Natural Language Processing-based Model for Log Anomaly Detection

Zezhou Li
JD Tech
Beijing, China
lizezhou1@jd.com

Jing Zhang
JD Tech
Beijing, China
zhangjing511@jd.com

Xianbo Zhang
JD Tech
Beijing, China
zhangxianbo1@jd.com

Feng Lin
JD Tech
Beijing, China
linfeng5@jd.com

Chao Wang
JD Tech
Beijing, China
wywangchao@jd.com

Xingye Cai
JD Tech
Beijing, China
caixingye1@jd.com

Abstract—**Logs are widely used in IT industry and the anomaly detection of logs is essential to identify the running status of systems. Conventional methods solving this problem require sophisticated rule-based regulations and intensive labor input. In this paper, we propose a new model based on natural language processing techniques. In order to modify the feature extraction and to improve the vector quality of log templates, Part-of-Speech (PoS) and Named Entity Recognition (NER) are employed in our model, which leads to the less involvement of regulation-based rule and a modification of the template vector thanks to the weight vector by NER. The PoS property of each token in the template is firstly analyzed, which also reduces labor involvement and helps for better weight allocation. The weight investigation on tokens of the template is introduced to modify the template vector. And the final detection based on the modified vector of templates is realized by deep neural networks (DNNs). The effectiveness of our model is tested on three datasets, and compared with two state-of-the-art models. The evaluation results prove that our model achieves better log anomaly detection.**

Keywords—log anomaly detection, natural language processing, deep neural networks

I. Introduction

Logs are one of the major approaches recording operation status and concomitant events in IT domains like operating systems [1,2]. Apparently, these logs containing noteworthy information of the operation process are essential resource to identify whether the system is working in a healthy state [2,3]. Thus, it is of great importance to have thoroughly insights into logs and to make accurate anomaly detection. There are three types of log anomaly in general, i.e., anomalous individual logs, anomalous log sequence, and anomalous log quantitative relationship [4]. In our paper, we focus on anomalous individual log identification, i.e. the logs containing specified tokens indicating anomaly.

Normally speaking, the anomaly detection on logs includes three steps, log parsing, feature extraction, and anomaly detection [4]. The templates extracted by parsing tools are text data which is supposed to be transformed into digital data thus can be fed into the following process. To this end, the feature extraction is necessary to obtain the digital representation of templates. As for template feature extraction, there are multiple approaches currently proposed to accomplish the task. One-hot encoding is among the earliest and easiest ways that can easily transform the text template into digital representation friendly for process [1,5]. Other than this handy encoding approach, more and more researchers are now interested in applying Natural Language Processing (NLP) techniques to achieve digital transformation, among which are methods like bag-of-words, word2vec, etc. [6,7]. Although the aforementioned methods can achieve the transformation from text data into digital data, they still face some drawbacks when it comes to the log anomaly detection. One-hot encoding is a less effective one which takes too much space to form the one-zero vector, and the valuable sematic information of templates is all neglected while using one-hot encoding. Bag-of-words and word2vec can effectively obtain the words vector thorough considering the semantic information of templates, though, they lack a consideration of the importance of each token appeared in the templates [9,10]. In other words, various weights should be allocated to different tokens in the templates [6,8]. In addition, deep neural networks (DNNs) are also reported to be applied in feature extraction of templates [11].

