2018 4th International Conference on Frontiers of Signal Processing

Anomaly Detection of System Logs Based on Natural Language Processing and Deep Learning

Mengying Wang[1,2], Lele Xu[1*], Lili Guo[1]
[1] Key Laboratory of Space Utilization, Technology and Engineering Center for space Utilization, Chinese Academy of Sciences, 100094, Beijing, China
[2] University of Chinese Academy of Sciences, 100049, Beijing, China
e-mail: xulele@csu.ac.cn

***Abstract*—System logs record the execution trajectory of the system and exist in all components of the system. Nowadays, the systems are deployed in a distributed environment and they generate logs which contain complex format and rich semantic information. Simple statistical analysis methods cannot fully capture log information for effective abnormal detection of software systems. In this paper, we propose to analyze the logs by combining feature extraction methods from natural language processing and anomaly detection methods from deep learning. Two feature extraction algorithms, Word2vec and Term Frequency-Inverse Document Frequency (TF-IDF), are respectively adopted and compared here to obtain the log information, and then one deep learning method named Long Short-Term Memory (LSTM) is applied for the anomaly detection. To validate the effectiveness of the proposed method, we compare LSTM with other machine learning algorithms, including Gradient Boosting Decision Tree (GBDT) and Naïve Bayes, the results show that LSTM can perform the best for anomaly detection of system logs with both of the two feature extraction methods, indicating that LSTM can capture contextual semantic information effectively in log anomaly detection and will be a promising tool for log analysis.**

Keywords-anomaly detection; Word2vec; TF-IDF; LSTM; log analysis; natural language processing

I. INTRODUCTION

With the rapid rise of cloud computing, software systems are gradually migrating to a distributed environment centered on large server clusters. Due to the increase of software complexity and the advancement of the version, there are more and more code defects in the software, which brings great challenges to the system abnormal detection. System logs record the execution trajectory of the system and exist in all components of the system, so it can detect abnormal behaviors of the system by mining a large number of logs which contains rich information [1].

Traditional approaches that leverage system logs for anomaly detection can be broadly classified into three groups as follows.

a) The rule-based approach: It is mainly to express expert knowledge as a series of rules, and the rules need to be written by developers in advance, such as Logsurfer [2].

b) The method based on association analysis: It is to discover the correlation from a large quantity of logs, find out the association rules, and use it for anomaly detection [3], the researches on this kind of anomaly detection problem are roughly divided into time series based association analysis method [4] and message sequence based association analysis method [5].

c) The classification-based method: It collects the statistical information of logs as features and then use classification methods to realize anomaly detection [6]. The classification-based methods could achieve great anomaly detection results and attract increasing attention in recent years.

Feature extraction and anomaly detection algorithms are two important parts in these methods. For the aspect of feature extraction, natural language processing can transform unstructured logs into structured data and extract the log semantic information; in terms of anomaly detection algorithms, deep learning methods are powerful and can effectively enhance the accuracy of anomaly detection. In this paper, we propose to perform anomaly detection of system logs based on natural language processing and deep learning algorithm. Firstly, preprocess the log by filtering, deduplication, tagging, etc. Secondly, use Word2vec or Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction. Finally, adopt Long Short-Term Memory (LSTM) for anomaly detection. For evaluating the effectiveness of the proposed method, we compare LSTM with other algorithms of Gradient Boosting Decision Tree (GBDT) and Naïve Bayes on each of the two feature extraction methods, including Word2vec and TF-IDF.

This paper is organized as follows. In section II, some related works are reviewed. Section III introduces our methods. Section IV demonstrates the experiment. Section V shows and discusses the results, and section VI concludes the paper.

II. RELATED WORK

A. Log Feature Extraction

System logs are generally unstructured, and their format can vary from system to system, and contain lots of duplicate information, which seriously affects the efficiency of log analysis. Therefore, we should parse unstructured or semi-structured logs into structured data and extract features before log analysis. Traditional methods include extracting log keywords, replacing irrelevant variables, and removing redundant information.

