**LETTER**

An Attention-Based GRU Network for Anomaly Detection from System Logs

Yixi XIE†, Lixin JI†, Nonmembers, and Xiaotao CHENG†, Member

**SUMMARY**  System logs record system states and significant events at various critical points to help debug performance issues and failures. Therefore, the rapid and accurate detection of the system log is crucial to the security and stability of the system. In this paper, proposed is a novel attention-based neural network model, which would learn log patterns from normal execution. Concretely, our model adopts a GRU module with attention mechanism to extract the comprehensive and intricate correlations and patterns embedded in a sequence of log entries. Experimental results demonstrate that our proposed approach is effective and achieve better performance than conventional methods.

**key words:** anomaly detection, GRU, attention-based model

1. Introduction

In recent years, the issue of anomaly detection from system logs has been a research hotspot of anomaly detection field [1]. As unstructured data, system logs are closely combined with text mining, statistics, machine learning and other domains [9]. Existing approaches are proposed, such as PCA based approaches over log message counters [2], invariant mining based methods to capture co-occurrence patterns between different log keys [3], and workflow based methods to identify execution anomalies in program logic flows [4]. Even though they are successful in certain scenarios, none of them is effective as a universal anomaly detection method.

In recent years, deep learning has been developing rapidly and constantly creating new application patterns, especially in the field of NLP (natural language processing). In this context, Du [5] considers system logs as natural language sequences and proposes DeepLog, a deep neural network model based on LSTM, which can learns log rules from normal log data. The experimental results show that DeepLog achieves high detection accuracy on several large data sets, and its overall performance is better than other log anomaly detection methods based on traditional data mining. However, due to the large amount of model parameters, the training of DeepLog is time-consuming. Hence, its efficiency needs to be improved.

To reduce the number of model parameters and improve the performance of anomaly detection, in this paper, we propose a novel attention-based GRU neural network model (ATT-GRU) for anomaly detection. Specifically, ATT-GRU first utilizes independent Gate Recurrent Unit (GRU) networks [7] to learn a model of log patterns from normal execution and flag deviations from normal system execution as anomalies, which has fewer parameters than DeepLog. Then an attention mechanism is integrated with GRU networks to improve the weights of meaningful log keys for achieving better performance. We evaluate ATT-GRU on several large system log data. Experimental results indicate that our proposed approach is effective and outperforms the conventional methods.

2. Methodology

Inspired by [5], we transform unstructured, free-text log entries into a structured representation. As shown by several prior work, we split every log entry into two parts, i.e. the string constant and variable. The string constant refers to the content from the print statement in the source code. For instance, in the Mysql log “InnoDB: Initializing buffer, size = 512.0M”, the string constant part is “InnoDB: Initializing buffer, size =”, and “512.0M” is the variable part. We consider the string constant as the type of log entry which is called log key [5]. The output of normal logs will follow a certain flow path and sequence, which is called execution path. As Log key sequence can represent the execution path of logs, extracting log key from log entries would be an effective log parsing method [6]. In our work, we use Spell [6] to parse each log entry. Besides, the variable part called parameters would be a valuable piece of information in log entries which reflects the performance of the system, and can be used as identifiers for some specific execution sequences.

Concretely, let $K = \{k_1, k_2, \ldots, k_n\}$ be the set of distinct log keys from a log-producing system, which is called log key vocabulary. For a certain log key sequence, the value $w_i$ at time $t$ would be strongly dependent on the most recent log keys that appeared prior to it. Therefore, anomaly detection can be considered as a multi-class classification problem, where each distinct log key defines a single class. To figure out the problem, we proposed an ATT-GRU model as a classifier. The input of ATT-GRU is a history of recent log keys $w = [w_{t-1}, w_{t-2}, \ldots, w_{t-l}]$, and the output is a probability distribution over each log key in $K$, i.e. $P(y_t = x_i | w)$, which describes the probability for each log key from $K$ to appear as the next log key value given the history. Our approach is to sort the possible log keys based on $P(y_t = x_i | w)$ as a candidate queue, and if the next log key $w_i$ is in the...
top $m$ of the queue, we would treat it as normal. Otherwise, the log key would be flagged as abnormal. Noted that $m$ is a hyper-parameter set by users. Figure 1 gives the overall architecture of proposed ATT-GRU.