Our model mainly focuses on the improvement of the feature extraction by considering both the semantic information and the weight allocation of each token since tokens are of different significance to the final detection. We utilize two NLP techniques to achieve the promotion, i.e. Part-of-Speech (PoS) and Named Entity Recognition (NER) [8,12,13]. And our model realizes the template feature extraction through the following steps. Specifically, the raw log messages are firstly parsed into log templates by means of an existing method, FT-Tree [14]. The templates are then processed via PoS tools to obtained the PoS property of each token in the template, which will be further utilized for weight vector calculation. At the same time, the tokens in templates are also vectored by word2vec into an initial template vector, and this initial vector will be modified furtherly by the weight vector. Those tokens tagged with important PoS tags that will contribute more to understanding the templates are retained while the others are abandoned. As for those tokens tagged with important PoS properties, some are of more importance while the others own less. NER is employed here to find out those more important tokens among all of the tokens tagged with important PoS properties, and a larger weight will be assigned to those key tokens recognized by NER. The initial template vector is then multiplied by this weight vector to generate a compound template vector which is used for the final anomaly detection by DNNs. The contributions of our work lie in the following several folds. In order to reduce labor input for log parsing and to prepare for the weight calculation, PoS analysis is adopted in our model and each token is tagged with a PoS property without sophisticated introduction of the regulations for templates extraction. Additionally, the importance of the tokens in templates is investigated for better weight allocation. And the digital vector of template is a compound consideration of both the sematic information of each token as well as the weight allocation.

The rest of this paper is organized as followings. Section 2 presents the study of related works, and the major techniques are presented in Section 3. Section 4 mainly talks about the experimental results and the comparison analysis, while Section 5 gives the final conclusions.

II. Related Works

The feature extraction procedure of the parsed templates is an essential step for anomaly detection. The main purpose of feature extraction is to transform the template of text format into a digital vector, and various methods have been proposed for template feature extraction.

A. One-hot Encoding

In DeepLog [1], each input log template from a group of k templates, t_i, $i \in [0,k)$, is encoded as a one-hot vector. In that case, a sparse k-dimension vector $V=[v_0,v_1,\ldots,v_{k-1}]$ is constructed for the log key t_i, such that $v_i=1$ and $v_j=0$ for all $j \neq i, j \in [0,k)$.

B. Natural Language Processing (NLP)

In order to extract the semantic information of log template and to transform it into a high-dimension vector, LogRobust [6] leverages off-the-shelf Fast-Text algorithm to extract the semantic information from the English vocabulary. Fast-Text can effectively capture the intrinsic relationship (i.e., semantic similarity) among words in natural language and map each word to a k-dimension vector. And various models using NLP techniques are also reported like word2vec and bag-of-words [6,9].

C. Deep Neural Network (DNN)

Different from NLP that uses word as a fine-grained unit such as word2vec or Fast-Text, LogCNN [11] produces log embeddings based on a 29x128 codebook which is a trainable layer optimized with gradient decent during the training of the whole configuration.

D. Novel Approaches

Template2Vec is a novel approach proposed in LogAnomaly [15] in order to effectively represent the words in templates based on synonyms and antonyms. And in LogClass [4], the feature construction of templates is realized by modifying the classical weighting method, TF-IDF [16,17], into TF-ILF which replaces the inverse document frequency by inverse location frequency.

III. Methodology

A piece of raw log message is a semi-structured text, such as an error log collected from an online payment application read as *HttpUtil – request connection [http: //172.24.57.18:80 / wx/v1/pay/prepay] failed, Read timeout at jave.net*. It normally consists of two parts, the variant (usually numbers) and the invariant (common incidents shared by multiple logs, also known as the template). As for anomaly detection of individual log, the purpose is to identify the existence of anomaly tokens or tokens indicating anomaly from the template parsed from initial logs. Our model proposes to use the PoS analysis as well as the NER techniques for more accurate and labor-saving detection. PoS helps to filter tokens tagged with unnecessary PoS properties, and NER aims at allocating importance to all of the tokens tagged with necessary PoS properties. A compound template vector is then obtained via the multiplication production of template vector and the weight vector.

Six steps are included in our model for log anomaly detection, i.e., template parsing, PoS analysis, initial vector construction, weight calculation based on NER, compound vector, and the final detection. The overall procedure of the detection is illustrated as Fig.1.