978-1-5386-7853-4/18/$31.00 ©2018 IEEE

As shown by several prior works [7], they assumed that each log contained a timestamp and thread ID or request ID to distinguish from different threads, and converted the unstructured log data into specific keyword formats, and then adopted these keyword sequences and log-related timing information for subsequent anomaly detection. Wang et al. replaced the variables in the original logs (such as numbers, file names, IP addresses, etc.). Then the logs were grouped by the edit distance, and the time threshold was set to filter the redundant logs [8].

However, these methods just extract the basic keywords and lose the semantic information or contextual information from the complex-format logs effectively.

B. Log-Based Anomaly Detection Methods

The existing log-based anomaly detection approaches mainly focus on rules-based, association-based, and classification-based approaches.

The rule-based approach is primarily to express expert knowledge as a set of rules that require developers to write in advance using scripts, the operator needs to specify two types of rules, one for regular expressions that extract certain text patterns from log messages, and one for performing simple aggregations on extracted patterns [9]. The method based on association analysis is to discover the correlation between items from a large amount of log data, find out the association rules, and use it for anomaly detection [10-11]. System anomaly detection based on classification model usually models abnormal or normal status of single categories. Beshastnikh et al. proposed a new model inference technique named as CSight, to solve the difficult problem of anomaly detection for concurrent system [12]. Although the above anomaly detection methods have been explored, for complex software systems, it is difficult to obtain comprehensive anomaly detection patterns, resulting in an excessively low abnormality detection rate.

III. METHODS

Due to the complex format and rich semantic information of logs in complex software systems, traditional keyword extraction and irrelevant variable substitution methods cannot fully capture log information. Furthermore, the traditional shallow anomaly detection methods can mine obvious abnormal patterns, but may miss rare exception patterns. Therefore, this paper firstly uses the Word2vec, as well as Term Frequency-Inverse Document Frequency (TF-IDF) methods in natural language processing to preprocess the text, and extract the structured text vector features. Further, the deep learning method of LSTM is adopted to recognize the abnormal state, and compared with other algorithms of Gradient Boosting Decision Tree (GBDT) and Naïve Bayes.

A. Log Feature Extraction

Since the log contexts are related to each other rather than independently, the vector characteristics of the logs can be obtained by mining the log context information. In this paper, we use Word2vec and TF-IDF respectively to convert the words in the log into dense vectors for feature extraction.

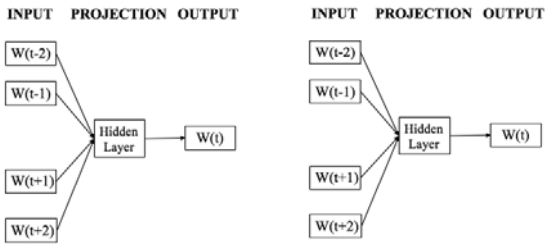
Word2vec is a model that can convert words in sentences into vector expressions. By training, the processing of log content is simplified to vector operations in K-dimensional vector space, and the similarity in vector space can be used to represent log semantics. It is mainly divided into two methods: Continuous Bags of Words (CBOW) and Skip-gram (Figure.1), where CBOW predicts the current word in the case of a given context, while Skip-gram attempts to predict the context in the case of a given word. Probability, CBOW is appropriate for small data, Skip-gram performs better in large corpora [13].

Term Frequency-Inverse Document Frequency (TF-IDF) [14] is a statistical method used to assess the importance of a word for a file set or one of the files in a corpus. The importance of a word increases proportionally with the number of times it appears in one file, but it also decreases inversely with the frequency it appears in the corpus. In other words, we can calculate the TF-IDF of all the nouns that appear in the article. The larger the TF-IDF, the higher the distinction between the noun and the article, and the more TF-IDF values. The specific formulas are as follows.

$$TF = \frac{the\ number\ of\ times\ it\ appears\ in\ the\ file}{the\ total\ number\ of\ words\ in\ the\ file} \quad (1)$$

$$IDF = log\frac{the\ total\ number\ of\ document\ in\ the\ corpus}{the\ number\ of\ document\ containing\ the\ word+1} \quad (2)$$

$$TF - IDF = TF \times IDF \quad (3)$$



(a) (b)

Figure 1. The models of Word2vec, (a) CBOW and (b) Skip-gram.

