A Theoretical Framework for Understanding the Relationship between Log Parsing and Anomaly Detection*

Donghwan Shin[1][0000−0002−0840−6449], Zanis Ali Khan[1][0000−0002−3935−2148], Domenico Bianculli[1][0000−0002−4854−685X], and Lionel Briand[1,2][0000−0002−1393−1010]

[1] University of Luxembourg, Luxembourg
{donghwan.shin,zanis-ali.khan,domenico.bianculli,lionel.briand}@uni.lu
[2] University of Ottawa, Canada

Abstract. Log-based anomaly detection identifies systems' anomalous behaviors by analyzing system runtime information recorded in logs. While many approaches have been proposed, all of them have in common an essential pre-processing step called log parsing. This step is needed because automated log analysis requires structured input logs, whereas original logs contain semi-structured text printed by logging statements. Log parsing bridges this gap by converting the original logs into structured input logs fit for anomaly detection.

Despite the intrinsic dependency between log parsing and anomaly detection, no existing work has investigated the impact of the "quality" of log parsing results on anomaly detection. In particular, the concept of "ideal" log parsing results with respect to anomaly detection has not been formalized yet. This makes it difficult to determine, upon obtaining inaccurate results from anomaly detection, if (and why) the root cause for such results lies in the log parsing step.

In this short paper, we lay the theoretical foundations for defining the concept of "ideal" log parsing results for anomaly detection. Based on these foundations, we discuss practical implications regarding the identification and localization of root causes, when dealing with inaccurate anomaly detection, and the identification of irrelevant log messages.

Keywords: Log Parsing · Log Analysis· Anomaly Detection.

1 Introduction

Logs record the critical state and events of the system at runtime, providing valuable information for monitoring and troubleshooting. Further, logs are often

* This work has received funding from the Celtic-Next project CRITISEC and NSERC of Canada under the Discovery and CRC programs. Donghwan Shin was partially supported by the Basic Science Research Programme through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (2019R1A6A3A03033444).

the only data available that record the system's runtime behavior. Therefore, to ensure the reliability of a system, *log-based anomaly detection* has been widely studied with the aim of automatically deciding if the logs contain any anomalous patterns that do not conform to the expected behavior of the system [8].

While many approaches have been proposed for log-based anomaly detection, all of them have in common an essential pre-processing step called *log parsing*. This step is needed because automated log analysis requires structured input logs, whereas collected logs are usually free-formed or semi-structured text strings printed by logging statements (e.g., `printf()`, `logger.info()`) included in the source code. Log parsing essentially consists in converting collected logs into structured input logs (e.g., by identifying the message templates corresponding to the various log entries); it is an active research topic that has received considerable attention [5, 8, 25].

Despite advances in log parsing and log-based anomaly detection, to the best of our knowledge, no existing work has thoroughly investigated the impact of the "quality" of log parsing results on anomaly detection. In particular, the concept of "ideal" log parsing results with respect to anomaly detection has not been defined and formalized yet. The lack of such a conceptual framework makes it difficult, when performing empirical studies on anomaly detection techniques, to determine if the root causes of inaccuracies in anomaly detection are due to the limitations of log parsing techniques.

In this short paper, we propose a theoretical framework for defining and discussing what are ideal log parsing results for anomaly detection. In particular, we consider log parsing as an information abstraction process that converts collected logs into structured logs. Since automated anomaly detection relies on structured logs, its best operating conditions are when the minimum amount of information that is necessary to distinguish normal from abnormal behaviors is present in such logs. To this end, we formally define the concepts of *distinguishability* and *minimality*, which further lead to the definition of *ideal* log parsing results. We also discuss practical implications related to log parsing and anomaly detection and present future research directions derived from our theoretical framework.

The rest of the paper is organized as follows. Section 2 explains the notations and basic definitions that will be used throughout the paper. Section 3 formalizes the key concepts regarding ideal log parsing results for anomaly detection. Section 4 discusses practical implications of the proposed theoretical framework. Section 5 discusses related work. Section 6 concludes the paper and provides directions for future work.

2 Preliminaries

This section introduces basic notations and concepts that will be used throughout the paper.

