2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) | 978-1-6654-0337-5/21/$31.00 ©2021 IEEE | DOI: 10.1109/ASE51524.2021.9678815

Log Anomaly to Resolution: AI Based Proactive Incident Remediation

Ruchi Mahindru
IBM T.J. Watson Research Center
Yorktown Heights, NY, USA
rmahindr@us.ibm.com

Harshit Kumar
IBM Research
New Delhi, India
harshitk@in.ibm.com

Sahil Bansal
IBM Research
New Delhi, India
sahilbansalkkr@gmail.com

Abstract—**Based on 2020 SRE report, 80% of SREs work on post-mortem analysis of incidents due to lack of provided information and 16% of toil come from investigating false positives/negatives. As a cloud service provider, the desire is to proactively identify signals that can help reduce outages and/or reduce the mean time to resolution. By leveraging AI for Operations (AIOps), this work proposes a novel methodology for proactive identification of log anomalies and its resolutions by sifting through the log lines. Typically, relevant information to retrieve resolutions corresponding to logs is spread across multiple heterogeneous corpora that exist in silos, namely historical ticket data, historical log data, and symptom resolution available in product documentation, for example. In this paper, we focus on augmented dataset preparation from multiple heterogeneous corpora, metadata selection and prediction, and finally, using these elements during run-time to retrieve contextual resolutions for signals triggered via logs. For early evaluation, we used logs from a production middleware application server, predicted log anomalies and their resolutions, and conducted qualitative evaluation with subject matter experts; the accuracy of metadata prediction and resolution retrieval are 78.57% and 65.7%, respectively.**

Index Terms—**incident triage, log anomaly detection, cloud, aiops**

I. Introduction

In general, when an outage is reported due to service failure, leading to alerts, anomalies, and/or the like, the role of the Site Reliability Engineer (SRE) is to ensure that the system returns to normal state as quickly as possible. The SREs would sift through hundreds or thousands of log lines to identify anomalous log lines, which is a tedious and time consuming process. Based on the anomalous log lines, SREs would manually formulate a query to retrieve resolutions from similar historical incidents [1], [2]. In AIOps, training a log anomaly pipeline require a huge amount of normal log data corresponding to normal/healthy state of the system. It may be the case that the client has the data, but it is not certain if the data is normal or abnormal or both.

This paper addresses the aforementioned problems that SRE faces: a) it proposes a system that detects a handful of log anomalies from hundreds and thousands of unlabeled log lines(normal or abnormal), (b) use them to automatically formulate a query to return resolutions. Compared to manual approach of sifting through log lines to identify anomalous log lines and then manually formulating a query to search for resolution, the proposed system not only reduces mean time to detect but also reduces mean time to resolution. That is, the proposed approach, Log Anomaly to Resolution (LA2R), is predictive in nature; it predicts anomalies from unlabeled logs which are available in real-time and maps them to the resolution URLs (i.e. documents containing resolution steps for problem diagnosis).

In a cloud environment, the various sources of signals are leveraged to triage an outage. As mentioned earlier, logs contain the signals that can be leveraged by the SREs for incident diagnosis and resolution. One of the challenges with logs is that they are noisy and voluminous, finding anomalous log line is akin to finding a needle in the haystack. The Log Anomaly Detection (LAD) algorithm analyze the underlying logs to predict anomalies. One of the initial steps in detecting log anomalies is parsing the logs to generate the templates [3], and the different methods include PCA [4], Invariant mining [5], [6], Log Clustering [7], [8], Neural Granger Causality [9] and Deep Learning based approaches [10]–[15]. The above cited approaches require labeled data over a long period of time, i.e. data corresponding to *normal* (healthy state), while the proposed LAD method does not impose such a requirement; it can identify log anomalies from unlabeled log data which contains both normal and abnormal data.

Proactive incident triage involves mapping the detected log anomalies to their resolutions. Rafiul et al. [16] provide an automated method for detecting log anomalies from a class of anomalies that are defined by normal values of key metrics and provide a resolution framework for such a class of metrics. Our approach, on the other hand, is generic and does not limit to a pre-defined class of anomalies or metrics. Fig 1 introduces the high level phases of LA2R and its functions:

1) Phase1 - Bootstrapping at day 0 constitutes 1) prepare augmented dataset from several heterogeneous datasets (Section II), 2) Log Anomaly Detection (Section III), and 3) Metadata Selection and Prediction Model (Section III).
2) Phase2 - System usage during steady state operations at day 1+ consisting of Act, Explain and Learn i.e. using the Log anomaly and Metadata Prediction model for query expansion, resolution retrieval, ranking and explainability (Section III). Finally, learning from system usage for incremental updates and fine-tuning (Section V).

