



Trace-Based Anomaly Detection with Contextual Sequential Invocations

Qingfeng Du¹, Liang Zhao^{1(✉)}, Fulong Tian², and Yongqi Han¹

¹ School of Software Engineering, Tongji University, Shanghai, China
{du_cloud,1931525,2011438}@tongji.edu.cn

² Di-Matrix(Shanghai) Information Technology Co., Ltd., Shanghai, China

Abstract. Nowadays, microservice architecture has been widely adopted in various real systems because of the advantages such as high availability and scalability. However, microservice architecture also brings the complexity of operation and maintenance. Trace-based anomaly detection is a key step in the troubleshooting of microservice systems, which can help to understand the anomaly propagation chain and then locate the root cause. In this paper, we propose a trace-based anomaly detection approach called TICAD. Our core idea is to group the invocations according to their microservice pairs and then perform anomaly detection individually. For each distinct microservice pair, we propose a neural network based on LSTM and self-attention to automatically learn the contextual pattern in the target invocation and previous invocations. Detected invocation anomalies can be further used to infer the trace anomalies. We have verified it on a public data set and the experimental results show that our proposed method is effective compared to the existing approaches.

Keywords: Trace · Anomaly Detection · AIOps · Deep Learning

1 Introduction

Trace-based anomaly detection is a key step in the troubleshooting of microservice systems because the structure of trace can help operators to understand the anomaly propagation chain and then locate the root cause.

Existing approaches can be divided into two categories: trace-level approaches and invocation-level approaches. For those trace-level approaches such as [3, 5], they need further analysis to locate the abnormal microservice. For those invocation-level approaches such as [2, 4], they can directly detect the invocation anomalies but highly depend on the accuracy.

We propose a supervised trace anomaly detection method called TICAD (Trace Invocation Callee Anomaly Detection), which can effectively learn the sequential patterns in the invocations and then infer the anomalies in the traces. Firstly, TICAD reorganizes the invocations of traces. Then, each invocation's callee state will be represented as a vector using multiple metrics. TICAD will

mine the inherent relationship between the previous invocations and the current one through a neural network based on LSTM and self-attention. After the invocation anomalies are detected, whether a trace is abnormal can be inferred from them.

Our main contributions are listed below: We propose the TICAD, which detects invocation anomalies and subsequently infers the anomalies of the traces. We further propose a neural network based on LSTM and self-attention to detect anomalies in the invocations, which can learn the contextual dependencies and patterns between the invocation vectors. We conduct extensive experiments on TICAD to verify its effectiveness on the public dataset.

2 Related Works

Supervised Machine Learning Approaches: MEPFL [8] is proposed to predict multiple tasks such as latent error detection in the trace log, which is collected from both normal and faulty versions of the application. And Seer [1] is presented to detect Qos violations in the massive trace data. A deep learning model, which contains CNN and LSTM layers, is trained in Seer to predict the abnormal microservices.

Unsupervised Machine Learning Approaches: Among all the unsupervised approaches, most of them are based on the normal assumption. AVEB [4] trains a variational autoencoder to learn the response time feature of normal cases for each microservice. Then the target data with significant reconstruct errors will be determined as anomalous. TraceAnomaly [3] also trains a variational autoencoder with posterior flow to model the normal pattern of trace. In [5], a multimodal LSTM model is proposed to learn the sequential pattern of the invocation type and response time.

3 TICAD Design

As shown in the Fig. 1, we first reorganize all the traces. After that, each invocation will be transformed into a vector according to the metrics. For each invocation, its vector will be fed into a neural network based on LSTM and self-attention along with those of the previous invocations. Finally, it will automatically learn the potential features associated with anomalies.

3.1 Trace Pre-processing

In this section, we process the original trace data and fine tune the data structure to better detect anomalies. We group all the invocations with the same microservice pair and reorder them by their timestamp. After that, the original dataset is divided into n_c datasets where n_c is the number of unique microservice pairs. In the following steps, invocations of different groups can be processed and