We use uppercase letters to denote *collections* (i.e., sets and sequences) and lowercase letters to denote *elements* in a collection. Specifically, $\{\dots\}$ denotes

Index	Log Message
1	`Receiving block blk_471078 src: /1.2.3.4:56 dest: /1.2.3.4:78`
2	`Receiving block blk_471078 src: /4.3.2.1:65 dest: /4.3.2.1:78`
...	...
14	`Verification succeeded for blk_471078`

Fig. 1. Example from HDFS Logs [9]

a set and $\langle \dots \rangle$ denotes a sequence. For simplicity, we use the same notation $|S|$ to denote the *cardinality* of a set S and the *length* of a sequence S.

Definition 1 (Log Messages and Logs). *A* log message m *is a string printed by a logging statement in the source code. A* log l *is a finite sequence of log messages, denoted by* $l = \langle m_1, m_2, \dots, m_n \rangle$.

For instance, Fig. 1 shows a simplified[3] example from the actual log produced by running HDFS [9]. In this case, the example log can be denoted by $l_{ex} = \langle m_1, m_2, \dots, m_{14} \rangle$ where m_{14} is the string "`Verification succeeded for blk_471078`".

Definition 2 (Normal and Abnormal Logs). *For a set of logs* L*, a log* $l \in L$ *is said to be* normal *if and only if it represents a normal behavior of the system. Otherwise,* l *is said to be* abnormal. *Normal and abnormal logs are denoted by* $L_n \subseteq L$ *and* $L_a \subseteq L$*, respectively; for a given* L *and the corresponding* L_n *and* L_a*, we have* $L_n \cap L_a = \emptyset$ *and* $L_n \cup L_a = L$.

Notice that Definition 2 is independent from the nature of anomalies (e.g., point and collective [2]). The definition only assumes that normal and abnormal behaviors of the system are known and distinguishable in logs. This assumption can be easily satisfied when the accuracy of anomaly detection techniques, including log parsing techniques, are evaluated in controlled experiments using known benchmarks; furthermore, enhancing the quality of logs to distinguish normal and abnormal behaviors has been actively studied [11, 22–24]. In the rest of the paper we adopt this definition and make its underlying assumption.

3 Ideal Log Parsing Results for Anomaly Detection

3.1 Log Parsing as Abstraction

Intuitively, log parsing is a process that converts original logs composed of free-formed messages into structured logs, by extracting key information from individual log messages. For example, some log parsing approaches, such as Drain [7]

[3] In general, logs may contain extra information, such as timestamps and logging levels (e.g., info, debug) for individual log messages. However, we omit such information since log parsing deals with log messages characterizing the states or events of the system.

and MoLFI [14], may extract key information from m_1 in Fig. 1 as an event template "`Receiving block <*> src: <*> dest <*>`" characterizing the event of receiving a block, where symbol `<*>` indicates the position of a parameter value determined at runtime. With respect to these approaches, all messages that match the template, such as m_2 in Fig. 1, represent the same event of receiving a block. In this sense, we can consider log parsing as an *abstraction* process that generates "abstract" key information that represents multiple "specific" messages. Notice that different log parsing approaches yield different abstraction results (not even necessarily in the form of templates). For example, a log parsing approach that simply counts the number of tokens would abstract m_1 as the integer 7 (as m_1 contains seven tokens). To keep our presentation general, we introduce an abstraction function τ that represents a log parsing approach.

Definition 3 (Log Parsing as an Abstraction Function). *Given a set of log messages M and a generic set of parsing results A, a log parsing approach can be represented as an abstraction function $\tau\colon M \to A$.*

Notice that the definition of A is left generic, to accommodate different types of results yielded by log parsing techniques.

Using the concept of τ, the results obtained by a parsing approach can be seen as an abstraction of the original log itself.

Definition 4 (Abstraction of Log). *Given an abstraction function τ representing a log parsing approach and a log $l = \langle m_1, m_2, \ldots, m_n \rangle$, the abstraction of l using τ, denoted by $\tau^*(l)$, is defined as $\tau^*(l) = \langle \tau(m_1), \tau(m_2), \ldots, \tau(m_n) \rangle$.*

In other words, τ^* can be considered as an abstraction function for a log, extended from τ. Similarly, we can further extend τ^* to consider a set of logs as follows.