Other major contributions include a user study to evaluate the

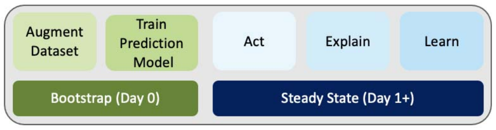


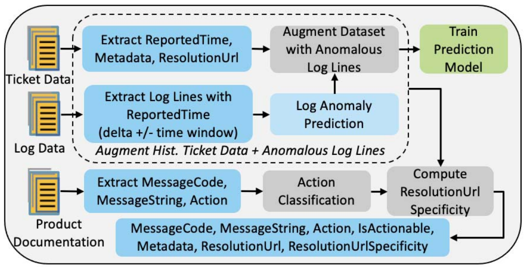
Fig. 1. Incident Triage using Log Anomaly to Resolution

metadata prediction and resolution retrieval (Section IV).

II. Dataset

We leverage information spread across three different datasets for an application server product: 1) Historical Ticket Data (including incidents), 2) Historical Log Data, and 3) Symptom / Resolutions available in standard Product Documents. There are several challenges to use these datasets for LA2R. Firstly, each dataset may exist in silos, each containing important information nuggets. For example, user reported incident *Description*, *ReportedTime*, *Product*, *ProductVersion*, *Category*, *Sub-Category*, *Severity* etc. are available in ticket data, while Product Documentation contains the *description* and official *Resolution*, and the symptoms (in form of log lines) from run-time are available in the log data. Such information nuggets when combined, provide a more holistic view of the incident, its context, and cognition process of the SRE that may have been involved in the resolution derivation. Secondly, log lines could run into hundreds or millions per system/application/component. Identifying specific log lines associated with the incident in historical ticket data is a challenging task. Algorithm below provides steps to prepare an augmented dataset, see Fig 2.

1) For each *incident*, extract the *ReportedTime*, *Metadata* (e.g., *Category*, *Sub-Category*, *Product*, *ProductVersion*, *Severity*), *ResolutionUrl*, where *ResolutionUrl* is a SRE identified resolution related to the symptom reported / detected.
2) For each *incident*, check if the corresponding logs for application/system/component in question are available.
   a) Use the *ReportedTime* from step 1 above to identify the time window within the +/- delta of the *ReportedTime* to extract logs of interest (Note: This may still be thousands of log lines).
   b) Run the extracted log lines through the unsupervised LAD model (Section III) to identify anomalous log lines (Output may be a couple of anomalous log lines).
3) Extract *MessageCode*, *MessageString*, and *Resolution* from the Product Documentation; using a rule-based approach, classify the *Resolution* as actionable or not. A new metadata attribute, *IsActionable* is set to **Y**, if the *Resolution* contains steps that can be acted upon by an SRE; otherwise, it is set to **N**.
4) Combine data extracted in Step 1, Step 2(b) and Step 3 using the common link which is *MessageCode*.
5) We observed that there could be several *ResolutionUrl*s for the same *MessageCode*. Using frequency distribution of *MessageCode*s for each *ResolutionUrl*, we introduce a field *ResolutionSpecificityScore(RSS)* that indicates the


Ticket Data
Extract ReportedTime, Metadata, ResolutionUrl
Augment Dataset with Anomalous Log Lines
Train Prediction Model
Log Data
Extract Log Lines with ReportedTime (delta +/- time window)
Log Anomaly Prediction
Augment Hist. Ticket Data + Anomalous Log Lines
Product Documentation
Extract MessageCode, MessageString, Action
Action Classification
Compute ResolutionUrl Specificity
MessageCode, MessageString, Action, IsActionable, Metadata, ResolutionUrl, ResolutionUrlSpecificity
Fig. 2. Architectural Diagram for Augmented Knowledge Base Creation

confidence on the *ResolutionUrl* for each incident. Assume that the frequency of *ResolutionUrl* u_i for $MessageCode_j$ (MC_j) is c_{ij}, then *RSS* is calculated as follows,

$$RSS = \frac{c_{ij}}{\sum_j c_j} * log \frac{|MC|}{|MC_j \in MC : url_i \in MC_j|} \tag{1}$$

where MC is a set of all *MessageCode*s in the dataset. The first term, $\frac{c_{ij}}{\sum_j c_j}$, captures the importance of the *ResolutionUrl* u_i for $MessageCode_j$ (MC_j). The second term, $\frac{|MC|}{|MC_j \in MC : url_i \in MC_j|}$, captures the importance of the $MessageCode_j$ (MC_j) for the *ResolutionUrl* u_i.