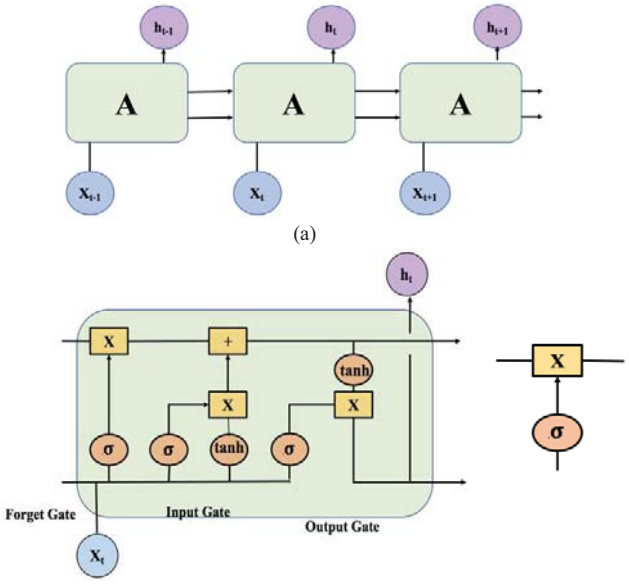
B. Anomaly Detection Methods

In this study, we choose Long Short-Term Memory (LSTM) for anomaly detection, and compare with Gradient Boosting Decision Tree (GBDT) and Naïve Bayes. LSTM is a recurrent neural network (RNN) architecture published in 1997 [15]. It is designed to solve long-term dependencies, and does not require special debugging hyper-parameters in a particularly complicated way, it can remember long-term information by default. Unlike traditional RNNs, an LSTM network is well-suited to learn from experience to classify, process when there are very long-time lags of unknown size between important events. Figure 2 shows schematic diagram of LSTM, we can find that LSTM contains three gates to control cell status and hidden layer status, namely the "forget gate", the "input gate" and the "output gate". From Figure 2b we can see, the "forget gate" is to decide what information we're going to pass to the current input from

the previous cell state. The "input gate" decides how much information we will add to the cell state from the current input. The "output gate" decides what we are going to output based on the updated cell state. The cell state flows across all the LSTM units and records the long-term history information of the input, thus it performs great in the sequence data analysis.

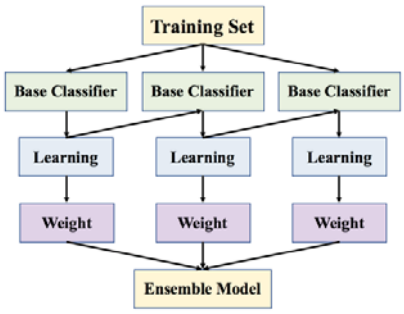
Gradient Boosting Decision Tree (GBDT) is one of the state-of-the-art ensemble learning algorithms based on boosting [16]. It typically uses Classification and Regression Tree (CART) as base learners, and fits a CART at each iteration step to pseudo-residuals of the previous step, and continuously improves the accuracy by reducing the bias of different CARTs.

Naïve Bayes is one of the classic machine learning classification algorithm based on probability theory. It classifies data by maximizing the posterior probability based on the Bayes' theorem. Schematic diagram of GBDT and Naïve Bayes are shown in Figure 3.

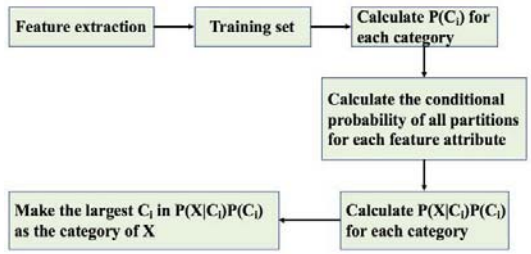


(a) (b) (c)

Figure 2. Schematic diagram of LSTM, (a) Repeating units in LSTM, (b) Unit structure in LSTM, (c) Gate structure



(a)



(b)

Figure 3. (a) Schematic diagram of GBDT and (b) Schematic diagram of Naïve Bayes

C. Evaluation Indicators

We use the accuracy, precision, recall rate, F1-score and the ROC (Receiver Operating Characteristic) curve [17] as evaluation indicators to evaluate and compare the detection results. The ROC curve is drawn according to the TPR (vertical axis) and FPR (horizontal axis), the area under the ROC curve (AUC) represents the performance of the classifier. The closer the AUC is to 1, the better the diagnostic effect is. The specific calculation formulas are as follows.