Definition 5 (Abstraction of Set of Logs). *Given an abstraction function τ representing a log parsing approach and a set of logs L, the abstraction of L using τ, denoted by $\tau^{**}(L)$, is defined as $\tau^{**}(L) = \{\tau^*(l) \mid l \in L\}$.*

Based on these definitions, $\tau^{**}(L)$ represents the results of using a log parsing approach (abstracted by τ) on a set of logs L; in our context, it represents the structured input logs provided as input to an anomaly detection approach.

Running Example To better understand the above definitions, let us consider a set of logs $L_{ex} = \{l_1, l_2, l_3\}$ where $l_1 = \langle m_a, m_b, m_c \rangle$, $l_2 = \langle m_b, m_a, m_c \rangle$, and $l_3 = \langle m_a, m_b, m_d \rangle$ and each message in $\{m_a, m_b, m_c, m_d\}$ is different from the others. Let us assume to use a log parsing approach that yields an integer value, such that both m_a and m_b are mapped to 1 while m_c and m_d are mapped to 2 and 3, respectively. We can represent the log parsing approach as the abstraction function τ_{ex} defined such that $\tau_{ex}(m_a) = \tau_{ex}(m_b) = 1$, $\tau_{ex}(m_c) = 2$, and $\tau_{ex}(m_d) = 3$. Using τ_{ex}, we can see that the abstraction of l_1 is $\tau^*_{ex}(l_1) = \langle \tau_{ex}(m_a), \tau_{ex}(m_b), \tau_{ex}(m_c) \rangle = \langle 1, 1, 2 \rangle$. Similarly, $\tau^*_{ex}(l_2) =$

$\langle 1,1,2\rangle$ and $\tau^*_{ex}(l_3) = \langle 1,1,3\rangle$. As a result, the abstraction of L_{ex} is $\tau^{**}_{ex}(L_{ex}) = \{\tau^*_{ex}(l_1), \tau^*_{ex}(l_2), \tau^*_{ex}(l_3)\} = \{\langle 1,1,2\rangle, \langle 1,1,3\rangle\}$.

This example shows that different logs (e.g., l_1 and l_2) can be *indistinguishable* when abstracted using a certain log parsing approach. This is directly related to one of the key concepts for defining the ideal log parsing results for anomaly detection, which will be detailed in the next section.

3.2 Ideal Log Parsing Results

As described above, log parsing abstracts L to $\tau^{**}(L)$, possibly resulting in different logs indistinguishable from each other as a result of abstraction. For the main anomaly detection step that takes $\tau^{**}(L)$ as input, if normal and abnormal logs are indistinguishable in $\tau^{**}(L)$, then it is impossible to correctly identify abnormal behaviors from it. It is thus important to formalize the concept of *distinguishability* of log parsing results:

Definition 6 (Distinguishability of Log Parsing Results). *Given a non-empty set of normal logs $L_n \subset L$ and a non-empty set of abnormal logs $L_a \subset L$ (where $L_n \cup L_a = L$ and $L_n \cap L_a = \emptyset$), an abstraction function τ* distinguishes *L_n and L_a if and only if $\tau^{**}(L_n) \cap \tau^{**}(L_a) = \emptyset$. In this case, $\tau^{**}(L)$ is called* d-maintaining *(maintaining the distinguishability) between L_n and L_a.*

In other words, *d-maintaining* log parsing results maintain the distinction between L_n and L_a after the abstraction of log parsing. For our running example used in Section 3.1, let us additionally consider $L_n = \{l_1, l_2\}$ and $L_a = \{l_3\}$. Since $\tau^{**}_{ex}(L_n) = \{\langle 1,1,2\rangle\}$ and $\tau^{**}_{ex}(L_a) = \{\langle 1,1,3\rangle\}$, $\tau^{**}_{ex}(L_n) \cap \tau^{**}_{ex}(L_a) = \emptyset$, and therefore $\tau^{**}_{ex}(L_{ex})$ is *d-maintaining* between L_n and L_a.

