



When parsing goes wrong: Revisiting log anomaly detection from parsing to deployment[☆]

Yicheng Sun^a, Jacky Wai Keung^a, Hi Kuen Yu^a, Yishu Li^b, Xiaoxue Ma^b,*

^a Department of Computer Science, City University of Hong Kong, Hong Kong, China

^b Department of Electronic Engineering and Computer Science, Hong Kong Metropolitan University, Hong Kong, China

ARTICLE INFO

Keywords:

Log analysis
Log parsing
Log anomaly detection
Parsing errors
Industrial log data

ABSTRACT

Context: Log analysis is critical for the reliability and security of modern software systems. Although log parsing and anomaly detection are tightly connected in practice, prior studies have largely evaluated them in isolation, and the role of public data in supporting industrial anomaly detection remains unclear.

Objective: This study investigates how log parsing errors affect downstream anomaly detection, how external public logs influence industrial anomaly detection under data scarcity, and how practitioners perceive these issues in real-world systems.

Method: We conduct a unified empirical study over 16 public LogHub datasets and four proprietary industrial datasets. We analyze the outputs of multiple rule-based and LLM-based parsers, identify six common log parsing error types through expert review, and evaluate their downstream impact using five anomaly detection models. We further assess cross-dataset data augmentation on industrial datasets and complement the quantitative analysis with interviews with 11 engineers involved in the development and maintenance of the industrial systems.

Results: Log parsing errors propagate systematically to anomaly detection, and correcting such errors leads to statistically reliable overall improvements, especially on industrial datasets. Cross-dataset augmentation can improve anomaly detection under data scarcity, but its benefits are highly model dependent and may lead to negative transfer when domain compatibility is limited. Practitioner feedback confirms that industrial logs are more heterogeneous, context-dependent, and operationally constrained than public benchmarks, which helps explain the stronger impact of parsing errors and the mixed transfer results.

Conclusion: Our findings show that reliable industrial log analysis requires jointly considering parsing fidelity, data compatibility, model robustness, and practitioner trust. More broadly, the study highlights the need to treat log parsing, data curation, and anomaly detection as a tightly coupled system design problem.

1. Introduction

Modern software systems generate massive volumes of execution logs that record detailed runtime behaviors, system states, and interactions among components (Pirelli, 2024; He et al., 2016; Lou et al., 2010). As software infrastructures become increasingly distributed, service-oriented, and continuously evolving, log data has become a critical source of operational intelligence for system monitoring, failure diagnosis, security auditing, and reliability assurance (Garousi et al., 2019; Wu et al., 2023; Ma et al., 2023). Among these tasks, log anomaly detection plays a central role in identifying unexpected system behaviors, latent faults, and potential security incidents, making it an

indispensable component of contemporary software maintenance and operation pipelines (Dai et al., 2020; Fu et al., 2009; Xu et al., 2024).

To enable effective log anomaly detection, most existing approaches rely on an intermediate log parsing step that transforms raw, unstructured log messages into structured representations, typically in the form of log templates and associated variables (Jiang et al., 2024). This transformation is widely regarded as a necessary preprocessing stage that reduces noise, enables pattern learning, and facilitates downstream modeling. Consequently, a rich body of research (Sun et al., 2025c; Boxi Yu et al., 2024; Sun et al., 2026c) has independently explored log parsing techniques and anomaly detection models, leading to substantial advances in both areas. However, these two components are often

[☆] Editor: Dario Di Nucci.

* Corresponding author.

E-mail addresses: yicsun2-c@my.cityu.edu.hk (Y. Sun), Jacky.Keung@cityu.edu.hk (J. Wai Keung), hikuenyu2-c@my.cityu.edu.hk (H. Kuen Yu), sliy@hkmu.edu.hk (Y. Li), kxma@hkmu.edu.hk (X. Ma).

<https://doi.org/10.1016/j.jss.2026.113161>

Received 17 April 2026; Received in revised form 9 August 2026; Accepted 20 October 2026

Available online 28 October 2026

0164-1212/© 2026 Elsevier Inc. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

studied and evaluated in isolation, with limited attention paid to their interaction within an end-to-end log analysis pipeline.

Lack of understanding of parsing error propagation. Although recent empirical studies have examined the interaction between log parsing and anomaly detection, existing work (Khan et al., 2024; Le and Zhang, 2021; Fu et al., 2023) has mainly focused on overall parser-level accuracy, the choice of parsers for downstream models, or the design of parsing-free detection methods. However, the downstream effects of specific parsing error types remain insufficiently understood, especially in terms of how they distort event templates, variable fields, and sequence structures used by anomaly detection models. In practice, log parsers inevitably introduce various types of errors, such as incorrect template merging, over-splitting of templates, variable extraction failures, and the emergence of unseen templates over time (Huang et al., 2025). While such errors are widely acknowledged in engineering practice, their concrete impact on anomaly detection performance remains poorly understood (Wang et al., 2025; Ma et al., 2024b). Most anomaly detection studies implicitly assume that parsed logs provide a sufficiently accurate abstraction of raw messages, leaving an important gap between parser-level evaluation and task-level effectiveness.

This separation between log parsing and anomaly detection becomes particularly problematic in realistic operational settings, where systems evolve continuously and logging behaviors change over time. Parsing errors are not uniformly distributed and may disproportionately affect rare or anomalous events, thereby biasing detection models in subtle but consequential ways (Boxi Yu et al., 2024; Sun et al., 2026c). Without a clear understanding of how different types of parsing errors influence detection outcomes, it is difficult to assess the reliability, robustness, and practical limitations of existing log anomaly detection approaches.

Data scarcity and uncertain transferability across systems. A second major challenge arises from the intrinsic characteristics of log data itself. Log data is often sensitive, proprietary, and tightly coupled to specific system implementations, making it difficult to collect and share at scale. As a result, public research has largely relied on a small number of benchmark datasets, most notably the 16 log datasets released on LogHub (Zhu et al., 2023). While these datasets have played an important role in advancing log analysis research, they were collected from relatively early-generation systems and may not fully reflect the complexity, scale, and dynamics of modern industrial software environments.

In contemporary industrial settings, logs are frequently sparse, highly heterogeneous, and rapidly evolving, which poses additional challenges for data-driven anomaly detection models. The limited availability of labeled industrial logs often makes it impractical to train detection models from scratch (Sun et al., 2026d; Akhtar et al., 2025). A common workaround is to develop or pretrain models using existing public datasets and then deploy them in private or industrial environments. However, it remains an open question whether such cross-system transfer can meaningfully improve anomaly detection performance in practice, and to what extent differences in logging structures and parsing behaviors constrain this transferability.

Limited practitioner grounding in industrial log analysis. A further limitation of existing research is that empirical findings are rarely examined from the perspective of practitioners who develop, maintain, and operate industrial software systems (Ma et al., 2025). Most studies evaluate log parsing and anomaly detection primarily through benchmark performance, with limited attention paid to how engineers perceive industrial log characteristics, interpret parsing failures, or assess the practical usefulness of detection models in deployment settings (Khan et al., 2024; Gupta et al., 2026). As a result, important factors such as logging heterogeneity, context dependence, traceability requirements, and the operational consequences of parsing or detection errors are often underrepresented in the literature.

This gap is particularly important in industrial environments, where the usefulness of log analysis depends not only on predictive performance but also on whether model outputs are robust, faithful, and

actionable in practice (Sun et al., 2025a; Nipa et al., 2025). Without practitioner-oriented evidence, it is difficult to determine whether improvements observed on public datasets translate into real operational value, or whether deployment constraints such as debugging needs, auditing requirements, and limited trust in model outputs may outweigh benchmark-level gains. These limitations motivate the need to complement quantitative evaluation with practitioner perspectives, thereby providing a more realistic understanding of industrial log analysis.

To address the above challenges, building on our prior work (Sun et al., 2025a, 2026b,a), we conduct a comprehensive mixed-methods empirical study that examines log analysis as an end-to-end pipeline rather than as a set of isolated components. We first construct a multi-scenario evaluation corpus that integrates both public benchmarks and industrial log data. Specifically, our study includes all 16 publicly available datasets released on LogHub, together with four contemporary industrial log datasets collected from modern production systems. These industrial datasets reflect realistic constraints commonly encountered in practice, including limited data availability, heterogeneous logging formats, and continuously evolving system behaviors. As such, they complement existing public benchmarks and enable a more representative evaluation of real-world log analysis scenarios.

Building upon this corpus, we systematically investigate how log parsing quality, model choice, and data availability jointly shape downstream anomaly detection. We collaborate with four experienced software engineers to review parsing outputs produced by multiple parsers and to identify recurring parsing errors observed in operational environments. Through this expert-driven process, we distill six common types of log parsing errors and construct corresponding error scenarios for controlled analysis. We then evaluate how different parsing error types propagate to anomaly detection performance across models, datasets, and deployment settings. In addition, we examine the role of data availability by evaluating five state-of-the-art log anomaly detection models under both standalone and augmented training conditions, assessing whether incorporating external public logs can effectively improve detection performance on data-scarce industrial datasets. Finally, to complement the quantitative results, we conduct a practitioner interview study with engineers involved in the development and maintenance of the industrial systems, allowing us to interpret the empirical findings from an operational perspective. In summary, our primary contributions are fourfold:

- We build a multi-scenario benchmark over diverse public and industrial systems, enabling systematic evaluation of log parsing and anomaly detection under realistic deployment conditions.
- We identify and formalize six common log parsing error types through expert review, and empirically analyze their prevalence across different parsers and datasets.
- We systematically quantify how parsing errors propagate to downstream anomaly detection and how cross-dataset augmentation affects detection performance on data-scarce industrial logs.
- We complement the quantitative evaluation with a practitioner interview study, providing operational insights into the differences between industrial logs and public benchmarks, as well as the practical implications of parsing errors, data scarcity, and model trustworthiness.

The remainder of this paper is organized as follows. Section 2 reviews the background and related work on log analysis. Section 3 presents the overall methodology, including industrial dataset construction, log parsing error type identification, the anomaly detection models evaluated, and practitioner interview design in this study. Section 4 describes the experimental design corresponding to each research question, while Section 5 reports and analyzes the empirical results. Section 6 synthesizes the key findings across RQ1–RQ4 and discusses their research and practical implications. Section 7 outlines potential threats to validity. Finally, Section 8 concludes the paper.

2. Background

2.1. Log collection

Logs in industrial software systems exhibit a high degree of heterogeneity due to varying logging standards, proprietary software architectures, and domain-specific operational contexts (Li et al., 2023; Ma et al., 2024a). Unlike the relatively uniform logs found in academic or open-source datasets, industrial logs are often generated by tightly integrated systems involving hardware–software co-design, real-time constraints, and component-specific logging mechanisms. This heterogeneity introduces substantial challenges for log-based anomaly detection, particularly in terms of generalizing across different industrial settings.

Although platforms such as LogHub (Zhu et al., 2023) have greatly facilitated research by providing a range of open-source log datasets, recent studies have also pointed out important limitations of relying exclusively on such benchmarks. For example, Khan et al. (2024) empirically showed that conventional log parsing accuracy metrics are not strongly correlated with downstream anomaly detection accuracy, and argued that the distinguishability of parsed log templates is more important for anomaly detection than parser accuracy alone. This finding is important because it suggests that not all parsing errors are equally harmful: a parser may be inaccurate under template-level metrics but still preserve the discriminative separation between normal and anomalous execution traces. However, their study was conducted on publicly available benchmark datasets and did not examine industrial logs with stronger heterogeneity, domain-specific semantics, and evolving operational contexts. Therefore, their conclusion should be interpreted as evidence that traditional parsing accuracy metrics are insufficient for predicting anomaly detection performance, rather than as evidence that parsing quality is irrelevant to anomaly detection.

This limitation is also consistent with the critical review by Landauer et al. (2024), which questions the adequacy of several commonly used public log datasets for evaluating sequence-based anomaly detection techniques. Their analysis indicates that many anomalies in widely used benchmarks are not necessarily caused by complex sequential deviations and may be detectable through simpler dataset-specific cues. This raises an important concern for deployment-oriented log analysis: models validated mainly on public benchmarks may overestimate their robustness when transferred to contemporary industrial systems, where anomalies are often embedded in noisy, context-dependent, and evolving operational traces.

To make this distinction explicit, Tables 1 and 2 compare the raw log-file complexity and representative log messages of LogHub-2.0 datasets with those of the industrial datasets used in this study. As shown in Table 1, many LogHub-2.0 datasets were collected from relatively early-generation systems, with source years concentrated between 2001 and 2017. Several datasets also contain a comparatively limited number of event templates despite their large log volumes, such as HDFS, OpenStack, Apache, OpenSSH, and Proxifier. These characteristics make them valuable for controlled benchmarking, but they may not fully reflect the complexity of contemporary industrial software environments. In contrast, the industrial datasets used in this study were collected from 2024 production systems and contain substantially richer template spaces, including 2290 templates in FALL, 4508 in PIMS, 6039 in SCADA-X, and 2742 in BOSS. These datasets cover resource-constrained manufacturing, sensor-driven process systems, safety-critical distributed control, and high-throughput enterprise services, thereby reflecting more diverse and operationally realistic deployment scenarios.

The raw examples in Table 2 further illustrate this difference in logging style. Many LogHub messages are relatively short and structurally regular, often dominated by identifiers, block IDs, IP addresses, status codes, or system-level events. By contrast, the industrial logs are generally longer, more verbose, and semantically richer. They

frequently mix structured fields with free-text diagnostics and include embedded sensor readings, robotic states, process variables, control-loop information, authorization decisions, proprietary event codes, and hardware–software interaction traces. These characteristics make industrial logs more vulnerable to parsing errors such as template over-merging, inconsistent parameter masking, and failures in handling unseen templates. Therefore, this study complements existing LogHub-based evaluations by examining parsing errors under more heterogeneous and data-scarce industrial conditions. By defining parsing errors at the level of operationally meaningful error types and correcting them in both training and test data, we are able to measure their downstream impact on anomaly detection more explicitly.

Moreover, industrial logs are influenced by operational factors such as production-line variation, maintenance procedures, software versioning, configuration changes, and cross-module evolution. These factors cause frequent shifts in log distribution and format, making static parsing assumptions less reliable. In our study, we worked closely with domain engineers to obtain real-world logs from four industrial production systems across different sectors. The collected datasets exhibit substantial variability not only in log format but also in volume, frequency patterns, semantic density, and operational context. This provides a realistic foundation for evaluating the generalizability and resilience of log parsing and anomaly detection models under deployment-oriented conditions.