III. Methodology : Log Anomaly to Resolution

This section describes the end-to-end run-time for LA2R, see Fig 3. Given incoming log lines from a log monitoring system (LogDNA), the system predicts log anomalies using the **LAD** model (explained below). For each log anomaly, the **Query Processing** extracts the *MessageCode* and masks the parameter values to remove any environment specific information. The **Metadata Prediction** predicts the metadata, such as *Category* and *Sub-Category*. The **Query Expansion** formulates a query consisting of predicted *Metadata* and *MessageCode*, if *MessageCode* is not present then it uses masked anomalous log line. Execute the query on the **Augmented Knowledge Base** to retrieve the top k results, re-ranked using the *ResolutionSpecificityScore*. To provide **Explainability** for each result, we expose predicted Metadata, along with the contextual Metadata, e.g. *Product*, *ProductVersion*, *Severity* sorted by min entropy and max correlation scores. Finally, leveraging the system usage logs, the **Feedback Learning** component would learn by collecting explicit feedback or implicit feedback.

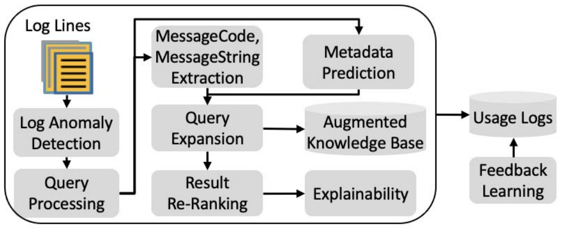


Fig. 3. Run-Time Architectural Diagram for Log Anomaly to Resolution

Log Anomaly Detection (LAD): LAD model is trained to detect anomalies from unlabeled data that may include both normal (healthy state of the system) or abnormal (unhealthy state of the system) log lines. LAD is based on the intuition that a log line is anomalous if it occurred very rarely or was never seen before in the training data. During the training phase, the input log data is fed to a dictionary-based binary error classifier. At this point, the model pathway splits itself into *Erroneous Pathway* (solid line in Figure 4) and *Non-Erroneous Pathway* (dotted line in Figure 4) for processing erroneous and non-erroneous log lines, respectively. The templatization allows to normalize a log line into a format that is invariant of the parameter values. Both the pathways templatize a log line into a log template [17], followed by a fuzzy clustering algorithm to form homogeneous groups known as log template clusters, **E**rroneous **L**og **T**emplate **C**lusters (ELTC) and **N**on-**E**rroneous **L**og **T**emplate **C**lusters (NELTC). That is, the ELTC contains clusters of erroneous log templates, and the NELTC contains clusters of non-erroneous log templates. Finally, the frequency thresholding module analyses the frequency distribution of the log template clusters to filter anomalous clusters where cluster frequency $<=$ threshold frequency of the cluster frequencies distribution. The threshold frequency is set to 3^{rd} quartile for the *Erroneous Pathway* and 1^{st} quartile for the *Non-Erroneous Pathway*. The intuition behind frequency thresholding is to capture the rareness aspect of a log template cluster. Each cluster in the ELTC and NELTC that has its size less than 3^{rd} quartile and 1^{st} quartile, respectively, is labeled as **A**nomalous **E**rroneous **L**og **T**emplate **C**luster (AELTC) and **A**nomalous **N**on-**E**rroneous **L**og **T**emplate **C**luster (ANELTC).

At the test-time, each incoming log line is classified into one of the two classes, erroneous or non-erroneous, followed by conversion into log template. The test-time further involves two pathways: *Erroneous Pathway* for erroneous log templates and the *Non-Erroneous Pathway* for the non-erroneous log templates. The *Erroneous Pathway* checks the membership of the erroneous log template with the AELTC. If it does, then label it as anomalous. Else, it checks for membership with the ELTC. If it does, then the label it as as non-anomalous. Else, the incoming log line is labeled as anomalous. The *Non-Erroneous Pathway* checks the membership of the non-erroneous log template with the ANELTC. If it does then label it as anomalous. Otherwise, check the membership with the NELTC. If it does not then it means that the log line was never seen before during training, and hence it is a possible anomalous candidate.