However, distinguishability is only a necessary condition for log parsing results to be ideal. For example, if we consider an abstraction function $\tau_=$ such that $\tau_=(m) = m$ for every message m, $\tau^{**}_=(L)$ is *d-maintaining* between arbitrary L_n and L_a (since $\tau^{**}_=(L_n) = L_n$, $\tau^{**}_=(L_a) = L_a$, and $L_n \cap L_a = \emptyset$ by definition). However, $\tau^{**}_=(L)$ does not represent the ideal log parsing results because $\tau_=$ does not produce an actual abstraction (since it is just defined as the identity function). Indeed, as long as log parsing results maintain the distinguishability between L_n and L_a, a higher degree of abstraction (i.e., mapping more messages to the same parsing result) leads to better operating conditions for anomaly detection, as it minimizes the "information" contained in the structured input logs (i.e., the log parsing results) that must be analyzed by the main anomaly detection step. Furthermore, since anomaly detection is largely based on Machine Learning (ML) [8], including Clustering, Support Vector Machine (SVM), and Long Short-Term Memory (LSTM), the abstraction of log parsing can significantly improve the learning performance of anomaly detection by reducing dimensionality (i.e., the number of features)[4]. Therefore, we should additionally consider the concept of *minimality* of the information contained in log parsing results.

[4] This is because distinct $\tau(m)$ for each message m that appear in L can lead to one or more dimensions. In ML, dimensionality reduction is an essential topic to improve predictive power [16].

To formalize the minimality concept, we first need to define the information contained in log parsing results. Since the core of log parsing is to abstract individual messages, we consider the information contained in $\tau^{**}(L)$ in terms of its unique entities (i.e., abstracted messages) as follows.

Definition 7 (Information in Log Parsing Results). *Given a set of logs L and an abstraction function τ, the information contained in L as abstracted by τ, denoted by $I(L,\tau)$, is defined as $I(L,\tau) = \bigcup_{l \in L}\{\tau(m) \mid m \in l\}$.*

For our running example, the information contained in L_{ex} abstracted by τ_{ex} is $I(L_{ex}, \tau_{ex}) = \{\tau_{ex}(m_a), \tau_{ex}(m_b), \tau_{ex}(m_c), \tau_{ex}(m_d)\} = \{1, 2, 3\}$, meaning that τ_{ex} reduces the information from $\{m_a, m_b, m_c, m_d\}$ to $\{1, 2, 3\}$ through abstraction.

Using $I(L,\tau)$, we can define the concept of ideal log parsing results by considering both distinguishability and minimality as follows.

Definition 8 (Minimal Distinguishable Log Parsing Results). *Given a non-empty set of normal logs $L_n \subset L$ and a non-empty set of abnormal logs $L_a \subset L$ (where $L_n \cup L_a = L$ and $L_n \cap L_a = \emptyset$), we say that $\tau^{**}(L)$ is* minimally d-maintaining *between L_n and L_a if and only if (1) $\tau^{**}(L)$ is* d-maintaining *between L_n and L_a and (2) there is no abstraction function τ' such that $\tau'^{**}(L)$ is* d-maintaining *and $|I(L,\tau')| < |I(L,\tau)|$.*

Taking the running example again, $\tau^{**}_{ex}(L_{ex})$ is *d-maintaining* (but not minimally) because there exists τ_{new} such that (1) $\tau^{**}_{new}(L_{ex})$ is *d-maintaining* and (2) $|I(L_{ex}, \tau_{new})| < |I(L_{ex}, \tau_{ex})|$. Specifically, if $\tau_{new}(m_a) = \tau_{new}(m_b) = \tau_{new}(m_c) = 12$ and $\tau_{new}(m_d) = 3$, then $\tau^{**}_{new}(L_n) = \{\langle 12, 12, 12\rangle\}$, $\tau^{**}_{new}(L_a) = \{\langle 12, 12, 3\rangle\}$, $I(L_{ex}, \tau_{new}) = \{12, 3\}$; therefore $\tau^{**}_{new}(L_n) \cap \tau^{**}_{new}(L_a) = \emptyset$ and $|I(L_{ex}, \tau_{new})| = |\{12, 3\}| = 2$ is less than $|I(L_{ex}, \tau_{ex})| = |\{1, 2, 3\}| = 3$. However, $\tau^{**}_{new}(L_{ex})$ is *minimally d-maintaining* because there is no τ' such that $|I(L,\tau')| = 1$ and $\tau'^{**}(L)$ is *d-maintaining*. As a result, $\tau^{**}_{new}(L_{ex})$ represents the ideal log parsing results for anomaly detection.

