Transformers for Log Key Anomaly Detection

Dat Quoc Ngo
Erik Jonsson School of Engineering and Computer Science
Univeristy of Texas at Dallas)
Richardson, United States
dqn170000@utdallas.edu

***Abstract*—Since the rise of Deep Learning, Recurrent Neural Network (RNN) variants for Natural Language Processing (NLP) have been extensively applied in log key anomaly detection. Inspired by the recent state-of-the-art performance of Attention-based architectures (e.g. Transformer, BERT) in many NLP tasks, we propose Transformer Encoder for anomaly detection in two tasks: next log-key prediction and fill in the gap. The results show that our Transformer Encoders underperform RNNs in log key anomaly detection. Naturally, Transformers are not designed for next log-key prediction and believed to require large scale datasets for training.**

***Index Terms*—Deep Learning, Log Key Anomaly Detection, Transformer**

I. Introduction

As systems and applications become more complex and sophisticated, they become easily vulnerable to bugs and attacks. Hence, anomaly detection becomes an essential and challenging task. Log records of system states and events have been leveraged in anomaly detection since they helped solve performance issues and perform root cause analysis. Each log entry is composed of the log key (a.k.a the unique constant print statement) and the parameter value (accompanied with the constant statement) which lead to log-key-based and parameter-value-based anomaly detection types. Our work focuses only on the log key anomaly detection.

To our best knowledge, DeepLog is the first state-of-the-art Deep Learning framework for log key anomaly detection. Based on the idea of NLP, DeepLog extracts a dictionary of log keys from incoming log entries and parses unstructured and free-text log entries into structured sequences of log keys. Stacked Long-Short-Term-Memory (LSTM), the DeepLog's core model, accept log-key sequences as inputs and predict the next log keys. The predicted log keys are expected to be normal and compared with incoming log keys from true log entries to detect abnormal execution processes [**?**].

Inspired by the state-of-the-art performance of Transformer architectures based on Attention mechanisms [**?**] in many NLP tasks, we propose to replace LSTMs in DeepLog with Transformer Encoder (TE) of BERT [**?**] to learn the representation of log keys to predict the next log key. The Attention mechanism in TE naturally neglects the sequential order of log keys and requires large-scale datasets for given training tasks to learn the log-key representation [**?**]. Hence, unlike LSTM in DeepLog which is naturally capable predicting the next log key, TE is not initially designed for the similar task. Our another approach is to reformulate the next log-key prediction task into filling in the gap. In this setting, TE is trained to reconstruct the entire log-key sequence and to predict the next log key which is masked in the input. The details of two approaches will be discussed in the next section.

II. Approach

This section is divided into following sections:

- **Log Parsing** is performed to parse the unstructured and free-text log entries into a structured and sequential format as input to Transformer Encoders.
- **Transformer Encoder**, unlike the stacked LSTM architecture in DeepLog, is not initially designed for the next log key prediction. Hence, besides simply replacing stacked LSTMs in DeepLog with Transformers, we propose to perform the Fill in Gap task that the next log key is the last missing item in the log entry sequence.
- **Experiments & Evaluation.** Two approaches were experimented on the OpenStack dataset and evaluated by two metrics: Accuracy and F-1 measure.

A. Log Parsing

Similar to DeepLog, our approach follows Natural Language Processing that a dictionary of unique log keys K={k_1, k_2,..., k_i} are extracted from log entries. Then, Spell [1], an unsupervised streaming parser based on LCS (longest common subsequence) is utilized to group or untangle log entries in concurrent processes into separate sequences of log keys.

For the next-log-key prediction task, the inputs are the *h-1* windows of recent log keys in each log-key sequence; and the targets are the next *h-th* log key in each sequence. For the fill-in-gap task, inputs share with targets the same *h* windows of log keys except the last *h-th* log key that is masked with *[MASK]* to denote the to-be-predicted next log key. Similar to BERT, each log key sequence is respectively wrapped by *CLS], [SEP]* at two ends.

B. Transformer Encoder

Both two tasks share the similar Transformer Encoder backbone in BERT [**?**] which contains N=3 Transformer blocks of Multi-Head (A=8 heads) Attention layers and H=256 Linear layers. The output block contains a single hidden linear layer with 128 nodes and the ReLU activation. For the next log-key prediction, the output is a K-length probability vector. For the fill-in-gap, the output is a $h \times K$ probability matrix. The loss function was cross-entropy for both two tasks.