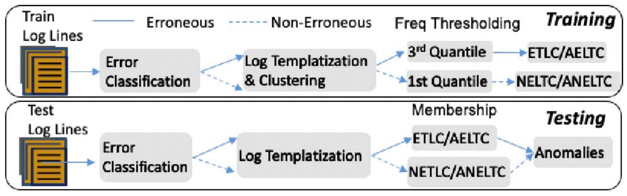


Fig. 4. Log Anomaly Training and Testing Pipeline

Metadata Selection and Prediction: We observed that a *ResolutionUrl* may contain information that may be leveraged to solve heterogeneous problems. There are 464 unique *ResolutionUrl*s spanning across approximately 1300 unique tickets, on an average the same *ResolutionUrl* is used for 5 unique *MessageCodes*. Therefore, knowing the context e.g., *Category*, *Sub-Category*, *Severity* etc. that SREs had used during problem diagnosis is critical, so that at run-time the context can be leveraged to automatically retrieve the most appropriate *ResolutionUrl*.

We applied entropy and information gain [18] to identify *Metadata* that could be useful for retrieving *ResolutionUrl* from the dataset. Entropy captures the randomness in the state of a *Metadata* attribute; high randomness implies low information gain. The entropy of *Severity*, *Category*, *Sub-Category*, *ResolutionTitle*, and *ResolutionUrl* is 1.62, 4.05, 6.75, 8.33 and 7.76, respectively. Although *Severity* has the lowest entropy but it has no correlation with *ResolutionUrl*, hence, it was not selected. Further, we computed the correlation coefficient amongst various *Metadata* and *ResolutionUrl* and, found a positive correlation between *Category*, *Sub-Category* with the *ResolutionUrl*. Also, *Category* / *Sub-Category* are correlated as they form a two-level taxonomy, i.e. *Sub-Category* value depends on the *Category* value. Hence, *Category* / *Sub-Category* are selected for training the prediction model. We experimented with two approaches to build the metadata prediction model:

1) Independent: Two Stochastic Gradient Descent (SGD) models are trained independent of each other, one to predict the *Category*, another to predict the *Sub-Category*.
2) Conditional: Two SGD models are trained, where the second SGD model is conditioned on the predictions of the first SGD model. First SGD is trained to predict the *Category* label, with log line as input. Second SGD is trained to predict *Sub-Category* label using log line and predicted *Category* as input.

The log data is split into training and validation sets in the ratio of 80:20, i.e. 7416 log lines and 167[1] erroneous log lines for training and validation, respectively. We experimented with the most common set of parameters, listed in Table I. Parameters were fine-tuned while building the *Category* prediction model. The learned parameter values are transferred to the *Sub-Category* prediction model under both the settings, Independent and Conditional. Table I shows the range of values considered and the best set of parameter values learned from the *Category* prediction model that achieved an accuracy of **(74.85%)** over the validation set. For the *Independent* and *Conditional* models, the accuracy of *Sub-Category* prediction model is **(52.1%)** and **(55.68%)** respectively. Further, we define *joint accuracy* as the percentage of log lines for which both the *Category* and *Sub-Category* were predicted correctly; joint accuracy of the *Independent* and *Conditional* models is **(50.9%)** and **(54.49%)**, respectively. For LA2R, we

[1] 167 erroneous log lines is a subset of 1852 log lines.

use *Conditional* model as it outperformed the corresponding *Independent* counterpart for each loss_type.

TABLE I
RANGE AND THE BEST SET PARAMETERS FOR CATEGORY PREDICTION

Parameter	Range	Best Value
loss_type	[log, hinge, modified huber, squared hinge, perceptron]	log
classifier_alpha	[0.00001, 0.000001]	0.000001
n-grams	[(1, 1), (1, 2)]	(1, 1)
vector norm	['l1', 'l2']	'l2'
use idf	[True, False]	True
max df	[0.5, 0.75, 1.0]	0.5

IV. EVALUATION

Dataset and Baseline: For evaluation of LA2R, we received 186K log lines from a production application server, spanning across 45 days. LAD identified 15 unique anomalous log windows and 16 unique log anomalies, each log window is defined as a group of log anomalies grouped by start-time. LA2R is compared against a baseline method which uses *MessageCode* to retrieve *ResolutionUrl*s - this is equivalent to what a SRE would use to find the resolution in the Product Documentation. For result validation, Five SREs who manage the Application Server product were consulted, majority voting was used to arrive at the final result.

