LogLR: A Log Anomaly Detection Method Based on Logical Reasoning

Kehan Zhang[1,2], Xiaoqiang Di[1,2,3](✉), Xu Liu[1,2], Bo Li[1,2], Luyue Fang[1,2], Yiping Qin[1,2], and Jinhui Cao[1,2]

[1] School of Computer Science and Technology, Changchun University of Science and Technology, Changchun, China
dixiaoqiang@cust.edu.cn

[2] Jilin Province Key Laboratory of Network and Information Security, Changchun, China

[3] Information Center, Changchun University of Science and Technology, Changchun 130022, China

Abstract. Logs are widespread in large and complex software-intensive systems. Log-based anomaly detection is used for system diagnosis and troubleshooting. Existing methods extract log sequences as temporal log vectors, preserving the timing information between logs. However, they lack a reasoning mechanism, which prevents the model from mining the logical relationship between logs and loses the logical association between logs. In this paper, we propose LogLR, a log anomaly detection method based on logical reasoning. LogLR extracts the logical relationship between temporal log vectors and improves detection accuracy by combining Logical Tensor Network (LTN) with LSTM. In order to overcome the problem of ignoring the logical relationship between logs in existing statistical methods for data annotation. LogLR uses LTN to capture the logical relationship between log sequences and obtains weak labels to train an LSTM model through the weak label estimation method, which saves time costs. We evaluate LogLR on two widely used public datasets and the results demonstrate the effectiveness of LogLR.

Keywords: Log anomaly detection · LTN · LSTM · Temporal log vectors · Weak label estimation

1 Introduction

Logs are important information that records system behavior. As more and more services appear, many attack behaviors and abnormal states of the system also

This work was supported in part by the Science and Technology Development Plan Project, Jilin, China, under Grant 20190302070GX, and in part by the Education Department of Jilin Province under Grant JJKH20220773KJ, and Cernet Next Generation IT Innovation project under Grant NGIICS20190503, and the National Natural Science Foundation of China under grant No. U21A20451.

increase. The log records the information generated when the system is running, and analyzing the log can help the system administrator to find the abnormal behavior of the system. An accurate and efficient anomaly detection method is the key to maintaining the normal operation of the system.

A structured log is called a log event, and multiple log events within a period of time are called a log sequence. The log sequence could reflect the order of task execution. Early PCA [24], IM [12], DT [8] and LogCluster [10], methods methods detect log sequence anomalies. Among them, DT [8] uses event count vectors and their labels to build decision trees. While achieving commendable detection results, the method relies on labeled data. In contrast, LogCluster [10] performs anomaly detection through clustering of unlabeled data, which gets rid of the time cost of obtaining labeled data, but the detection result is lower than that of supervised learning methods. PLELog [25] proposes a semi-supervised anomaly detection method, which saves time cost while ensuring detection accuracy. However, existing semi-supervised methods based on statistical methods ignore the logical relationship between logs, resulting in a high error rate of data annotation.

Methods in the field of natural language processing (NLP) extract timing information in time series, and since log sequences are time series, many methods in the field of NLP are used for anomaly detection. DeepLog extracts the timing information between log events by inputting each log event into LSTM Cell at different time steps. PLELog uses the GRU model to observe the temporal log vectors of log sequences and detect anomalies by binary classification. These methods improve detection accuracy by obtaining timing information of log sequences. Although the existing log-based anomaly detection models are effective, they lack an inference mechanism, which leads to the loss of the logical relationship between log sequences and the reduction of detection accuracy.

To overcome the above challenges, this paper proposes LogLR, a log anomaly detection method based on logical reasoning. LogLR adds a reasoning mechanism by introducing LTN, captures the logical relationship between log sequences, and uses weak labels to assign probability values to log sequences, which not only saves time, but also maintains the effectiveness of supervised learning.

The main contributions of this paper are as follows:

1) We point out the problem that the existing data annotation methods based on statistical methods cannot extract the logical relationship between log sequences, and extract the logical information of log sequences through the weak label estimation method, which improves the accuracy of data annotation.
2) We propose LogLR, a log anomaly detection method based on logical reasoning. LogLR introduces a reasoning mechanism, and simultaneously extracts the timing information and logical information between log sequences for the first time, which improves the detection accuracy.
3) We evaluate the effectiveness of LogLR on two publicly available datasets, and the results confirm that our method outperforms existing state-of-the-art methods.

