Tomaž Martinčič
Šklendrovec 7b, 1414 Podkum, Slovenija
Study programme: Computer and Data Science, MAG
Enrollment number: 63160211

Committee for Student Affairs
Univerza v Ljubljani, Fakulteta za računalništvo in informatiko
Večna pot 113, 1000 Ljubljana

The master's thesis topic proposal
Candidate: Tomaž Martinčič

I, Tomaž Martinčič, a student of the 2nd cycle study programme at the Faculty of computer and information science, am submitting a thesis topic proposal to be considered by the Committee for Student Affairs with the following title:

Slovenian: **Samonadzorovano odkrivanje anomalij v produkcijskih dnevniških zapisih**
English: **Self-supervised anomaly detection in production log streams**

This topic was already approved last year: ***NO***

I declare that the mentors listed below have approved the submission of the thesis topic proposal described in the remainder of this document.

I would like to write the thesis in English with the following reason: I have a foreign co-mentor who does not speak Slovenian.

I propose the following mentor:

Name and surname, title: Slavko Žitnik, doc. dr.
Institution: Univerza v Ljubljani, Fakulteta za računalništvo in informatiko
E-mail: slavko.zitnik@fri.uni-lj.si

I propose the following co-mentor:

Name and surname, title: Álvaro García Faura, M.Sc.
Institution: XLAB d. o. o.
E-mail: alvaro.garcia.faura@xlab.si

Ljubljana, 4. december 2021.

Proposal of the masters thesis topic

1 The narrow field of the thesis topic

Natural language processing, anomaly detection, machine learning

2 Key-words

NLP, anomaly detection, production logs, self-supervised machine learning, AIOps

3 Detailed thesis proposal

Past approvements of the proposed thesis topic:
The proposed thesis has not been submitted nor approved in previous years.

3.1 Introduction and problem formulation

Computer-generated log messages are a very valuable source of information to represent the current status of a system or application. Log messages are precisely generated to provide application developers and system operators with information that could help them understand execution paths, find bugs or solve incidents. Generally speaking, when a problem occurs, logs are often relied upon for investigation.

In an operational phase, the automatic analysis of these logs could thus provide valuable insights regarding the current and past status of the monitored assets, but the analyses are nevertheless still mostly performed manually. Many research works have tackled this problem recently, but there are still open research questions and further improvements possible, especially by inclusion of individual log-entry semantics and its context. Furthermore, in most cases, current approaches are developed in a closed-world setting and anomalies are usually not reported per log-entry, thus their application to real-world use cases is limited.

Traditionally, to automatically detect certain security threats, popular frameworks such as Wazuh[1] collect log data and use a rule-based approach to match them to existing known patterns. The validity of this approach is only supported by the handcrafted rule set,

[1] `https://wazuh.com/`

which is certainly limited to known threats. Similarly, for general-purpose production-ready log anomaly detection, ML tools from ELK stack (Logstash, Elastic, Kibana) are commonly used[2] which only focus on numerical log parameters or aggregations (e.g., windowed log counts) and thus translates the problem to anomaly detection on time-series data.

Alternatively, we propose to model the stream log data as a language, and thus leverage the advances in the field of NLP seen in the recent years, by including the individual log-entry semantics, as well as its context, captured in the preceding sequences of logs. The proposed solution captures the normal behavior of whatever system or application is being monitored in a self-supervised fashion, thus enabling the possibility of detecting deviations into abnormal patterns that may potentially be the symptom of an underlying security or any other abnormality incident, with labeled data only needed for the evaluation purposes.

3.2 Related work

There are a few research works that tackle the problem of anomaly detection in logs. Each of them has shortcomings that prevent them from being widely used in a real-world production setting. Below, we present a high-level overview of those methods and expose the biggest constraints.

Most of the approaches usually divide the process into three main steps, namely, (i) log preprocessing and log template mining, (ii) representation of the individual log messages/templates, or the sequences of logs/templates as vectors in a multi-dimensional space (embeddings), which are chronologically stacked and used for (iii) anomaly detection.

Different approaches are used to preprocess log message and transform them into normalized representation. Some of them transform the raw log messages into normalized representation (log templates) by detecting and masking parameters [1, 2, 3]. The parameters are sometimes additionally used as supplementary information for anomaly detection. Log templates are represented as a categorical variable or additionally encoded to capture semantic meaning with well-known methods from natural language processing like Word2Vec [4] or BERT [5]. Semantic embeddings are in some methods obtained from raw log messages without prior template extraction [6].

Training of log anomaly detection models is in the literature most often done in a supervised [7, 8, 6] or self-supervised [9, 10, 11] fashion. It is challenging to incorporate all of the necessary functionalities when using self-supervised learning. One of the biggest problems is dealing with new log messages not seen in a training phase. However, it is difficult to expect labeled training data when applying a model to a production environment.

[2] `https://www.elastic.co/guide/en/machine-learning/current/xpack-ml.html`

Therefore, it is still preferred to go with a self-supervised approaches.

