et al., "Detecting large-scale system problems by mining console logs," in *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles SOSP 09*, pp. 117–113, Helena, MT, USA, October 2009.

[7] W. Shang, Z. M. Jiang, H. Hemmati et al., "Assisting developers of big data analytics applications when deploying on hadoop clouds," in *Proceedings of the International Conference on Software Engineering*, pp. 402–411, San Francisco, CA, USA, May 2013.

[8] M. Chen, A. X. Zheng, J. Lloyd et al., "Failure diagnosis using decision trees," in *Proceedings of the Autonomic Computing IEEE*, pp. 36–43, New York, NY, USA, May 2004.

[9] T. Reidemeister, M. Jiang, and P. A. S. Ward, "Mining unstructured log files for recurrent fault diagnosis," in *Proceedings of the International Symposium on Integrated Network Management*, pp. 377–384, Dublin, Ireland, August 2011.

[10] C. Yuan, N. Lao, J. R. Wen et al., "Automated known problem diagnosis with event traces," in *Proceedings of the European Conference on Computer Systems*, pp. 375–388, London, UK, August 2006.

[11] S. He, J. Zhu, P. He et al., "Experience report: system log analysis for anomaly detection," in *Proceedings of the International Symposium on Software Reliability Engineering*, pp. 207–218, Ottawa, Canada, April 2016.

[12] L. Tangl and C. S. Perng, "LogSig: generatingsystemevents from raw textual logs," in *Proceedings of the ACM International Conference on Information and Knowledge Management*, pp. 785–794, New York, NY, USA, October 2011.

[13] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proceedings of the International Conference on Machine Learning*, pp. 160–167, Helsinki, Finland, July 2008.

[14] T. Mikolov, K. Chen, G. Corrado et al., "Efficient estimation of word representations in vector space," 2013, https://arxiv.org/abs/1301.3781.

[15] J. Han and M. Kamber, *Data Mining: Concept and Technology*, Mechanical Industry Press, Beijing, China, 2012.

[16] S. J. OlinerA, "What supercomputers say: a study of five system logs," in *Proceedings of the The International Conference on Dependable Systems and Networks*, pp. 575–584, Washington D.C.,USA, June 2007.

[17] Y. Liang, Y. Zhang, H. Xiong et al., "Failure prediction in IBM Bluegene/L event logs," in *Proceedings of the Seventh IEEE International Conference on Data Mining*, pp. 583–588, Omaha, NE, USA, December 2008.