4 Applications

4.1 Localization of the Causes of Inaccurate Anomaly Detection

When anomaly detection accuracy is not 100% (i.e., some abnormal behaviors are not correctly detected or some normal behaviors are incorrectly detected as abnormal), it is important to know exactly where the problem lies (in the log parsing step, in the main anomaly detection step, or in both), in order to improve the results. Using our theoretical framework, we can localize the cause of inaccurate anomaly detection results. Specifically, for a set of normal logs L_n and a set of abnormal logs L_a, we can distinguish the following three cases.

Case 1. If the log parsing results is minimally d-maintaining between L_n and L_a, the main anomaly detection step must be the cause of the inaccuracy, because the log parsing results are ideal for anomaly detection.

Case 2. If the log parsing results is d-maintaining between L_n and L_a but not minimally so, a perfect anomaly detection approach could achieve pinpoint accuracy. However, as discussed in Section 3.2, making the log parsing results minimally d-maintaining could significantly increase anomaly detection accuracy.

Case 3. Otherwise, inaccurate anomaly detection results are inevitable due to the low-quality log parsing results. We can further investigate the issue of log parsing results by focusing on exactly what prevents the log parsing results from being d-maintaining between L_n and L_a.

The above characterization has important implications for researchers who want to assess the accuracy of their anomaly detection approaches. As non-ideal log parsing results decrease anomaly detection accuracy, *it is recommended to use ideal log parsing results in controlled experiments* to properly assess the performance of a technique, independently of log parsing. Also, if possible, using various log parsing results including minimally d-maintaining, d-maintaining but not minimal, and non-d-maintaining ones, would provide a better picture on how anomaly detection would work in practice, depending on the quality of the log parsing results.

4.2 Removal of Unnecessary Log Messages for Anomaly Detection

As discussed in Section 3.1, some messages become indistinguishable through the abstraction of log parsing. One may wonder whether simply removing some of the messages could contribute to further reduce the amount of information contained in the log parsing results. Indeed, in our running example, $\tau_y^{**}(L_{ex})$ remains minimally d-maintaining even if we remove m_a and m_b from L_{ex}. However, this is not always true. For example, consider a normal log $l_n = \langle m_x, m_y \rangle$ and an abnormal log $l_a = \langle m_y \rangle$. While we can consider an abstraction function τ such that $\tau(m_x) = \tau(m_y)$ and $\tau^*(l_n) \neq \tau^*(l_a)$, removing m_x from the logs makes l_n and l_a indistinguishable. This example shows that, though there are messages that can be abstracted to the same entity, it does not necessarily mean that one of them can be removed without affecting anomaly detection accuracy.

Notice that existing log parsing techniques do not reduce the length of individual logs[5]. However, we know, as discussed above, that having minimal information necessary to distinguish normal and abnormal logs is the best operating condition for anomaly detection. In this sense, further research is needed to develop an automated approach for "greedy" log parsing techniques that not only abstract but also remove log messages to achieve minimality while maintaining the distinguishability of the results.

[5] Though the length of logs can be reduced in a pre-processing step by omitting certain messages or events based on domain knowledge, this is independent from log parsing, which just abstracts messages.

5 Related Work

To the best of our knowledge, there is no existing work that provides a framework to formalize the concept of ideal log parsing results for anomaly detection. This is mainly because most of the existing log parsing approaches, including AEL [10], Drain [7], IPLoM [13], LenMa [18], LFA [17], LogCluster [20], LogMine [6], LogSig [19], MoLFI [14], SHISO [15], SLCT [21], Spell [4], and Logram [3], have been proposed as general-purpose approaches rather than specialized for anomaly detection. The accuracy of all these approaches has been assessed with respect to the logging statements that produce individual messages. For example, the execution of the logging statement `printf("retry " + i)` in the source code, when the program variable `i` evaluates to 1, will generate the log message "`retry 1`". Then a log parsing approach is expected to reconstruct the form of the logging statement as a template "`retry <*>`" without accessing the source code, where symbol "`<*>`" indicates the position of the parameter value (i.e., "`1`"). In other words, the ground truth used to assess the accuracy of general-purpose log parsing is determined based on the logging statements that generated the input logs. On the other hand, there is no ground truth that guarantees the best operating conditions for anomaly detection. To address this challenge, we provide a theoretical foundation to precisely define key concepts, including the distinguishability and minimality of ideal log parsing results.