The traditional methods are predominately based on LSTM networks [9, 10]. The methods presented in [9, 10] are self-supervised and utilize a candidate set approach for anomaly detection. In the candidate set approach, all of the known log keys are sorted based on the probability of it occurring as the next log key in a sequence. The next log key is marked as normal if it is among the top N candidates. Otherwise, it is flagged as being abnormal. In DeepLog method [9] they also utilize the parameters that are extracted with Spell [1] preprocessing method, but the method ignores the semantic meaning of the log templates. Contrary, in LogAnomaly method [10] the model ignores parameters, but the log templates embeddings preserve semantic meaning. The LogRobust method [7] introduces an architectural change, describing a Bidirectional LSTM network to capture information from sequences in both directions. The method tackles the problem of anomaly detection as a binary classification methodm but the training is supervised, therefore requiring labeled data.

Recently, work has been presented that utilizes Transformer [12] architectures from the NLP domain [8, 6, 11]. The methods presented in [8, 6] are trained in a supervised fashion, dealing with the problem as a binary classification. The HitAnomaly method [8] semantically embeds log messages, which are then combined into a sequence of log messages. It also makes use of the parameter values, but again partially uses supervised objectives. The NeuralLog method [6] represents a similar approach, but it ignores the parameters, while the LogBERT method [11] cannot handle parameter values nor does it semantically embed the log messages. It is also not capable of handling new log templates without retraining.

3.3 Expected contributions

Our main contribution will be the development of a method for the detection of anomalies in computer-generated log sequences. The method will (i) jointly model both, the sequence, as well as individual log templates with representations that preserve semantic meaning, in a self-supervised fashion. This significantly differs from the current self-supervised approaches that only model the (a) sequence of log templates [11] or (b) individual log templates [13]. This complete representation will also enable (ii) handling of log templates not seen in the training phase, which besides supervised objectives of the current approaches [8], represents one of the crucial limitations of the current approaches to be deployed in a real-world setting.

3.4 Methodology

The log anomaly detection pipeline consists out of preprocessing and anomaly detection parts. The preprocessing represents an important part and can significantly influence the performance. We will use established approaches in the literature (e.g. Drain[2]), which are also used by the competing methods [7, 8, 11]. The preprocessing transforms the raw log data into the normalized representation (log templates), which will be later on used in the proposed method to semantically embed them using the pretrained Transformer-based language models [4, 5]. A chronological sequence of log template embeddings will then be separately modeled to capture the temporal dependencies. Finally, this model will be used to compute anomaly score on per log basis.

The method will be tested on multiple open datasets [14], which are widely used for benchmarking current state-of-the-art log anomaly detection methods. Each dataset contains millions of logs with per log anomaly annotations, thus making the evaluation possible in terms of established performance metrics (e.g. F1 score). The competing approaches presented in Section 3.2 are evaluated on those datasets, thus making a comparison in terms of anomaly detection performance possible. There is currently no published work that would demonstrate the handling of unseen log entry data during the deployment of the system, which is crucial for real-world operation (e.g. system upgrades, new modules), thus we also plan to perform closed evaluation of our method on the real-world proprietary log datasets provided by XLAB.

3.5 References

[1] M. Du, F. Li, Spell: Streaming parsing of system event logs, in: 2016 IEEE 16th International Conference on Data Mining (ICDM), IEEE, 2016, pp. 859–864.

[2] P. He, J. Zhu, Z. Zheng, M. R. Lyu, Drain: An online log parsing approach with fixed depth tree, in: 2017 IEEE international conference on web services (ICWS), IEEE, 2017, pp. 33–40.

[3] S. Zhang, W. Meng, J. Bu, S. Yang, Y. Liu, D. Pei, J. Xu, Y. Chen, H. Dong, X. Qu, et al., Syslog processing for switch failure diagnosis and prediction in datacenter networks, in: 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS), IEEE, 2017, pp. 1–10.

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

[5] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[6] V.-H. Le, H. Zhang, Log-based anomaly detection without log parsing, arXiv preprint arXiv:2108.01955 (2021).

[7] X. Zhang, Y. Xu, Q. Lin, B. Qiao, H. Zhang, Y. Dang, C. Xie, X. Yang, Q. Cheng, Z. Li, 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, 2019, pp. 807–817.

[8] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, Z. Luan, Hitanomaly: Hierarchical transformers for anomaly detection in system log, IEEE Transactions on Network and Service Management 17 (4) (2020) 2064–2076. doi:10.1109/TNSM.2020.3034647.

[9] M. Du, F. Li, G. Zheng, V. 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, 2017, pp. 1285–1298.

[10] W. Meng, Y. Liu, Y. Zhu, S. Zhang, D. Pei, Y. Liu, Y. Chen, R. Zhang, S. Tao, P. Sun, et al., Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs., in: IJCAI, Vol. 19, 2019, pp. 4739–4745.

[11] H. Guo, S. Yuan, X. Wu, Logbert: Log anomaly detection via bert (2021). arXiv:2103.04475.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[13] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, O. Kao, Self-attentive classification-based anomaly detection in unstructured logs, in: 2020 IEEE International Conference on Data Mining (ICDM), IEEE, 2020, pp. 1196–1201.

[14] A. Oliner, J. Stearley, What supercomputers say: A study of five system logs, in: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), 2007, pp. 575–584. doi:10.1109/DSN.2007.103.