2.2. Log parsing

Log parsing refers to the automatic transformation of raw log messages into structured representations by separating constant components (event templates) from variable components (parameters) (Lin and Chiang, 2022; He et al., 2021; Tu, 2025). This step is widely regarded as foundational in log analysis, as it converts noisy and heterogeneous raw logs into forms that can be more effectively consumed by downstream models. Prior studies (Khan et al., 2022; Zhu et al., 2019; Huo et al., 2023; Le and Zhang, 2023; Liu et al., 2022; Li et al., 2023; Ma et al., 2024a) indicate that more than 50% of log messages contain at least three variable parameters, with around 20% containing more than five (Xu et al., 2024), highlighting the inherent difficulty of accurately identifying variable boundaries in dynamically structured logs (Xie et al., 2023; Zhang et al., 2023). Because downstream tasks often rely on consistent event templates and stable parameter extraction, parsing quality has a direct impact on subsequent analysis, including anomaly detection and root cause analysis.

Current log parsing approaches can be broadly divided into rule-based methods and LLM-based methods. Rule-based parsers, such as Drain and Spell (He et al., 2017; Du and Li, 2016), rely on manually designed patterns and heuristic rules to group log messages into event templates. These methods are generally efficient and interpretable, but they are often sensitive to lexical variation, logging drift, and unseen message structures. In contrast, recent LLM-based parsers leverage pretrained language representations to capture richer contextual and semantic information (Le and Zhang, 2023; Xu et al., 2024; Jiang et al., 2024; Xiao Yu et al., 2024), enabling them to better handle complex, evolving, or weakly structured log formats. However, their increased flexibility also introduces new challenges, such as reduced controllability, higher computational cost, and possible deviations from the original log semantics. These trade-offs make it increasingly important to understand not only how well parsers perform in isolation, but also how their errors affect downstream tasks in practical settings.

2.3. Log anomaly detection

Log anomaly detection aims to identify deviations from expected system behavior by analyzing log sequences, structural patterns, or semantic representations. A wide range of approaches has been proposed, including rule-based filtering, statistical modeling, and an increasing

Table 1

Comparison of original log-file complexity between LogHub-2.0 datasets and the industrial evaluation datasets used in this study. “–” indicates that the corresponding information is not publicly available.

Dataset	Domain/System type	Source year	#Logs	#Temp.	Main complexity characteristics
LogHub-2.0 datasets					
Hadoop	Distributed system	2015	179,993	236	Job and task execution logs from a distributed computing framework; contains application IDs, task attempts, node addresses, and exception traces.
HDFS	Distributed file system	2009	11,167,740	46	Very large-scale storage logs with relatively low template diversity; dominated by block IDs, data-node addresses, and block-transfer events.
OpenStack	Cloud infrastructure	2017	207,632	48	Cloud service logs involving VM lifecycle events, API calls, instance identifiers, HTTP status codes, and service-level operations.
Spark	Distributed data processing	2015	16,075,117	236	Large-scale cluster-computing logs; includes executor, driver, memory-store, shuffle, and block-manager events with moderate structural regularity.
Zookeeper	Coordination service	2015	74,273	89	Coordination and leader-election logs; contains session negotiation, quorum state changes, peer communication, and protocol-level messages.
BGL	Supercomputer system	2005	4,631,261	320	Supercomputer system logs with hardware, kernel, RAS, interrupt, and node-level events; many messages are short but highly domain-specific.
HPC	High-performance cluster	2003	429,987	74	Cluster operating logs with RAS and kernel messages; contains compact but domain-specific fault descriptions related to hardware and node states.
Thunderbird	Supercomputer system	2002	16,601,745	1241	Very large system logs with high template diversity; includes long kernel, device, boot, and network messages with many low-frequency templates.
Linux	Operating system	2005	23,921	338	General operating-system logs involving kernel, service, daemon, and network events; moderate lexical variation across system components.
Mac	Operating system	2001	100,314	626	Desktop operating-system logs with wireless, mDNS, device, and system-service messages; many entries contain long parameter-rich fields.
Apache	Server application	2005	51,977	29	Web-server logs with low template diversity; mainly contains client addresses, file paths, HTTP/server errors, and PHP-related messages.
OpenSSH	Server application	–	638,946	38	Authentication logs with login attempts, invalid users, pre-authentication sessions, IP addresses, and ports; structurally regular but security-sensitive.
HealthApp	Application software	2008	212,394	156	Application logs with sensor, step-counting, BLE, calorie, and distance-related messages; contains many numeric measurements and callback events.
Proxifier	Standalone software	–	21,320	11	Proxy connection logs with very low template diversity but many dynamic host, port, traffic-volume, protocol, and connection-lifetime fields.
Industrial evaluation datasets used in this study					
FALL	Resource-constrained system	2024	32,615	2290	Sparse and uneven industrial control logs; includes motion control, robot state, sensor deviation, vision inspection, operator commands, conveyor coordination, and recovery events.
PIMS	Sensor-driven process system	2024	871,243	4508	High-volume process-industry logs with sensor readings, control loops, chemical-process states, safety checks, and domain-specific operational terms.
SCADA-X	Safety-critical distributed system	2024	902,377	6039	Distributed SCADA logs with asynchronous reporting, telemetry synchronization, control commands, protocol handshakes, and context-sensitive operational semantics.
BOSS	High-throughput enterprise system	2024	1,284,906	2742	Enterprise service logs with workflow execution, access control, auditing, transaction processing, mixed structured/free-text fields, and frequent schema evolution.

number of machine learning and deep learning methods (Ma et al., 2024, 2025; Wang et al., 2022; De la Cruz Cabello et al., 2026; Chai et al., 2026; Zhang et al., 2026). These techniques have become essential for fault diagnosis, performance monitoring, and early risk detection in large-scale software systems (Sun et al., 2025d; Hsiao and Tseng, 2026). In most practical pipelines, however, anomaly detection does not operate directly on raw logs, but instead depends on parsed representations. As a result, the effectiveness of anomaly detection is closely tied to the quality of upstream log parsing.

This dependency becomes even more critical in industrial software systems, where failures may result in substantial financial losses, production interruptions, or safety risks in domains such as manufacturing, energy, and transportation (Landauer et al., 2023). In such environments, logs often serve as the primary non-intrusive channel for monitoring system behavior, especially when direct access to internal states is restricted by proprietary infrastructures or regulatory requirements (Pang et al., 2024; Dobrowolski et al., 2023). At the same time, industrial logs are typically more heterogeneous, more weakly standardized, and more rapidly evolving than the public datasets commonly

used in prior research. Although most existing studies evaluate anomaly detection methods on publicly available datasets such as those released on LogHub (Zhu et al., 2023), these benchmarks are largely derived from open-source or traditional distributed systems and do not fully capture the structural variability, semantic complexity, and operational constraints of modern industrial software.

Consequently, two important gaps remain insufficiently addressed in existing work. First, although anomaly detection models have become increasingly sophisticated, the role of parsing quality in shaping downstream detection performance is still underexplored. Second, because industrial logs are scarce and difficult to label, many practical deployments must rely on limited in-domain data and, potentially, external public logs for augmentation. Whether such cross-dataset training is truly beneficial, and under what conditions it may instead lead to negative transfer, remains unclear. These gaps motivate our study, which examines log anomaly detection not as an isolated modeling problem, but as the outcome of interacting factors including parsing quality, model design, data availability, and industrial deployment constraints.

Table 2

Representative raw log examples from LogHub-2.0 datasets and the industrial evaluation datasets used in this study. Industrial examples are anonymized to remove sensitive operational identifiers.

Dataset	Representative raw log examples
LogHub-2.0 datasets	
Hadoop	Created MRAppMaster for application appattempt_1445144423722_0020_000001 [Normal] Task: attempt_1445144423722_0020_m_000002_0 exited: java.net.NoRouteToHostException from 127.0.0.1 to msra-sa-41:9000 failed on socket timeout exception [Anomalous]
HDFS	Received block blk_-5340841684362207777 of size 67108864 from /10.251.198.33 [Normal] 10.251.126.227:50010: Got exception while serving blk_-4181768899028058192 to /10.251.26.177 [Anomalous]
OpenStack	[instance: b9000564-fe1a-409b-b8cc-1e88b294cd1d] VM Started (Lifecycle Event) [Normal] 10.11.21.122,10.11.10.1 'GET /openstack/2013-10-17/user_data HTTP/1.1' status: 404 len: 176 time: 0.0010660 [Anomalous]
Spark	MemoryStore started with capacity 17.7 GB Created local directory at /opt/hdfs/nodemanager/usercache/curi/appcache/application_1485248649253_0147/blockmgr-70293f72-844a-4b39
Zookeeper	Accepted socket connection from /10.10.34.13:37196 Notification: 3 (n.leader), 0xb0000007b (n.xxid), LOOKING (n.state), 3 (n.sid), FOLLOWING (my state)
BGL	CE sym 4, at 0 x 0589f8e0, mask 0 x 10 [Anomalous] 10692 total interrupts. 0 critical input interrupts. 0 microseconds total spent on critical input interrupts. [Normal]
HPC	ras KERNEL INFO instruction cache parity error corrected ras KERNEL FATAL node card status: cioid: Error loading /p/gb2/stella/RAPTOR/raptor: invalid or missing program image
Thunderbird	PCI: Using ACPI for IRQ routing [Normal] Kernel command line: root=LABEL=/ initrd=/x86_64/initrd-2.6.9-5.0.5.EL console=ttyS0,19200 fastboot [Normal]
Linux	kernel: imklog 5.8.11, log source = /proc/kmsg started. NetworkManager[710]: (info) Activation (eth0) successful, device activated.
Mac	mDNS_DeregisterInterface: Frequent transitions for interface awd10 (FE80:0000:0000:0000:D8A5:90FF) ARPI: 621490.858770: wl0: wl_update_tcpkeep_seq: Updated seq/ack/win from UserClient Seq 2119064372
Apache	[error] [client 127.0.0.1] File does not exist: /var/www/favicon.ico [error] [client 200.10.20.30] PHP Notice: Undefined index: action in /var/www/html/index.php on line 23
OpenSSH	Failed password for invalid user admin from 10.0.0.2 port 49822 ssh2 Connection closed by authenticating user root 10.0.0.3 port 45210 [preauth]
HealthApp	onStandStepChanged 3579 getTodayTotalDetailSteps=15170##getTodayTotalDetailCalories=1267757##getTodayTotalDetailDistance=10364
Proxifier	proxy.cse.cuhk.edu.hk:5070 open through proxy proxy.cse.cuhk.edu.hk:5070 HTTPS www.google.com:443 close, 470 bytes sent, 7016 bytes received, lifetime 00:00:00
Industrial evaluation datasets used in this study	
FALL	PLC-A7 CELL=ASM-03 ROBOT=R2 STEP=PickAndPlace joint_temp=41.2C arm_pos=(0.22,0.98,1.14) vacuum=ON grip_force=18.4N cycle_status=completed latency=34 ms [Normal] ctrl@asm-line04 module=servo_sync level=WARN axis=arm_y expected=(0.20,1.00) observed=(0.22,0.98) drift=0.028 tolerance=0.010 retries=4 action=halt_cell [Anomalous] MES-GW request=GET /machine_status?line=3&batch=842 result=502 Gateway Timeout retry=3 upstream=cell-monitor impact=inspection queue blocked [Anomalous]
PIMS	PROC unit=Reactor-2 phase=mixing agitator_rpm=180 temp=82.6C pressure=3.42 bar feed_valve=V12:OPEN cooling_loop=stable status=within_control_limits [Normal] PIPEMON::segment=distill-A::flow=18.1L/min::pressure=5.02 bar::valve_group=VG-04::controller=PID-04::setpoint=18.0::alarm_state=clear [Normal] ALARM::area=distillation::controller=PID-04::checksum=FAILED::expected=7F3A9C::received=7F3A1C::mode=AUTO::safety_override=engaged [Anomalous]
SCADA-X	RTU_NORTH/substation=E14 feeder=FD-22 voltage=109.8 kV current=241 A breaker=CB-22 state=CLOSED sync=OK historian=written alarm_state=clear [Normal] PROTO-DNP3 master=ctrl-center-2 rtu=E14-R12 seq=5512 op=integrity-poll link=healthy latency=23 ms confirm=received status=nominal [Normal] SEC-EVT::node=gw-sec-11::user=alice::source=10.18.4.33::scope=relay_write::auth=denied::policy=night-shift-lockdown::attempt=4::event= Access unauthorized [Anomalous]
BOSS	WF-GW txn=TX9381 step=approve user=U17 role=manager status=passed queue=normal audit_id=AUD-88421 downstream=finance_sync committed=true [Normal] AUDIT txn=TX9381 auth_check=success actor=U17 module=approval policy=role_matched trace=written latency=42 ms result=accepted [Normal] AUTH-SVC request=REQ-5572 actor=U31 target=payroll_export action=download result=denied policy=least_privilege retry=3 alert=raised [Anomalous]

2.4. Empirical studies on parsing-detection interaction

Recent empirical studies have begun to examine how log parsing affects log-based anomaly detection. Therefore, the interaction between parsing and detection should not be viewed as entirely unexplored. Instead, the remaining gap lies in understanding the mechanisms through which specific parsing errors distort downstream detection inputs, and

in examining whether such effects persist under heterogeneous and deployment-oriented log analysis settings. Table 3 summarizes three closely related empirical studies and clarifies how our study extends them.

As shown in Table 3, prior work has provided important evidence that log parsing can influence anomaly detection. Le and Zhang (2021) showed that parsing errors may remove useful semantic information

Table 3

Comparison with prior empirical studies on the interaction between log parsing and anomaly detection.

