Guidelines for Assessing the Accuracy of Log Message Template Identification Techniques

Zanis Ali Khan
University of Luxembourg
Luxembourg, Luxembourg
zanis-ali.khan@uni.lu

Donghwan Shin
University of Luxembourg
Luxembourg, Luxembourg
donghwan.shin@uni.lu

Domenico Bianculli
University of Luxembourg
Luxembourg, Luxembourg
domenico.bianculli@uni.lu

Lionel Briand
University of Luxembourg
Luxembourg, Luxembourg
University of Ottawa
Ottawa, Canada
lionel.briand@uni.lu

ABSTRACT

Log message template identification aims to convert raw logs containing free-formed log messages into structured logs to be processed by automated log-based analysis, such as anomaly detection and model inference. While many techniques have been proposed in the literature, only two recent studies provide a comprehensive evaluation and comparison of the techniques using an established benchmark composed of real-world logs. Nevertheless, we argue that both studies have the following issues: (1) they used different accuracy metrics without comparison between them, (2) some ground-truth (oracle) templates are incorrect, and (3) the accuracy evaluation results do not provide any information regarding incorrectly identified templates.

In this paper, we address the above issues by providing three guidelines for assessing the accuracy of log template identification techniques: (1) use appropriate accuracy metrics, (2) perform oracle template correction, and (3) perform analysis of incorrect templates. We then assess the application of such guidelines through a comprehensive evaluation of 14 existing template identification techniques on the established benchmark logs. Results show very different insights than existing studies and in particular a much less optimistic outlook on existing techniques.

CCS CONCEPTS

• **General and reference** → **Metrics**; **Evaluation**; • **Software and its engineering** → **Maintaining software**.

KEYWORDS

logs, template identification, metrics

ACM Reference Format:
Zanis Ali Khan, Donghwan Shin, Domenico Bianculli, and Lionel Briand. 2022. Guidelines for Assessing the Accuracy of Log Message Template Identification Techniques. In *44th International Conference on Software Engineering (ICSE '22), May 21–29, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3510003.3510101

ICSE '22, May 21–29, 2022, Pittsburgh, PA, USA

1 INTRODUCTION

Software logs are essential for various software engineering tasks, such as model inference [13, 24] and anomaly detection [3, 17], since logs are often the only data available that record the run-time behavior of a software system.

In general, a log is a sequence of log messages generated by log printing statements in the source code. For example, the execution of the log printing statement `log("retry " + counter)`, when the program variable `counter` evaluates to 1, will generate the log message "`retry 1`". While such log messages contain valuable run-time information, they cannot be directly processed by log-based analysis techniques that require structured input logs instead of free-formed log messages.

Log message template identification aims to address the issue by decomposing log messages into fixed parts called message templates (templates, in short), characterizing the event types, and variable parts containing the parameter values of the events, which are determined at run time. For the above example message, the event template would be "`retry <*>`", where symbol "`<*>`" indicates the position of the parameter value ("`1`") in the variable part. Log template identification is straightforward when one has access to the source code, because one can derive message templates from the log printing statements. However, often the source code is not available, e.g., because the system is composed of 3rd-party components, and, as a result, many automated log template identification techniques have been proposed in the literature (e.g., AEL [9], Drain [6], IPLoM [11], LenMa [19], LFA [16], LKE [4], LogCluster [22], LogMine [5], LogSig [20], MoLFI [14], SHISO [15], SLCT [21], Spell [2]) to identify templates using only log messages [26].

Given the number of available techniques for log message template identification, it is important to evaluate and rank them using different criteria (e.g., accuracy, execution time). However, most of the proposed techniques have been evaluated, in terms of accuracy, in isolation or with respect to a few alternatives, often using ad-hoc benchmarks. A notable exception is the study by Zhu et al. [26], which provides the first comprehensive evaluation of the 13 aforementioned techniques using a benchmark composed of 16 log

datasets collected from real-world systems. Dai et al. [1] recently used the same benchmark to compare their new technique (Logram) with the five techniques that achieved the highest accuracy scores as reported by Zhu et al. [26]. Nevertheless, in both studies, we observed three important issues regarding the accuracy evaluation of template identification techniques.

First, both studies used different accuracy metrics, namely Grouping Accuracy (GA)[1] and Parsing Accuracy (PA), leading to different rankings for several techniques. However, these evaluation results have not been compared with each other yet, making it hard to clearly understand how and why the rankings of template identification techniques vary when using different accuracy metrics. Furthermore, some accuracy metrics can return misleading results since they are sensitive to the number of repeated log messages. This can be a significant issue for systems whose logs contain repeated log messages (e.g., heartbeat messages) that are not important in terms of the system's business logic, a common situation.

Second, both studies determined the oracle templates (which represent the ground truth of template identification results) by manually inspecting log messages, which is an error-prone process. Though such a process is unavoidable when the source code is not accessible, the impact of using incorrect oracle templates on the assessment of the accuracy of template identification techniques is unclear for the various accuracy metrics.

Third, both studies did not provide any information regarding the templates incorrectly identified by template identification techniques. Due to such lack of information, we cannot understand the weaknesses and limitations of the various techniques for log template identification, making it difficult to select the best technique for a given application context.

Contributions. In this paper, we address the above issues by (1) providing guidelines for assessing the accuracy of log message template identification techniques and (2) assessing the application of such guidelines through a comprehensive evaluation of 14 existing template identification techniques.

More specifically, to address the issue related to the choice of accuracy metrics, we first describe new metrics, called *Template Accuracy* (TA) metrics, that, unlike GA and PA, are not sensitive to the number of log messages. We then discuss which metrics (GA, PA, or TA) are more appropriate depending on the nature of the software engineering task in which log message templates are used. The idea behind TA metrics is that template identification should be regarded as an information retrieval process in which message templates are identified from a collection of log messages. Therefore, based on standard information retrieval metrics, i.e., precision and recall, we define *Precision-TA* (PTA) and *Recall-TA* (RTA) metrics for template identification. As for determining oracle templates, we propose a set of heuristic rules for correcting oracle templates in order to minimize the negative bias in experimental results introduced by manually generated oracle templates. Furthermore, to provide additional information about incorrectly identified templates for a detailed analysis, we propose a technique for analyzing incorrect templates.

[1]The metric is called Parsing Accuracy in the original paper [26]; in this paper, we follow the naming convention proposed by Dai et al. [1].

From an empirical software engineering point of view, we assess the application of the proposed guidelines by investigating, using a benchmark composed of 14 existing template identification techniques, the following research questions:

RQ1: How does the ranking of techniques vary when using different accuracy metrics?

RQ2: What is the impact of oracle template correction on different accuracy metrics?

RQ3: Can the analysis of incorrect templates provide any insight to improve template identification techniques?

Significance. Log message template identification is an essential pre-processing step for automated log analysis research and practice. Therefore, adopting an adequate evaluation and comparison methodology for log message template identification is of great importance for both researchers and practitioners. The contributions of this paper can impact the field of log analysis by providing practical guidelines for accurately selecting the most adequate template identification techniques for a given software engineering application. Specifically, our empirical evaluation results show that following the guidelines is indeed critical in assessing and comparing the accuracy of log template identification techniques. The results also shed very different insights than existing studies and in particular a much less optimistic outlook on existing techniques. The insights will be useful for improving log template identification techniques in future research.

Paper structure. The rest of the paper is organized as follows. Section 2 reviews the state of the art and motivates our work with a running example. Section 3 describes the guidelines for assessing the accuracy of log template identification techniques. Section 4 reports on the evaluation results and Section 5 concludes the paper.

2 MOTIVATION

Given the number of available log template identification techniques, it is important to evaluate and rank them using all relevant criteria (e.g., accuracy, execution time); in this paper, we focus on the evaluation of the *accuracy*, informally defined as a measure of the ability of a technique to correctly identify log templates. In particular, our starting point are the two notable studies of Zhu et al. [26] and Dai et al. [1], which represent the state of art in empirical software engineering in terms of assessment of the accuracy of template identification techniques.