2 Related Work

Supervised Learning: Supervised learning methods use labeled data to assist in anomaly detection, so supervised learning methods perform well in detecting anomalies. Statistical models such as LR [21], DT [8], SVM [9], etc. are widely used in classification tasks and are trained using event count vectors and their labels to distinguish normal and abnormal log events. Inspired by SVM [9], methods such as OC-SVM [18], SVDD [19], etc. obtain spherical boundaries around the dataset to distinguish normal and abnormal log events. Considering the temporal relationship between log events, many RNN-based methods are used to extract temporal information between log events. LogRobust [26] extracts the semantic information of log events and detects anomalies using an attention-based Bi-LSTM model, capturing the contextual information of log sequences. OC4Seq [20] jointly detects anomalies using a local representative RNN model and a global representative RNN model, focusing on local and global information in the sequence, respectively. Methods [6,22] use the inference mechanism of Bayesian network for anomaly detection, LogGAN [23] uses GAN network to infer data to infer similar data. However, there is currently no method to effectively combine the inference mechanism with the temporal characteristics of log sequences. We propose a logical reasoning log anomaly detection method LogLR, which effectively combines the reasoning mechanism and the time-series characteristics of log sequences to improve the detection accuracy.

Unsupervised Learning and Semi-supervised Learning: Unsupervised learning and semi-supervised learning methods use unlabeled or a small amount of labeled data to assist in anomaly detection, which is more in line with practical application production environments. Methods such as PCA [24], IM [12], and LogCluster [10] perform anomaly detection by mining the similarity or linear relationship between data of the same category. Different from the widely used TFIDF [17] method, LogClass [14] proposes a new feature representation method, TFILF, and verifies the effectiveness of this method using classical machine learning methods. DeepLog [3] uses an LSTM model to preserve timing information between log events and detect anomalies when log patterns deviate from models trained under normal log execution. LogAnomaly [15] combines sequential and quantitative detection for the first time to improve detection performance. PLELog [25] uses some labeled data to label the remaining training data through probabilistic label estimation, using only a small amount of labeled data to take advantage of supervised learning. In contrast, LogLR adopts the weak label estimation method based on logical reasoning, and applies LTN to data annotation, which improves the accuracy of data annotation.

3 Methodology

In order to overcome the problem of the lack of reasoning mechanism in existing anomaly detection methods, which prevents the model from mining the logical relationship between logs. We propose LogLR, a log anomaly detection method based on logical reasoning. Figure 1 shows the overview of LogLR. LogLR consists of the following four parts: log parsing, vectorization, weak label estimation and anomaly detection.