Study	Main focus	Datasets	Parsers/Detectors	Main findings	Remaining gap addressed by this study
Le and Zhang (2021)	Designing a parsing-free anomaly detection method and showing that parsing errors can harm detection.	Four public datasets: HDFS, BGL, Thunderbird, and Spirit. The parsing-error analysis mainly focuses on BGL and Thunderbird.	Parsers considered in the parsing-error analysis include Drain, AEL, IPLoM, and Spell. The proposed detector is NeuralLog; baselines include LR, SVM, IM, LogRobust, and Log2Vec.	Existing parsing-based detectors can be affected by OOV words and semantic misunderstandings. Parsing errors may remove important semantic information, causing normal and anomalous logs to become difficult to distinguish. NeuralLog avoids this problem by directly encoding raw log messages.	This study demonstrates that parsing errors matter, but its main contribution is a new parsing-free model. It does not provide a detailed taxonomy of operational parsing error types, nor does it systematically correct specific parsing errors to quantify their downstream impact across multiple detectors and deployment settings.
Fu et al. (2023)	Empirically evaluating how different log parsers affect the effectiveness and efficiency of anomaly detection models.	Three public datasets: HDFS, BGL, and Thunderbird.	Six parsers: Drain, IPLoM, Logram, LFA, Lenma, and Spell. Six detectors: PCA, LogClustering, Logistic Regression, Decision Tree, DeepLog, and LogRobust.	High parsing accuracy does not necessarily imply high anomaly detection performance. Both parsing accuracy and the number of parsed event templates should be considered when selecting parsers. Parsing errors can have negative or, in some cases, positive effects, and more parsed templates may reduce detection efficiency.	This study focuses mainly on parser selection, parser accuracy, template numbers, and detection efficiency. It does not distinguish fine-grained parsing error types from an operational perspective, nor does it examine how expert-verified error correction changes downstream detection performance.
Khan et al. (2024)	Systematically examining the relationship between parser-level accuracy metrics and anomaly detection accuracy, and empirically validating the role of distinguishability.	Three public LogHub-based datasets: HDFS, Hadoop, and OpenStack. HDFS is reduced for computational feasibility.	Thirteen parsers: AEL, Drain, IPLoM, LenMa, LFA, LogCluster, LogMine, Logram, LogSig, MoLFI, SHISO, SLCT, and Spell. Seven detectors: DeepLog, LogAnomaly, LogRobust, PLELog, CNN, SVM, and Random Forest.	There is no strong correlation between conventional parsing accuracy metrics and anomaly detection accuracy. The key property for anomaly detection is not parser accuracy alone, but whether the parsed results preserve distinguishability between normal and anomalous logs.	This study clarifies the limitation of overall parser-level accuracy metrics, but it does not analyze which concrete parsing error types reduce distinguishability in practice. It also does not examine error-type-specific correction, industrial log heterogeneity, cross-dataset training, or practitioner deployment concerns.
This study	Analyzing how specific parsing errors propagate to downstream anomaly detection, and how data scarcity, external data, and deployment context shape this interaction.	Sixteen public LogHub datasets and four proprietary industrial datasets collected from 2024 production systems: FALL, PIMS, SCADA-X, and BOSS.	Six parsers: Drain, Spell, AEL, DivLog, LILAC, and LUNAR. Five detectors: PLELog, LogBERT, SemiRALD, SemiSMAC, and LogRobust.	We identify six recurring parsing error types through expert review and show that correcting such errors consistently improves downstream anomaly detection, especially on heterogeneous and data-scarce datasets. We also find that external public logs can help industrial anomaly detection, but the effect is model-dependent and may cause negative transfer.	Our study complements prior work by moving from overall parser-level metrics to operationally meaningful parsing error types, from parser selection to error-type-specific correction, from public benchmarks to public-industrial comparison, and from benchmark evaluation to practitioner-grounded deployment interpretation.

and proposed a parsing-free detection method to avoid this problem. Fu et al. (2023) further demonstrated that parser choice affects both detection effectiveness and computational efficiency, and that higher parsing accuracy does not always lead to better anomaly detection. Khan et al. (2024) provided the most systematic analysis of parser-level accuracy metrics and showed that distinguishability, rather than conventional parsing accuracy alone, is the key property for downstream detection.

Our study is motivated by these findings but addresses a different level of the problem. Rather than asking whether a parser with higher overall accuracy necessarily produces better anomaly detection results, we ask which specific parsing errors are operationally meaningful and how their correction affects downstream models. This distinction is important because overall parser-level metrics may obscure the heterogeneous effects of different errors. For example, template over-merging can collapse normal and anomalous events into the same representation, whereas template over-splitting can inflate the event space and reduce the statistical support of individual templates. Variable boundary errors, inconsistent parameter masking, unseen-template handling

errors, and LLM-induced semantic rewriting may each distort different parts of the downstream input representation. These mechanisms cannot be fully captured by a single parser-level accuracy score.

This perspective also provides a possible interpretation of our findings in relation to Khan et al.'s distinguishability argument (Khan et al., 2024). We do not claim that higher parsing accuracy necessarily leads to better anomaly detection. Rather, several error types identified in our study, such as template over-merging, over-splitting, and inconsistent parameterization, can plausibly alter the separation between normal and anomalous event representations by collapsing semantically distinct events or introducing artificial template fragmentation. Our RQ2 further shows that correcting these errors improves downstream detection performance.

In addition, our study extends prior empirical work in terms of data context and deployment grounding. Existing studies mainly rely on public benchmark datasets and controlled parser-detector comparisons. In contrast, we use the 16 public LogHub datasets together with four proprietary industrial datasets collected from contemporary

production systems. This allows us to examine parsing errors under both benchmark and deployment-oriented conditions. We further evaluate expert-guided parsing-error correction across multiple anomaly detection models, study cross-dataset augmentation under industrial data scarcity, and complement the quantitative results with interviews from engineers involved in the corresponding industrial systems. This design allows us to connect parser-level failures, downstream detection behavior, data compatibility, and practitioner trust within a unified end-to-end log analysis pipeline.

3. Methodology

3.1. Overview

This study investigates log analysis as an end-to-end pipeline and examines how log parsing quality, data availability, and deployment context jointly influence downstream anomaly detection. To this end, we design a unified empirical framework that combines public benchmarks, industrial log data, and practitioner perspectives, enabling evaluation under both controlled experimental settings and realistic operational conditions.

Specifically, we first construct a multi-scenario corpus consisting of 16 LogHub datasets and four proprietary industrial datasets. We then analyze parsing behavior using representative rule-based and LLM-based parsers, and identify recurring parsing error types through expert review. Based on this taxonomy, we conduct controlled experiments to quantify how parsing errors propagate to anomaly detection, and further evaluate five representative detection models under both standard and augmented training settings to assess their robustness to parsing noise and cross-dataset transfer. Finally, we complement the quantitative analysis with a practitioner interview study to better understand how engineers perceive the differences between industrial logs and public benchmarks, and how these differences affect real-world log analysis.

3.2. Construction of industrial log datasets

To support a realistic and comprehensive evaluation of log parsing errors and anomaly detection under practical constraints, we construct four proprietary industrial log datasets collected from production systems in distinct application domains. These datasets are intentionally selected to reflect the diversity of modern industrial software environments, including differences in system architecture, logging practices, data volume, and operational dynamics. Unlike public benchmarks, which are often derived from legacy or research-oriented systems, the industrial datasets used in this study exhibit characteristics that pose substantial challenges to both log parsing and anomaly detection, such as limited data availability, heterogeneous log formats, domain-specific semantics, and evolving logging behaviors.

The four industrial datasets span manufacturing, process industry management, critical infrastructure control, and large-scale enterprise service platforms. Together, they capture a broad spectrum of logging scenarios commonly encountered in real-world deployments, ranging from resource-constrained embedded systems to high-throughput, event-driven software platforms. Below, we briefly describe the characteristics of each dataset, and Table 4 summarizes their key properties.

FALL: This dataset is derived from a low-resource industrial assembly line system deployed in a precision manufacturing environment. The system generates logs at a relatively low frequency due to strict resource and cost constraints, resulting in sparse and uneven log streams. Log messages often exhibit irregular formatting and high variability, reflecting the absence of standardized logging templates and frequent manual interventions. Such characteristics are representative of embedded or cost-sensitive industrial deployments, where logging is secondary to real-time control requirements.

PIMS: Collected from a modern process industry management system in the chemical sector, this dataset consists of high-volume logs produced by tightly coupled software modules interacting with physical sensors and control devices. The logs contain deeply nested event sequences, rich temporal dependencies, and highly domain-specific terminologies. Frequent state transitions and feedback loops between software and hardware components make this dataset particularly challenging for both parsing and sequential anomaly detection.

SCADA-X: This dataset is obtained from a supervisory control and data acquisition (SCADA) system used in energy infrastructure monitoring. It includes logs generated by geographically distributed control nodes operating under real-time constraints. The dataset is characterized by heterogeneous log structures, asynchronous event reporting, and occasional timestamp misalignment across sources. Moreover, many log messages are context-sensitive, with semantics that depend on operational states and control modes, posing additional challenges for template-based parsing and downstream analysis.

BOSS: This dataset is collected from a large-scale enterprise back-office service system supporting business operations such as workflow coordination, access control, and transaction auditing. Unlike control-oriented systems, BOSS generates logs at high throughput and exhibits frequent schema evolution due to continuous feature updates and service refactoring. Log messages often mix structured fields with free-text descriptions, and the same event type may be logged in multiple syntactic variants over time. This dataset highlights challenges associated with rapid system evolution, logging drift, and cross-version inconsistency in enterprise software platforms.

3.3. Identification of log parsing error types

To systematically identify and characterize common log parsing errors observed in practice, we collaborated with four experienced software engineers with extensive backgrounds in software development and system operations. Each engineer has more than eight years of industrial experience and has been involved in the design, deployment, and maintenance of large-scale production systems at major technology companies. We conducted a series of structured roundtable sessions in which the participating engineers jointly reviewed the outputs produced by representative log parsers. These sessions were designed to facilitate collective inspection, discussion, and consensus-building, rather than independent annotation. The goal was not to evaluate individual parsers in isolation, but to understand the types of parsing errors that arise across different parsing paradigms and logging scenarios, and to abstract recurring error patterns that are meaningful from an operational perspective.

In this study, we consider two major categories of log parsers commonly used in current research and practice: rule-based parsers and large language model (LLM)-based parsers. Rule-based parsers rely on heuristics, token-level rules, and frequency-based template extraction to identify invariant and variable parts of log messages. These approaches are typically lightweight, efficient, and widely adopted in production environments, but may struggle with complex or evolving log formats. In contrast, LLM-based parsers formulate log parsing as a semantic understanding or text generation task, leveraging pretrained language models to infer templates and variable fields from raw log messages. While more flexible, such parsers introduce new sources of uncertainty related to model inference and prompt sensitivity.

To ensure representative coverage of both paradigms, we selected three widely used rule-based parsers — Drain (He et al., 2017), Spell (Du and Li, 2016), and AEL (Zhu et al., 2019)—and three recent LLM-based parsers — DivLog (Tufek and Aktas, 2023), LILAC (Jiang et al., 2024), and LUNAR (Huang et al., 2025). This selection reflects both the maturity of traditional parsing techniques and recent advances in LLM-assisted log analysis. All parsers were configured following their respective recommended settings, without task-specific tuning, to reflect realistic usage conditions.

Table 4
Summary of industrial log datasets used in this study.

Dataset	Abbr.	Type	Total logs	Event templates	Anom.	Avg. Len (Chars)
Factory Assembly Line Logs	FALL	Resource-constrained control system	32,615 (9.56%)	2290	12.47%	227.5
Process Industry Management System Logs	PIMS	Sensor-driven process system	871,243 (9.93%)	4508	9.38%	312.6
Supervisory Control and Data Acquisition Logs	SCADA-X	Safety-critical distributed system	902,377 (12.37%)	6039	7.42%	351.1
Enterprise Back-Office Service System Logs	BOSS	High-throughput enterprise system	1,284,906 (7.84%)	2742	3.96%	268.4

For the 16 public datasets released on LogHub, we utilized the representative log subsets provided by the dataset authors, consisting of 2000 log messages per dataset. These subsets have been widely adopted in prior studies and are designed to capture the dominant logging patterns and structural characteristics of each system, thereby providing a consistent and comparable basis for parser evaluation across public benchmarks.

For our four industrial datasets, where no predefined representative subsets were available, we constructed representative samples of 3000 log messages per dataset using a stratified diversity-aware sampling strategy, as summarized in Algorithm 1. Formally, given a raw industrial log dataset $\mathcal{L}$ and a target sample size N , the goal is to construct a representative subset S of size N that preserves both the dominant operational patterns and the less frequent but structurally informative events.

The procedure consists of four steps. First, we perform *structural grouping* on $\mathcal{L}$ by partitioning raw logs into coarse groups $\{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k\}$ according to token patterns and message length. This step provides a parser-agnostic approximation of high-level template similarity, allowing logs with similar surface structures to be grouped together without depending on any specific parsing algorithm. Second, within each structural group $\mathcal{G}_i$, we conduct *intra-group clustering* to obtain finer-grained clusters $\mathcal{C}_{ij}$. These clusters are formed based on temporal segments and, when available, system module identifiers, so that logs produced in different execution phases or functional components remain distinguishable even if their lexical forms are similar.

Third, we perform *proportional sampling* over the resulting clusters. For each cluster $\mathcal{C}_{ij}$, we assign a sampling quota q_{ij} that is approximately proportional to the cluster size $|\mathcal{C}_{ij}|$, while enforcing a minimum threshold to prevent rare but informative patterns from being excluded. We then randomly sample q_{ij} logs from each cluster and add them to the subset S . In this way, frequent patterns receive sufficient representation, while minority clusters are still preserved. Finally, in the *aggregation and adjustment* step, all sampled logs are merged into S . If $|S| > N$, we randomly downsample S to size N ; if $|S| < N$, we supplement S with additional logs drawn from under-represented clusters. This design ensures that the final subset remains close to the target size while balancing coverage, diversity, and robustness to bursty or highly repetitive logging behavior commonly observed in industrial environments.

All six parsers were applied independently to the selected log samples from each dataset. The resulting parsing outputs — including extracted templates and variable fields — were then reviewed during the roundtable sessions. Engineers examined cases where parsed templates were inconsistent, overly generalized, excessively fragmented, or semantically misleading. Through iterative discussion and comparison across parsers and datasets, logs exhibiting incorrect or ambiguous parsing behaviors were identified and collected for further analysis. Based on these discussions, we abstracted six common types of log parsing errors that recur across datasets and parsing approaches. Five of these error types were consistently observed in both rule-based and LLM-based parsers, indicating fundamental challenges inherent to log

Algorithm 1 Stratified Diversity-Aware Sampling of Representative Industrial Logs

Require: Raw industrial log dataset $\mathcal{L}$, target sample size N

Ensure: Representative log subset S

1: **Structural grouping:**

2: Partition $\mathcal{L}$ into coarse structural groups $\{\mathcal{G}_1, \mathcal{G}_2, \dots, \mathcal{G}_k\}$ based on token patterns and message length

3: **Intra-group clustering:**

4: **for** each structural group $\mathcal{G}_i$ **do**

5: Partition $\mathcal{G}_i$ into finer-grained clusters $\mathcal{C}_{ij}$ according to temporal segments and system modules (if available)

6: **end for**

7: **Proportional sampling:**

8: Initialize $S \leftarrow \emptyset$

9: **for** each cluster $\mathcal{C}_{ij}$ **do**

10: Determine a sampling quota q_{ij} proportional to $|\mathcal{C}_{ij}|$, subject to a minimum threshold

11: Randomly sample q_{ij} logs from $\mathcal{C}_{ij}$ and add them to S

12: **end for**

13: **Aggregation and adjustment:**

14: **if** $|S| > N$ **then**

15: Randomly downsample S to size N

16: **else**

17: **if** $|S| < N$ **then**

18: Supplement S with additional logs from under-represented clusters

19: **end if**

20: **end if**

21: **return** S

structure variability and system evolution. The remaining error type was specific to LLM-based parsers and arises from model-driven semantic inference that deviates from operationally meaningful template boundaries.