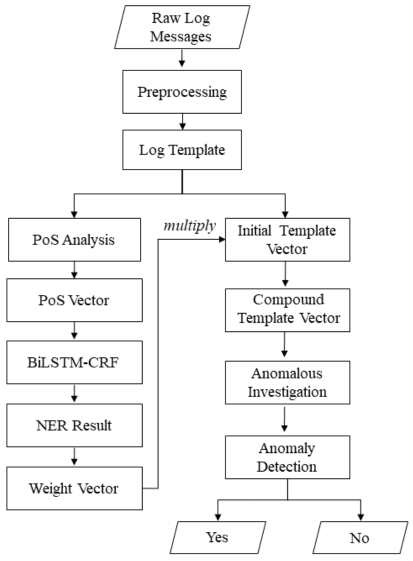
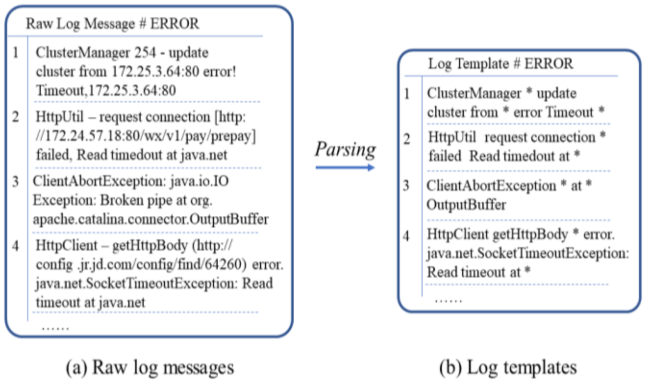


Fig. 1. Overall procedure of our model proposed for log anomaly detection

A. Template Parsing

Initial logs are semi-structured texts. They are read-unfriendly, and contain some unnecessary information which could cause confusion or obstruction for log investigations. Therefore, a preprocessing step is always required to leave out the variants like some numbers or symbols and to extract the invariant, i.e. the template. Take the aforementioned log message as an example. The template of raw log *HttpUtil – request connection [http: //172.24.57.18:80 / wx/v1/pay/prepay] failed, Read timeout at jave.net.* can be extracted as *HttpUtil request connection * failed Read time out at **. We use the simple yet effective way, the FT-Tree, to realize log parsing. It is noteworthy here that we do not introduce sophisticated rule-based regulations to take away those less important tokens, like the stop words, belonging to the invariant of a template, and PoS will do it. Therefore, engineers do not have to pay attention to the sophisticated regulation design.



```
Raw Log Message # ERROR
1  ClusterManager 254 - update
   cluster from 172.25.3.64:80 error!
   Timeout,172.25.3.64:80
2  HttpUtil – request connection [http:
   //172.24.57.18:80/wx/v1/pay/prepay]
   failed, Read timedout at java.net
3  ClientAbortException: java.io.IO
   Exception: Broken pipe at org.
   apache.catalina.connector.OutputBuffer
4  HttpClient – getHttpBody (http://
   config .jr.jd.com/config/find/64260) error.
   java.net.SocketTimeoutException: Read
   timeout at java.net
   ……
```

```
Log Template # ERROR
1  ClusterManager * update
   cluster from * error Timeout *
2  HttpUtil  request connection *
   failed  Read timedout at *
3  ClientAbortException * at *
   OutputBuffer
4  HttpClient getHttpBody * error.
   java.net.SocketTimeoutException:
   Read timeout at *
   ……
```

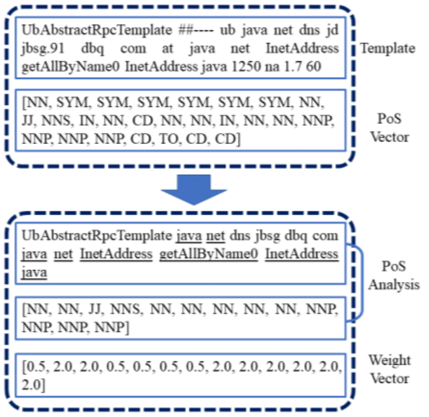
Fig. 2. Raw log messages parsing into log templates

B. PoS Analysis

Only English words, phrases, and some non-native words are remained in the well-parsed templates. These tokens are of various PoS properties, such as VB and NN. Table I lists some commonly used PoS tags and the related meanings. According to our observations on a large number of log templates, some PoS properties are important to understanding the meaning conveyed by the template while the others can be ignored. As illustrated in Fig.3, the word "at" in the parsed template is theoretically inessential, and the corresponding PoS tag "IN" is also inessential. In other word, we can still tell whether the template is normal or not even if we took out the tokens of IN. So, after we obtain the PoS vector, we can easily simplify the template by abandoning those tokens of specific PoS properties. The remained tokens are significant, at least are helpful, for better understanding of the template.