For each anomalous log line, a row in Table II shows the obfuscated *MessageCode* and the last character 'E', 'W', 'I' indicates whether it is an error, warning or informational (column A), number of retrieved *ResolutionUrl*s using baseline method (B), number of retrieved *ResolutionUrl*s using LA2R (C), number of retrieved *ResolutionUrl*s using LA2R that contain steps that can be executed by SRE for problem diagnosis and resolution (D), validation of column D by SRE (E), actual category label (F), predicted category label (G), SRE validation of F and G (H). For instance, for a given anomalous log line, *MC15W: The res1 could not be found for element with id attribute res2*, the extracted *MessageCode* is *MC15W*, the metadata prediction model predicted *Category* is *System*. The formulated query is *MessageCode*='MC15W' and *Category*='System' that returns 3 results using LA2R. Whereas, for the same query, the baseline returns 0 results, as the *MessageCode* is not present in the dataset. However, the masked log line and the predicted *Category* helps LA2R to find the matching records in the dataset. Note that among the 3 returned results, all are actionable (i.e. executable by SRE). However, SREs suggested that only 1 result out of 3 is relevant; a result is relevant if it helps resolve the problem.

Metadata Prediction and Resolution Retrieval: For each of the unique anomalous log line, *Category* is predicted using the Conditional Metadata Prediction model. Overall accuracy of the predicted *Category* is (**68.75%**) i.e. 11/16. For the *MessageCode*s **MC2E** and **MC5E**, SREs indicated that the ground truth *Category* labels were incorrect highlighting that the model is trained on imperfect (subjective) data. Hence, curated training data may lead to improved accuracy of prediction model. We recomputed the *Category* prediction accuracy on valid ground truth data only, then the predicted category accuracy is (**78.57%**), i.e. 11/14. The joint accuracy of the *Category* and *Sub-Category* prediction is 62%.

TABLE II
EVALUATION OF METADATA PREDICTION (CATEGORY)

(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)
MC1E	12	2	0	-	JDBC	JDBC	Y
MC2E	9	2	0	-	Security	Security	N
MC3E	7	3	3	3	System	System	Y
MC4W	3	2	2	2	JDBC	JDBC	Y
MC5E	1	1	0	-	Transac	Transac	N
MC6E	1	1	1	1	Perf	JMS	N
MC7W	1	1	1	1	System	WebServ	Y
MC8W	2	2	0	-	Classloader	HTTP	Y
MC9I	1	1	1	1	-	JDBC	Y
MC10I	1	1	0	-	-	Security	N
MC11E	1	1	0	-	-	System	Y
MC12W	1	1	1	1	-	Perf	Y
MC13I	1	1	1	1	-	JPA	Y
MC14W	0	3	1	0	-	Security	Y
MC15W	0	3	3	1	-	Security	N
MC16I	0	3	3	0	-	Security	Y
Total	41	28	17	11	-	-	-

In Table II, across 16 unique log lines, LA2R returned 28 (C) results as compared to 41 (B) returned by the baseline, a reduction of (**31%**) in the result set; this is because LA2R leverages contextual *Metadata* during query formulation, resulting in a more targeted result set. Using rule-based approach for resolution classification (Step 3 during augmented dataset preparation in Section II), 17/28 (**60%**) resolutions are actually actionable (i.e. executable). Hence, it is clear that LA2R would derive productivity improvements by reducing cognitive burden for the users as they do not have to go through non-actionable resolutions (e.g. reference to another repository or messages like problem solved or no action needed or contact support). Finally, per SRE evaluation 11/17 (**64.7%**) results were found relevant; and based on observations discussed above, there is a room for improvement. Furthermore, *MessageCodes* of type informational and warning are predictive in nature, indicating that an incident may occur in near future. Given that LA2R leverages log data for LAD, the system identifies anomalous log windows containing such predictive signals; based on the *MessageCode*s in Table II, 62% are informational and warning that can aid in proactive remediation of incidents.