Zhu et al. [26] provided the first comprehensive evaluation of 13 log template identification techniques using 16 real-world log datasets. They made the replication package publicly available, including the implementation of the log template identification techniques, the real-world log datasets, and the oracle templates for the logs. Using these artifacts, Dai et al. [1] compared their new technique (called Logram) with the five techniques that achieved the highest accuracy scores reported by Zhu et al. [26]. Dai et al. [1] also proposed a new accuracy metric after raising an issue about the accuracy metric previously used by Zhu et al. [26]. Though both studies made essential steps towards a comprehensive assessment of the accuracy of log template identification techniques, we observed three important issues: (1) the choice of accuracy metrics, (2) determining oracle templates, and (3) incorrectly identified

templates. The three issues are discussed in detail in the following subsections.

2.1 The Choice of Accuracy Metrics

The choice of accuracy metrics is naturally of great importance in evaluating the accuracy of log template identification techniques. Initially, Zhu et al. [26] used the *Grouping Accuracy* (GA) metric to assess the accuracy of log template identification. The idea behind the GA metric is that template identification can be regarded as a clustering process in which log messages with different log events are clustered into different groups. Therefore, this metric checks if log messages that are grouped together by having the same identified template indeed form the same group as in the ground truth. Specifically, the GA metric is defined as the ratio of "correctly parsed" log messages (thanks to the identified templates) over the total number of log messages, where a log message is considered "correctly parsed" if and only if it is grouped with other log messages in a way consistent with the ground truth. However, Dai et al. [1] used another metric, called *Parsing Accuracy* (PA), since the GA metric does not consider whether the identified templates are identical to the oracle ones but only accounts for how the identified templates support the log message grouping activity. The PA metric is defined as the ratio of "correctly parsed" log messages over the total number of log messages (the same as for the GA metric), where a log message is considered to be "correctly parsed" if and only if "all its static text and dynamic variables (i.e., fixed and variables parts) are correctly identified" [1, p. 7]. As a result, the GA and PA metrics return different results in assessing the accuracy of log template identification techniques.

To better understand the GA and PA metrics, let us consider the scenario in which a software engineer (or a researcher) wants to assess the accuracy of two template identification techniques (called A and B), using the set of example log messages $M_{ex} = \{m_1, \ldots, m_6\}$ and the corresponding set of oracle (i.e., ground truth) templates $O = \{o_1, \ldots, o_5\}$ in Table 1; the example is based on a simplified version of real log messages and templates extracted from the log datasets provided by Zhu et al. [26]. Running the implementation of each of the two techniques on M_{ex} yields the corresponding set of identified templates $T_A = \{t_1^A, \ldots, T_4^A\}$ and $T_B = \{t_1^B, \ldots, T_5^B\}$, as shown Table 1. Notice that one template matches one or more messages, e.g., template t_1^A matches both m_1 and m_2.

According to the definition of correctly parsed log messages adopted for computing the GA metric, messages m_1 and m_2 can be correctly parsed by technique A because , according to t_1^A, they are exactly grouped together as they would be by o_1 (i.e., the oracle template of m_1 and m_2). However, message m_5 cannot be correctly parsed using t_3^A since the latter groups m_5 with another message m_4 while o_4 (i.e., the oracle template of m_5) does not. For readability, the correctly parsed log messages are marked under column *GA* in Table 1. Overall, only four messages (i.e., m_1, m_2, m_3, and m_6) out of six can be correctly parsed using the templates in the set $T_A = \{t_1^A, \ldots, t_4^A\}$, resulting in a GA score for technique A equal to $\frac{4}{6} \approx 0.67$. Similarly, the GA score for technique B is $\frac{2}{6} \approx 0.33$ since only two messages (i.e., m_3 and m_6) out of six can be correctly parsed using the templates in the set $T_B = \{t_1^B, \ldots, t_5^B\}$. Just by looking at the GA score, one could think that technique A is better than technique B.

However, even if template t_4^A can be used to correctly parse message m_6 in terms of the GA metric, it is quite different from oracle template o_5. More specifically, template o_5 contains string `adr` in the fixed part, informally suggesting that this template matches log messages that record different values for the memory location `adr`. On the other hand, the fixed part of template t_4^A is represented by string `0xff`, meaning that this template will match log messages that record the value `0xff` at any memory location. Such a limitation of the GA metric is addressed in the PA metric, which considers both the fixed and variable parts of identified templates. In the same running example, the PA score for technique A is $\frac{2}{6} \approx 0.33$ since two messages (i.e., m_1 and m_2) out of six can be correctly parsed by the template t_1^A (since t_1^A is identical to its oracle template o_1 in both fixed and variable parts). Similarly, the PA score for technique B is $\frac{2}{6} \approx 0.33$ since two messages (i.e., m_3 and m_6) out of six can be correctly parsed by the templates t_3^B and t_5^B. For readability, the log messages correctly parsed in terms of PA are marked under column *PA* in Table 1. By looking at the PA score, one could think that both of the techniques A and B are equally accurate in log template identification, which is clearly different from the result obtained when using the GA metric.

While the GA and PA metrics are clearly different, both metrics share the same issue: they are sensitive to the number of messages contained in the log (and not to the number of templates). This can be problematic for systems whose logs contain repeated log messages that are not important in terms of the system's business logic. For example, if a log contains many heartbeat messages that are repeated every second, then the GA and PA scores of a template identification technique can appear sufficiently high even if the technique is only able to correctly identify one template for the heartbeats. In our running example, we can already see that, though technique A correctly identifies only one template (i.e., t_1^A), the PA score is $\frac{2}{6} \approx 0.33$ since two messages (i.e., m_1 and m_2) out of six can be correctly parsed by the template t_1^A. If we focus on the number of correctly identified templates, technique B is "better" than technique A since technique B correctly identifies two templates (t_3^B and t_5^B).

We want to note that choosing a template identification technique based on the GA metric is not an issue per se when a template identification technique is only used for grouping log messages based on the identified templates. For example, for performance anomaly detection [8, 17], it is sufficient to monitor occurrence patterns (e.g., a repetition of the same event in a short period of time) of the (events corresponding to) log messages, without considering the message parameter values. However, in other scenarios in which the parameter values of messages (e.g., 1 in m_1) matter—such as model inference with guard conditions [24] and advanced anomaly detection using parameter values [3]—it is important to choose a log template identification technique based on another accuracy metric that accounts for the correctness of both fixed and variable parts in identified templates. § 3.1.3 further describes how to choose appropriate metrics in detail.

To address the limitations of the GA and PA metrics, in section 3.1 we propose to use alternative metrics for assessing the accuracy of template identification techniques. We then replicate

Table 1: Log messages, oracle templates, and templates identified by two techniques *A* and *B* in our running example

	Template			GA		PA	
Message (M_{ex})	Technique A (T_A)	Technique B (T_B)	Oracle (O)	Technique A	Technique B	Technique A	Technique B
(m_1) `retry 1`	(t_1^A) `retry <*>`	(t_1^B) `retry 1`	(o_1) `retry <*>`	✓	✗	✓	✗
(m_2) `retry 2`	(t_1^A) `retry <*>`	(t_2^B) `retry 2`	(o_1) `retry <*>`	✓	✗	✓	✗
(m_3) `x is 3.5`	(t_2^A) `x is <*>.<*>`	(t_3^B) `x is <*>`	(o_2) `x is <*>`	✓	✓	✗	✓
(m_4) `tot 4 MB`	(t_3^A) `<*> <*> MB`	(t_4^B) `<*> <*> MB`	(o_3) `tot <*> MB`	✗	✗	✗	✗
(m_5) `usd 1 MB`	(t_3^A) `<*> <*> MB`	(t_4^B) `<*> <*> MB`	(o_4) `usd <*> MB`	✗	✗	✗	✗
(m_6) `adr=0xff`	(t_4^A) `<*>=0xff`	(t_5^B) `adr=<*>`	(o_5) `adr=<*>`	✓	✓	✗	✓

the comprehensive evaluations of Zhu et al. [26] and Dai et al. [1] in section 4.2, additionally using different accuracy metrics, including the proposed alternative ones, to better understand how and why the ranking of log template identification techniques vary when using different techniques.

2.2 Determining Oracle Templates

Oracle templates are essential to evaluate the accuracy of template identification techniques. Both of the existing studies used the same oracle templates generated by Zhu et al. [26]. The oracle templates were determined by manually inspecting log messages. However, when we compare them with the corresponding log printing statements in the source code, the oracle templates are not always correct. For example, for a log message "`status is false`" generated by a log printing statement `log.info("status is " + var)`, an engineer with no access to the source code has mistakenly defined an oracle template "`status is false`" instead of "`status is <*>`", by incorrectly assuming that the token "`false`" was hard-coded in the log printing statement.