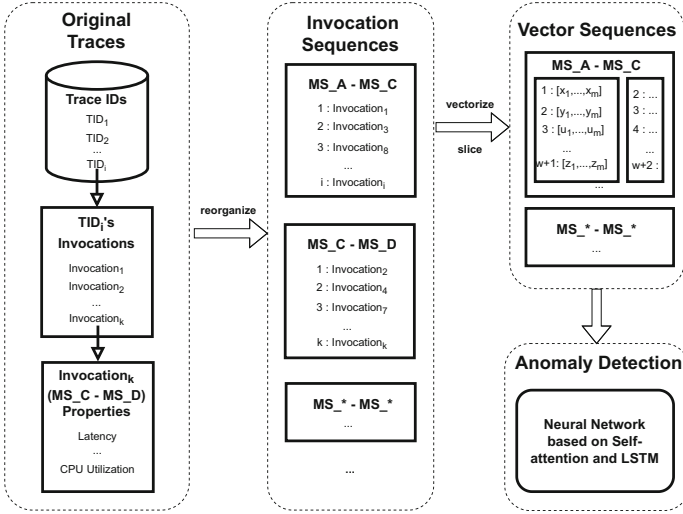


Fig. 1. The framework of TICAD

learned in parallel without affecting each other. Next, we vectorize the invocations from different perspectives. More precisely, we vectorize the callee state of the invocation, which means the label represents whether the callee is normal when the invocation occurs. To avoid the problem caused by using the latency alone, we use additional resource utilization metrics of the callee to enhance the representation of the invocation. We directly concatenate the latency and resource utilization metrics to form the vector of the invocation and standardize the values of each dimension in the vector.

3.2 Anomaly Detection

After the vectorization of the invocations, we will detect whether each invocation is abnormal. For the target invocation, in addition to its own feature vector information, we also use the extra vectors of the previous invocations to enrich the current information. Instead of relying only on the vector of the target invocation, this kind of learning method can help to decrease the false positives caused by the noise data. In practice, a reasonable window will be selected to slice the invocations. For each invocation waiting to be detected, the input is a matrix X consists of $w + 1$ vectors:

$$X_i = [v_{i-w}, v_{i-w+1}, \dots, v_i]^\top \quad (1)$$

where v_i is the vector of current invocation and w is the window length.

Now that we have the input matrix X , TICAD demands a neural network to automatically learn the relation between the input and the fact whether the current invocation is anomalous. Therefore, we propose a neural network based on both LSTM and self-attention [6]. Briefly, The same input matrix will be

learned with LSTM and self-attention neural network separately, and the output of target invocation will be concatenated to detect anomalies.

For the self-attention part, a neural network will take the input and use the multi-head scaled dot-product attention mechanism to aggregate the information. Instead of directly using the whole encoder of Transformer, we only utilize a few parts which are easy but effective. We first scale the input and add positional information to the original vector:

$$X_h = dropout(\sqrt{h}(X_i W^h) + W^{pos}) \quad (2)$$

where h is a scalar representing the hidden size. $W^h \in R^{n_f \times h}$ is the weight of the linear transformation and n_f is the number of features, i.e., the size of original invocation vector. $W^{pos} \in R^{(w+1) \times h}$ represents the learned positional embedding.

Then the X_h will be fed into the multi-head scaled dot-product attention layer, which can aggregate the information according to the attention scores. In practise, multiple heads can be calculated in parallel. The Q_i , K_i and V_i of each head, which are the indispensable elements of attention mechanism, will be transformed from the same input X_h :

$$Q_i = X W_i^Q, K_i = X W_i^K, V_i = X W_i^V \quad (3)$$

where $W^Q, W^K, W^V \in R^{h \times d_{head}}, d_{head} = h/n_{heads}$, n_{heads} is the number of heads.

For each head, scaled dot-product attention mechanism will calculate the attention scores and then get the weighted sum of values, which is shown in the following equation:

$$head_i = softmax\left(\frac{Q_i K_i^\top}{\sqrt{d_{head}}}\right) V_i \quad (4)$$

All the results of the heads will be concatenated and transformed to X'_h which is shown below:

$$X_{h'} = (head_1 \oplus head_2 \oplus \dots \oplus head_i) W^{h'} \quad (5)$$

where $W^{h'} \in R^{h \times h}$.

The final part of self-attention consists of layer normalization and residual dropout and the aggregated vector of target invocation is represented as v_s , which is shown in the following equation:

$$X_f = LayerNormalization(X_h + dropout(X_{h'})) \quad (6)$$

$$v_s = X_f[w] \quad (7)$$

For the LSTM part, we adopt a variant of LSTM called Bi-LSTM (Bidirectional Long Short Term Memory), whose detailed structure is shown in the Fig. 2. Each row of X_h will be input into the Bi-LSTM model at each time step.