V. CONCLUSION AND FUTURE WORK

We have set forward our vision for an end-to-end system for proactive incident triaging using heterogeneous data in AIOps environment. We have presented an end-to-end LA2R system that uses predictive models for incident remediation. We have demonstrated the efficacy of our system on a production Application Server dataset. Based on SRE feedback, prediction accuracy is 78% and that 64% of resolutions are actionable and relevant. Further, 62% predictive log anomalies contain information and warning *MessageCode*s that can aid in proactive incident remediation.

Finally, we envision a closed loop system that can learn from the LA2R system usage in AIOps pipeline to further fine-tune the models to improve the quality of the anomalies detected and the results retrieved.

References

[1] Junjie Chen, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. Continuous incident triage for large-scale online service systems. In *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 364–375. IEEE, 2019.

[2] Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, et al. Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 304–314, 2020.

[3] Animesh Nandi, Atri Mandal, Shubham Atreja, Gargi Banerjee Dasgupta, and Subhrajit Bhattacharya. Anomaly detection using program control flow graph mining from execution logs. In Balaji Krishnapuram, Mohak Shah, Alexander J. Smola, Charu C. Aggarwal, Dou Shen, and Rajeev Rastogi, editors, *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 215–224. ACM, 2016.

[4] Wei Xu, Ling Huang, Armando Fox, David A. Patterson, and Michael I. Jordan. Detecting large-scale system problems by mining console logs. In Jeanna Neefe Matthews and Thomas E. Anderson, editors, *Proceedings of the 22nd ACM Symposium on Operating Systems Principles 2009, SOSP 2009, Big Sky, Montana, USA, October 11-14, 2009*, pages 117–132. ACM, 2009.

[5] JianGuang Lou, Qiang Fu, Shengqi Yang, Ye Xu, and Jiang Li. Mining invariants from console logs for system problem detection. In Paul Barham and Timothy Roscoe, editors, *2010 USENIX Annual Technical Conference, Boston, MA, USA, June 23-25, 2010*. USENIX Association, 2010.

[6] Zhen Ming Jiang, Ahmed E Hassan, Gilbert Hamann, and Parminder Flora. Automatic identification of load testing problems. In *2008 IEEE International Conference on Software Maintenance*, pages 307–316. IEEE, 2008.

[7] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, and Xuewei Chen. Log clustering based problem identification for online service systems. In Laura K. Dillon, Willem Visser, and Laurie A. Williams, editors, *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016 - Companion Volume*, pages 102–111. ACM, 2016.

[8] Max Landauer, Markus Wurzenberger, Florian Skopik, Giuseppe Settanni, and Peter Filzmoser. Dynamic log file analysis: An unsupervised cluster evolution approach for anomaly detection. *computers & security*, 79:94–116, 2018.

[9] Huida Qiu, Yan Liu, Niranjan A Subrahmanya, and Weichang Li. Granger causality for time-series anomaly detection. In *2012 IEEE 12th international conference on data mining*, pages 1074–1079. IEEE, 2012.

[10] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1285–1298. ACM, 2017.

[11] Shayan Hashemi and Mika Mäntylä. Detecting anomalies in software execution logs with siamese network, 2021.

[12] Siyang Lu, Xiang Wei, Yandong Li, and Liqiang Wang. Detecting anomaly in big data system logs using convolutional neural network. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, pages 151–158, 2018.

[13] Weibin Meng, Ying Liu, Yichen Zhu, Shenglin Zhang, Dan Pei, Yuqing Liu, Yihao Chen, Ruizhi Zhang, Shimin Tao, Pei Sun, and Rong Zhou. Loganomaly: Unsupervised detection of sequential and quantitative anomalies in unstructured logs. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 4739–4745. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[14] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. Self-attentive classification-based anomaly detection in unstructured logs, 2020.

[15] Xu Zhang, Qingwei Lin, Qingwei Lin, and ... Robust log-based anomaly detection on unstable log data. In *FSE'19*, August 2019.

[16] Rafiul Ahad, Eric Chan, and Adriano Santos. Toward autonomic cloud: Automatic anomaly detection and resolution. In *2015 International Conference on Cloud and Autonomic Computing*, pages 200–203. IEEE, 2015.

[17] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*, pages 33–40. IEEE, 2017.

[18] Nikhil R Pal and Sankar K Pal. Entropy: A new definition and its applications. *IEEE transactions on systems, man, and cybernetics*, 21(5):1260–1270, 1991.