2.1 Execution Path Anomaly Detection

As shown by several prior work, log data can be regarded as a special natural language. Log key sequence is equivalent to a natural sentence, and each log key in the sequence would be considered as a single world. As a consequence, in a log key sequence, the current log key $w_t$ would depend on the former log keys just like human language, which means the potential dependency relationships exist in the context of log key sequence. Therefore we can model log anomaly detection problem according to the method of language modeling. Inspired by [5], we design and implements a neural language model based on GRU with attention mechanism, which can detect anomalies through long-term sequence dependencies. Different from [5], we use GRU instead of LSTM. GRU is similar to LSTM with simpler inner structure and parameters, which exhibits a higher training efficiency than LSTM.

In our model, we adopt independent GRU network with attention to capture the potentially non-linear and high dimensional dependencies among log key sequence. Supposed that given a history log key sequence $w = \{w_t, h_{t-1}, w_{t-2}, \ldots, w_{t-1}\}$, and the propose is calculate the current log key $w_t$. As depicted in Fig. 1, in the input layer, every log key $w_t$ in $W$ is mapped to a $d$-dimensional vector $w_t \in \mathbb{R}^d$ via a log key embedding matrix. Then we feed these embeddings into a GRU networks module [7].

GRU network is a famous variant of LSTM, which synthesizes the forgetting gate and input gate to a single update gate and also mixes cell state and hidden state. So, the final GRU network is simpler and faster than the standard LSTM network. Especially when training large log data sets, it can save a lot of time with small performance difference from that of standard LSTM networks. The main mechanism of GRU is to adopt various gates to decide the degree of information that GRU should keep and memorize in long-term transmission. The internal structure of GRU model is shown in Fig. 2, where $z_t$ represents update gate and $r_t$ represents reset gate.

At time $t$, the GRU calculates the new state as:

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$$

(1)

This is to compute a linear interpolation between the previous state $h_{t-1}$ and the current candidate state $h'_t$ with the new sequence information. The update gate $z_t$ decides to keep how much past information and to add how much new information. It controls the extent to which the information of the previous state is brought into the current state. The larger the value of $z_t$, the more information of the previous state is brought in. The state of $z_t$ is updated as:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z)$$

(2)

where $x_t$ is the sample vector at time $t$, $h'_t$ is the candidate state computed in the same way as the hidden layer of traditional RNN network:

$$h'_t = \tanh(w_h x_t + r_t \odot (u_h h_{t-1}) + b_h)$$

(3)

where $r_t$ denotes a reset gate which controls how much the previous state contributes to the current candidate state $h'_t$. The smaller the $r_t$ value, the smaller the contribution from the previous state. If $r_t = 0$, it will forget the previous state. The reset gate is updated as

$$r_t = \sigma(W_r x_t + u_r h_{t-1} + b_r)$$

(4)

The model is therefore able to exploit information from the history, in which the architecture is shown in Fig. 3.

Then an average-pooling operator runs over all the hidden states $h_t$ to obtain the log key sequence vector $W$. It is observed that not all log keys in a sequence contribute equally to the semantic meaning for the current log key $w_t$. Therefore, we introduce attention mechanism [8] to achieve more beneficial semantic sequence vectors through improving the weights of meaningful log keys, which would enhance the contribution of important log keys. Given the hidden states $\{h_1, h_2, \ldots, h_d\}$ corresponding to the log keys in
sequence $W$, the representation vector $W$ is obtained as follow.

$$W = \frac{1}{n} \sum_{i=1}^{t} \alpha_i h_i$$  

(5)

where $\alpha_i$ are the attention weights, which are defined as:

$$\alpha_i = \frac{\exp(e(h_i))}{\sum_j \exp(e(h_j))}$$  

(6)

In Eq. (6), $e(\cdot)$ is a measure function which reflects the relevance between each log key and corresponding the purpose vector. Inspired by [8], $e(\cdot)$ is defined as:

$$e(h_i) = h_i \cdot v \cdot P$$  

(7)

where $v$ is a weighted diagonal matrix and $P$ is the representation vector of the specific purpose which would be learned in training phrase. Afterwards, we would acquire the representations of each log key sequence. Finally, a softmax layer follows to perform the specific classification task.