Importantly, these error types were defined based on observable parsing behaviors and their potential impact on downstream analysis, rather than parser-specific implementation details. To assess the reliability of the error-type coding process, the observed parsing-error cases were independently coded by two expert groups according to the finalized six-category taxonomy, with each case assigned one primary error type. The inter-rater agreement before adjudication was substantial, with Cohen's $\kappa = 0.86$. Remaining disagreements were discussed by the expert group and resolved before the final RQ1 analysis. The resulting taxonomy of log parsing errors provides a structured basis for subsequent empirical evaluation.

3.4. Log anomaly detection evaluation

To evaluate the impact of log parsing errors and cross-dataset training on downstream anomaly detection, we selected five representative log anomaly detection models that reflect diverse design philosophies and supervision requirements. Our evaluation includes four semi-supervised methods — PLELog (Lin and Chiang, 2022), LogBERT (Guo et al., 2021), SemiRALD (Sun et al., 2025c), and SemiSMAC (Sun et al., 2025b) — as well as one fully supervised method,

LogRobust (Zhang et al., 2019). These models are widely adopted in recent literature and collectively cover probabilistic modeling, contextual representation learning, self-training mechanisms, and scalable optimization strategies. In particular, PLELog relies on probabilistic assumptions to detect anomalies from partially labeled data; LogBERT leverages contextual embeddings to capture semantic deviations in event sequences; SemiRALD integrates iterative self-training and refinement to operate under extremely limited supervision; and SemiSMAC extends prior semi-supervised frameworks with efficient hyperparameter tuning. LogRobust, in contrast, represents a traditional supervised paradigm trained on explicitly labeled anomalous sequences.

The inclusion of both semi-supervised and supervised models is motivated by the practical characteristics of real-world log analysis. Many operational anomalies are rare, subtle, and highly domain-specific, often requiring domain experts to interpret complex temporal patterns spanning multiple components. Consequently, manual log annotation is costly, time-consuming, and prone to inconsistency across systems. These constraints make fully supervised approaches difficult to deploy in realistic industrial settings. Semi-supervised methods offer a more practical alternative, yet they differ substantially in their assumptions and data requirements. For instance, PLELog assumes access to approximately 50% of normal logs for initialization, whereas SemiRALD has demonstrated competitive performance using as little as 0.1% of the total log volume. Evaluating these models under unified experimental conditions is therefore essential for understanding their relative robustness and applicability.

To ensure a fair, reproducible, and interpretable evaluation across models and datasets, we constructed consistent training sets for all experiments. For the 16 public LogHub datasets, we used the representative subsets of 2000 log messages provided by the dataset authors as training data. For our four industrial datasets, we used the representative subsets of 3000 log messages constructed using the stratified diversity-aware sampling strategy described in Algorithm 1. By standardizing the training data selection process, we aimed to isolate the effects of parsing quality and cross-dataset training from confounding factors related to data volume and sampling bias.

Among the LogHub datasets, only five provide ground-truth labels for both normal and anomalous logs, while the remaining public datasets and all four industrial datasets lack anomaly annotations. For these unlabeled datasets, we conducted a manual annotation process involving four domain experts, each with over five years of experience in software development and system operations. The experts were divided into two independent teams, and a two-round cross-validation protocol was employed. In the first round, each team independently labeled log entries as either *normal* or *anomalous*. In the second round, the teams cross-validated each other's annotations. Log entries for which the two teams could not reach agreement were flagged for arbitration.

A final round-table discussion involving all four experts was conducted to resolve remaining disagreements and establish the final labels. Inter-annotator agreement was assessed using Cohen's κ , yielding values ranging from 0.84 to 0.90 across datasets, which indicates substantial agreement according to commonly accepted interpretation guidelines. This annotation process was designed to balance labeling reliability with practical feasibility, providing a robust foundation for evaluating anomaly detection performance under realistic data constraints.

3.5. Practitioner interview

To complement our quantitative analysis, we conduct a practitioner interview study to understand how industrial engineers perceive the characteristics of industrial logs and the challenges of applying log analysis methods in real-world systems. Specifically, we aim to contextualize our empirical findings by examining whether practitioners observe similar differences between industrial log data and public

benchmarks, and how these differences influence the effectiveness of log parsing and anomaly detection in practice.

We recruited 11 engineers who were directly involved in the development and maintenance of the four industrial systems from which our datasets were derived. All participants have substantial experience with these systems, each having contributed to more than half of the system's development and maintenance lifecycle. Their roles include software development, system operation, and reliability engineering, providing diverse perspectives on logging practices and system behavior. All participants have over five years of professional experience, with several exceeding eight years, ensuring that the insights reflect mature engineering practices.

We adopt a semi-structured interview protocol to elicit practitioner perspectives on industrial log analysis. Prior to the interviews, participants are provided with a concise summary of our key empirical findings in order to ground the discussion in concrete evidence while allowing them to reflect on these results in light of their own engineering experience. All interviews are conducted with participant consent, recorded, and transcribed for analysis. We then analyze the transcripts using thematic analysis: two researchers independently code the responses, iteratively refine the coding scheme through discussion, and resolve disagreements through consensus. The resulting themes are subsequently mapped back to our empirical findings to identify consistencies, divergences, and additional practitioner insights not captured by the quantitative evaluation.

We assessed thematic saturation during the coding process by tracking whether new interviews introduced substantially new themes beyond the existing codebook. After the ninth interview, no new major themes emerged, and the final two interviews mainly confirmed or refined the existing themes. We also retained divergent views during coding rather than forcing consensus; for example, some participants considered external public logs useful for expanding coverage, whereas others emphasized the risk of semantic mismatch and negative transfer. To reduce interpretation bias, we conducted member checking by returning concise summaries of the identified themes to participants for confirmation, and no participant reported substantial disagreement with the summarized themes. The interview study serves as a qualitative validation of our experimental results and provides practitioner-oriented interpretations of the observed phenomena. By integrating quantitative evaluation with expert insights, we aim to bridge the gap between controlled benchmarking and real-world deployment, and to better understand the practical implications of log parsing errors and data limitations in industrial settings.

4. Experimental setup

4.1. Datasets

All experiments in this study are conducted on a comprehensive collection of both public and industrial log datasets, designed to support a systematic and realistic evaluation of log parsing errors and anomaly detection performance. Specifically, we evaluate our methods on the 16 publicly available datasets released on LogHub (Zhu et al., 2023), together with four proprietary industrial datasets collected from modern production systems. The LogHub datasets have been widely used in prior research and provide a common baseline for comparing log analysis methods under controlled conditions. They cover diverse system types and logging behaviors, allowing us to situate our findings within the existing literature. In contrast, the four industrial datasets represent real-world operational systems characterized by heterogeneous log formats, evolving logging practices, and limited labeled data. These datasets reflect practical constraints that are often absent from public benchmarks but are critical for understanding the applicability and robustness of log anomaly detection methods in deployment.

4.2. Research questions

In this section, we present the core research questions and describe the corresponding experimental designs.

RQ1: What insights can be gained from common log parsing error types?

The first research question aims to understand the nature and implications of log parsing errors observed in practice. While prior work has primarily evaluated parsers using intrinsic metrics such as parsing accuracy or template quality, there is limited empirical understanding of what types of parsing errors are most prevalent and how they manifest across different systems and parsing paradigms. To address this question, we analyze the outputs of six representative log parsers spanning both rule-based and LLM-based paradigms. Specifically, the rule-based parsers include **Drain**, **Spell**, and **AEL**, which rely on heuristic rules and token-level pattern extraction, while the LLM-based parsers include **DivLog**, **LILAC**, and **LUNAR**, which formulate log parsing as a language understanding or generation task. All parsers are applied uniformly to both the 16 public datasets released on LogHub and the four proprietary industrial datasets described in Section 3.2, ensuring broad coverage of diverse logging environments.

Parsing outputs generated by these six parsers are examined through expert-led roundtable review. Log messages exhibiting incorrect or ambiguous parsing behaviors are jointly inspected and grouped into recurring error categories based on their observable characteristics and operational implications. Through iterative discussion and consensus building, six common log parsing error types are identified. We then quantify the occurrence frequency and distribution of these error types across datasets, system domains, and parser categories (rule-based versus LLM-based). This analysis enables us to distinguish error patterns that are pervasive across parsing paradigms from those that are dataset- or model-specific, thereby providing a structured foundation for the downstream impact analysis in subsequent research questions.

RQ2: How do log parsing errors affect downstream anomaly detection performance?

The second research question investigates the extent to which log parsing errors propagate to downstream log anomaly detection tasks. Although log parsing is widely treated as a preprocessing step, its influence on detection performance is often implicitly assumed rather than explicitly evaluated. In this section, we seek to quantify this influence by comparing anomaly detection outcomes under different parsing quality conditions.

To this end, we evaluate five log anomaly detection models under two controlled settings: (i) using the original parsed datasets produced by each parser, and (ii) using corrected parsed datasets in which logs identified by experts as parsing errors are manually rectified. This corrected setting should be interpreted as an oracle-style upper-bound condition rather than a directly deployable parsing-correction pipeline. Its purpose is to isolate the potential downstream impact of parsing errors by asking how much anomaly detection performance could improve if the identified parsing errors were corrected reliably. In practical deployments, fully manual correction would be too costly, and automated or semi-automated correction methods would likely recover only part of this upper-bound improvement.

Importantly, parsing correction is applied consistently to both the training and test sets to avoid introducing distributional mismatch. The original and corrected conditions also use exactly the same sequence-construction procedure, window boundaries, chronological data splits, and downstream evaluation metrics. Thus, the comparison changes the parsing representation while keeping the downstream evaluation protocol unchanged. However, because the expert-guided correction is derived from the same error-identification process used to construct the parsing-error taxonomy, the correction stage is not fully independent of error identification. The corrected condition should therefore be interpreted as a controlled oracle-style intervention rather than an independently generated alternative parsing condition.

RQ3: Does incorporating additional training data improve anomaly detection on industrial datasets?

The third research question examines whether augmenting limited industrial training data with external public logs can improve anomaly detection performance in practice. Industrial log datasets are often small, proprietary, and difficult to label, which constrains the training of data-driven detection models. A common strategy is to incorporate additional logs from public datasets; however, it remains unclear whether such cross-dataset training yields tangible benefits given differences in logging formats and system semantics.

To study this question, we aggregate the representative 2000-log subsets from the 16 LogHub datasets and apply the stratified diversity-aware sampling strategy described in Algorithm 1 to select a diverse subset of candidate logs. Domain experts further review this subset to remove highly similar or redundant samples, resulting in a curated auxiliary training set of 7000 log messages. This dataset is then used as additional training data for each of the five anomaly detection models. For each of the four industrial datasets, we compare model performance under two training conditions: (i) using only the 3000 representative industrial logs as training data, and (ii) using the same industrial training data augmented with the 7000 external logs. Detection performance is evaluated on the corresponding industrial test sets. Through this comparison, we aim to assess whether and to what extent external public logs can enhance anomaly detection under data-scarce industrial conditions, and whether such benefits are consistent across models.

To further examine the stability of the augmentation effect, we conducted an additional ablation study by varying both the size and composition of the auxiliary public-log set. For size ablation, we constructed auxiliary sets containing 3000, 5000, and 7000 logs, while keeping the same diversity-aware sampling strategy. For composition ablation, we fixed the auxiliary size at 7000 logs and compared four sampling strategies. Random LogHub sampling uniformly selects logs from the pooled LogHub candidates without considering their similarity to the target industrial datasets. Diversity-aware sampling first groups candidate logs according to structural patterns, parsed templates, and source datasets, and then samples from different groups to maximize coverage while reducing near-duplicate logs. Domain-compatible sampling ranks candidate logs by their lexical similarity, template-level similarity, and domain relevance to the target industrial dataset, and preferentially selects logs with higher compatibility scores. In contrast, mismatch-heavy sampling intentionally selects logs with low lexical, template-level, or domain similarity to the target dataset, serving as a stress-test setting for examining negative transfer. All configurations use the same downstream training and evaluation protocol to ensure fair comparison.

RQ4: How do practitioners perceive the differences between industrial logs and public datasets, and their impact on log analysis?

In this section, we conduct a semi-structured interview study with 11 engineers involved in the development and maintenance of the four industrial systems considered in this work. Prior to the interviews, participants are provided with a summary of our key empirical findings, including (i) the taxonomy of six log parsing error types identified in RQ1, (ii) the observed impact of parsing errors on anomaly detection performance in RQ2, and (iii) the mixed effects of cross-dataset augmentation in RQ3. This design ensures that participants can reflect on concrete evidence while relating the findings to their own experience.

During the interviews, we focus on three aspects: (1) perceived differences between industrial logs and public datasets in terms of structure, variability, and evolution; (2) the relevance and prevalence of the identified parsing error types in real-world systems; and (3) the applicability and limitations of existing anomaly detection approaches, including the use of external data for training. Each interview lasts approximately 30–45 min and is conducted online.

The collected responses are analyzed through thematic coding, and the resulting themes are used to interpret and contextualize the quantitative findings from RQ1–RQ3. This design allows us to bridge empirical evaluation with practitioner insights and to better understand the practical implications of log analysis techniques in industrial environments.

4.3. Evaluation protocol

In this section, we describe the log sequence construction, data splitting, and evaluation metrics used for anomaly detection. Before training anomaly detection models, raw log entries were converted into log sequences using a consistent windowing protocol. Following common practice in log anomaly detection (He et al., 2016; Du et al., 2017; Yang et al., 2021), we adopted a hybrid strategy according to dataset characteristics. For datasets with explicit execution identifiers, such as block IDs, application/task IDs, instance IDs, or request IDs, we used session-based windows, where all log events sharing the same identifier were grouped into one log sequence. For datasets without reliable session identifiers, including the four industrial datasets, we used a fixed event-count sliding window with a window size of 20 log events and a stride of 10 log events. We used event-count windows rather than fixed time windows for the industrial datasets because their logs are often sparse, bursty, and asynchronously generated across modules, making time-based windows less stable across systems.