Nevertheless, manually determining oracle templates (in the context of the study by Zhu et al. [26]) was unavoidable since some of the benchmark log datasets were collected from real-world systems that do not provide access to the source code. Furthermore, from a more pragmatic perspective, this strategy is acceptable considering the fact that manually generated oracle templates are, in most cases, very similar to the actual templates extracted from the source code. However, the impact of using slightly incorrect oracle templates on the assessment of the accuracy of template identification techniques may vary depending on the accuracy metrics used. For example, the GA metric only accounts for how log messages are grouped and does not consider whether the Technique Under Evaluation (TUE) correctly identifies templates by accurately decomposing fixed and variable parts; therefore, one could speculate that the impact of using slightly incorrect oracle templates could be limited. Proving such assumptions is an open problem.

To correct such potentially-incorrect oracle templates without accessing the source code, in section 3.2, we propose a set of heuristic rules that can automatically correct many oracle templates. Furthermore, we empirically investigate the impact of using the oracle template correction on different accuracy metrics in section 4.3.

2.3 Incorrectly Identified Templates

Regardless of the accuracy metric used, simply ranking the log template identification techniques by their accuracy scores may not be sufficient. It is also important to analyze why the techniques incorrectly identify some templates. For example, let us consider a technique that has a low accuracy score only because it does not correctly recognize IP addresses as variable parts. In practice, such an issue can be easily addressed by modifying the TUE to support user-defined regular expressions for pre-processing structured text strings like IP addresses. However, without additional information about incorrectly identified templates, the TUE could be perceived as being severely inaccurate, even if this is not really the case.

However, both existing studies did not provide any information regarding the templates incorrectly identified by template identification techniques. Such missing information hinders the possibility to improve existing techniques by addressing the limitations discovered by experimental assessment.

To address this issue, in section 3.3, we propose a technique for analyzing incorrect templates that can help software engineers and researchers understand in which way templates are incorrectly identified. We also discuss the insights one can get by applying the proposed analysis technique to the empirical evaluation results presented in section 4.4.

3 GUIDELINES

In this section, to address the issues discussed in section 2, we present three guidelines for assessing the accuracy of log template identification techniques:

(1) Do use appropriate accuracy metrics (§ 3.1),
(2) Do perform oracle template correction (§ 3.2), and
(3) Do perform analysis of incorrect templates (§ 3.3).

3.1 Do Use Appropriate Metrics

As discussed in section 2.1, since existing accuracy metrics rely on different definitions for "correctly parsed" log messages, it is essential to use a metric that adequately assesses the accuracy of template identification techniques in the context of the targeted use cases. Before we present our guidelines for choosing an appropriate accuracy metric considering a use case, we first introduce new metrics to address the common limitations of the existing metrics.

3.1.1 Template Accuracy (TA) Metrics. Recall that both GA and PA are sensitive to the number of messages contained in the log, which is possibly misleading when there are many repetitive yet useless messages, commonly found in real-world system logs. To address this, we propose alternative accuracy metrics that take into account how many *templates* are "correctly" identified by the TUE.

First, we define the concept of *correctly identified* templates. A template (determined by a template identification technique) is *correctly identified* from log messages if and only if it is identical

(token-by-token) to the oracle template(s) of the log messages. In other words, the correctness of an identified template must be determined by comparison to its "corresponding" oracle template(s).

More precisely, let M be a set of log messages, O be the set of oracle templates for M, and T_v be a set of templates for M identified by a TUE v. We introduce two auxiliary functions: $msg\colon T_v \rightarrow 2^M$, which, given an identified template t, returns the set of messages from M whose template is t; $ot\colon M \rightarrow O$, which, given a message m, returns the oracle template of m included in O. We define the *set of corresponding oracle templates in O for a template t*, denoted and defined as $corrOT(t) = \{ot(m) \mid m \in msg(t)\}$. For instance, in our running example M_{ex}, O_{ex}, and T_A shown in Figure 1, $corrOT(t_1^A)$ is $\{o_1\}$ because $msg(t_1^A) = \{m_1, m_2\}$ and $ot(m_1) = ot(m_2) = o_1$.

Based on the definition of corresponding oracle templates, we say that a template $t \in T_v$ is *correctly identified* (*correct*, for simplicity) by TUE v if and only if $corrOT(t) = \{t\}$; otherwise, we say t is *incorrectly identified* (*incorrect*, for simplicity) by v. For the running example above, t_1^A is correct since $corrOT(t_1^A) = \{o_1\} = \{t_1^A\}$; on the other hand, t_3^A is incorrect since $corrOT(t_3^A) = \{o_3, o_4\} \neq \{t_3^A\}$.

Notice that our definition of correctly identified template relies on the set of oracle templates corresponding to a certain message. It is not enough that the identified template is included in the set of oracle templates O; to be correct, the identified template has to be identical to the one and only oracle template *corresponding* to the messages for which it was identified. For instance, in our running example, even if there were an additional oracle template $o_6 \in O'_{ex}$ (where $O'_{ex} = O_{ex} \cup \{o_6\}$) that is identical to t_4^A such that $t_4^A \in T_A \cap O'_{ex}$, t_4^A could not be considered correct because $corrOT(t_4^A) = \{o_5\} \neq \{t_4^A\}$.

Using the definition of correctly identified template, we introduce two TA metrics: *Precision-TA* (PTA) and *Recall-TA* (RTA) — collectively denoted by PTA‡RTA — which are based on the standard metrics *precision* and *recall* used in the information retrieval domain. PTA is defined as $\frac{|\{t \in T_v | corrOT(t) = \{t\}\}|}{|T_v|}$, the *ratio of correctly identified templates over the total number of identified templates*, indicating the *precision* of the TUE at the template level. RTA is defined as $\frac{|\{t \in T_v | corrOT(t) = \{t\}\}|}{|O|}$, the *ratio of correctly identified templates over the total number of oracle templates*, indicating the *recall* of the TUE at the template level.

These two metrics range between 0 and 1. For our running example, in the case of technique A, its PTA score is $\frac{1}{4} = 0.25$ because only one template (i.e., t_1^A) out of the four in T_A is correct; the RTA score is $\frac{1}{5} = 0.2$ since the total number of oracle templates in O_{ex} is five. In the case of technique B, its PTA score is $\frac{2}{5} = 0.4$ because two templates (i.e., t_3^B and t_5^B) out of the five in T_B are correct; the RTA score is $\frac{2}{5} = 0.4$.

3.1.2 Relationship among the Metrics. There are interesting relationships among GA, PA, and PTA‡RTA. Specifically, when PTA = RTA = 1 we have GA = 1 because log message groups determined by oracle templates must be the same as the groups determined by identified templates, given that all identified templates are correct and the total number of identified templates is the same as the number of oracle templates. For the same reason, when PA = 1 we have GA = 1. On the other hand, having GA = 1 does not imply PTA = 1, RTA = 1, or PA = 1 since completely incorrect templates can group messages in the same way as oracle templates do (like t_4^A in our running example). Meanwhile, since PTA‡RTA and PA consider the correctness of fixed and variable parts of identified templates (though PA counts correctly parsed log messages while PTA‡RTA count correctly identified templates), having PTA = RTA = 1 imply PA = 1, and vice versa.

3.1.3 How to Choose Appropriate Metrics. Given a target use case, one should choose an appropriate metric to evaluate the accuracy of the TUE. The two important criteria to consider when performing this choice are (1) whether the variable parts of log messages are used and (2) whether the importance of a message is proportional to its frequency.

When the variable parts of log messages are not used in the target use case, then one should use the GA metric. For example, the log key anomaly detection model of DeepLog [3] aims to detect behavioral anomalies using the system's event sequences recorded in the log. It uses log template identification to convert a sequence of log messages into a sequence of log keys (i.e., template ids). Therefore, the GA metric — that only considers message grouping — is the best to assess the accuracy of template identification in this use case.