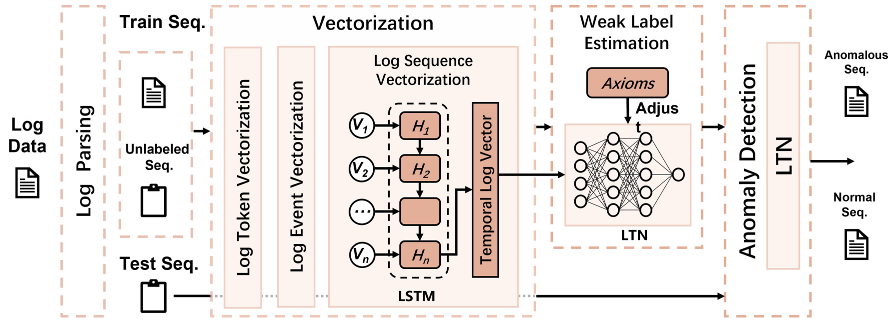


Fig. 1. Overview of LogLR

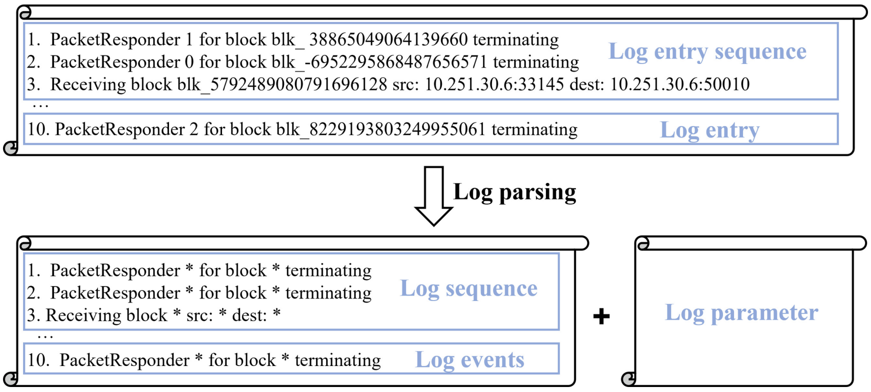
3.1 Log Parsing

Since logs are unstructured data, they contain a lot of special information (e.g., IP addresses, file names, etc.) that prevents the model from automatically detecting. It is necessary to extract this special information before using the raw logs as input to an anomaly detection model. We call the processed raw logs log events, and the step of extracting special new ones is log parsing. In this paper, we use Drain [7], which can parse logs in a streaming and timely manner. To accelerate the parsing process, Drain uses a fixed depth parse tree, which encodes specially designed rules for parsing. For example, in Fig. 2, the first log entry *"Receiving block blk_5792489080791696128 src: 10.251.30.6:33145 dest: 10.251.30.6:50010"* is parsed into the log event *"Receiving block * src: * dest: *"*. Through log parsing, unstructured raw log are transformed into structured log events.

3.2 Vectorization

The vectorization step converts the structured log events into digital vectors. Since the anomaly detection model requires an input of numeric vectors, log events need to be vectorized before being fed into the anomaly detection model. It consists of three parts: log token vectorization, log event vectorization and log sequence vectorization.

Log Token Vectorization. Treat log events as natural language sentences, each word in the sentence is called a log token, and the context between log tokens can better describe the sentence. To extract semantic information between log tokens, LogLR first splits matching words in log events into separate words according to Camel Case [2], and removes non-character tokens and stop words in



```
1.  PacketResponder 1 for block blk_ 38865049064139660 terminating
2.  PacketResponder 0 for block blk_-6952295868487656571 terminating
3.  Receiving block blk_5792489080791696128 src: 10.251.30.6:33145 dest: 10.251.30.6:50010
 …
10. PacketResponder 2 for block blk_8229193803249955061 terminating
```

```
1.  PacketResponder * for block * terminating
2.  PacketResponder * for block * terminating
3. Receiving block * src: * dest: *
 …
10.  PacketResponder * for block * terminating
```

Fig. 2. Overview of log parsing

log events to preprocess log events. LogLR then uses the FastText algorithm [16] to vectorize each log token in the log event. FastText performs word vectorization through the context of each log token to obtain a log token vector. After the log token vectorization, each log token is converted into a fixed-dimensional vector.

Log Event Vectorization. To extract semantic information between log events, LogLR performs weighted summation of the token vector in the log event to obtain the log event vector, The log event vector V be calculated by Eq. 1:

$$V = \frac{1}{N}\sum_{i=1}^{N} w_i \cdot v_i \tag{1}$$

where N is the number of log tokens in the log event, v_i is the log token vector, and w_i is the weight of each log token.

LogLR uses TF-IDF, a weighting technique commonly used in information retrieval and data mining, to calculate the weight w_i for each log token, where TF is the term frequency and IDF is the inverse text frequency index. TF is the frequency of occurrence of each log token in log events, calculated as $\frac{\#w}{\#N}$, and IDF is a measure of the general importance of a word, calculated as $log(\frac{\#L}{\#L_w})$, where #w is the number of log token w in log events, #N is the total number of log tokens in log events, #L is the total number of different log events, and $\#L_w$ is the log containing log token w number of events. The weight ω is calculated as $TF \times IDF$. Through weighted summation, LogLR obtains a log event vector containing semantic information.

Log Sequence Vectorization. After obtaining the log event vector containing semantic information, LogLR uses LSTM to solve the problem of gradient disappearance and explosion during long-sequence training, and extract the log sequence vector. Figure 3 shows the overview of An LSTM Cell.