6 Conclusion and Future Research Directions

In this short paper, we proposed a theoretical framework that formalizes the concepts of distinguishability and minimality, showing that log parsing results that minimally maintain distinguishability between normal and abnormal logs provide the best operating conditions for anomaly detection. Using our theoretical framework, we also identified practical implications for researchers regarding the root causes for inaccuracy in anomaly detection and the removal of log messages that are unnecessary for anomaly detection.

Several future research directions can be derived from our theoretical framework.

Efficient Ideal Log Parsing for Experiments. We saw that having ideal log parsing results is important in controlled experiments to properly identify the cause of inaccurate anomaly detection results. However, getting ideal log parsing results for a given set of logs is not that simple since, for the logs containing n unique log messages, the number of all possible log parsing results (i.e., the number of all possible abstraction functions) is equal to the Bell number B_n (i.e., the number of all partitions of a set of size n) [1]. Indeed, the problem of identifying the ideal log parsing results can be regarded as an optimization problem to minimize the amount of information contained in the log parsing results while maintaining distinguishability between normal and abnormal logs. Also, there is additional information that is potentially relevant to address this problem, such

as the similarity between messages. Therefore, developing an efficient approach is an appealing research direction. We plan to extend the theoretical framework to measure the degree of distinguishability and minimality of given log parsing results and use such measures as fitness functions in meta-heuristic search algorithms [12] to find optimal log parsing results for anomaly detection.

Log Parsing Approaches Tailored to Anomaly Detection. Since ideal log parsing results for anomaly detection require different properties than those sought by general-purpose log parsing approaches, solutions tailored to anomaly detection are called for. Notice that, to be used in practice, these approaches should be able to generate near-ideal log parsing results without being provided with normal and abnormal labels.

Empirical Studies. While we formalize the concepts of ideal log parsing results for anomaly detection, the impact of log parsing on anomaly detection in practice remains unclear. Therefore, to better understand such impact in real-world applications, more empirical studies investigating the relationship between log parsing and anomaly detection results are required.

Run-time Applications. The theoretical framework can also be used in an online setting where logs are produced at runtime (e.g., in the context of stream verification). For example, online log parsing techniques, such as Drain [7], dynamically update log parsing results given streaming logs. Therefore, as long as normal and abnormal behaviors are distinguished in logs, one could monitor the quality of log parsing results for anomaly detection at runtime by considering distinguishability and minimality. Opportunities for applications in such diverse settings are open.

References

1. Aigner, M.: A characterization of the bell numbers. Discrete Mathematics **205**(1), 207–210 (1999). https://doi.org/10.1016/S0012-365X(99)00108-9
2. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: A survey. ACM Comput. Surv. **41**(3) (Jul 2009). https://doi.org/10.1145/1541880.1541882
3. Dai, H., Li, H., Chen, C.S., Shang, W., Chen, T.: Logram: Efficient log parsing using n-gram dictionaries. IEEE Transactions on Software Engineering pp. 1–1 (2020). https://doi.org/10.1109/TSE.2020.3007554
4. Du, M., Li, F.: Spell: Streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining (ICDM). pp. 859–864. IEEE, Barcelona, Spain (2016). https://doi.org/10.1109/CNSM.2015.7367331
5. El-Masri, D., Petrillo, F., Guéhéneuc, Y.G., Hamou-Lhadj, A., Bouziane, A.: A systematic literature review on automated log abstraction techniques. Information and Software Technology **122**, 106276 (2020). https://doi.org/10.1016/j.infsof.2020.106276
6. Hamooni, H., Debnath, B., Xu, J., Zhang, H., Jiang, G., Mueen, A.: Logmine: Fast pattern recognition for log analytics. In: 25th ACM International