TABLE I. COMMONLY USED POS TAGS AND THE MEANINGS

PoS	Meaning	Examples
NN	Noun, singular or mass	error springframework message record
VB	Verb, base form	retry connect resolve distribute
JJ	Adjective	local unknown unable concurrent
IN	Preposition or conjunction	in, on, at
TO	"to" as preposition or infinitive marker	to
DT	Determiner	the
CC	Coordination	and, either, or



```
UbAbstractRpcTemplate ##---- ub java net dns jd jbsg.91 dbq com at java net InetAddress getAllByName0 InetAddress java 1250 na 1.7 60
```

```
[NN, SYM, SYM, SYM, SYM, SYM, SYM, NN, JJ, NNS, IN, NN, CD, NN, NN, IN, NN, NN, NNP, NNP, NNP, NNP, CD, TO, CD, CD]
```

```
UbAbstractRpcTemplate java net dns jbsg dbq com java net InetAddress getAllByName0 InetAddress java
```

```
[NN, NN, JJ, NNS, NN, NN, NN, NN, NN, NNP, NNP, NNP, NNP]
```

```
[0.5, 2.0, 2.0, 0.5, 0.5, 0.5, 0.5, 2.0, 2.0, 2.0, 2.0, 2.0, 2.0]
```

Fig. 3. PoS vector and weight vector

In our model, we utilize an off-the-shelf tool, *nltk* [18,19], to finish PoS vector construction. The input to *nltk* is the template parsed from raw log messages, and the output is the corresponding PoS vector as shown in Fig.3.

C. Initial Template Vector Construction

While obtaining the PoS vector, the template is also encoded into a digital vector. In order to take the sematic information into account, the word2vec is employed in our model to construct the initial vector of the template. This initial vector will be multiplied with the weight vector obtained in the next step, which arrives at a compound and optimized representation of the template.

Various models or tools have been developed for the generation of word vector, and all of them are able to transform the text data into digital data. In our model, we select the prevailing method, word2vec, to transform the template into the initial vector. Word2vec is able to explore and to exploit the intrinsic relationships among words in natural language and to map each word into a *d*-dimension vector (*d*=100 in our model). Suppose the template is a text vector *T* containing n tokens, and *T* can be described as (1). Word2vec transforms *T* into a word vector list, *V'*, shown as (2).

$$T = [t_0, t_1, \ldots, t_{n-1}],\ where\ n = len(T) \tag{1}$$

$$V' = [v'_0, v'_2, \ldots, v'_{n-1}],\ where\ v'_i \in R^d, i \in [0, n) \tag{2}$$

D. Weight Calculation

Tokens in the template are firstly processed by the PoS analysis, and those with little significance are abandoned. As for the remaining tokens, some are key words that convey essential information such as the server actions, health status, and so on. And the other ones might own less important information, like the object of the action, the warning level, and so on. In order to amplify those important ones, we construct a weight vector for the standout of those important tokens. NER technique is utilized for this purpose. With an input of the defined important entities, the tool is able to learn to pick out all of the tokens tagged as important ones. The process is shown in the Fig.4. In order to ensure the length match, the position of the abandoned tokens should pad with zero.