Each log sequence was assigned a binary label. A sequence was labeled as anomalous if it contained at least one anomalous log entry; otherwise, it was labeled as normal. After sequence construction, each dataset was split chronologically into training, validation, and test sets with a ratio of 60%/20%/20%. We used chronological splitting rather than random shuffling to better simulate deployment, where models are trained on historical logs and evaluated on future logs, and to reduce temporal data leakage. For RQ2, the original and corrected parsing settings used exactly the same window boundaries and data splits. For RQ3, the Base and Aug settings used the same industrial training, validation, and test splits; the auxiliary public logs were added only to the training set and were never included in validation or test data.

Log anomaly detection is treated as a classification problem, and to assess the effectiveness of models in detecting anomalies, we rely on Precision, Recall, and F1-Score metrics to evaluate model performance. The accuracy metric, which calculates the proportion of all predictions correctly made by the anomaly detection model out of the total number of samples, is not used. This is because, in most datasets, normal logs make up the majority, while anomalous logs constitute only a small portion, which is characteristic of well-qualified software. Consequently, a model could attain a high accuracy score even if it predicts all logs as normal. The definitions of each metric are as follows:

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Specifically, TP (True Positive) is the count of anomalous log sequences correctly detected by the model. FP (False Positive) is the count of normal log sequences incorrectly identified as anomalies. TN (True Negative) is the count of normal logs that are correctly classified. FN (False Negative) is the count of anomalous log sequences that the model failed to detect.

5. Results and analysis

5.1. Taxonomy and prevalence of log parsing errors (RQ1)

Using the expert-led roundtable protocol described in Section 3.3, we identified six recurring log parsing error types across the 16 LogHub datasets and four industrial datasets. Five error types were consistently observed in both rule-based and LLM-based parsers, reflecting fundamental challenges inherent to log structure variability and logging drift. One additional error type was found to be specific to LLM-based parsers, stemming from model-driven semantic rewriting. Table 5 summarizes the taxonomy and the observed occurrence frequency of each error type.

As shown in Table 5, the five shared error types (E1–E5) together account for more than 90% of observed parsing errors, indicating that most failures arise from structural ambiguity and logging variability rather than parser-specific implementation details. Among these, template over-splitting (E2) and over-merging (E1) are the most prevalent, jointly accounting for nearly half of all observed errors. Engineers emphasized that these two error types are frequently encountered during routine system monitoring and often go unnoticed until downstream analysis exhibits degraded performance. To further illustrate how different parsing error types manifest in practice, Table 6 presents representative examples observed across datasets, summarized and anonymized based on expert review.

Template over-merging (E1) occurs when distinct log events are incorrectly collapsed into a single template. A representative example observed across multiple datasets involves error and success messages that share a common prefix but differ in a critical status token or error code. For instance, log messages such as “request completed with status OK” and “request completed with status FAILED” were occasionally merged into the same template, with the status token treated as a variable. Engineers noted that this seemingly minor abstraction can have severe consequences: by masking semantically meaningful distinctions, over-merging suppresses weak anomaly signals and increases false negatives, particularly in scenarios where failures are rare and only differ from normal behavior by a small set of tokens.

In contrast, template over-splitting (E2) fragments a single event type into many templates due to minor lexical or formatting variations. Common examples include differences in whitespace, optional fields, or dynamically generated substrings embedded in log messages. For example, a recurring operational message may be split into dozens of templates simply because of varying parameter order or formatting styles across software versions. Engineers described this as a “template explosion” phenomenon, which inflates the event space and reduces the statistical support for each individual template. Downstream anomaly detectors operating on sequences or frequency statistics are then forced to learn from sparse and unstable representations, leading to increased variance and degraded generalization.

Variable boundary and extraction errors (E3), inconsistent parameter masking (E4), and unseen-template handling errors (E5) are closely related failure modes that frequently co-occur in evolving software systems. Engineers repeatedly characterized these errors as “silent but harmful”, as they often preserve superficial template similarity while breaking critical structural invariants. In several observed cases, identifiers such as request IDs, container hashes, or session tokens were only partially extracted (e.g., truncated prefixes or suffixes), or inconsistently masked across log occurrences. For example, the same request identifier might be treated as a variable in some log messages but retained as a constant in others, causing two messages corresponding to the same operational event to be parsed into different templates or parameter sets. Such inconsistencies undermine the assumption — implicit in many anomaly detection models — that certain fields remain stable across normal execution traces, thereby distorting learned sequence patterns and representations.

Table 5

Taxonomy of common log parsing error types and their observed occurrence frequency (%).

Error type	Description (Operational symptom)	All	Rule	LLM
E1: Template over-merging	Distinct events are collapsed into one template, reducing discriminative power for downstream detection.	21.4	17.2	25.6
E2: Template over-splitting	Minor lexical variations lead to excessive template fragmentation and sparsity.	24.7	31.8	17.5
E3: Variable boundary/Extraction error	Variable spans are partially or incorrectly extracted, corrupting key identifiers.	18.9	22.6	15.3
E4: Inconsistent parameter masking	The same parameter is inconsistently treated as constant or variable across occurrences.	14.6	15.9	13.2
E5: Unseen template (OOV) handling error	New or rare templates are mis-matched or collapsed during logging drift.	12.3	12.5	12.0
E6: LLM semantic rewriting/Hallucinated template	(LLM-only) The parser paraphrases or introduces tokens not present in the raw log.	8.1	–	16.4

Table 6

Representative examples of common log parsing error types (E1–E6).

Type	Raw log message (Simplified)	Incorrect parsed template	Correct parsing intuition
E1	request completed with status OK request completed with status FAILED operation finished with code 0/1 connection established to node A connection timeout to node A	request completed with status (*) operation finished with code (*) connection (*) to node (*)	Distinguish success and failure events as separate templates; status and error codes should not be abstracted away. Connection success and timeout represent different control paths and should not be merged.
E2	load config file = /etc/app.yaml load config file = /etc/app.json load config file = /etc/app.conf user login from ip = 10.0.0.1 user login from ip = 10.0.0.2	load config file = /etc/app.yaml load config file = /etc/app.json load config file = /etc/app.conf user login from ip = 10.0.0.1 user login from ip = 10.0.0.2	File path should be treated as a variable to avoid template explosion. IP addresses are parameters and should be consistently masked.
E3	request id = abc12345xyz timeout request id = abc67890xyz timeout container = nginx-7f9c8d restart container = nginx-8a1e3f restart	request id = abc123 (*) timeout container = nginx-* restart	The full request identifier should be captured as a single variable. Container hashes should be fully parameterized, not partially truncated.
E4	session = 9f3a login success session = 9f3a login success node = worker-01 heartbeat ok node = worker-01 heartbeat ok	session = (*) login success session = 9f3a login success node = (*) heartbeat ok node = worker-01 heartbeat ok	The same field should be consistently treated as either constant or variable. Inconsistent masking breaks template stability across identical events.
E5	new feature flag enabled for user feature flag updated for user warning: cache size limit reached (v2) warning: cache limit reached	unknown template unknown template warning (*) reached	New templates should be created instead of collapsing into a generic bucket. Minor version differences should not force new logs into overly generic templates.
E6	operation timed out after 30 s retry limit exceeded	operation failed due to network latency request aborted after multiple failures	Parser should preserve original wording without semantic inference or rewriting. LLM-based parser should not introduce inferred causes not present in the raw log.

These issues are further amplified under system evolution, where unseen-template (OOV) handling errors (E5) become prominent. Engineers reported multiple instances in which newly introduced log statements following software updates or configuration changes were either force-matched to existing templates or collapsed into a generic “unknown” category. In one industrial dataset, a warning message introduced after a minor version update differed from an existing template only by an additional contextual field; however, it was consistently mapped to a generic template, causing the anomaly detector to flag large volumes of benign logs as anomalous. In another case, a new informational log emitted during a feature rollout was partially matched to an error-related template due to shared lexical tokens, leading to spurious anomaly signals. Together, these examples illustrate how parameterization errors and inadequate handling of unseen templates can blur the boundary between genuine anomalies and normal system evolution, resulting in both inflated false positives and suppressed detection of subtle abnormal behaviors.

While rule-based and LLM-based parsers share the majority of error types, their error profiles differ in systematic ways. Rule-based parsers exhibit higher rates of over-splitting (E2) and boundary-related errors (E3), reflecting their sensitivity to token-level variation and rigid heuristics. LLM-based parsers, on the other hand, reduce some superficial fragmentation by leveraging semantic similarity but are more prone to over-merging (E1) and semantic generalization. Engineers observed

that LLM-based outputs often appear cleaner and more readable, which can obscure subtle but consequential deviations from the raw log content.

A distinctive error mode unique to LLM-based parsers is semantic rewriting or hallucinated templates (E6), accounting for approximately 16% of LLM-related errors. In these cases, the parser paraphrases log messages, expands abbreviations, or introduces inferred causal language that does not exist in the original log. For example, a raw message indicating a timeout might be rewritten as “operation failed due to network latency”, introducing assumptions that are not explicitly supported by the log. Engineers consistently expressed concern over this behavior, emphasizing that operational debugging and forensic analysis rely on strict traceability between parsed templates and raw log messages. Even small semantic deviations can hinder log search, correlation, and compliance auditing.

 **Answer to RQ1:** The findings illustrate that log parsing errors are not merely cosmetic preprocessing artifacts but structured failure modes with concrete operational consequences. Different error types distort event templates, variable fields, and sequence structure in distinct ways, suggesting that their impact on anomaly detection is heterogeneous rather than uniform.

Table 7

The F1-score (%) of five anomaly detection models on 16 LogHub datasets and 4 industrial datasets, evaluated on original parsed logs (Orig.) and corrected parsed logs (Corr.). $\uparrow$ indicates absolute F1-score change in percentage points.

Dataset	PLELog (Orig./Corr.†)	LogBERT (Orig./Corr.†)	SemiRALD (Orig./Corr.†)	SemiSMAC (Orig./Corr.†)	LogRobust (Orig./Corr.†)
HDFS	78.6/81.2(+2.6%)	85.1/88.0(+2.9%)	93.8/95.4(+1.6%)	87.5/90.3(+2.8%)	61.4/63.0(+1.6%)
Hadoop	74.3/77.1(+2.8%)	82.6/85.7(+3.1%)	94.2/95.8(+1.6%)	85.9/89.0(+3.1%)	59.8/61.5(+1.7%)
Spark	69.1/72.8(+3.7%)	78.4/82.6(+4.2%)	91.0/93.6(+2.6%)	82.3/86.4(+4.1%)	54.6/57.4(+2.8%)
Zookeeper	72.5/75.0(+2.5%)	81.9/84.7(+2.8%)	94.6/96.0(+1.4%)	84.1/86.8(+2.7%)	56.8/58.4(+1.6%)
OpenStack	61.8/65.7(+3.9%)	70.4/75.0(+4.6%)	88.2/91.0(+2.8%)	77.6/82.4(+4.8%)	48.9/52.0(+3.1%)
BGL	66.4/70.2(+3.8%)	75.8/79.9(+4.1%)	90.1/92.9(+2.8%)	80.9/85.1(+4.2%)	50.7/53.5(+2.8%)
HPC	58.7/62.9(+4.2%)	67.9/72.5(+4.6%)	86.3/89.8(+3.5%)	74.8/79.7(+4.9%)	45.2/48.4(+3.2%)
Thunderbird	52.1/56.9(+4.8%)	61.7/67.3(+5.6%)	83.5/87.9(+4.4%)	70.6/76.8(+6.2%)	41.6/45.2(+3.6%)
Windows	63.5/67.4(+3.9%)	72.2/76.5(+4.3%)	88.9/91.7(+2.8%)	78.5/83.1(+4.6%)	49.4/52.6(+3.2%)
Linux	65.0/68.8(+3.8%)	63.9/68.1(+4.2%)	89.6/92.4(+2.8%)	79.4/83.8(+4.4%)	50.1/53.2(+3.1%)
Mac	60.3/64.2(+3.9%)	69.5/74.0(+4.5%)	87.2/90.3(+3.1%)	76.9/81.7(+4.8%)	47.6/50.8(+3.2%)
Android	62.7/66.5(+3.8%)	71.1/75.4(+4.3%)	88.0/91.2(+3.2%)	78.0/82.6(+4.6%)	46.8/50.1(+3.3%)
HealthApp	48.9/52.8(+3.9%)	57.4/62.1(+4.7%)	82.4/86.9(+4.5%)	68.7/74.5(+5.8%)	39.2/42.6(+3.4%)
Apache	67.8/71.6(+3.8%)	76.5/80.7(+4.2%)	91.2/93.8(+2.6%)	81.9/86.2(+4.3%)	52.9/55.6(+2.7%)
OpenSSH	59.6/63.7(+4.1%)	68.2/72.9(+4.7%)	88.7/91.9(+3.2%)	77.1/82.2(+5.1%)	46.3/49.4(+3.1%)
Proxifier	54.8/59.3(+4.5%)	63.5/68.8(+5.3%)	85.1/89.6(+4.5%)	72.4/78.7(+6.3%)	42.7/46.6(+3.9%)
FALL	44.9/48.2(+3.3%)	52.8/56.1(+3.3%)	65.7/68.9(+3.2%)	58.3/62.4(+4.1%)	34.6/36.9(+2.3%)
PIMS	39.8/42.9(+3.1%)	48.6/52.1(+3.5%)	61.9/65.2(+3.3%)	55.4/59.2(+3.8%)	31.7/33.8(+2.1%)
SCADA-X	31.2/34.6(+3.4%)	46.9/51.3(+4.4%)	58.4/62.6(+4.2%)	42.1/46.8(+4.7%)	25.9/28.1(+2.2%)
BOSS	53.7/56.8(+3.1%)	60.1/63.5(+3.4%)	70.8/73.6(+2.8%)	66.2/69.9(+3.7%)	41.3/43.7(+2.4%)

5.2. Downstream sensitivity of anomaly detection to parsing errors (RQ2)

In this section, we quantify how log parsing errors propagate to downstream anomaly detection by comparing model performance on (i) the original parsed logs and (ii) the corrected parsed logs, where expert-verified parsing errors are manually rectified in both training and test sets. Table 7 reports F1-scores for five detection models across 20 datasets, together with the relative improvement after correction. It is important to note that the corrected results in Table 7 represent an upper-bound estimate under expert-guided correction. They should not be interpreted as the performance that an automated parser-repair method would necessarily achieve in deployment. Instead, the gap between the original and corrected settings quantifies the maximum recoverable performance loss attributable to the expert-identified parsing errors under our experimental protocol.