When the variable parts of log messages matter in the target use case, one should additionally consider the second criterion above: if the importance of a log message is proportional to its frequency, then PA should be used; otherwise, PTA‡RTA should be used. This is because PA is sensitive to the proportion of log messages whereas PTA‡RTA are not. For example, DeepLog [3] has another model for detecting irregular parameter values, which is not detected by the above log key anomaly detection model. Since the irregular parameter detection model needs different parameter values (i.e., variable parts) of log messages accurately extracted by log template identification, GA cannot be used to adequately assess the accuracy of template identification in this case. Furthermore, if the importance of system events recorded in the log does not depend on the frequency of the event messages, then using PA can yield misleading accuracy evaluation results, as PA counts the number of log messages parsed by correctly identified templates. In this case, PTA‡RTA should be used to assess the accuracy of the TUE.

To summarize, by evaluating the two aforementioned criteria in the context of the target use cases, one can choose an appropriate metric among GA, PA, and PTA‡RTA to assess the accuracy of log template identification techniques.

3.2 Do Perform Oracle Template Correction

As discussed in section 2.2, when oracle templates are manually determined (e.g., because there is no access to the source code), they may be incorrect. To address this issue, we propose to perform oracle template correction using heuristic rules.

3.2.1 Heuristic-based Oracle Template Correction. The idea of correcting oracle templates is based on our analysis of manually determined oracle templates provided by Zhu et al. [26], a publicly available, widely-used template identification benchmark. After a detailed analysis of the oracle templates, we noticed that some of them

contain fixed parts that are unlikely to be hard-coded in log printing statements. For example, one of the oracle templates for the Hadoop system in LogHub is "`nodeBlacklistingEnabled:true`"; hard-coding the value "`true`" in the log printing statement seems wrong. Indeed, we checked the source code of Hadoop and found that the actual log printing statement is "`LOG.info("nodeBlacklisting-Enabled:" + nodeBlacklistingEnabled)`", meaning that the correct oracle template is "`nodeBlacklistingEnabled:<*>`". Note that such an incorrect oracle template is not an issue for the GA metric, as long as the incorrect oracle template can group log messages in the same way as the correct oracle template does. However, the presence of such an incorrect oracle template may introduce a bias in the computation of the PA and TA metrics, which directly compare identified and oracle templates. Therefore, we propose a set of heuristic rules, based on the manual investigation of the oracle templates in LogHub, that converts fixed parts into variable parts using regular expressions. The regular expressions are provided in our replication package (see § 4.7).

Table 2 shows the rules, their descriptions, and simple application examples. The DS (Double Space) rule replaces any double (or more) whitespace with a single whitespace to eliminate trivial whitespace differences between oracles and identified templates. The DG (DiGit), BL (BooLean), PS (Path String), and US (User-defined String) rules replace a fixed token, such as digits, Boolean values, paths, and user-specific strings, with the placeholder symbol "`<*>`"; the rationale behind these rules is that the type of token they try to represent is rarely hard-coded in log printing statements. Unlike the others, rule US is made for users who can indicate specific strings that should be replaced with "`<*>`". For example, a string "`idle`" could be hard-coded in a log printing statement in a system, such as `log.info("system status is idle")`; however, for another system, "`idle`" could be a value for variable "`ret`" in a log printing statement like `log.info("status=" + ret)`. If all log messages containing "`idle`" have the form "`...=idle`", one could be drawn to consider token "`idle`" as a variable part in the template. In this case, "`idle`" can be indicated as one of the user-defined strings for the application of rule US. The MT (Mixed Token), CV (Consecutive Variables), and DV (Dot-separated Variables) rules replace a token containing variable parts (such as "`v<*>`", "`<*><*>`", and "`<*>.<*>`") with "`<*>`" because the latter catches all the strings caught by the former, without loss of information. For instance, "`v<*>`" is used in templates to indicate version information (e.g., "`v2`"), "`<*><*>`" usually indicates a number with a unary operator (e.g., "`-1`"), and "`<*>.<*>`" usually indicates a floating-point number (e.g., "`0.5`"). In all cases, these tokens can be simply replaced by "`<*>`", which captures all the strings captured by the original tokens.

Table 3 shows the number of corrected oracle templates for each dataset in the benchmark provided by Zhu et al. [26], when each of our heuristic rules is applied to the manually determined oracle templates (columns DS, DG, BL, PS, US, MT, CV, and DV); column $|O|$ shows the number of manually determined oracle templates for each dataset in the benchmark. In summary, for a total of 1363 oracle templates in the benchmark, each rule corrected from a minimum of 13 (CV) to a maximum of 316 (DV) oracle templates. We manually checked the soundness of all the corrected oracle templates. Nevertheless, we cannot guarantee that they are truly consistent with the corresponding log printing statements in the source code, which is unknown to us. Section 4.6 will further discuss this issue.

3.2.2 Using Oracle Template Correction. The proposed oracle template correction is meant to be used when oracle templates are manually determined (e.g., when the source code is inaccessible). In such a case, the oracle template correction should be used to make possibly error-prone oracle templates more consistent with the general form of log printing statements in the source code.

When manually determining oracle templates, the heuristic rules for oracle template correction can be already taken into account. For example, given a log message "`status is false`", one can identify an oracle template "`status is <*>`" by considering the rationale behind the BL rule.

3.3 Do Perform Analysis of Incorrect Templates

As discussed in section 2.3, simply ranking different log template identification techniques by their accuracy scores may not be sufficient, since such a ranking does not provide any insight about the reason for which a TUE got a certain accuracy score. We propose to perform an *analysis of incorrect templates* to further analyze the incorrectly identified templates to understand in which way they are incorrect; this analysis may provide insights to engineers and researchers on how to improve the TUE.

3.3.1 Analysis of Incorrect Templates. This analysis is based on the observation that template identification can also be seen as the process of generalizing log messages by converting fixed parts into variable parts. In our running example, shown in Figure 1, technique A generalizes the two messages m_1 ("`retry 1`") and m_2 ("`retry 2`") to the template t_1^A ("`retry <*>`") by converting the second token of both messages into a variable. Thus, incorrectly identified templates can be seen as incorrectly generalized templates. Based on this idea, we classify incorrect templates into three types: *Over-Generalized* (OG), *Under-Generalized* (UG), and *MiXed* (MX). By doing this, we can understand whether the TUE suffers from over-generalization, under-generalization, or both.

The core of this three-way classification is the *generalization* relationship between templates, which is defined in terms of the formal language defined by the templates. Since a template can be seen as a regular expression[2], we denote by $lang(t)$ the regular language defined by a template t. We say that a template t_x is *more general* than a template t_y if and only if $lang(t_y) \subset lang(t_x)$. In our running example, o_2 is more general than t_2^A because all potential log messages that match t_2^A will also match o_2 but the opposite does not hold (e.g., message "`x is 1`" is in $lang(o_2)$ but not in $lang(t_2^A)$).

Based on the generalization relationship between templates, we can classify an incorrectly identified template as OG, UG, or MX. Specifically, an incorrectly identified template t is classified as OG if t is more general than all oracle templates in $corrOT(t)$, i.e., t is always more general than its corresponding oracle templates. Similarly, t is classified as UG if all oracle templates in $corrOT(t)$ are more general than t, i.e., t is always less general than its corresponding oracle templates. If t is classified neither as OG nor UG,

[2]Recall that "`<*>`" represents any characters except whitespace.