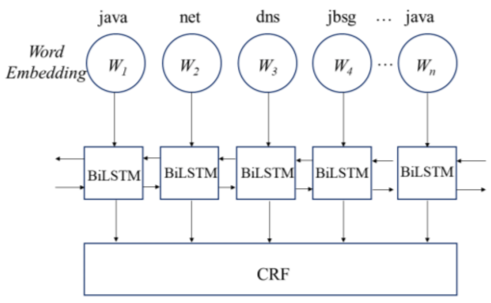


Fig. 4. NER model learning process

CRF is a normally used tool for NER [20,21], and it is also employed in our model for recognition of significant tokens. That is to say, the model could learn to recognize those tokens identified as important ones by providing it with tokens tagged as important. Once the tokens in the template are recognized by CRF, the corresponding position is assigned with a weight value (2.0). Therefore, we get a weight vector *W*.

$$W=[w_0,w_1,\ldots,w_{n-1}],w_i=\begin{cases}0.0 & \textit{if } t_i \textit{ abandoned in PoS analysis}\\ 2.0 & \textit{if } t_i \textit{ detected as important token}\\ 0.5 & \textit{if } t_i \textit{ detected as unimportant token}\end{cases} \quad (3)$$

E. Compound Vector

After obtaining the weight vector *W*, a compound and optimized vector *V* representing the template can be achieved by multiplying the initial vector *V'* by the weight vector W. Important tokens are allocated with more attention while the others with less.

$$V=V'\cdot W \quad (4)$$

F. Anomaly Detetion

The compound vector *V* obtained in step E is then fed into the final fully connected layers for anomaly detection. the output of the fully connected layers gives 0 or 1 representing normal or anomalous respectively.

IV. Experimental Analysis

In this section, experiments are carried out to verify the improvement of anomalous individual log detection by our model. Two datasets available on public archive are adopted, and one dataset collected from a globally top e-commerce corporation is also adopted to justify the practical ability. We compare our results with two advanced models proposed for log anomaly detection, Deeplog and LogClass.

The framework of CANet is constructed in PyTorch and trained on a server with an Intel(R) Xeon(R) GPU Tesla P40. We select SGD (Stochastic Gradient Descent) as the optimizer throughout the 35 training epochs. Learning rate is set to be 2e-4. All of the hyper-parameters are trained from scratch..

A. Datasets

Both BGL and HDFS are two commonly used datasets for log analysis.

1) HDFS: HDFS is collected from more than 200 Amazon EC2 nodes running Hadoop-based jobs [22]. It consists of 11,175,629 raw log messages and 16,838 are labeled as anomalous.

2) BGL: BGL is collected from a BlueGene/L supercomputer system [23,24], and it contains 4,747,963 raw log messages, among which 348,469 logs are anomalous. Each log message was manually labeled as anomaly or not.

3) Dataset A: Dataset A is collected from real-world operating systems for practical verification. It contains 915,577 raw log messages and 210,172 abnormal logs are manuually labled.

B. Baselines

We compare our model with two state-of-the-art models over the three datasets.

1) DeepLog: DeepLog is a deep neural network-based model utilizing Long Short-Term Memory (LSTM) to achieve the detection. One-hot encoding is employed in DeepLog as the template vectorizing approach.

2) LogClass: LogClass proposes a novel method, Inverse Location Frequency (ILF), to weight words of logs in feature construction. This novel weight method varies from the prevailing method, Inverse Document Frequency (IDF).

C. Evaluation Metrics

Precision, Recall, and F1-score are used to evaluate the accuracy of anomaly detection of the two baselines and our model. They are depicted as (5), (6), and (7).

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

$$Recall=\frac{TP}{TP+TN} \quad (6)$$

$$F1\text{-}score=\frac{2*Precision*Recall}{Precision+Recall}, \quad (7)$$

where TP, FP, and FN stand for true positive, false positive, and false negative.

D. Evaluation on HDFS

Fig.5 shows the performance of our model and the other two baselines on HDFS. Our model gets the highest F1-score, 0.981, against that of DeepLog and LogClass, 0.960 and 0.978. In addition, our model performs the best in recall, too. LogClass achieves the best score in precision, a bit higher than ours.