$$TP — true\ positives \qquad TN — true\ negatives$$

$$FP — false\ positives \qquad FN — false\ negatives$$

$$Accuracy = \frac{TP+TN}{TP+TN+FN+FP} \tag{4}$$

$$P(Precision) = \frac{TP}{TP+FP} \tag{5}$$

$$R(Recall\ rate) = \frac{TP}{TP+FN} \tag{6}$$

$$F1 = \frac{2*P*R}{P+R} \tag{7}$$

$$TPR = \frac{TP}{TP+FN} \tag{8}$$

$$FPR = \frac{FP}{TN+FP} \tag{9}$$

IV. Experiments

A. Dataset

In this paper, we use the system logs generated by the Thunderbird supercomputer. This supercomputer was installed at Sandia National Laboratories in Albuquerque, New Mexico, USA. The logs used in this article are derived from publicly available log datasets, which you can find at https://www.usenix.org/cfdr/.

In Thunderbird, an example of logs it generates is shown in Figure 4. We identify log records containing ALERT, FATAL and FAILED as exceptions, and others as normal events.

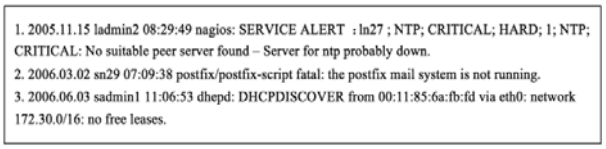


Figure 4. An example of logs generated by Thunderbird

B. Experimental Settings

Because the data samples are not balanced, we first use the oversampling method to obtain the equilibrium samples, and then divide the data set into training set and test set with a ratio of 8:2. Next, we select the skip-gram model of Word2vec to train the word vectors by setting the vector dimension to 100.

For anomaly detection, we use the Long Short-Term Memory (LSTM) algorithm, Gradient Boosting Decision Tree (GBDT) algorithm and Naïve Bayes. In LSTM, the batch size is 32 and one embedding layer is added. In order to prevent over-fitting, we use dropout with the probability of 0.5 for regularization. In GBDT, the number of base classifiers is 200, the weight reduction factor for each base classifier is 0.1, and the maximum depth of decision tree is 5. As to Naïve Bayes, Gaussian Naïve Bayes is applied after Word2vec because of the continuous features extracted by Word2vec, while Multinomial Naïve Bayes after TF-IDF as this model is characterized by words, and the value is term frequency.

C. Algorithm Comparison

The experiments are compared from two aspects. In terms of feature extraction, we compare Word2vec with TF-IDF. For anomaly detection, LSTM, GBDT and Naïve Bayes are performed and compared. Accuracy, precision, recall rate, F1-score and the ROC curve are adopted for evaluating the performance of all the models.

V. Results and Discussion

A. The Results of Preprocessing

Before the feature extraction algorithms applied on the logs, we first perform data cleaning on the original logs to remove unnecessary information, and use the over-sampling method to obtain the equilibrium samples. Furthermore, mark the logs containing “fatal”, “failed” and “alert” as the exception class “1” and the rest as normal class “0”. Through the preprocessing of the original logs generated by the Thunderbird, the number of valid logs is 16507988 and the number of invalid logs is 8251462.

B. The Results of Feature Extraction

In this paper, we firstly use the skip-gram model of Word2vec and TF-IDF for feature extraction, they can transform words into vectors, and then the vectors obtained by the two models are input to anomaly detection models for training respectively. The comparisons of different models with accuracy, precision, recall rate and F1-score are shown in Table III and Figures 5-6.

As it can be seen in the results (Table III and Figures 5-6), the accuracy of Naïve Bayes, GBDT and LSTM are 0.93, 0.94 and 0.99 respectively after feature extraction of TF-IDF, 0.98, 0.95 and 0.99 after feature extraction by Word2vec. LSTM performs great based on both of the two feature extract methods. This may because that LSTM has a strong ability to capture the relationship between contexts and thus are insensitive to different features. GBDT and Naïve Bayes perform better based on the features extracted from Word2vec than those from TF-IDF, indicating that feature extraction methods are important in the shallow machine learning algorithms, and also demonstrating that Word2vec can mine more effective semantic information of logs for anomaly detection than TF-IDF.