Table 2: Heuristic rules for oracle template correction

Rule	Description	Example (before → after)
Double Spaces (DS)	Replace double spaces with a single space	`Input:__<*>` → `Input:_<*>`
Digit (DG)	Replace digit tokens with variables	`euid=0` → `euid=<*>`
Boolean (BL)	Replace True/False with a variable	`cancel=false` → `cancel=<*>`
Path String (PS)	Replace a path-like token with a variable	`/lib/tmp started` → `<*> started`
User-defined String (US)	Replace user-defined strings with variables	`status=idle` → `status=<*>`
Mixed Token (MT)	Replace a token containing both fixed and variable parts with a variable	`python v<*>` → `python <*>`
Consecutive Variables (CV)	Replace consecutive variables as a single variable	`value=<*><*>` → `value=<*>`
Dot-separated Variables (DV)	Replace dot separated variables as a single variable	`<*>.<*>` seconds → `<*>` seconds

Table 3: Number of oracle templates corrected using our heuristic rules for the benchmark provided by Zhu et al. [26]

Dataset	|O|	DS	DG	BL	PS	US	MT	CV	DV
HDFS	14	1	0	0	7	0	0	0	0
Hadoop	114	6	4	2	3	1	3	0	5
Spark	36	0	0	0	0	0	0	0	0
Zookeeper	50	1	0	0	10	0	1	0	0
OpenStack	43	0	0	0	0	0	0	0	22
BGL	120	0	2	0	6	0	2	0	6
HPC	46	7	0	0	0	0	3	0	0
Thunderbird	149	0	4	0	9	10	5	3	37
Windows	50	0	0	2	0	1	2	0	10
Linux	118	12	13	0	3	3	6	2	41
Mac	341	20	0	5	9	13	6	4	167
Android	166	10	0	63	0	12	0	4	26
HealthApp	75	3	0	10	0	0	0	0	2
Apache	6	0	0	0	0	0	0	0	0
OpenSSH	27	1	0	0	0	4	0	0	0
Proxifier	8	0	0	0	0	0	0	0	0
Total	1363	61	23	82	47	44	28	13	316

it is classified as MX, meaning t is over-generalized in some cases while under-generalized in others.

In our running example, template t_2^A is UG because $corrOT(t_2^A) = \{o_2\}$ and o_2 is more general than t_2^A. On the other hand, template t_3^A is OG because $corrOT(t_3^A) = \{o_3, o_4\}$ and t_3^A is more general than both o_3 and o_4. Template t_4^A is MX because $corrOT(t_4^A) = \{o_5\}$ and neither t_4^A nor o_5 is more general than the other. Note that, with respect to o_5, t_4^A is over-generalized because of the position of `<*>` but is also under-generalized because of the presence of token `0xff`. Based on this analysis, the three incorrect templates of technique A cover all the three types, meaning that technique A needs to be improved in all aspects.

3.3.2 Using Incorrect Template Analysis. The proposed incorrect template analysis can be performed without requiring additional inputs other than oracle templates and identified templates, which are already used to compute accuracy scores. Furthermore, as it is based on the formal definition of the generalization relationship between templates, it can be easily automated. We remark that the proposed incorrect template analysis is independent from the choice of accuracy metrics.

4 EVALUATION

In this section, we assess the application of the guidelines proposed in the previous section, using 14 existing template identification techniques. Specifically, we answer the following research questions, already introduced in Section 1:

RQ1: How does the ranking of techniques vary when using different accuracy metrics?
RQ2: What is the impact of oracle template correction on different accuracy metrics?
RQ3: Can the analysis of incorrect templates provide any insight to improve template identification techniques?

4.1 Benchmark and Settings

To answer the research questions, we used the publicly available benchmark proposed by Zhu et al. [26] to assess different template identification techniques. This benchmark is based on the LogHub benchmark [7], which contains a large collection of log messages from 16 real-world systems including distributed systems, operating systems, mobile systems, and standalone programs. To determine the ground truth in terms of templates, Zhu et al. [26] randomly sampled 2000 messages for each system and manually labeled them with oracle templates. Table 4 provides an overview of the 16 datasets (each of them containing 2000 log messages and the corresponding manually generated templates) in the benchmark; column $|O|$ shows the number of oracle templates for each dataset. To support reproducibility, Zhu et al. [26]'s benchmark also includes the implementation of 13 log template identification techniques (i.e., AEL [9], Drain [6], IPLoM [11], LenMa [19], LFA [16], LKE [4], LogCluster [22], LogMine [5], LogSig [20], MoLFI [14], SHISO [15], SLCT [21], Spell [2]).

In our evaluation, we used the same logs, oracle templates, implementations of template identification techniques, and parameter settings (for each template identification technique) used by Zhu et al. [26]. We also added the implementation of Logram [1] to our benchmark and used the parameter values suggested by the authors. In total, we considered 14 log template identification techniques. Additionally, as discussed in § 3.2, the use of the US (User String) rule requires user-defined strings as input. Based on our analysis of the templates in the Zhu et al. [26]'s benchmark, we used the following tokens as user-defined strings: `null`, `root`, and `admin`.

Table 4: Log datasets (each dataset has 2000 messages, |O| indicates the number of oracle templates)

Dataset	Description	$\|O\|$
HDFS	Hadoop distributed file system log	14
Hadoop	Hadoop mapreduce job log	114
Spark	Spark job log	36
ZooKeeper	ZooKeeper service log	50
OpenStack	OpenStack software log	43
BGL	Blue Gene/L supercomputer log	120
HPC	High performance cluster log	46
Thunderbird	Thunderbird supercomputer log	149
Windows	Windows event log	50
Linux	Linux system log	118
Mac	Mac OS log	341
Android	Android framework log	166
HealthApp	Health app log	75
Apache	Apache server error log	6
OpenSSH	OpenSSH server log	27
Proxifier	Proxifier software log	8

4.2 RQ1: Ranking of Techniques

Methodology. To answer RQ1, we executed the 14 log template identification techniques on each dataset in the benchmark and used their output to compute the accuracy in terms of the GA, PA, and TA (PTA and RTA) metrics. For each execution (of a technique on a dataset), we set a timeout of 24 hours. To account for the randomness of LKE and MoLFI, we repeated the corresponding experiments 30 times and computed the average results. Notice that we did not use oracle template correction because it will be investigated in RQ2.

Results. All techniques completed their execution on all datasets, except in one case: the execution of IPLoM on the Android dataset timed out. Figure 1 shows the GA, PA, and PTA‡RTA score distributions of the individual techniques across all datasets for which the techniques completed their execution. Overall, it is clear that GA scores tend to be higher than PA and PTA‡RTA scores for all techniques. This means that, for all techniques, there are many identified templates that are incorrect but happen to be able to group log messages in a way consistent with the ground truth. This implies that the GA metric is inadequate to measure the accuracy of a template identification technique when identifying correct templates (both fixed and variable parts) is important.

The left-hand side of Table 5 shows the ranking of the different techniques in terms of the GA, PA, and PTA‡RTA scores. The techniques are sorted, in descending order, by average accuracy score (computed over the 16 datasets). We can see that Drain is ranked 1st for GA and PTA‡RTA, but not for PA. In other words, Drain outperforms all other techniques, both at correctly grouping messages and at identifying correct templates. However, SLCT outperforms Drain in terms of PA, meaning that SLCT is better than Drain at identifying correct templates when the number of log messages corresponding to the correctly identified templates is also considered. Notice that SLCT is ranked 10th for GA but 1st for PA. This is not the only case where the ranking of the techniques differs depending on the metric used; for example, LenMa is better than AEL in terms of GA, but AEL is better than LenMa in terms of PA and PTA‡RTA. This happens because LenMa correctly groups more log messages than AEL, even though AEL correctly identifies more templates than LenMa. Such results strongly imply that the choice of the accuracy metric matters a lot when comparing log template identification techniques.

To assess how the ranking of the techniques varies when using the different accuracy metrics, we measured Spearman's rank correlation between the rankings of each pair of accuracy metrics. A rank correlation coefficient ρ ranges between 0 and 1; the more similar the compared rankings, the higher the value of ρ. We found that, though the resulting rank correlation varies depending on the accuracy metrics compared, ranging from 0.455 (between PA and GA) to 0.846 (between GA and RTA), the average rank correlation remains moderate (0.666), implying that the ranking of different log template identification techniques can significantly vary depending on the accuracy metrics used.

To see whether the difference between Drain and the other individual techniques is statistically significant in terms of the GA and PTA‡RTA metrics, we additionally performed the Wilcoxon signed-rank test [25], which is a non-parametric statistical hypothesis test for paired samples (i.e., in our context, pairs of techniques' accuracy scores across log datasets). Given a level of significance $\alpha = 0.05$, Drain is significantly better than the others only in terms of GA; the p-values comparing Drain and AEL in terms of PTA and RTA are 0.1519 and 0.0663, respectively. As a result, we can conclude that Drain is preferable only in terms of correctly grouping log messages as it is significantly better than the other techniques with respect to GA. However, we cannot say that Drain is significantly better than the others in terms of identifying correct templates.