2.2 Parameter Value Anomaly Detection

The log key sequence is useful for detecting execution path anomalies. However, some anomalies are not shown as a deviation from a normal execution path, but as an irregular parameter value [5]. In this subsection, we using ATT-GRU to detect parameter value anomaly.

As mentioned above, each log entry can be parsed into log key and parameters. Since the interval time between two log entries would measure the performance of system, in this paper we adopt the interval between two adjacent timestamps and parameters parsed from log keys to construct a parameter value vector. These parameter value vectors for a same log key would form a parameter value vector sequence, which would be input into our ATT-GRU model. In this way, we convert the parameter value anomaly detection to a time series prediction problem. By comparing the predicted value with the observed value, we can estimate whether the system is abnormal.

We adopt ATT-GRU to model a multi-variate time series data, with following adjustments. Note that a separate ATT-GRU network is built for the parameter vector sequence of each distinct log key. In the input layer, we firstly normalize the values in each vector by the average and the standard deviation of all values from the same position from the training data. Then the input at each time step is simply the preprocessed parameter value vector from that timestamp. The output is a real value vector as a prediction for the next parameter value vector, based on a sequence of parameter value vectors from recent history. During training phrase, mean square loss is used to minimize the error between a prediction and an observed value in order to adjust the weights of model.

In detecting mode, if the error between a prediction and an observed value vector is within a high-level of confidence interval of the above Gaussian distribution, the current parameter value vector is considered normal, otherwise is considered abnormal.

3. Experiments

Extensive experiments have been carried out to verify that our proposed ATT-GRU model is effective for anomaly detection, which consist of two parts: execution path anomaly detection and parameter value anomaly detection. In our work, ATT-GRU is implemented using Keras with Tensorflow as the backend.

3.1 Execution Path Anomaly Detection

We compare our approach against following widely applied methods, which are PCA (principal component analysis) [2], IM (invariant mining) [3], DeepLog [5], GRU-only and LogGAN [10]. In this section, we evaluate our model comprehensively through comparing the precision and running velocity. HDFS log dataset [2] which is generated through running Hadoop-based jobs on 203 Amazon’s EC2 nodes is adopt to perform the experiments. We group dataset into 575,062 sessions by block_id. Considering the duplication of data, we choose 1% normal sessions as training data for ATT-GRU and DeepLog, and adopt whole dataset as test data. The hyper-parameter history window size $h$ is set to be 10.

Table 1 demonstrates the results of ATT-GRU and other five methods on HDFS. We can find that although PCA has the highest accuracy, its F1 and Recall are much lower than other methods. Compared to other five methods, ATT-GRU obtains a higher F1 than others, and acquires about the same accuracy with DeepLog and LogGAN. According to the comparison with GRU-only, it is illustrated that the attention mechanism could obtain a considerable improvement on performance. This is because when the attention mechanism is adopted, the weights of meaningful log keys in anomaly detection are enhanced against other trivial parameters. In addition, Table 2 compares the running speed of DeepLog, LogGAN and ATT-GRU. It is indicated that due to fewer parameters, our ATT-GRU is faster than DeepLog or LogGAN.

| #log entry | Total Time (s) | Average Time (s) |
|------------|----------------|------------------|
| DeepLog    | 11,757,629     | 11,399           | 1.02             |
| LogGAN     | 11,757,629     | 12,508           | 1.12             |
| ATT-GRU    | 31,375,829     | 9,858            | 0.86             |

Table 1: The performance of anomaly detection for HDFS log datasets.

| Method    | Accuracy(%) | Recall(%) | F1(%) |
|-----------|-------------|-----------|-------|
| PCA       | 97.6        | 67.8      | 80.0  |
| IM        | 88.0        | 92.7      | 90.3  |
| DeepLog   | 96.1        | 98.4      | 96.2  |
| GRU-only  | 96.0        | 98.2      | 96.1  |
| LogGAN    | 96.4        | 96.3      | 96.3  |
| ATT-GRU   | 96.5        | 98.3      | 96.7  |
3.2 Parameter Value Anomaly Detection