C. The Results of Anomaly Detection

After feature extraction, we select LSTM, GBDT and Naïve Bayes algorithms for anomaly detection respectively. The in-depth comparisons using accuracy, precision, recall rate, F1-score and the ROC curve of these models are shown in the Table III and Figures 5-7.

From the results (Table III and Figures 5-6) we can know, after feature extraction of TF-IDF, LSTM achieves a better performance (the accuracy, precision, recall rate and F1-score are all 0.99) than GBDT (the accuracy, precision, recall rate and F1-score are 0.94, 0.92, 0.91 and 0.91) and Naïve Bayes (the accuracy, precision, recall rate and F1-score are 0.93, 0.89, 0.91 and 0.90). After feature extraction of Word2vec, LSTM also shows reasonable performance, its accuracy, precision, recall rate and F1-score are all 0.99, which is much better than GBDT (the accuracy of 0.95, precision of 0.96, recall rate of 0.95 and F1-score of 0.95) and Naïve Bayes (the accuracy of 0.98, precision of 0.98, recall rate of 0.97 and F1-score of 0.97). For these three algorithms, LSTM achieves the best performance in anomaly detection based on both of the feature extraction methods.

The ROC curve (Figure 7) shows that, the AUC value of LSTM for anomaly detection are 0.9994 with TF-IDF and 1.0 with Word2vec, the AUC value of GBDT are 0.9681 based on TF-IDF and 0.9878 based on Word2vec, the AUC value of Naïve Bayes are 0.7819 with TF-IDF and 0.9897 with Word2vec. As the closer the AUC is to 1, the better the diagnostic effect is, using LSTM for anomaly detection with both of the two feature extraction methods could achieve the best and encouraging results.

In general, LSTM could achieve the best results in anomaly detection of system logs based on both of the feature extraction methods, especially the Word2vec method.

TABLE I. Evaluation metrics of different methods on Thunderbird dataset

Feature extraction	Learning algorithms	Accuracy	Precision	Recall rate	F1-score
TF-IDF	Naïve Bayes	0.93	0.89	0.91	0.90
	GBDT	0.94	0.92	0.91	0.91
	LSTM	**0.99**	**0.99**	**0.99**	**0.99**
Word2vec	Naïve Bayes	0.98	0.98	0.97	0.97
	GBDT	0.95	0.96	0.95	0.95
	LSTM	**0.99**	**0.99**	**0.99**	**0.99**

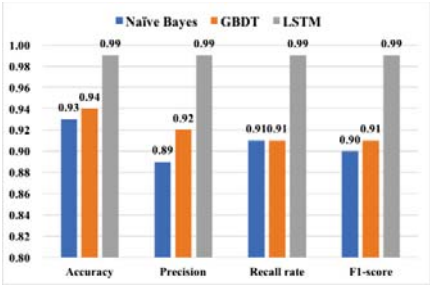


Figure 5. Results based on feature extraction of TF-IDF.

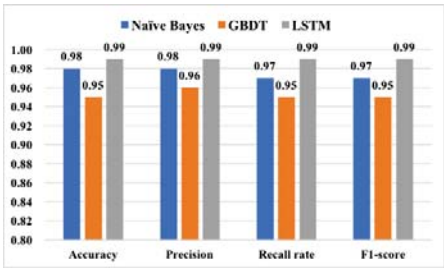


Figure 6. Results based on feature extraction of Word2vec.

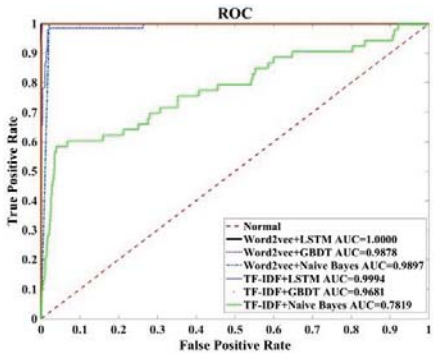


Figure 7. The ROC curve of different methods on Thunderbird dataset.