To conclude, the answer to RQ1 is that the ranking of template identification techniques can vary a lot depending on the choice of accuracy metrics, since different metrics consider different aspects of template identification. When accuracy is measured using the PA and PTA‡RTA metrics, all template identification techniques achieve a low accuracy score (less than 50%), thus providing a much worse assessment than what was reported in existing studies [1, 26]. This implies that empirical studies using the GA metric were too optimistic due to the use of message-level grouping for calculating the accuracy scores of template identification techniques. This therefore calls for further research on accurate template identification techniques; some insights on how to improve existing techniques can be drawn from the results of our analysis of incorrect templates. In section 4.4, we will show the analysis results and discuss the insights that can be drawn from them.

4.3 RQ2: Impact of Oracle Template Correction

Methodology. To answer RQ2, we assessed how the GA, PA, and PTA‡RTA scores change when using oracle template correction. To compute the accuracy scores, we followed the same methodology and settings used for RQ1, and *corrected the oracle templates* applying the rules described in § 3.2. Then, we calculated the difference between the accuracy scores obtained, hereafter denoted with the subscript X_{otc} (where $X \in \{GA, PA, PTA, RTA\}$), and those computed as part of answering RQ1, hereafter denoted with the subscript X_{org}. We computed the following differences:

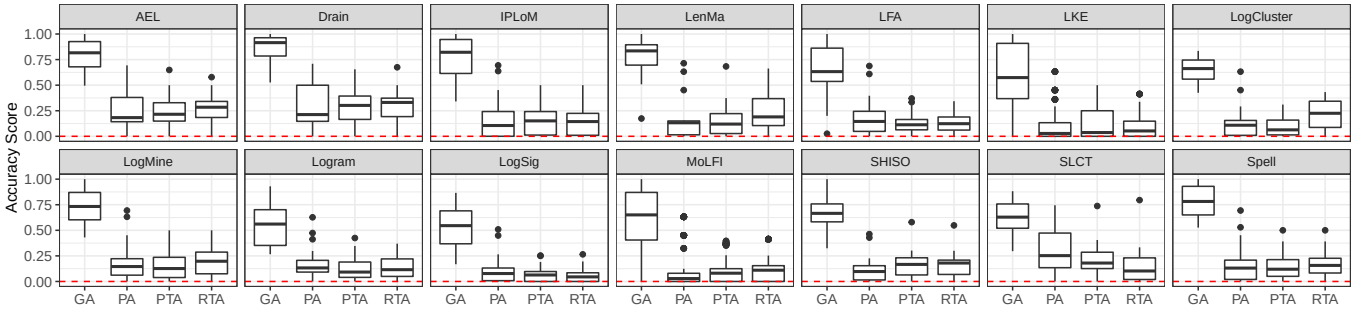

Figure 1: Accuracy results using the different metrics

Table 5: Ranking of log identification techniques based on the GA, PA, and PTA‡RTA metrics

Ranking without Oracle Template Correction								Ranking with Oracle Template Correction							
GA		PA		PTA		RTA		GA		PA		PTA		RTA	
Technique	Score	Technique	Score	Technique	Score	Technique	Score	Technique	Score	Technique	Score	Technique	Score	Technique	Score
Drain	0.87	SLCT	0.30	Drain	0.27	Drain	0.29	Drain	0.86	Drain	0.34	Drain	0.29	Drain	0.31
Spell	0.79	Drain	0.29	AEL	0.25	AEL	0.27	AEL	0.80	AEL	0.28	AEL	0.24	AEL	0.27
AEL	0.79	AEL	0.26	SLCT	0.22	LenMa	0.23	Spell	0.79	SLCT	0.27	SLCT	0.19	LenMa	0.22
LenMa	0.77	LogMine	0.20	SHISO	0.16	LogCluster	0.22	LenMa	0.76	LFA	0.24	Spell	0.16	LogCluster	0.19
IPLoM	0.76	LFA	0.20	LenMa	0.16	LogMine	0.20	IPLoM	0.76	LogMine	0.23	IPLoM	0.16	LogMine	0.19
LogMine	0.74	Logram	0.19	IPLoM	0.16	Spell	0.17	LogMine	0.74	Spell	0.20	LenMa	0.15	Spell	0.19
SHISO	0.68	IPLoM	0.19	LogMine	0.16	SHISO	0.17	SHISO	0.68	IPLoM	0.19	SHISO	0.15	SHISO	0.16
LogCluster	0.65	Spell	0.18	Spell	0.15	SLCT	0.16	LogCluster	0.65	Logram	0.17	LogMine	0.15	IPLoM	0.15
LFA	0.64	LenMa	0.18	LFA	0.14	IPLoM	0.15	LFA	0.65	LenMa	0.16	LFA	0.15	LFA	0.15
SLCT	0.63	LogCluster	0.15	Logram	0.13	Logram	0.14	LKE	0.63	LogCluster	0.13	Logram	0.13	Logram	0.14
MoLFI	0.62	SHISO	0.13	LKE	0.12	LFA	0.14	SLCT	0.63	LogSig	0.11	LKE	0.10	SLCT	0.14
LKE	0.56	LogSig	0.13	LogCluster	0.10	MoLFI	0.11	MoLFI	0.63	SHISO	0.11	LogCluster	0.09	MoLFI	0.11
Logram	0.55	MoLFI	0.09	MoLFI	0.09	LKE	0.09	Logram	0.55	LKE	0.09	MoLFI	0.08	LKE	0.10
LogSig	0.53	LKE	0.08	LogSig	0.08	LogSig	0.07	LogSig	0.53	MoLFI	0.09	LogSig	0.07	LogSig	0.06

$\Delta^{GA}_{otc} = GA_{otc} - GA_{org}$, $\Delta^{PA}_{otc} = PA_{otc} - PA_{org}$, $\Delta^{PTA}_{otc} = PTA_{otc} - PTA_{org}$, $\Delta^{RTA}_{otc} = RTA_{otc} - RTA_{org}$.

Results. Figure 2 shows the distributions of the Δ^{GA}_{otc}, Δ^{PA}_{otc}, Δ^{PTA}_{otc}, and Δ^{RTA}_{otc} values for the individual techniques on all datasets, except IPLoM whose execution timed out on the Android dataset. For all $X \in \{GA, PA, PTA, RTA\}$, the difference between X_{otc} and X_{org} is statistically insignificant across log datasets based on the Mann–Whitney U test [12], with a level of significance $\alpha = 0.05$. This means that the impact of the oracle template correction on the GA, PA, and PTA‡RTA metrics is statistically insignificant, though there are some outliers as visible in the figure.

Given that many oracle templates (28.5% on average for all datasets) were incorrect and thus modified by applying oracle template corrections, it is surprising to see such a limited impact of the corrections. Through a more thorough analysis, we found that this was the case because many templates identified by the techniques are different enough from the corresponding oracle templates to such an extent that they are "incorrect" regardless of the corrections that can be applied to the oracle templates. This implies that the impact of oracle template corrections would be much larger if the techniques fared better at correctly identifying templates.

Notice that Δ^{GA}_{otc} is non-zero for many techniques, even though the GA metric ignores whether the identified templates are correct in terms of their fixed and variables parts. This is because the number of oracle templates (and therefore the number of log message groups generated by the oracle templates) can be changed by the oracle template correction step when two or more incorrect oracle templates become the same oracle template, after applying a correction.