For parameter value anomaly detection, the client log data of SmartBI is adopted as experiment dataset. We simulate a scenario that the system is under attack through controlling network speed. As results, 49,287 client log entries are collected. Figure 4 shows partial parameter value vectors parsed from log entries. In experiment, parameter value vectors are grouped into different datasets by log keys, and each dataset would be separated into training, validation and testing sets in the proportion of 1 : 1 : 1. The confidence interval P-value is chosen to be 98%.

In the experiment, our ATT-GRU can recognize all the abnormal log entry. Figure 5 demonstrates anomaly detection results of partial log keys. Concretely, in the left line chart, three outliers are identified, which are all caused by the long run time of system. In the right line chart, no abnormal values are recognized by ATT-GRU, which is because log key No.17 does not involve communication between client and server.

In conclusion, proposed ATT-GRU is effective for execution path anomaly detection and parameter value anomaly detection.

4. Conclusion

In this paper, we propose a novel neural network model for anomaly detection, ATT-GRU, which realizes automatically learning the comprehensive and intricate correlations and patterns embedded in a sequence of log entries. Specifically, ATT-GRU utilizes independent GRU networks with attention to improve the weights of meaningful log keys. Experimental results indicate that our approach is effective in execution path anomaly detection and parameter value anomaly detection, and outperforms conventional methods.

Acknowledgments

This work is supported by the National Key Research and Development Program of China (2018YFB0804004), the Foundation for Innovative Research Groups of the National Natural Science Foundation of China (61521003), the Foundation of the National Natural Science Foundation of China (61602509).

References

[1] T.B. Adhi and R. Kyung-Hyune, “HFSTE: Hybrid feature selections and tree-based classifiers ensemble for intrusion detection system,” IEICE Trans. Inf. & Syst., vol.E100-D, no.8, pp.1729–1737, 2017.
[2] W. Xu, L. Huang, A. Fox, D. Patterson, and M.I. Jordan, “Detecting large-scale system problems by mining console logs,” Proc. ACM Symposium on Operating Systems Principles, pp.117–132, 2009.
[3] J.-G. Lou, Q. Fu, S. Yang, J. Li, and B. Wu, “Mining program workflow from interleaved traces,” Proc. ACM SIGKDD, 2010.
[4] X. Yu, P. Joshi, J. Xu, et al., “CloudSeer: Workflow monitoring of cloud infrastructures via interleaved logs,” Proc. ACM International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), pp.489–502, 2016.
[5] M. Du, F. Li, G. Zheng, and V. Srikumar, “Deeplog: anomaly detection and diagnosis from system logs through deep learning,” 2017 ACM SIGSAC Conference on Computer and Communications Security, pp.1285–1298, 2017.
[6] M. Du and F. Li, “Spell: Streaming Parsing of System Event Logs,” Proc. IEEE International Conference on Data Mining, pp.859–864, 2016.
[7] G. Cheng, P. Zhang, and J. Xu, “Automatic Speech Recognition System with Output-Gate Projected Gated Recurrent Unit,” IEICE Trans. Inf. & Syst., vol.E102-D, no.2, pp.355–363, 2019.
[8] Y. Lin, S. Shen, Z. Liu, H. Luan, and M. Sun, “Neural relation extraction with selective attention over instances,” Proc. ACL, pp.2124–2133, 2016.
[9] K. Limthong, K. Fukuda, Y. Ji, and S. Yamada, “Unsupervised Learning Model for Real-Time Anomaly Detection in Computer Networks,” IEICE Trans. Inf. & Syst., vol.E97-D, no.8, pp.2084–2094, 2014.
[10] B. Xia, J. Yin, J. Xu, and Y. Li, “LogGAN: A Sequence-Based Generative Adversarial Network for Anomaly Detection Based on System Logs,” Science of Cyber Security, Lecture Notes in Computer Science, Springer, 2019.