VI. CONCLUSION

For large-scale complex software systems, various sources, different log formats and incompatibility increase the difficulty of abnormal detection. We propose a method for anomaly detection that combines natural language processing methods, such as Word2vec and TF-IDF and deep learning algorithms of LSTM, and verify its effectiveness and accuracy with the system logs generated by the Thunderbird supercomputer.

For feature extraction, the skip-gram model of Word2vec could capture more effective semantic information of logs in converting words into vectors expressions for anomaly detection than TF-IDF. Furthermore, the anomaly detection results show that LSTM performs better than Naïve Bayes and GBDT algorithms on both of the two feature extraction methods, demonstrating that LSTM has strong ability in capturing the contextual semantic information of logs, is insensitive to different features, and will be a powerful and promising tool in system logs anomaly detection analysis.

Although the proposed method in this study can perform excellent in anomaly detection, it may also overlook the anomalous mode which never appears in the training set. Future work should further take this problem into account.

ACKNOWLEDGMENT

The authors thanks for all the valuable comments provided by the reviewer. We wish to appreciate all the members of Key Laboratory of Space Utilization, thanks for their support and helpful discussion.

REFERENCES

[1] Anderson J P. Computer Security Threat Monitoring and Surveillance. For Washington: James P. Anderson Co, 1980.

[2] Prewett JE. Analyzing cluster log files using logsurfer. In: Proc. of the 4th Annual Conf. on Linux Clusters, 2003.

[3] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. Execution anomaly detection in distributed systems through unstructured log analysis. In Proc. IEEE International Conference on Data Mining (ICDM). 149–158. 2009.

[4] Yamanishi K, Maruyama Y. Dynamic syslog mining for network failure monitoring. In: Proc. of the 11th ACM SIGKDD Int'l Conf. on Knowledge Discovery in Data Mining. 2005.

[5] Xu W, Huang L, Fox A, Patterson D, Jordan M. Online system problem detection by mining patterns of console logs. In: Proc. of the 9th IEEE Int'l Conf. on Data Mining (ICDM). 2009.

[6] Mariani L, Pastore F. Automated identification of failure causes in system logs. In: Proc. of the 19th Int'l Symp. on Software Reliability Engineering (ISSRE). 2008.

[7] Min Du and Feifei Li. Spell: Streaming Parsing of System Event Logs. In Proc. IEEE International Conference on Data Mining (ICDM). 859–864, 2016.

[8] Wang Weihua, Ying Shi, Jia Xiangyang, Wang Bingming, Cheng Guoli. A Multi-type Failure Prediction Method Based on Log Clustering. Computer Engineering, 2017.

[9] Prewett JE. Analyzing cluster log files using logsurfer. In: Proc. of the 4th Annual Conf. on Linux Clusters, 2003.

[10] Lim C, Singh N, Yajnik S. A log mining approach to failure analysis of enterprise telephony systems. In: Proc. of the IEEE Int'l Conf. on Dependable Systems and Networks with FTCS and DCC, 2008.

[11] Lou JG, Fu Q, Yang S, Xu Y, Li J. Mining invariants from console logs for system problem detection. In: Proc. of the USENIX Annual Technical Conf , 2010.

[12] Beschastnikh I, Brun Y, Ernst MD, Krishnamurthy A. Inferring models of concurrent systems from logs of their behavior with CSight. In: Proc. of the 36th Int'l Conf. on Software Engineering, 2014.

[13] Le, Mikolov. Distributed representation of sentences and documents. ICML,2014.

[14] Ke Zhang, Jianwu Xu, Martin Renqiang Min, Guofei Jiang, Konstantinos Pelechrinis, and Hui Zhang. Automated IT system failure prediction: A deep learning approach. In Proc. IEEE International Conference on Big Data (ICBD). 1291–1300, 2016.

[15] Sepp Hochreiter and Ju¨rgen Schmidhuber. Long short-term memory. Neural computation, 1997.

[16] Jerome H. Friedman.Greedy Function Approximation: A Gradient Boosting Machine,2002.

[17] Spackman, K.A.Signal detection theory: Valuable tools for evaluating inductive learning. In Processing of the 6th International Workshop on Machine Learning (IWML),160-163, Ithaca, NY, 1989.