Though the impact of the oracle template correction on the metrics is limited in terms of accuracy scores, the ranking of the techniques varies due to oracle template correction. The right-hand side of Table 5 shows the ranking of the different techniques based on the accuracy scores with corrected oracle templates. We can see that Drain is now ranked 1st independently from the metric used to determine accuracy. Furthermore, based on the Wilcoxon signed-rank test [25] with a significance level $\alpha = 0.05$, the difference between Drain and the other individual techniques is statistically significant for all the metrics. Compared to the results without the oracle template correction, the difference between Drain and the others becomes statistically significant for PA and PTA‡RTA, mainly because the score of Drain has increased more than that of the other techniques; for example, through the oracle template

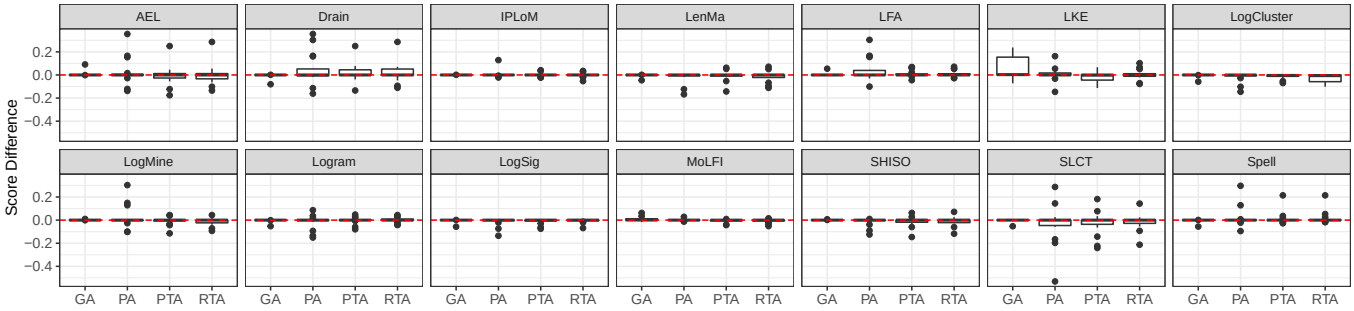

Figure 2: Impact of Oracle Template Correction

correction, the PA score of Drain has increased from 29% to 34% while that of SLCT has decreased from 30% to 27%. This happens because Drain identifies more templates that are identical to correct oracle templates, whereas SLCT identifies more templates that are identical to slightly incorrect oracle templates. As a result, we can conclude that Drain is preferable in all circumstances if the oracle template correction is used.

The answer to RQ2 is that, for all template identification techniques, enabling the oracle template correction has a limited impact on their accuracy scores in terms of GA, PA, and PTA‡RTA metrics. This is mainly because the templates identified by the techniques are, in general, quite different from oracle templates and therefore oracle template corrections have a limited effect. In other words, the limited impact of corrections are due to the relatively poor accuracy of the techniques, to various degrees. Nevertheless, the ranking of some template identification techniques, most particularly that of the best ones, changes when applying the oracle template correction. This implies that the oracle template correction may indeed be important for properly ranking template identification techniques based on their accuracy, especially when the latter tends to be high.

4.4 RQ3: Insights from the Analysis of Incorrect Templates

Methodology. To answer RQ3, we followed the same methodology and settings used for RQ1; in addition, we classified incorrectly identified templates into OG (Over-Generalized), UG (Under-Generalized), and MiXed (MX) types, as described in § 3.3.

Results. Table 6 shows the average percentage of OG, UG, and MX templates for the individual techniques on all datasets, except IPLoM whose execution timed out on the Android dataset. We can see that different techniques have different distributions of OG, UG, and MX types. The results provide insights on the main limitations of the techniques and how to improve them. For example, LFA generated over-generalized (OG) templates in 52.7% of the cases; in other words, LFA's main limitation is over-generalization. This information can then be used to improve the algorithm, e.g., by adjusting it to make it less likely to convert fixed tokens into variables. On the other hand, LogCluster — which generated under-generalized (UG) templates in 72.9% of the cases — could be improved by making it more likely to convert fixed tokens into variables. Techniques with many MX templates, such as MoLFI (82.5%), LogSig (77.7%), and IPLoM (69.0%), generated both under- and over-generalized templates; they require a more thorough analysis to determine where to intervene for improving the underlying algorithms.

Table 6: Average Percentage of OG, UG, and MX Types

Technique	OG (%)	UG (%)	MX (%)
AEL	21.7	38.2	15.3
Drain	19.4	32.6	20.6
IPLoM	4.6	10.5	69.0
LFA	52.7	16.2	17.5
LKE	0.1	32.8	55.4
LenMa	5.5	44.5	33.8
LogCluster	0.0	72.9	17.0
LogMine	6.8	36.0	41.4
LogSig	0.1	14.2	77.7
Logram	27.3	26.4	33.2
MoLFI	0.0	8.4	82.5
SHISO	6.4	44.4	32.8
SLCT	17.5	28.9	31.7
Spell	7.7	13.1	64.2

To conclude, the answer to RQ3 is that the analysis of incorrect templates can provide insights to improve template identification techniques by highlighting the limitations of individual techniques through the percentages of OG, UG, and MX types among identified templates.

4.5 Discussion

In RQ1, the results obtained for the GA metric are exactly the same as those computed by the replication package[3] of the original study [26]. However, the results obtained for the PA metric are quite different from those reported by Dai et al. [1]. For example, on average for all datasets, Drain achieved a PA score of 29% (without oracle template correction, see Table 5), which is much lower than the PA score of 75% reported by Dai et al. [1]. We could not reproduce their results since (1) they manually checked the parsing results to calculate PA values and (2) their replication package[4] contains an accuracy evaluation script only for calculating GA scores, not PA scores.

[3]https://github.com/logpai/logparser (accessed: 2021-07-21)
[4]https://github.com/BlueLionLogram/Logram (accessed: 2021-07-21)

One of the most surprising results from our evaluation is the significant gap between GA and the other metrics in terms of accuracy scores for the same template identification techniques; for example, on average for all datasets, Drain achieved a GA score of 87%, but its PA and PTA‡RTA scores were only 29%, 27%, and 29%, respectively. In other words, by taking into account the correctness of fixed and variable parts of identified templates, we get a very different picture of the accuracy of the techniques in comparison to the existing study [26] relying on the GA metric. The results emphasize the importance of using adequate metrics to assess template identification results in the context of target use cases. For example, if the parameter values of log messages are used by a log-based analysis task, e.g., model inference with guard conditions [24] and advanced anomaly detection using parameter values [3], accuracy should be assessed using PA or TA scores since they adequately measure a technique's ability to correctly identify the variable parts of log messages. However, one might argue that PA and TA metrics are too strict to fairly indicate the usefulness of template identification techniques as no degree of correctness is accounted for individual templates, even for the cases where the variable parts of log messages matter. More studies are therefore required to better understand the relationship between various accuracy scores of template identification techniques and their actual usefulness in specific applications.

For some of the techniques (e.g., Drain), the importance of oracle template corrections is confirmed by the evaluation results. However, overall, the impact of such corrections on accuracy scores is limited. Based on our analysis, we reckon that the more accurate the technique, the more likely its ranking is to be significantly affected by oracle template corrections. Since all techniques, to various degrees, fare relatively poorly, as discussed above, this result in limited impact on accuracy. In the future, as template identification techniques are expected to improve, oracle template corrections will increasingly become important.

We also show that the analysis of incorrectly identified templates can provide insights on how to improve individual techniques by understanding their limitations in terms of the over- and under-generalization of templates. Nevertheless, since decreasing the over-generalization may increase under-generalization and vice versa, how to practically improve the accuracy of the techniques remains an open problem for future work.

4.6 Threats to Validity

Using a specific set of log datasets is a potential threat to external validity. However, the benchmark contains 16 log datasets from real-world systems and the corresponding oracle templates, and has been used in previous studies focused on log template identification techniques [1, 26]. Nevertheless, further experiments with different benchmarks are required to improve the generalizability of our results.

The implementations of the template identification techniques used in the evaluation may introduce threats to internal validity. To mitigate such threats, we used the same set of implementations used in the previous studies (i.e., [1, 26]). Furthermore, as mentioned above, we checked that the GA scores obtained by our experiments are exactly the same as those computed by the replication package of Zhu et al. [26].

The heuristic rules for oracle template correction can also be potential threats to internal validity. Ideally, correct oracle templates extracted from the source code are mandatory to validate the heuristic rules. At the same time, such extraction task is very challenging, even for open-source programs. This is because different versions of the same program may have different log printing statements while the logs provided in the benchmark [26] do not have the corresponding program version information. Therefore, for each program, one should manually check many previous versions to find log printing statements that would generate the given log messages; moreover, older versions may no longer be available. Though this process is extremely time-consuming, we tried our best to find the matching versions for the open-source program logs. We could find them only for HDFS, Hadoop, and OpenStack logs, and confirmed that the oracle templates modified by our heuristic rules are indeed correct with respect to the logging statements in the source code.

4.7 Data Availability

The replication package of our empirical evaluation — including the Python implementations for computing TA metrics and applying oracle template correction and incorrect template analysis — is available on Figshare [10].