Across the 20 datasets and five detection models, corrected parsing produced positive F1-score gains in most dataset-detector pairs. This pattern indicates that parsing errors are not merely cosmetic artifacts at the preprocessing stage; instead, they systematically distort the event template space and variable representations that detection models rely on. The gains are particularly pronounced on datasets with heterogeneous formats and higher structural variability (e.g., Thunderbird, HealthApp, Proxifier, and several industrial datasets). A notable trend is that industrial datasets exhibit larger relative improvements after correction (typically +6% to +10%) compared to many public LogHub datasets (often +3% to +8%). Industrial datasets are typically data-scarce; when training data is limited, parsing noise consumes a larger fraction of the effective learning signal, making models more sensitive to template distortions.

Among the evaluated models, SemiRALD achieves the best overall performance across nearly all datasets under both original and corrected parsing conditions. It also exhibits comparatively smaller relative gains after correction, indicating stronger robustness to parsing noise. This behavior can be attributed to SemiRALD's use of a more complex hybrid language modeling architecture, which integrates multiple deep learning models to capture both local token patterns and longer-range contextual dependencies in log sequences. In contrast, LogRobust consistently yields the lowest F1-scores across datasets, and its improvements from parsing correction are modest relative to semi-supervised alternatives. This outcome is expected in realistic settings where anomalies are rare and labels are limited or noisy.

We further observe that models based on contextual representation learning, such as LogBERT and SemiRALD, consistently outperform

PLELog across most datasets, particularly those with richer event semantics and longer contextual dependencies (e.g., Hadoop, Thunderbird, and the industrial datasets). This advantage stems from their stronger representational capacity to capture semantic and sequential patterns, making them less sensitive to local distortions caused by imperfect parsing. SemiSMAC generally tracks closely behind SemiRALD, and its improvements after correction indicate that cleaner parsed logs primarily enhance the quality of pseudo-labels and grouping structures, thereby stabilizing the optimization process rather than fundamentally increasing model expressiveness.

To further examine whether the observed improvements are statistically reliable, we conducted paired non-parametric tests between the original and corrected parsing settings. Each dataset-detector pair was treated as one paired observation, resulting in 100 paired comparisons. Since the paired F1-score differences were not assumed to be normally distributed, we applied the Wilcoxon signed-rank test and reported the matched-pairs rank-biserial correlation r as the effect-size measure. Holm correction was used for multiple comparisons.

As shown in Table 8, corrected parsing led to a statistically significant improvement over original parsing overall ($p_{\text{Holm}} < 0.001$, $r = 1.00$). The rank-biserial correlation of $r = 1.00$ reflects the fact that all 100 dataset-detector pairs exhibited positive signed differences under the expert-correction condition; it should not be interpreted as evidence of perfect predictive association or as an independent validation of the correction procedure. The mean and median F1-score increases were 3.61 and 3.55 percentage points, respectively, with improvements observed on both LogHub-2.0 and industrial datasets. This unusually consistent direction of improvement should also be interpreted in light of the oracle-style experimental design. The corrected representations were derived from expert-identified parsing errors and were intended to estimate the recoverable downstream loss when these identified errors are rectified. Because error identification and correction were not fully independent stages, the uniform positive direction may partly reflect the controlled nature of this intervention. Therefore, the result provides strong evidence of downstream sensitivity to the identified parsing errors under our correction protocol, but should not be interpreted as an independently validated estimate of the effect that would necessarily arise under an external or automated correction procedure.

At the detector level, all five anomaly detection models showed positive average gains after parsing correction. The gains were relatively smaller for SemiRALD, which already achieved strong baseline performance, but the improvement remained statistically significant.

Table 8

Statistical robustness analysis for RQ2. Paired comparisons are conducted between corrected parsed logs and original parsed logs. Each dataset–detector pair is treated as one paired observation. $\Delta F1$ is reported in percentage points. Average relative improvement is computed as the mean of $(F1_{corr.} - F1_{orig.})/F1_{orig.}$ across paired observations.

Comparison	n	+ / 0 / -	Mean $\Delta F1$	Median $\Delta F1$	Avg. Rel. Imp.	p_{Holm}	Rank-biserial r
Overall	100	100/0/0	+3.61	+3.55	+5.8%	<0.001	1.00
LogHub-2.0 datasets	80	80/0/0	+3.68	+3.80	+5.5%	<0.001	1.00
Industrial datasets	20	20/0/0	+3.32	+3.30	+7.0%	<0.001	1.00
PLELog	20	20/0/0	+3.65	+3.80	+6.5%	<0.001	1.00
LogBERT	20	20/0/0	+4.14	+4.25	+6.3%	<0.001	1.00
SemiRALD	20	20/0/0	+3.05	+2.95	+3.8%	<0.001	1.00
SemiSMAC	20	20/0/0	+4.45	+4.50	+6.2%	<0.001	1.00
LogRobust	20	20/0/0	+2.77	+2.95	+6.2%	<0.001	1.00

Table 9

The F1-score (%) on four industrial datasets with and without external data augmentation. $\uparrow$ indicates absolute F1-score change in percentage points.

Dataset	PLELog (Base/Aug $\uparrow$)	LogBERT (Base/Aug $\uparrow$)	SemiRALD (Base/Aug $\uparrow$)	SemiSMAC (Base/Aug $\uparrow$)	LogRobust (Base/Aug $\uparrow$)
FALL	48.2/56.8(+8.6%)	56.1/71.2(+15.1%)	68.9/74.2(+5.3%)	62.4/68.8(+6.4%)	36.9/48.6(+11.7%)
PIMS	42.9/48.3(+5.4%)	52.1/58.4(+6.3%)	65.2/68.1(+2.9%)	59.2/66.7(+7.5%)	33.8/51.3(+17.5%)
SCADA-X	34.6/49.3(+14.7%)	51.3/49.2(-2.1%)	62.6/67.0(+4.4%)	46.8/57.2(+10.4%)	28.1/42.8(+14.7%)
BOSS	56.8/62.4(+5.6%)	63.5/61.6(-1.9%)	73.6/73.1(-0.5%)	69.9/72.5(+2.6%)	43.7/55.1(+11.4%)

This suggests that robust detectors can partially tolerate parsing errors, but may not fully eliminate their negative effect.

Answer to RQ2: Parsing errors systematically affect downstream anomaly detection. Corrected parsing brings statistically significant F1-score improvements across dataset–detector pairs, especially on industrial datasets. This suggests that parser quality is a critical factor for reliable log anomaly detection.

5.3. Effects of external data augmentation on industrial anomaly detection (RQ3)

In this section, we evaluate whether augmenting data-scarce industrial training sets with an auxiliary set of 7000 curated logs from LogHub improves anomaly detection performance. Table 9 reports F1-scores for five models under two training conditions: *Base* (training on the 3000 industrial logs only) and *Aug* (training on the same industrial logs augmented with the 7000 external logs). The relative change is computed as $(Aug - Base)/Base$.

As shown in Table 9, augmenting the industrial training data with 7000 curated external logs improves F1-score across all four industrial datasets for most models. The gains are substantial for PLELog and LogRobust (typically around +5% to +15% relative improvement), suggesting that carefully selected external logs can act as an effective regularizer under data scarcity. An interesting outcome is that **SemiRALD** shows comparatively smaller gains from augmentation, despite being the strongest overall model. This suggests that SemiRALD already extracts robust, transferable features from the limited industrial training data, leaving less headroom for improvement from additional external logs. In contrast, **LogRobust** benefits from augmentation with consistent improvements. This is notable because LogRobust is fully supervised and typically constrained by the limited volume of labeled training data available in industrial settings. The observed improvement indicates that augmenting the training set with additional logs increases overall sample diversity and coverage of normal operational patterns, thereby reducing variance during training and enabling the model to learn a more stable decision boundary. Although the reliance on labeled anomalies remains a practical bottleneck, the increase in effective training data volume helps mitigate overfitting to small industrial datasets and leads to more reliable detection performance.

Despite the overall benefits of data augmentation, we observe a subtle but consistent negative transfer effect for **LogBERT** on certain

Table 10

Statistical robustness analysis for RQ3. Paired comparisons are conducted between the settings with and without external data augmentation. Each dataset–detector pair is treated as one paired observation. $\Delta F1$ is reported in percentage points. *Note.* The dataset-level rows are reported descriptively because each dataset contains only five paired observations. Therefore, statistical significance is tested only at the overall level.

Scope	n	+ / 0 / -	Mean $\Delta F1$	Median $\Delta F1$	p	Rank-biserial r
Overall	20	17/0/3	+7.30	+6.35	<0.001	0.94
FALL	5	5/0/0	+9.42	+8.60	–	1.00
PIMS	5	5/0/0	+7.92	+6.30	–	1.00
SCADA-X	5	4/0/1	+8.42	+10.40	–	0.87
BOSS	5	3/0/2	+3.44	+2.60	–	0.60

industrial datasets, notably SCADA-X and BOSS. In these cases, augmenting the training set with external logs leads to a small reduction in F1-score (approximately 2%). A plausible explanation is that LogBERT learns normal system behavior by modeling contextual regularities and constructing a compact representation of normal log sequences. When additional external logs with mismatched semantic or structural patterns are introduced, the learned normal representation may become overly broad, causing domain-specific but informative deviations in the target system to be absorbed into the expanded notion of normality. As a result, subtle anomalies that are distinctive within the industrial system become harder to separate from normal behavior. This observation underscores that data augmentation for contextual representation learning models requires careful attention to domain compatibility and targeted curation, rather than indiscriminate inclusion of external logs.

To assess whether external data augmentation led to statistically reliable improvements, we conducted a paired Wilcoxon signed-rank test using each dataset–detector pair as one paired observation. This resulted in 20 paired comparisons across four industrial datasets and five anomaly detection models. We report the mean and median F1-score differences, the number of positive/tied/negative pairs, and the matched-pairs rank-biserial correlation as the effect-size measure.

As shown in Table 10, external data augmentation produced positive gains in 17 out of 20 paired comparisons. The overall improvement was statistically significant ($p < 0.001$), with a large rank-biserial effect size ($r = 0.94$). However, the magnitude and direction of improvement varied across datasets and models. The gains were strongest on FALL, PIMS, and SCADA-X, while BOSS showed smaller improvements and more negative-transfer cases. These results suggest that external data

Table 11

Ablation study on the size and composition of the auxiliary public-log set for RQ3. Results are averaged over four industrial datasets and five anomaly detection models. $\Delta F1$ is reported in percentage points relative to the Base setting without external augmentation.

Ablation type	Auxiliary set	Aux. size	Mean F1	Mean $\Delta F1$	Median $\Delta F1$	+0/−
Baseline	No external logs	0	52.83	–	–	–
Size	Diversity-aware sampling	3000	58.48	+5.65	+4.97	16/0/4
Size	Diversity-aware sampling	5000	59.62	+6.79	+5.86	17/0/3
Size	Diversity-aware sampling	7000	60.13	+7.30	+6.35	17/0/3
Composition	Random LogHub sampling	7000	58.74	+5.91	+4.88	15/0/5
Composition	Domain-compatible sampling	7000	59.72	+6.89	+6.10	17/0/3
Composition	Diversity-aware sampling	7000	60.13	+7.30	+6.35	17/0/3
Composition	Mismatch-heavy sampling	7000	55.18	+2.35	+1.42	12/0/8

augmentation is generally beneficial under the current curated setting, but the degree of benefit remains dataset- and model-dependent.

To examine whether the augmentation effect depends on the fixed 7000-log auxiliary set, we conducted an ablation study by varying auxiliary-set size and composition. Specifically, we tested 3000, 5000, and 7000 logs under diversity-aware sampling, and compared four 7000-log composition settings: random LogHub sampling, domain-compatible sampling, diversity-aware sampling, and mismatch-heavy sampling, as summarized in Table 11. The size ablation shows that external augmentation remains beneficial across different auxiliary-set sizes, but the gains increase only moderately as more external logs are added. Compared with the Base setting, using 3000 external logs improves the mean F1-score from 52.83 to 58.48, with positive gains in 16 out of 20 dataset–detector pairs. Increasing the auxiliary set to 5000 logs further improves the mean F1-score to 59.62. The 7000-log setting achieves the best overall result, with a mean F1-score of 60.13 and an average improvement of 7.30 percentage points. These results suggest that the observed benefit is not an artifact of a single auxiliary-set size, although the improvement begins to saturate as the auxiliary set becomes larger.

The composition ablation further shows that the usefulness of external logs depends on both domain compatibility and sample diversity. Random LogHub sampling improves over the Base setting, but it is less effective than the curated sampling strategies. Domain-compatible sampling achieves a mean F1-score of 59.72, which is close to but slightly lower than diversity-aware sampling. This indicates that lexical, template-level, and domain similarity are useful for selecting auxiliary logs, but overly focusing on similarity may reduce coverage of diverse operational patterns. Diversity-aware sampling achieves the best overall result, with a mean F1-score of 60.13 and positive gains in 17 out of 20 dataset–detector pairs, suggesting that balancing compatibility and diversity is more effective than relying on compatibility alone. In contrast, mismatch-heavy sampling produces much smaller gains, with a mean F1-score of 55.18 and eight negative-transfer cases. This indicates that negative transfer is not a random anomaly in the fixed 7000-log setting, but becomes more pronounced when the auxiliary data contain logs that are structurally or semantically distant from the target industrial system. Therefore, cross-dataset augmentation should be treated as a compatibility- and diversity-sensitive strategy rather than a simple data-scaling approach.

Answer to RQ3: External log augmentation brings statistically reliable overall gains on data-scarce industrial datasets, but the benefits remain model-, dataset-, and auxiliary-data-dependent. Ablation results show that balancing domain compatibility and sample diversity is more effective than simply increasing auxiliary data size, while mismatch-heavy logs increase negative transfer. Therefore, cross-dataset training is promising but conditional.

5.4. Practitioner perspectives on industrial logs and public benchmarks (RQ4)

To complement the quantitative findings in RQ1–RQ3, we analyzed the interview data from 11 engineers and summarized the major

practitioner-reported themes regarding industrial log analysis. Table 12 summarizes the six major themes identified from the practitioner interviews, together with their frequency and their relationship to the preceding research questions.

As shown in Table 12, the most frequently mentioned concerns center on structural heterogeneity, parsing reliability, and the limited transferability of public logs. The interview results strongly support the main quantitative findings of this study, while also offering important contextual explanations that are difficult to obtain from benchmark-based experiments alone. In particular, practitioners consistently emphasized that industrial logs differ from public datasets not only in scale or label availability, but more fundamentally in their structural instability, semantic coupling with system context, and operational consequences when parsing or detection fails. These observations help explain why industrial datasets in our experiments exhibited both lower baseline anomaly detection performance and higher sensitivity to parsing correction.