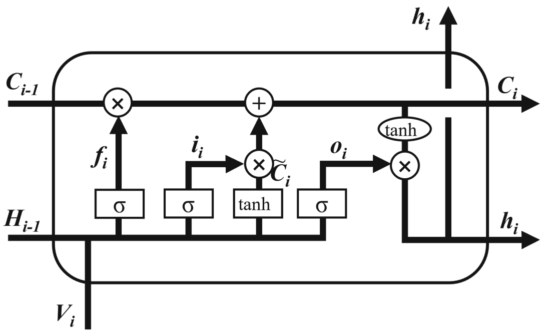


Fig. 3. Overview of An LSTM Cell

LSTM uses the gating unit to combine the LSTM state of the previous time step with the input data of this time step to generate the LSTM state of this time step. The gating unit is calculated as the Eq. 2:

$$\begin{aligned}
f_t &= \delta(W_f \cdot [h_{t-1}, x_t] + b_f) \\
i_t &= \delta(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\widetilde{C}_t &= tanh(W_c \cdot [h_{t-1}, x] + b_c) \\
o_t &= \delta(W_o \cdot [h_{t-1}, x_t] + b_o)
\end{aligned} \tag{2}$$

The LSTM state at this time step is calculated as Eq. 3:

$$\begin{aligned}
C_t &= f_t * C_{t-1} + i_t * \widetilde{C}_t \\
h_t &= o_t * tanh(C_t)
\end{aligned} \tag{3}$$

In order to extract the timing information of the log sequence, LogLR connects multiple LSTM Cell, inputs the log events in the log sequence into different LSTM Cell in turn, and uses the final hidden h_t as the log sequence vector of the sequence, which is called the temporal log vector.

3.3 Weak Label Estimation

After log-vectorization, LogLR uses LTN [1], a framework that combines tensor networks with first-order multivalued logical inference, to label unlabeled data. The structure of LTN is shown in Fig. 4.

Some objects are associated with a set of quantitative properties, represented by a real-valued n-tuple $G(o_i) \in R_n$, which we call grounding, where o_i belongs to an infinite set of objects $O = \{o_1, o_2, ...\}$. LogLR uses the vectorization process as the ground, x_+ are the normal examples, x_- are the abnormal examples input into $G(A|\theta) : x \rightarrow sigmoid(MLP(x))$, where MLP is a multilayer perceptron with one output neuron whose parameter θ needs to be learned. Through $G_\theta(A)$, LogLR obtains a probability value as the label of the input example, and labels

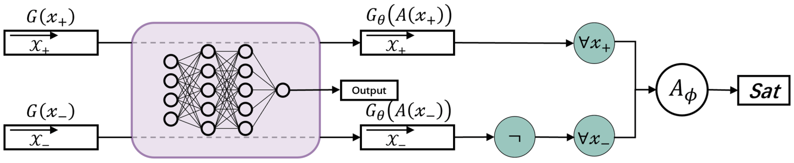


Fig. 4. Overview of LTN

weak labels for the examples at the boundary of normal examples and abnormal examples, reducing the impact of data annotation errors on the model.

LTN introduces the inference mechanism by setting the axioms, and in the back-propagation stage, the model parameters are adjusted by setting the loss function using the axioms. The axioms are set as shown in Eq. 4:

$$\begin{aligned} &\forall x_+ A(x_+) \\ &\forall x_- \neg A(x_-) \end{aligned} \tag{4}$$

K is a set of closed first-order logic formulas. The objective function with $K = \{\forall x_+ A(x_+), \forall x_- \neg A(x_-)\}$ is denoted as $SatAgg_{\phi \in K} G_{\theta, x \leftarrow D}(\phi)$. The value of the objective function represents the satisfaction of the knowledge base and the confidence that all examples are correctly classified. The loss function is calculated as 1 minus the value of the objective function. The objective function of LTN is calculated as Eq. 5:

$$\begin{aligned} &\underset{\phi \in K}{SatAgg} G_\theta(\phi) = \\ &1 - \frac{1}{2}(1 - (1 - (\frac{1}{|G(x_+)|} \sum_{v \in G(x_+)} (1 - sigmoid(MLP_\theta(v)))^2)^{\frac{1}{2} \cdot 2}) \\ &+ 1 - (1 - (\frac{1}{|G(x_-)|} \sum_{v \in G(x_-)} (sigmoid(MLP_\theta(v)))^2)^{\frac{1}{2} \cdot 2}))^{\frac{1}{2}} \end{aligned} \tag{5}$$

The notation $G_{x \leftarrow D}(\phi(x))$ means that the variable x is grounded with the data D when grounding $\phi(x)$.