on Conference on Information and Knowledge Management (CIKM). pp. 1573–1582. Association for Computing Machinery, Indianapolis, IN, USA (2016). https://doi.org/10.1145/2983323.2983358
7. He, P., Zhu, J., Zheng, Z., Lyu, M.R.: Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services (ICWS). pp. 33–40. IEEE, Honolulu, HI, USA (2017). https://doi.org/10.1109/ICWS.2017.13
8. He, S., He, P., Chen, Z., Yang, T., Su, Y., Lyu, M.R.: A survey on automated log analysis for reliability engineering. ACM Comput. Surv. **54**(6) (Jul 2021). https://doi.org/10.1145/3460345
9. He, S., Zhu, J., He, P., Lyu, M.R.: Loghub: A large collection of system log datasets towards automated log analytics (2020), https://arxiv.org/pdf/2008.06448.pdf
10. Jiang, Z.M., Hassan, A.E., Flora, P., Hamann, G.: Abstracting execution logs to execution events for enterprise applications. In: 2008 The Eighth International Conference on Quality Software (QSIC). pp. 181–186. IEEE, Oxford, UK (2008). https://doi.org/10.1109/QSIC.2008.50
11. Liu, Z., Xia, X., Lo, D., Xing, Z., Hassan, A.E., Li, S.: Which variables should i log? IEEE Transactions on Software Engineering pp. 1–1 (2019). https://doi.org/10.1109/TSE.2019.2941943
12. Luke, S.: Essentials of Metaheuristics. Lulu, second edn. (2013), available for free at http://cs.gmu.edu/∼sean/book/metaheuristics/
13. Makanju, A.A., Zincir-Heywood, A.N., Milios, E.E.: Clustering event logs using iterative partitioning. In: 15th ACM International conference on Knowledge discovery and data mining (SIGKDD). pp. 1255–1264. Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1557019.1557154
14. Messaoudi, S., Panichella, A., Bianculli, D., Briand, L., Sasnauskas, R.: A search-based approach for accurate identification of log message formats. In: 2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC). pp. 167–16710. ACM, Association for Computing Machinery, Gothenburg, Sweden (2018). https://doi.org/10.1145/3196321.3196340
15. Mizutani, M.: Incremental mining of system log format. In: 2013 IEEE International Conference on Services Computing (SCC). pp. 595–602. IEEE, Santa Clara, CA, USA (2013). https://doi.org/10.1109/SCC.2013.73
16. Mohri, M., Rostamizadeh, A., Talwalkar, A.: Foundations of Machine Learning. The MIT Press (2012), https://dl.acm.org/doi/book/10.5555/3360093
17. Nagappan, M., Vouk, M.A.: Abstracting log lines to log event types for mining software system logs. In: 2010 7th IEEE Working Conference on Mining Software Repositories (MSR 2010). pp. 114–117. IEEE, IEEE, Cape Town, South Africa (2010). https://doi.org/10.1109/MSR.2010.5463281
18. Shima, K.: Length matters: Clustering system log messages using length of words (2016), https://arxiv.org/abs/1611.03213
19. Tang, L., Li, T., Perng, C.S.: Logsig: Generating system events from raw textual logs. In: 20th ACM international conference on Information and knowledge management (CIKM). pp. 785–794. ACM, New York, NY, USA (2011). https://doi.org/10.1145/2063576.2063690
20. Vaarandi, R., Pihelgas, M.: Logcluster - a data clustering and pattern mining algorithm for event logs. In: 2015 11th International Conference on Network and Service Management (CNSM). pp. 1–7. IEEE, Barcelona, Spain (2015). https://doi.org/10.1109/CNSM.2015.7367331

21. Vaarandi, R.: A data clustering algorithm for mining patterns from event logs. In: 3rd IEEE Workshop on IP Operations & Management (IPOM). pp. 119–126. IEEE, Kansas City, MO, USA (2003). https://doi.org/10.1109/IPOM.2003.1251233
22. Yuan, D., Park, S., Huang, P., Liu, Y., Lee, M.M., Tang, X., Zhou, Y., Savage, S.: Be conservative: Enhancing failure diagnosis with proactive logging. In: 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12). pp. 293–306. USENIX Association, Hollywood, CA (Oct 2012), https://www.usenix.org/conference/osdi12/technical-sessions/presentation/yuan
23. Yuan, D., Zheng, J., Park, S., Zhou, Y., Savage, S.: Improving software diagnosability via log enhancement. ACM Trans. Comput. Syst. **30**(1) (Feb 2012). https://doi.org/10.1145/2110356.2110360
24. Zhao, X., Rodrigues, K., Luo, Y., Stumm, M., Yuan, D., Zhou, Y.: Log20: Fully automated optimal placement of log printing statements under specified overhead threshold. In: 2017 26th Symposium on Operating Systems Principles (SOSP). p. 565–581. SOSP '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132747.3132778
25. Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., Lyu, M.R.: Tools and benchmarks for automated log parsing. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP). pp. 121–130. IEEE, Madrid, Spain (2019). https://doi.org/10.1109/ICSE-SEIP.2019.00021