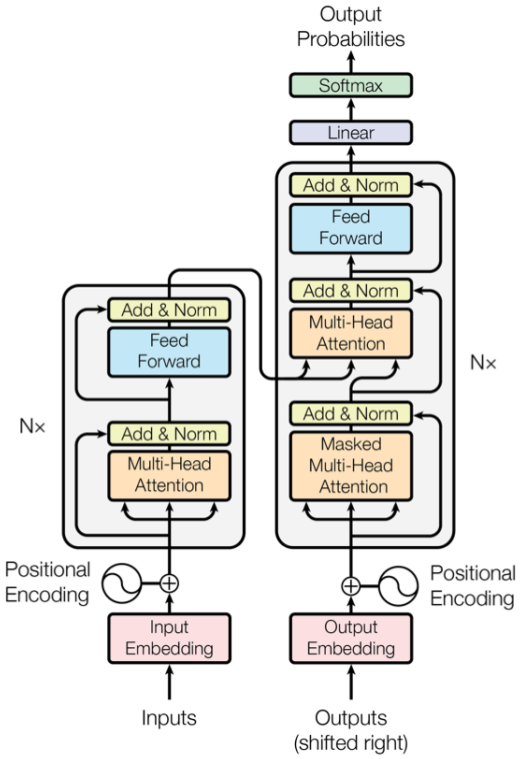

Figure 1: The Transformer - model architecture.

Fig. 1. Transformer architecture consisting of Encoder (left) and Decoder (right).

Unlike DeepLog which directly inputs raw log keys into LSTM [**?**], our approach sums the word and positional embeddings of log-key sequences as input to Transformer Encoders in Fig. 1. Since the Attention mechanism neglects the input order, the positional embedding is necessary in providing the location information of log keys to Transformer [**?**].

C. Experiments & Evaluation

Two tasks were trained and evaluated on the OpenStack dataset that has 386 sessions parsed from raw log entries, 50359 sequences of the window size *h=5*, and the vocabulary of 1156 unique log keys. Both tasks were evaluated by Accuracy and F-1 accuracy for the imbalanced log-key classes.

III. Results

TABLE I
Log Key Anomaly Detection Accuracy

	Accuracy	**F1-Measure**
DeepLog(paper)	N/A	0.98
DeepLog(reimplemented)	0.94	0.76
Next Log-Key Prediction	0.53	0.53
Fill-in-Gap	0.13	0.13

Table 1 shows accuracy and F1-measure reported by the DeepLog's authors, by a non-official re-implementation found on Github, and our two approaches. Since there exists no official implementation of DeepLog; our approaches strictly follow processing and training steps in the DeepLog re-implementation. Looking at the table 1, it is obvious that Transformers Encoders in both two tasks underperformed LSTMs in DeepLog with accuracies at 53% and 13%. Similarly, the F-1 measures of two Transformer Encoders are lower than those of DeepLogs with values at 53% and 13%.

In the DeepLog re-implementation, there exists a sustainable gap between accuracy and F1-measure (94% and 76%) which is likely due to the class imbalance. Also, there exists the difference in F1-measure between the DeepLog original report and re-implementation (98% and 76%). This observation is likely due to different ratios of the OpenStack dataset used for training in two implementations that we could not find the official DeepLog implementation.

IV. Discussion

The lower performance of Transformer Encoder in two tasks than LSTMs in DeepLog is unexpected. LSTMs in DeepLog are naturally designed to recurrently learn the log key representation in a specific direction [**?**] that is capable of sequentially predicting the next log key. On the other hand, the Attention mechanism in Transformer naturally ignores the input order that requires the additional location information given by the positional embedding [**?**]. Regardless of the additional positional embedding, Transformer Encoder has been extensively utilized for classification problems rather than next-item prediction.

To force Transformer Encoder to learn the input representation in sequential orders, BERT is pre-trained on two tasks, Fill-in-Gap and Next-Sentence-Prediction which requires large-scale datasets (i.e. BookCorpus of 74M sentences [**?**]). Our second proposed task, Fill-in-Gap, is inspired by BERT's pretraining tasks. However, the scale of the OpenStack dataset is smaller than that of BookCorpus (50K sequences) which likely leads to the least performance of Fill-in-Gap in Fig. 1.

References

[1] M. Du and F. Li, "Spell: Streaming Parsing of System Event Logs," Proc. IEEE International Conference on Data Mining (ICDM), pp. 859-864, 2016.
[2] M. Du, F. Li, G. Zheng, V. Srikumar, " DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning," 2016.
[3] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, J. Jones, A. Gomez, L. Kaiser, I. Polosukhin, "Attention Is All You Need," 31st Conference on Neural Information Processing System (NIPS), 2017.
[4] J. Devlin, M. Chang, K. Lee, K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," Conference of the North American Chapter of the Association for Computational Linguistics (NAACL), 2019.
[5] S. Hochreiter, and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, pp.. 1735-1780, 1997.