In the weak label estimation stage, LogLR obtains a true value for the input sample in the interval [0, 1] as the label value of the unlabeled data. The samples at the classification boundary are easily mislabeled, and directly classifying the samples with a large label error rate will change the distribution of the samples. LogLR uses probability values as weak labels to increase training data and improve the detection accuracy of the model without changing the overall distribution of samples as much as possible.

3.4 Anomaly Detection

In the anomaly detection step, we use the session as the basic unit of classification of the anomaly detection model. A session is a process of information exchange between a client and a server. A session is established within a period of time, during which multiple information transfers are involved. We use two hyperparameters to divide sessions into log sequences, which are fed into the log anomaly detection model. A session is considered normal when all log sequences in the session are classified as normal by the anomaly detection model, but is considered abnormal when at least one log sequence in it is detected as abnormal.

LogLR detects anomalies using the LTN detection model. After the weak label estimation stage, LogLR retrains the LTN model with weak labels. LogLR uses log sequences marked with 0 or 1, where 0 indicates that the log sequence is abnormal and 1 indicates that the log sequence is normal. Different from the weak label estimation stage, the anomaly detection stage compares the true value between [0, 1] obtained by the LTN model with the preset threshold. LogLR detects the log sequence as normal when the true value of the output is greater than the threshold, otherwise it is detected as abnormal.

Overall, LogLR processes unstructured logs and converts them into structured log events. Secondly, construct the log event sequence, and extract the timing information and logical information in the log sequence and convert it into a log vector. Then, the training data is increased by the weak label estimation method, and finally the LTN model is used to extract the logical relationship between log sequences to improve the accuracy of anomaly detection.

4 Evaluation

4.1 Datasets

We evaluate our approach on two publicly available log datasets, including the HDFS dataset and the BGL dataset.

HDFS dataset: It is generated through running Hadoop-based map-reduce jobs on more than 200 Amazon's EC2 nodes, and labeled by Hadoop domain experts [3]. HDFS dataset has 11197954 log entries, according to the identifiers, the log sequences are divided into 575061 identifiers. Each identifier is annotated by domain experts. Among them, 4855 normal log sequences and 1638 abnormal sequences are selected as the training dataset, and the rest are used as the test dataset for testing.

BGL dataset: It is generated by the Blue Gene/L supercomputer, which consisted of 128K processors and was deployed at Lawrence Livermore National Laboratory(LLNL) [15]. BGL dataset has 4747963 log entries, each log entry is labeled by domain experts as normal or abnormal, and 348460 logs are labeled as abnormal. Divide log entries into log sequences, 44054 normal log sequences and 4050 abnormal sequences are selected as the training dataset, and the rest are used as the test dataset for testing.

To execute anomaly detection approaches, we group log entries into different sessions by an identifier field which for HDFS log is block_id and for BGL log is the sliding window. We divide each dataset into training, validation, and test sets with a ratio of 6:1:3 to evaluate the performance of log-based anomaly detection methods. To evaluate the annotation accuracy of the semi-supervised method LogLR, we sample 50% of the training data as known log sequences and the remaining log sequences in the training data as unlabeled log sequences to simulate a semi-supervised scenario.

4.2 Measurements

In this paper, we use Precision, Recall and F1-score scores to measure the effectiveness of abnormal detection based on log-based abnormal detection. Precision, Recall and F1-score is calculated as $\frac{TP}{TP+FP}$, $\frac{TP}{TP+FP}$, $\frac{2\cdot(Precision\cdot Recall)}{Precision+Recall}$, where TP, FP, and FN refer to the number of true positives(An abnormal log sequence is detected as an abnormal sequence), false positives(A normal log sequence is detected as an abnormal sequence), and false negatives(An abnormal log sequence is detected as a normal sequence), respectively.

4.3 Results and Analysis

Comparison with Statistical Methods. Figure 5 shows the superiority of using LTN for data annotation. Compared with statistical methods, LTN gradually regulates the classification boundary between normal and abnormal log sequences by automatic learning, and has improved the accuracy of annotation through the reasoning mechanism to reason log sequences. The experimental results show that the accuracy of using LTN for data annotation reaches 97.1%, which is higher than the existing statistical method PCA [4], K-Means [11], MST [5] and HDBSCAN [13]. LogLR uses weak label estimation method to provide a probability value for the log sequence labeled by error, thereby reducing the impact of error annotation on the detection model.