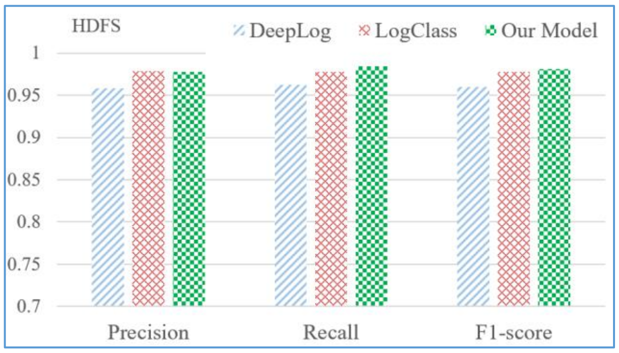


Fig. 5. Evaluation on HDFS

E. *Evaluation on BGL*

Fig.6 presents the performance of our model, DeepLog, and LogClass on BGL. Our model performs the best in recall (0.991) and F1-score (0.986) among the three models, while slightly lower than LogClass in precision. LogClass also gets satisfying results and the results are slightly lower than our results.

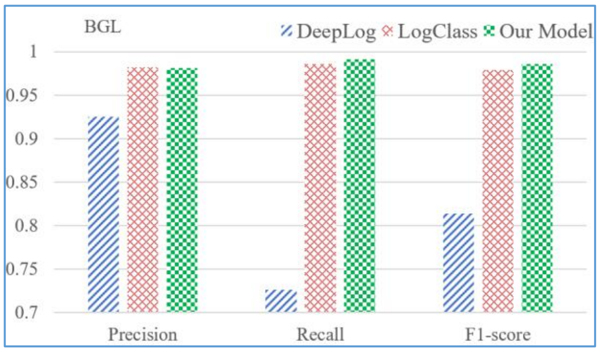


Fig. 6. Evaluation on BGL

F. *Evaluation on Dataset A*

Fig.7 presents the performance of the three models on Dataset A. Our model achieves the best performance followed by LogClass. An apparent discrepancy between DeepLog and others is observed, which might be resulted from the simple design of both feature extraction and network configuration.

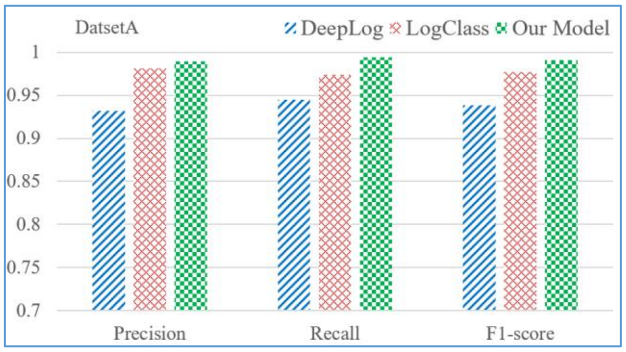


Fig. 7. Evaluation on Dataset A

Details of the specific values are listed in Table II. Our model has the best performance of F1-score in all datasets. We also get the highest recall rate against the other two baselines, which means that less confusion would be caused by our model.

TABLE II. EVALUATIONS OF EACH MODEL ON THREE DATASETS

	DeepLog			LogClass			Our Model		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
HDFS	0.958	0.962	0.960	**0.979**	0.978	0.978	0.978	**0.984**	**0.981**
BGL	0.925	0.726	0.814	**0.982**	0.986	0.979	0.981	**0.991**	**0.986**
Dataset A	0.932	0.945	0.938	0.981	0.974	0.977	**0.989**	**0.994**	**0.991**

V. CONCLUSION

In this paper, our model used for anomalous individual log detection is proposed in order to reduce labor input and to modify the template vector through weight consideration. The PoS property of the template tokens is investigated and a PoS vector is obtained to get rid of the objection by those tokens offering little reference to final detection, which is a labor-saving method. NER is then used for weight calculation of the tokens of important PoS properties. Based on the weight allocation, the template vector is optimized and promotes the model to reach a better detection results. Final results tested on two public datasets and one real-world dataset prove the superiority of our model against the other two advanced models.

Logs are usually semi-structured, so the semi-structured log-like texts are the major concern in this paper. As for those unstructured data that do not preserve case, punctuation, spaces, etc., additional measures would be necessary when preprocessing the input data and this can be studied in future.