Theme 1: Industrial logs are more heterogeneous, context-dependent, and evolving than public datasets. The most frequently mentioned theme, raised by 10 out of 11 engineers, is that industrial logs are substantially more heterogeneous than the public LogHub datasets. Participants repeatedly noted that industrial systems often combine logs from multiple modules, services, devices, or deployment environments, each following different logging conventions. Two engineers explained:

“Our logs are not consistent even within the same system. Different modules log in different styles, and those styles also change when teams update components independently.”

“Compared with benchmark datasets, industrial logs are much less standardized. Some messages are structured, some are semi-structured, and some are almost free text.”

This perception directly aligns with the differences observed between the four industrial datasets and the 16 LogHub datasets. For example, datasets such as HDFS and Hadoop contain relatively stable event patterns and have long been used as parsing benchmarks, whereas our industrial datasets — especially SCADA-X and BOSS — exhibit stronger format drift, asynchronous event reporting, and module-specific syntactic variation. In SCADA-X, engineers noted that the same control-state transition may be logged differently by different subsystems, for example: [CTRL_A] mode=SYNC state_change ID=2048 from=STANDBY to=ACTIVE and [MON_SYNC] device 2048 entered ACTIVE after sync_ack delay=28 ms. Although both messages describe the same operational transition, one is emitted by a local controller and the other by a synchronization monitor, and the lexical forms, field order, and contextual information are markedly different. In BOSS, a single business transaction may also generate multiple logs across modules, such as [WF] ticket=TX9381 step=approve user=U17 status=passed

Table 12
Summary of practitioner-reported themes in RQ4.

Theme	Freq.	Related findings
Industrial logs are more heterogeneous, context-dependent, and evolving than public datasets	10/11	RQ1, RQ2
Parsing errors are common and have direct downstream consequences	9/11	RQ1, RQ2
Industrial logs differ from LogHub not only in form, but also in semantics and operational meaning	8/11	RQ1, RQ3
Public data can improve coverage, but only to a limited extent	8/11	RQ3
Data scarcity and labeling constraints shape what is practically feasible	9/11	RQ2, RQ3
Robustness, faithfulness, and traceability matter more than marginal performance gains	7/11	RQ1, RQ2, RQ3

and [AUDIT] txn TX9381 auth_check success actor=U17 module=approval. Here, both logs refer to the same underlying workflow event, yet they use different syntactic structures and module-specific vocabularies. Practitioners emphasized that this heterogeneity is not merely cosmetic: two log messages with similar surface forms may correspond to very different operational states, while messages describing the same event may appear in noticeably different formats over time. This helps explain why industrial datasets in RQ2 begin with lower baseline F1-scores and why parsing correction produces a larger downstream effect in these settings.

Theme 2: Parsing errors are common in practice and have direct downstream consequences. Nine participants explicitly stated that parsing errors are frequent in industrial systems and often have consequences that extend far beyond the parsing stage itself. One engineer noted:

“We often see cases where identifiers are only partially extracted, or where two different events are grouped into one template. Once that happens, the downstream monitoring logic becomes unreliable.”

Another engineer stated:

“Parsing errors are not just parser-level problems. They can directly change what the anomaly detector sees, so the final alerts are also affected.”

This practitioner perspective provides strong qualitative support for RQ2. In our experiments, correcting parsing outputs consistently improved anomaly detection, and the interview data clarifies why. Engineers stressed that the most harmful parsing errors are those that alter the effective event space seen by downstream models. For instance, when failure and recovery messages are incorrectly merged into the same template, anomaly detectors lose a critical signal that separates abnormal breakdown from normal system recovery. In practice, logs such as [ERR] disk_failure node=12 code=E204 retry=0 and [INFO] disk_recovered node=12 after_retry=1 may be grouped into a single template like disk_{status} node={id}, effectively masking the transition from failure to recovery. As a result, the sequential pattern of “failure → recovery” — which is often a key indicator of normal system behavior — becomes indistinguishable from anomalous sequences where recovery does not occur.

Similarly, inconsistent extraction of variable fields disrupts the stability of event representations. For example, logs such as [REQ] id=8fa2_ttl user=U17 action=commit and [REQ] id=9bc4 user=U17 action=commit should ideally map to a consistent template with a parameterized request ID. However, if one parser extracts the ID correctly while another partially retains it (e.g., id=*_ttl vs. id=*), the same logical event may be represented as different templates. This inconsistency breaks the regularity of event sequences and introduces artificial variability, which can mislead models that rely on frequency patterns or contextual embeddings.

Engineers further emphasized that these issues are amplified in industrial settings, where log streams are both large in volume and highly heterogeneous. In large-scale monitoring environments, a single malformed template may affect thousands of log entries and thereby

distort the statistical patterns learned by downstream models. One practitioner explained:

“If a common event is split into multiple templates, the model may treat all of them as rare events, which can easily trigger false alarms.”

Conversely, when semantically distinct events are merged into the same template, genuinely anomalous behaviors may be absorbed into the learned representation of normal operation. As a result, both false positives and false negatives become more likely. These observations provide a practical explanation for the results of RQ2, where parsing errors consistently lead to degraded anomaly detection performance, with the effect being especially pronounced in industrial datasets characterized by high log volume, strong structural variability, and evolving system behavior.

Theme 3: Industrial logs differ from LogHub not only in form, but also in semantics and operational meaning. A more nuanced theme, raised by 8 out of 11 participants, is that the gap between industrial logs and public benchmark logs is semantic as well as structural. One practitioner explained:

“Even when a public log looks similar to ours, the meaning may be completely different. The same token can indicate a harmless state in one system and a warning sign in another.”

Another participant remarked:

“Benchmark logs are useful for testing general methods, but they do not capture the operational context we rely on when interpreting industrial logs.”

This insight helps explain the mixed results observed in RQ3. Public datasets may still provide useful surface-level coverage of generic operational events such as retries, timeouts, authentication failures, or file access patterns. For example, logs such as [WARN] retry connection id=42 attempt=3 or [ERROR] timeout while reading file path=/data/tmp commonly appear in both public benchmarks and industrial systems, allowing models to learn general failure patterns. However, these similarities are often limited to lexical or syntactic overlap and do not necessarily reflect the same operational meaning across systems.

Engineers repeatedly emphasized that industrial log interpretation is highly context-dependent. For instance, in SCADA-X, a message such as [SYNC] node=12 status=OK delay=35 ms may appear to indicate normal synchronization behavior. However, under specific timing conditions or system states, this same message may actually signal a control-state inconsistency when synchronization delays exceed acceptable thresholds. Similarly, in BOSS, logs such as [AUTH] user=U17 action=approve status=success and [WF] txn=TX9381 step=approve result=ok may appear benign in isolation, yet their operational meaning depends on transaction context (e.g., approval sequences, permission hierarchies, or workflow dependencies). Without access to this contextual information, models may misinterpret such events as normal patterns.

As a result, external public logs are only partially transferable. While they can help expand the coverage of common operational patterns, they may also introduce semantically mismatched signals into the training data. For example, a retry or synchronization event that is benign in a public dataset may correspond to an early warning signal in an industrial system. When such patterns are incorporated without context, the model may broaden its notion of normal behavior, making it more difficult to distinguish subtle anomalies specific to the target system. This phenomenon provides a practical explanation for the mixed outcomes in RQ3, where augmentation improves performance in some cases but leads to mild negative transfer in others.

Theme 4: Public data can help with coverage, but only to a limited extent. Eight engineers acknowledged that external public logs can be useful when industrial data is scarce, but they consistently described this benefit as conditional rather than universal. One engineer stated:

“If you only have a small amount of industrial logs, external data can help the model see more normal patterns. But it only works when those patterns are actually relevant.”

Another said:

“Public logs can be helpful as a supplement, but they are not a substitute for real industrial logs. If the external data is too different, it may confuse the model instead of helping it.”

These observations map closely to the augmentation results in RQ3. Practitioners generally agreed that external logs can improve coverage of generic operational behaviors, which may explain why models such as PLELog, SemiSMAC, and LogRobust benefit from augmentation in several industrial datasets. For example, public logs such as [WARN] retry connection id=42 attempt=3 or [ERROR] timeout while reading file path=/tmp/data capture common failure patterns that also appear in industrial systems. When such patterns are added to the training data, models can better learn baseline behaviors such as retry cycles, transient failures, or access errors, thereby improving detection performance when industrial data is limited.

At the same time, practitioners also anticipated the possibility of negative transfer, particularly for models that are sensitive to semantic regularities. This is consistent with our observation that LogBERT shows mild degradation on certain datasets after augmentation. Engineers explained that similar-looking log messages may carry different meanings across systems. For instance, a public log such as [SYNC] status=OK node=12 delay=20 ms may indicate normal synchronization behavior, whereas in SCADA-X, a log like [SYNC] node=12 status=OK delay=35 ms may signal a degraded or unstable state when the delay exceeds system-specific thresholds. Similarly, logs such as [AUTH] user=U17 action=approve status=success may represent routine operations in public datasets, but in BOSS, the same pattern may be part of a multi-step workflow where its interpretation depends on transaction context and sequence position.

Engineers interpreted this behavior as a symptom of over-broad normality learning. When external logs introduce structurally plausible but semantically mismatched patterns, models may expand the learned notion of normal behavior to include signals that are actually anomalous in the target system. As a result, the boundary between normal and anomalous behavior becomes less precise. This effect is particularly pronounced for representation-learning models, which rely on contextual similarity: once semantically inconsistent patterns are embedded close to normal events, subtle anomalies become harder to distinguish. This provides a concrete explanation for the mixed outcomes observed in RQ3, where augmentation improves performance in some cases but leads to mild degradation in others.

Theme 5: Data scarcity and labeling constraints shape what is practically feasible. Nine participants emphasized that data scarcity is not simply a matter of dataset size, but a broader issue involving annotation cost, uncertain anomaly boundaries, and the rarity of meaningful failure cases. One engineer commented:

“We rarely have enough labeled anomalies to train a supervised model well. Most of the time, anomalies are sparse, and verifying them requires domain experts.”

Another participant commented:

“The real bottleneck is not just collecting logs, but deciding which logs truly represent anomalies and making sure the labels are correct.”

This theme offers an important explanation for the model-level differences observed in RQ2 and RQ3. Practitioners consistently viewed semi-supervised methods as more realistic for industrial deployment because fully supervised training rarely aligns with the realities of industrial maintenance. In practice, anomalies are both rare and heterogeneous. For example, logs such as [ALERT] device=12 code=E782 pressure_drop=17% or [ERROR] workflow TX9381 step=commit failed reason=state_mismatch may occur only a handful of times in a large log stream, and often under slightly different conditions each time. Engineers emphasized that such events are difficult to label consistently, as their interpretation depends on system state, historical context, and domain knowledge.

As a result, even when supervised models have access to additional data, their effectiveness remains constrained by the limited number of reliable anomaly labels. One engineer noted that “we may see thousands of normal logs for every anomaly, but only a few anomalies are clearly labeled and verified”. In such settings, logs that appear similar at the surface level — such as [WARN] retry transaction TX9381 attempt=2 versus [ERROR] retry transaction TX9381 attempt=5 exceeded_limit — may correspond to normal retry behavior in one case and true anomalies in another. Without sufficient labeled examples, supervised models struggle to learn robust decision boundaries, especially when anomaly patterns are diverse and context-dependent.

This perspective is consistent with our findings that LogRobust improves with external augmentation but still remains substantially behind stronger semi-supervised alternatives. In contrast, semi-supervised models can leverage large volumes of unlabeled logs to learn normal behavior patterns, making them less sensitive to the scarcity of anomaly labels. This also helps explain why SemiRALD, which is designed for low-resource settings, remains robust even when training data is limited: by focusing on modeling normal log sequences and refining representations iteratively, it can better capture system regularities even when anomalies are sparse and weakly labeled.

Theme 6: Robustness, faithfulness, and traceability matter more than marginal performance gains. A final important theme, raised by 7 out of 11 engineers, is that practitioners value robustness and traceability more than small improvements in benchmark performance. One engineer noted:

“In production, I would prefer a parser or detector that is stable and explainable over one that is slightly more accurate but behaves unpredictably.”

Another engineer emphasized:

“We need to trace an alert back to the original logs. If the parsing output rewrites the meaning or hides key details, it becomes harder to trust the system.”

This concern resonates strongly with our RQ1 findings, especially the identification of LLM-specific semantic rewriting errors. While engineers appreciated the flexibility of LLM-based methods, several also expressed concern that semantically polished outputs may be less faithful to the raw logs. For example, a raw log such as [ERR] device=12 code=E204 temp=87.3C threshold=85C shutdown=1 may be rewritten into a more fluent representation like “Device 12 exceeded temperature limits and was shut down”. Although the rewritten version is easier to read, it omits critical fields such as the exact error code and threshold values, which are essential for diagnosing the root cause and verifying system behavior.

A similar issue arises in more complex operational logs. For instance, a raw message such as [AUTH] user= U17 role=admin action=approve txn=TX9381 result=fail detailed reason=permission_mismatch may be summarized as “Approval failed due to insufficient permissions”. While this summary captures the high-level outcome, it removes structured information such as user roles, transaction identifiers, and specific failure reasons, which are often required for auditing and traceability. Engineers emphasized that such transformations may obscure subtle distinctions between different failure modes, especially when multiple conditions lead to similar high-level descriptions.

From a practical perspective, these observations highlight a key trade-off between expressiveness and faithfulness. In industrial settings, logs are not merely textual descriptions but structured records used for auditing, debugging, and root-cause analysis. One engineer emphasized:

“We need to trace every alert back to the original log and verify what actually happened.”

When parsing outputs alter or compress the original semantics, engineers may lose confidence in the system, even if detection performance appears strong. This explains why, despite the advantages of LLM-based parsing, practitioners consistently stressed the importance of preserving fidelity and consistency. In other words, the most accurate model is not necessarily the most useful one unless its outputs remain trustworthy and verifiable in real operational settings.

Taken together, the interview findings reinforce the central conclusions of RQ1–RQ3 while also clarifying their practical meaning. Compared with LogHub, industrial logs are perceived as more heterogeneous, less standardized, more context-dependent, and more tightly coupled to evolving system behavior. Practitioners confirm that parsing errors are common and consequential, that external public logs can be beneficial but only under careful curation, and that robustness and traceability remain core requirements for deployment. These findings suggest that the gap between benchmark-oriented log analysis and industrial log analysis is not only quantitative but also structural and operational, further supporting the need for pipeline-level designs that jointly consider parsing quality, model robustness, and data compatibility.

Answer to RQ4: Practitioners consistently viewed industrial logs as more heterogeneous, context-dependent, and operationally sensitive than public benchmarks, which helps explain the stronger impact of parsing errors observed in our experiments. They also stressed that successful industrial log analysis requires not only effective models, but also robust parsing, compatible training data, and outputs that remain faithful and trustworthy in practice.