Comparison with Anomaly Detection Methods. Figure 6 shows the superiority of LogLR over other semi-supervised and unsupervised learning methods. LogLR captures the logical relationship of temporal log vectors through preset axioms, extracts the logical information of log sequences, and achieves better detection results. DeepLog and LogAnomaly outperform BGL on HDFS dataset, This is because there are more unstable data in BGL due to its longer time span compared with HDFS. More specifically, the BGL dataset is unstable, there are a lot of data in the test data that did not appear during training. DeepLog and LogAnomaly predict log events based on log sequences, are sensitive to unseen log events, and detect unseen log events as anomalies. PLELog only performs simple binary classification processing on the time log vector, ignoring the logical relationship between log events. Compared with LogGAN, LogLR achieves better results by pre-extracting the temporal characteristics of log sequences.

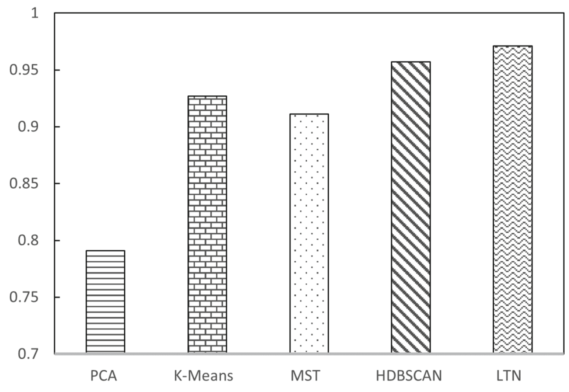
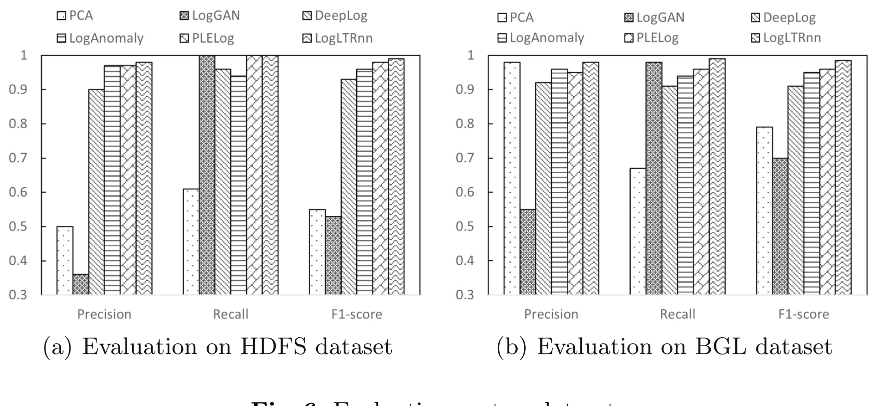


Fig. 5. Experimental results of HDFS dataset data label accuracy

LogLR outperforms existing state-of-the-art unsupervised and semi-supervised learning methods. Table. 1 shows the comparison of LogLR with the state-of-the-art supervised learning methods. Although there is a gap between LogLR and LogRobust, the gap between the three metrics is very small. This shows that LogLR combines the advantages of supervised learning well with weak label estimation methods. Moreover, as LogRobust depends on a large amount of manually labeled training data, LogLR has greater usability in practice.



(a) Evaluation on HDFS dataset (b) Evaluation on BGL dataset

Fig. 6. Evaluation on two datasets

Table 1. Comparison with supervised learning method

	LogRobust-HDFS	LogRobust-BGL	LogLR-HDFS	LogLR-BGL
Precision	0.98	1.00	0.98	0.98
Recall	1.00	1.00	0.99	1.00
F1-score	0.99	1.00	0.99	0.99

5 Conclusion

Over the years, many log-based anomaly detection methods have been proposed, but they lack inference mechanisms that prevent models from mining logical relationships between logs. In this paper, we propose LogLR, a log anomaly detection method based on logical reasoning. LogLR extracts the temporal and logical information of log sequences by effectively combining LTN and LSTM. LogLR uses LTN to detect anomalies while applying LTN to data annotation, which not only saves time costs, but also maintains the accuracy of supervised learning. Finally, we demonstrate the effectiveness of LogLR on the two most widely used public datasets, demonstrating that LogLR outperforms current state-of-the-art methods.