REFERENCES

[1] Du, M., Li, F., Zheng, G., & Srikumar, V. "Deeplog: Anomaly detection and diagnosis from system logs through deep learning." Proceedings of the 2017 ACM SIGSAC conference on computer and communications security. 2017.

[2] Zhao, N., Wang, H., Li, Z., Peng, X., Wang, G., Pan, Z., ... & Pei, D. "An empirical investigation of practical log anomaly detection for online service systems." Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2021.

[3] Chen, Z., Liu, J., Gu, W., Su, Y., & Lyu, M. R. "Experience Report: Deep Learning-based System Log Analysis for Anomaly Detection." arXiv preprint arXiv:2107.05908 (2021).

[4] Meng, W., Liu, Y., Zhang, S., Zaiter, F., Zhang, Y., Huang, Y., ... & Pei, D. "Logclass: Anomalous log identification and classification with partial labels." IEEE Transactions on Network and Service Management 18.2 (2021): 1870-1884.

[5] Jie, L., Jiahao, C. H. E. N., Xueqin, Z., Yue, Z. H. O. U., & Jiajun, L. I. N. "One-hot encoding and convolutional neural network based anomaly detection." Journal of Tsinghua University (Science and Technology) 59.7 (2019): 523-529.

[6] Zhang, X., Xu, Y., Lin, Q., Qiao, B., Zhang, H., Dang, Y., ... & Zhang, D. " Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. 2019.

[7] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. 2016. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:1612.03651 (2016).

[8] Lv, Dan, Nurbol Luktarhan, and Yiyong Chen. "ConAnomaly: Content-Based Anomaly Detection for System Logs." Sensors 21.18 (2021): 6125.

[9] Mikolov, T., Chen, K., Corrado, G., & Dean, J. "Efficient estimation of word representations in vector space." arXiv preprint arXiv:1301.3781 (2013).

[10] Zhang, Yin, Rong Jin, and Zhi-Hua Zhou. "Understanding bag-of-words model: a statistical framework." International Journal of Machine Learning and Cybernetics 1.1 (2010): 43-52.

[11] Lu, Siyang, 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.

[12] Pi, A., Chen, W., Zeller, W., & Zhou, X. "It can understand the logs, literally." 2019 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). IEEE, 2019.

[13] Li, J., Sun, A., Han, J., & Li, C. (2020). "A survey on deep learning for named entity recognition." IEEE Transactions on Knowledge and Data Engineering 34.1 (2020): 50-70.

[14] Zhang, S., Meng, W., Bu, J., Yang, S., Liu, Y., Pei, D., ... & Song, L. et al. "Syslog processing for switch failure diagnosis and prediction in datacenter networks." 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS). IEEE, 2017.

[15] Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., ... & Zhou, R. "LogAnomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs." IJCAI. Vol. 19. No. 7. 2019.

[16] Soucy, Pascal, and Guy W. Mineau. "Beyond TFIDF weighting for text categorization in the vector space model." IJCAI. Vol. 5. 2005.

[17] Salton, Gerard, and Christopher Buckley. "Term-weighting approaches in automatic text retrieval." Information processing & management 24.5 (1988): 513-523.

[18] Loper, Edward, and Steven Bird. "Nltk: The natural language toolkit." arXiv preprint cs/0205028 (2002).

[19] Perkins, Jacob. Python text processing with NLTK 2.0 cookbook. PACKT publishing, 2010.

[20] Lafferty, John, Andrew McCallum, and Fernando CN Pereira. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data." (2001).

[21] Sutton, Charles, and Andrew McCallum. "An introduction to conditional random fields." Foundations and Trends® in Machine Learning 4.4 (2012): 267-373.

[22] W. Xu, L. Huang, A. Fox, D. Patterson, and M. I. Jordan, “Detecting large-scale system problems by mining console logs,” in SOSP. ACM, 2009.

[23] Adam Oliner and Jon Stearley. 2007. “What supercomputers say: A study of five system logs”. In 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN’07). IEEE, 575–584.

[24] BGL Dataset. https://www.usenix.org/cfdr-data.