5 CONCLUSION AND FUTURE WORK

In this paper, we provide three guidelines for evaluating the accuracy of log message template identification techniques. We also assess the application of such guidelines through a comprehensive evaluation of 14 log template identification techniques using a benchmark composed of 16 real-world log datasets, previously used in the literature. The evaluation results show that the accuracy assessment results for different techniques may significantly vary depending on whether the guidelines are followed.

As part of future work, we plan to further refine the accuracy metrics, for example by incorporating the concept of edit distance to compute the similarity between identified templates and their corresponding oracles, leveraging the idea proposed by Nedelkoski et al. [18]. We also plan to apply template identification techniques to specific applications, such as anomaly detection, and measure the impact of incorrectly identified templates on the performance of such applications.

ACKNOWLEDGMENTS

This work has received funding from the Celtic-Next project CRITISEC and from the Natural Sciences and Engineering Research Council of Canada (NSERC) under the Discovery and CRC programs. The experiments presented in this paper were carried out using the HPC facilities of the University of Luxembourg [23] — see hpc.uni.lu. The authors would also like to thank the anonymous referees for their valuable comments and helpful suggestions.

REFERENCES

[1] H. Dai, H. Li, C. S. Chen, W. Shang, and T. Chen. 2020. Logram: Efficient Log Parsing Using n-Gram Dictionaries. *IEEE Transactions on Software Engineering (TSE)* (2020), 1–1. https://doi.org/10.1109/TSE.2020.3007554

[2] Min Du and Feifei Li. 2016. Spell: Streaming parsing of system event logs. In *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, IEEE, Los Alamitos, CA, USA, 859–864. https://doi.org/10.1109/CNSM.2015.7367331

[3] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *2017 ACM Conference on Computer and Communications Security (SIGSAC)* (Dallas, Texas, USA) *(CCS '17)*. Association for Computing Machinery, New York, NY, USA, 1285–1298. https://doi.org/10.1145/3133956.3134015

[4] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In *2009 IEEE international conference on data mining (ICDM)*. IEEE, IEEE, Los Alamitos, CA, USA, 149–158. https://doi.org/10.1109/ICDM.2009.60

[5] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. 2016. Logmine: Fast pattern recognition for log analytics. In *25th ACM International on Conference on Information and Knowledge Management (CIKM)*. Association for Computing Machinery, New York, NY, USA, 1573–1582. https://doi.org/10.1145/2983323.2983358

[6] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R Lyu. 2017. Drain: An online log parsing approach with fixed depth tree. In *2017 IEEE International Conference on Web Services (ICWS)*. IEEE, IEEE, Los Alamitos, CA, USA, 33–40. https://doi.org/10.1109/ICWS.2017.13

[7] Shilin He, Jieming Zhu, Pinjia He, and Michael R. Lyu. 2020. Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics. arXiv:2008.06448 [cs.SE] https://arxiv.org/pdf/2008.06448.pdf

[8] Tong Jia, Lin Yang, Pengfei Chen, Ying Li, Fanjing Meng, and Jingmin Xu. 2017. Logsed: Anomaly diagnosis through mining time-weighted control flow graph in logs. In *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*. IEEE, IEEE, Los Alamitos, CA, USA, 447–455. https://doi.org/10.1109/CLOUD.2017.64

[9] Zhen Ming Jiang, Ahmed E Hassan, Parminder Flora, and Gilbert Hamann. 2008. Abstracting execution logs to execution events for enterprise applications. In *2008 The Eighth International Conference on Quality Software (QSIC)*. IEEE, IEEE, Los Alamitos, CA, USA, 181–186. https://doi.org/10.1109/QSIC.2008.50

[10] Zanis Ali Khan, Donghwan Shin, Domenico Bianculli, and Lionel Briand. 2022. Artifact for "Guidelines for Assessing the Accuracy of Log Message Template Identification Techniques". https://doi.org/10.6084/m9.figshare.18858332

[11] Adetokunbo AO Makanju, A Nur Zincir-Heywood, and Evangelos E Milios. 2009. Clustering event logs using iterative partitioning. In *15th ACM SIGKDD international conference on Knowledge discovery and data mining (SIGKDD)*. Association for Computing Machinery, New York, NY, USA, 1255–1264. https://doi.org/10.1145/1557019.1557154

[12] H. B. Mann and D. R. Whitney. 1947. On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60. http://www.jstor.org/stable/2236101

[13] L. Mariani, M. Pezzè, and M. Santoro. 2017. GK-Tail+ An Efficient Approach to Learn Software Models. *IEEE Transactions on Software Engineering (TSE)* 43, 8 (2017), 715–738. https://doi.org/10.1109/TSE.2016.2623623

[14] Salma Messaoudi, Annibale Panichella, Domenico Bianculli, Lionel Briand, and Raimondas Sasnauskas. 2018. A search-based approach for accurate identification of log message formats. In *2018 IEEE/ACM 26th International Conference on Program Comprehension (ICPC)*. ACM, Association for Computing Machinery, New York, NY, USA, 167–16710. https://doi.org/10.1145/3196321.3196340

[15] Masayoshi Mizutani. 2013. Incremental mining of system log format. In *2013 IEEE International Conference on Services Computing (SCC)*. IEEE, IEEE, Los Alamitos, CA, USA, 595–602. https://doi.org/10.1109/SCC.2013.73

[16] Meiyappan Nagappan and Mladen A Vouk. 2010. Abstracting log lines to log event types for mining software system logs. In *2010 7th IEEE Working Conference on Mining Software Repositories (MSR)*. IEEE, IEEE, Los Alamitos, CA, USA, 114–117. https://doi.org/10.1109/MSR.2010.5463281

[17] Animesh Nandi, Atri Mandal, Shubham Atreja, Gargi B. Dasgupta, and Subhrajit Bhattacharya. 2016. Anomaly Detection Using Program Control Flow Graph Mining From Execution Logs. In *22nd ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD)* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery (ACM), New York, NY, USA, 215–224. https://doi.org/10.1145/2939672.2939712

[18] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. 2020. Self-Supervised Log Parsing. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track - European Conference, ECML PKDD 2020, Proceedings, Part IV (Lecture Notes in Computer Science, Vol. 12460)*. Springer, Cham, 122–138.

[19] Keiichi Shima. 2016. Length Matters: Clustering System Log Messages using Length of Words. arXiv:1611.03213 [cs.OH] https://arxiv.org/abs/1611.03213

[20] Liang Tang, Tao Li, and Chang-Shing Perng. 2011. LogSig: Generating system events from raw textual logs. In *20th ACM international conference on Information and knowledge management (CIKM)*. ACM, New York, NY, USA, 785–794. https://doi.org/10.1145/2063576.2063690

[21] Risto Vaarandi. 2003. A data clustering algorithm for mining patterns from event logs. In *3rd IEEE Workshop on IP Operations & Management (IPOM)*. IEEE, IEEE, Los Alamitos, CA, USA, 119–126. https://doi.org/10.1109/IPOM.2003.1251233

[22] R. Vaarandi and M. Pihelgas. 2015. LogCluster - A data clustering and pattern mining algorithm for event logs. In *2015 11th International Conference on Network and Service Management (CNSM)*. IEEE, Los Alamitos, CA, USA, 1–7. https://doi.org/10.1109/CNSM.2015.7367331

[23] S. Varrette, P. Bouvry, H. Cartiaux, and F. Georgatos. 2014. Management of an Academic HPC Cluster: The UL Experience. In *Proc. of the 2014 Intl. Conf. on High Performance Computing & Simulation (HPCS 2014)*. IEEE, Los Alamitos, CA, USA, 959–967.

[24] Neil Walkinshaw, Ramsay Taylor, and John Derrick. 2016. Inferring extended finite state machine models from software executions. *Empirical Software Engineering* 21, 3 (01 Jun 2016), 811–853. https://doi.org/10.1007/s10664-015-9367-7

[25] Frank Wilcoxon. 1992. *Individual Comparisons by Ranking Methods*. Springer New York, New York, NY, 196–202. https://doi.org/10.1007/978-1-4612-4380-9_16

[26] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R Lyu. 2019. Tools and benchmarks for automated log parsing. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. IEEE, IEEE, Los Alamitos, CA, USA, 121–130. https://doi.org/10.1109/ICSE-SEIP.2019.00021