References

1. Badreddine, S., Garcez, A.d., Serafini, L., Spranger, M.: Logic tensor networks. Artif. Intell. **303**, 103649 (2022)
2. Dit, B., Guerrouj, L., Poshyvanyk, D., Antoniol, G.: Can better identifier splitting techniques help feature location? In: 2011 IEEE 19th International Conference on Program Comprehension, pp. 11–20. IEEE (2011)
3. Du, M., Li, F., Zheng, G., Srikumar, DeepLog: anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, pp. 1285–1298 (2017)
4. Dunia, R., Qin, S.J.: Multi-dimensional fault diagnosis using a subspace approach. In: American Control Conference. Citeseer (1997)
5. Gower, J.C., Ross, G.J.: Minimum spanning trees and single linkage cluster analysis. J. Roy. Stat. Soc.: Ser. C (Appl. Stat.) **18**(1), 54–64 (1969)
6. Gu, J., Lu, S.: An effective intrusion detection approach using SVM with naïve bayes feature embedding. Comput. Secur. **103**, 102158 (2021)
7. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: an online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS), pp. 33–40. IEEE (2017)
8. He, S., Zhu, J., He, P., Lyu, M.R.: Experience report: system log analysis for anomaly detection. In: 2016 IEEE 27th international symposium on software reliability engineering (ISSRE), pp. 207–218. IEEE (2016)
9. Liang, Y., Zhang, Y., Xiong, H., Sahoo, R.: Failure prediction in IBM bluegene/l event logs. In: Seventh IEEE International Conference on Data Mining (ICDM 2007), pp. 583–588. IEEE (2007)

10. Lin, Q., Zhang, H., Lou, J.G., Zhang, Y., Chen, X.: Log clustering based problem identification for online service systems. In: Proceedings of the 38th International Conference on Software Engineering Companion, pp. 102–111 (2016)
11. Lloyd, S.: Least squares quantization in PCM. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
12. Lou, J.G., Fu, Q., Yang, S., Xu, Y., Li, J.: Mining invariants from console logs for system problem detection. In: USENIX Annual Technical Conference, pp. 1–14 (2010)
13. McInnes, L., Healy, J.: Accelerated hierarchical density based clustering. In: 2017 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 33–42. IEEE (2017)
14. Meng, et al.: LogClass: anomalous log identification and classification with partial labels. IEEE Trans. Netw. Serv. Manage. **18**(2), 1870–1884 (2021)
15. Meng, W., et al.: LogAnomaly: unsupervised detection of sequential and quantitative anomalies in unstructured logs. In: IJCAI, vol. 19, pp. 4739–4745 (2019)
16. Pennington, J., Socher, R., Manning, C.D.: Glove: global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532–1543 (2014)
17. Salton, G., Buckley, C.: Term-weighting approaches in automatic text retrieval. Inf. Process. Manag. **24**(5), 513–523 (1988)
18. Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)
19. Tax, D.M., Duin, R.P.: Support vector data description. Mach. Learn. **54**(1), 45–66 (2004)
20. Wang, Z., Chen, Z., Ni, J., Liu, H., Chen, H., Tang, J.: Multi-scale one-class recurrent neural networks for discrete event sequence anomaly detection. In: Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining, pp. 3726–3734 (2021)
21. Wright, R.E.: Logistic regression. (1995)
22. Wu, D., et al.: LSTM learning with Bayesian and Gaussian processing for anomaly detection in industrial Iot. IEEE Trans. Industr. Inf. **16**(8), 5244–5253 (2019)
23. Xia, B., Bai, Y., Yin, J., Li, Y., Xu, J.: LogGAN: a log-level generative adversarial network for anomaly detection using permutation event modeling. Inf. Syst. Front. **23**(2), 285–298 (2021)
24. Xu, W., Huang, L., Fox, A., Patterson, D., Jordan, M.: Largescale system problem detection by mining console logs. In: Proceedings of SOSP 2009 (2009)
25. Yang, L., et al.: Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), pp. 1448–1460. IEEE (2021)
26. Zhang, X., et al.: Robust log-based anomaly detection on unstable log data. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pp. 807–817 (2019)