As shown in the figure, $hf_w \in R^{h/2}$ and $hb_0 \in R^{h/2}$ are the hidden state vectors at the last time step, which will be concatenated to represent the result of Bi-LSTM:

$$v_l = h_w^l \oplus h_0^r \quad (8)$$

Finally, v_s and v_l will be concatenated to calculate the anomaly probability:

$$Anomaly_Probability = \sigma((v_s \oplus v_l)^T W^a + b^a) \quad (9)$$

where σ represents the sigmoid function and $W^a \in R^{2*h \times 1}$.

If a trace has at least one abnormal invocation, the trace will be judged as abnormal.

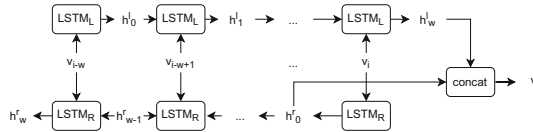


Fig. 2. The structure of LSTM based neural network

4 Evaluation

4.1 Datasets and Criteria

Datasets. To make the experiments more convincing, we use the public dataset which is proposed in TraceRCA [2] to evaluate the effectiveness of TICAD. This dataset collected traces from the Train Ticket [7] system, which is one of the largest open source microservice systems.

Baselines. To demonstrate the effectiveness of TICAD, We compare it with TraceAnomaly [3] and MEPFL-RF [8]. MEPFL-RF refers to the Random Forest version of MEPFL. Parameters of them are set best for accuracy.

Evaluation Metrics. As with previous researches, we use three evaluation metrics: *precision*, *recall* and *F1 score*, which are calculated as follows: Precision = $TP / (TP + FP)$, Recall = $TP / (TP + FN)$, F1 score = $(2 * Precision * Recall) / (Precision + Recall)$.

4.2 Preparation Experiments

For TICAD, if we directly divides all the invocations, it's likely that there will be no complete trace in the test set. This will result in the inability to compare the effectiveness of methods because TICAD can't infer trace-level results from incomplete trace invocations. Therefore, we randomly select 5% normal trace ID and 5% abnormal trace ID as the reserved ID, which means all the invocations of these traces will be reserved for test set. For supervised methods such as TICAD and MEPFL, we directly copy the abnormal traces or invocations to solve the lack of positive samples.

4.3 Experiments on Trace-Level Anomaly Detection

In this section, we use different methods to perform trace-level anomaly detection, which can show the effectiveness of different methods on whole traces. The results are shown in the Table 1. It can be seen that TICAD proposed in this paper achieved the highest F1 score(0.974) and the highest recall(0.986). Although the precision is not the highest, it doesn't lag far behind other methods. In general, TICAD shows its availability and effectiveness in anomaly detection tasks at the level of trace.

Table 1. Trace-level Anomaly Detection Results

	Precision	Recall	F1 score
TICAD	0.961	0.986	0.974
TraceAnomaly	0.901	0.159	0.270
MEPFL-RF	0.987	0.953	0.970

5 Conclusion

In this paper, we propose an end-to-end trace anomaly detection method called TICAD. It has the ability to effectively learn the sequential patterns in the invocations and then infer the anomalies in the traces. TICAD will mine the inherent relationship between the current invocation and the previous ones through a neural network based on LSTM and self-attention.

References

1. Gan, Y., Zhang, Y., Hu, K., Cheng, D., Delimitrou, C.: Seer: leveraging big data to navigate the complexity of performance debugging in cloud microservices. In: the Twenty-Fourth International Conference (2019)
2. Li, Z., Chen, J., Jiao, R., Zhao, N., Wang, Z., Zhang, S., Pei, D.: Practical root cause localization for microservice systems via trace analysis. In: IWQOS (2021)
3. Liu, P., Xu, H., Ouyang, Q., Jiao, R., Pei, D.: Unsupervised detection of microservice trace anomalies through service-level deep Bayesian networks. In: 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE) (2020)
4. Nedelkoski, S., Cardoso, J., Kao, O.: Anomaly detection and classification using distributed tracing and deep learning. In: 2019 19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (2019)
5. Nedelkoski, S., Cardoso, J., Kao, O.: Anomaly detection from system tracing data using multimodal deep learning. In: 2019 IEEE 12th International Conference on Cloud Computing (CLOUD) (2019)
6. Vaswani, A., et al.: Attention is all you need. arXiv (2017)
7. Zhou, X., et al.: Fault analysis and debugging of microservice systems: industrial survey, benchmark system, and empirical study. *IEEE Trans. Softw. Eng.* 1 (2018)
8. Zhou, X., et al.: Latent error prediction and fault localization for microservice applications by learning from system trace logs. In: 27th ACM Joint Meeting on European Software Engineering Conference (ESEC)/Symposium on the Foundations of Software Engineering (FSE) (2019)