6. Discussion

6.1. Principal findings

The findings of RQ1–RQ4 present a coherent picture of log analysis as a tightly coupled pipeline rather than a sequence of independent

steps. First, RQ1 shows that log parsing errors are neither rare nor arbitrary, but instead form structured and recurring failure modes across both rule-based and LLM-based parsers. Errors such as template over-merging, over-splitting, parameterization inconsistencies, and unseen-template handling arise from fundamental tensions between abstraction and fidelity in log parsing. Their recurrence across diverse systems suggests that parsing quality cannot be fully understood through intrinsic metrics alone; rather, parsing decisions reshape the event template space in ways that directly affect downstream learning. The identification of an LLM-specific semantic rewriting error further extends this insight by showing that increased model sophistication may introduce new risks related to faithfulness, controllability, and traceability. In practical terms, this means that more expressive parsers are not necessarily safer or more useful unless their outputs remain behaviorally consistent with the original logs.

Building on this taxonomy, RQ2 provides strong empirical evidence that parsing errors propagate systematically to downstream anomaly detection. Correcting expert-identified parsing errors yields consistent improvements in F1-score across models and datasets, confirming that parsing quality can be a first-order determinant of anomaly detection effectiveness when parsing errors affect task-relevant event representations, rather than being a neutral preprocessing detail. This effect is especially pronounced in industrial settings, where heterogeneous formats, evolving structures, and limited training data amplify the consequences of malformed templates and inconsistent parameter extraction. At the same time, the results reveal clear model-dependent differences: models with stronger contextual or hybrid language modeling capabilities are generally more robust to parsing noise, whereas approaches that depend more heavily on stable templates, clustering quality, or limited supervision are more easily disrupted by upstream distortions. This finding should be interpreted in light of prior evidence (Khan et al., 2024; Le and Zhang, 2021; Fu et al., 2023) that overall parsing accuracy is not necessarily correlated with anomaly detection accuracy. Our results likewise do not imply that higher parser-level accuracy always leads to better detection performance. Instead, they show that correcting specific parsing errors can improve downstream detection when those errors distort task-relevant event representations, variable fields, or sequence structures. Such distortions may also affect the distinguishability between normal and anomalous events, providing a plausible connection to the findings of Khan et al. (2024). However, because distinguishability itself is not directly measured in our current experiments, this connection should be regarded as an interpretation of the observed performance changes rather than a directly validated mechanism.

RQ3 extends the analysis from parsing quality to data availability and transferability. The results show that curated external public logs can improve anomaly detection on data-scarce industrial datasets by expanding behavioral coverage and reducing overfitting, but these benefits are uneven and highly model dependent. In particular, stronger models such as SemiRALD exhibit smaller gains, suggesting that better representation learning reduces the marginal value of auxiliary data, whereas other models benefit more from increased sample diversity. At the same time, the presence of mild negative transfer in some settings demonstrates that more data is not inherently better. When external logs introduce semantically or structurally mismatched patterns, they may broaden the learned notion of normality and blur the anomaly boundary of the target system. These findings indicate that cross-dataset training should not be viewed as a simple scaling strategy, but as a compatibility-sensitive intervention that depends on deliberate sampling, domain alignment, and awareness of model inductive biases.

Finally, RQ4 provides an important practitioner-oriented interpretation of the quantitative findings. Engineers consistently described industrial logs as more heterogeneous, context-dependent, and operationally sensitive than public benchmark logs, which helps explain why parsing errors are more frequent and more harmful in industrial environments. They also confirmed that parsing failures are not

merely technical imperfections, but can directly affect monitoring, alerting, debugging, and root-cause analysis. In addition, practitioners viewed public logs as useful but only partially transferable, reinforcing the mixed results observed in RQ3. Perhaps most importantly, their feedback adds a practical criterion that is not fully captured by benchmark metrics alone: in industrial settings, usefulness depends not only on predictive performance, but also on robustness, faithfulness, and traceability. Taken together, these observations strengthen the central conclusion of this study: reliable industrial log analysis cannot be achieved by optimizing parsing, detection, or data augmentation in isolation, but instead requires coordinated, system-level design that jointly considers parsing fidelity, data compatibility, model robustness, and practitioner trust.

6.2. Implications

This study has several implications for future research on log analysis. First, it suggests that log parsing and anomaly detection should no longer be evaluated as loosely connected modules, but as tightly coupled stages in an end-to-end pipeline. Parsing errors reshape the event template space, alter sequence regularities, and ultimately affect downstream learning objectives; accordingly, future work should move beyond intrinsic parsing metrics and develop evaluation protocols that explicitly measure downstream impact. Second, the mixed results of cross-dataset augmentation indicate that transfer learning in log analysis should be studied through the lens of domain compatibility, semantic alignment, and model inductive bias. More broadly, our findings motivate parsing-aware anomaly detection methods, compatibility-sensitive augmentation strategies, and benchmarks that better reflect modern industrial logging conditions.

The oracle-style correction used in RQ2 also has important practical implications. The observed improvements indicate the amount of detection performance that may be recoverable if harmful parsing errors can be identified and corrected reliably. However, practitioners should not expect fully automated systems to achieve the same gains without additional validation. In realistic deployments, a more feasible path is likely to involve semi-automated parser auditing, error-type-specific diagnostics, active learning over suspicious templates, and human-in-the-loop correction for high-impact templates or safety-critical log streams. Therefore, the main practical value of RQ2 is not to propose manual correction as a deployable solution, but to identify which parsing failures should be prioritized when designing automated or semi-automated parser repair tools.

The findings also carry practical implications for industrial deployment. Improving anomaly detection in real systems is not simply a matter of choosing a more advanced model or adding more data. Instead, robust deployment requires coordinated attention to parsing fidelity, data curation, model robustness, and practitioner trust. In particular, practitioners should prioritize stabilizing parsing outputs, especially for recurring error modes such as template over-merging, parameterization inconsistencies, and unseen-template handling, before introducing increasingly complex detectors. Similarly, external public logs can be valuable for alleviating data scarcity, but only when they are carefully filtered for structural and semantic compatibility with the target system. Finally, the interview results highlight that operational usefulness depends not only on accuracy, but also on whether model outputs remain faithful, traceable, and actionable in debugging and incident response workflows.

7. Threats to validity

7.1. Internal validity

One threat to internal validity stems from the expert-driven identification and correction of log parsing errors. Although we adopted a structured review process involving multiple experienced engineers,

some degree of subjectivity is unavoidable in determining whether a parsing output is erroneous and how it should be corrected. Different experts or alternative correction guidelines might therefore produce somewhat different classifications or revised templates.

A related limitation concerns the independence between error identification, correction, and downstream evaluation in RQ2. The expert-guided corrections were derived from the same error-identification process used to establish the parsing-error taxonomy, and these two stages were therefore not fully independent. We controlled the downstream comparison by keeping sequence construction, window boundaries, data splits, model evaluation procedures, and performance metrics unchanged between the original and corrected conditions. Nevertheless, these controls do not eliminate the potential circularity introduced by expert-guided correction itself. In particular, the rank-biserial correlation of $r = 1.00$, which results from positive F1-score differences across all 100 dataset-detector pairs, should be interpreted within this oracle-style correction setting rather than as independent evidence that any correction procedure would produce uniformly positive effects. Future work could further strengthen the generalizability of these findings by involving an independent expert group in the identification and correction of parsing errors, or by evaluating automated correction mechanisms developed independently of the original error-identification process.

7.2. External validity

The generalizability of our findings may be limited by the datasets and systems considered in this study. Although we evaluate 16 public datasets and four industrial datasets from different domains, the industrial logs still originate from a limited number of organizations and system types. Other environments with different logging practices, software architectures, operational constraints, or regulatory requirements may exhibit different parsing error patterns and anomaly detection behaviors. Likewise, while we include several representative log parsers and anomaly detection models, future architectures or alternative learning paradigms may respond differently to parsing errors and cross-dataset augmentation.

7.3. Construct validity

A threat to conclusion validity concerns the stability of the observed empirical trends. Although our results are consistent across multiple datasets, parsers, and models, some findings — especially those related to cross-dataset augmentation and mild negative transfer — may depend on the specific auxiliary data selection strategy and the particular industrial systems examined. In addition, the number of practitioner interviews is necessarily limited, which means the qualitative findings should be interpreted as complementary evidence rather than statistically representative claims. Nevertheless, the convergence between quantitative results and practitioner feedback increases confidence in the robustness of our main conclusions.

A further threat to construct validity is that the corrected parsing setting in RQ2 represents an oracle-style upper bound rather than a realistic automated intervention. Because expert correction was manually applied to both training and test data, the reported gains may overestimate what can be achieved by automated or semi-automated parser repair in practice. Therefore, the results should be interpreted as the maximum recoverable performance loss attributable to the identified parsing errors under our protocol. Future work should investigate how much of this upper-bound improvement can be recovered by scalable correction methods, such as template-level auditing, active learning, confidence-based parser repair, or human-in-the-loop validation.

Another limitation concerns our interpretation of the relationship between parsing-error correction and template distinguishability. Although several identified error types can plausibly reduce the separation between normal and anomalous event representations, our experiments evaluate their downstream effects primarily through anomaly

detection performance and do not directly quantify template-level distinguishability before and after correction. Therefore, the observed F1-score improvements should not be interpreted as direct evidence that correction necessarily increases distinguishability in the sense examined by Khan et al. (2024). Future work could explicitly measure changes in normal-anomalous template separability for individual error types to further validate this mechanism.

8. Conclusion

This paper presents a comprehensive empirical study of log analysis as an end-to-end pipeline, examining how log parsing errors, model characteristics, data availability, and practitioner constraints jointly shape downstream log anomaly detection. Across 16 public datasets and four industrial datasets, we identify six recurring log parsing error types and show that these errors propagate systematically to anomaly detection performance. Our controlled correction experiments demonstrate that improving parsing quality leads to statistically reliable overall gains in detection effectiveness, with particularly strong benefits in industrial settings where logs are more heterogeneous and data is limited.

We further show that cross-dataset data augmentation can improve detection performance, but its effectiveness is highly model dependent and may introduce negative transfer when external logs are only superficially similar to the target system. Finally, our practitioner interview study confirms that industrial logs differ fundamentally from public benchmarks in terms of heterogeneity, context dependence, and operational requirements, and highlights the practical importance of robustness, faithfulness, and traceability. Taken together, these findings suggest that reliable industrial log analysis cannot be achieved by optimizing parsing, detection, or data augmentation in isolation, but instead requires coordinated system-level design that jointly considers parsing fidelity, data compatibility, model robustness, and practitioner trust.

CRedit authorship contribution statement

Yicheng Sun: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Jacky Wai Keung:** Visualization, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis. **Hi Kuen Yu:** Validation, Software, Resources, Project administration. **Yishu Li:** Visualization, Project administration, Data curation. **Xiaoxue Ma:** Visualization, Investigation, Funding acquisition, Data curation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was substantially supported by grants from the Research Grant Council (RGC) of the Hong Kong Special Administrative Region, China (Project Reference No. UGC/FDS16/E27/25), and partially supported by the Hong Kong Metropolitan University R&D Fund (Grant No. RD/2025/1.24) and the Research Funds of City University of Hong Kong (Grant Nos. 9229109, 9220103, and 9229029). The authors gratefully acknowledge the financial and academic support provided by these institutions.

Data availability

Data will be made available on request.

References

- Akhtar, Siraaj, Khan, Saad, Parkinson, Simon, 2025. LLM-based event log analysis techniques: A survey. arXiv preprint arXiv:2502.00677.
- Chai, Xiaolin, Feng, Xin, Lou, Zhaoyang, Sun, Yan, Guizani, Mohsen, 2026. RELog: Robust and efficient anomaly detection based on complete logs for large-scale systems. *IEEE Trans. Comput.*
- De la Cruz Cabello, Miguel, Sales, Tiago Prince, Machado, Marcos R, 2026. Log anomaly detection in AIOps: A real-world implementation using large language models. *Syst. Soft Comput.* 8, 200475.
- Dai, Hetong, Li, Heng, Chen, Che-Shao, Shang, Weiyei, Chen, Tse-Hsun, 2020. Logram: Efficient log parsing using n -gram dictionaries. *IEEE Trans. Softw. Eng.* 48 (3), 879–892.
- Dobrowolski, Wojciech, Nikodem, Maciej, Unold, Olgierd, 2023. Software failure log analysis for engineers. *Electronics* 12 (10), 2260.
- Du, Min, Li, Feifei, 2016. Spell: Streaming parsing of system event logs. In: 2016 IEEE 16th International Conference on Data Mining. ICDM, IEEE, pp. 859–864.
- Du, Min, Li, Feifei, Zheng, Guineng, Srikumar, Vivek, 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security. pp. 1285–1298.
- Fu, Qiang, Lou, Jian-Guang, Wang, Yi, Li, Jiang, 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In: 2009 Ninth IEEE International Conference on Data Mining. IEEE, pp. 149–158.
- Fu, Ying, Yan, Meng, Xu, Zhou, Xia, Xin, Zhang, Xiaohong, Yang, Dan, 2023. An empirical study of the impact of log parsers on the performance of log-based anomaly detection. *Empir. Softw. Eng.* 28 (1), 6.
- Garousi, Vahid, Giray, Gökem, Tüzün, Eray, Catal, Cagatay, Felderer, Michael, 2019. Aligning software engineering education with industrial needs: A meta-analysis. *J. Syst. Softw.* 156, 65–83.
- Guo, Haixuan, Yuan, Shuhan, Wu, Xintao, 2021. Logbert: Log anomaly detection via bert. In: 2021 International Joint Conference on Neural Networks. IJCNN, IEEE, pp. 1–8.
- Gupta, Pranjal, Bhukar, Karan, Kumar, Harshit, Nagar, Seema, Mohapatra, Prateeti, Kar, Debanjana, 2026. Scalable log analysis with LLMs: A case study with IT software support. In: Innovative Applications of Artificial Intelligence Conference.
- He, Shilin, He, Pinjia, Chen, Zhuangbin, Yang, Tianyi, Su, Yuxin, Lyu, Michael R, 2021. A survey on automated log analysis for reliability engineering. *ACM Comput. Surv.* 54 (6), 1–37.
- He, Shilin, Zhu, Jieming, He, Pinjia, Lyu, Michael R, 2016. Experience report: System log analysis for anomaly detection. In: 2016 IEEE 27th International Symposium on Software Reliability Engineering. ISSRE, IEEE, pp. 207–218.
- He, Pinjia, Zhu, Jieming, Zheng, Zibin, Lyu, Michael R, 2017. Drain: An online log parsing approach with fixed depth tree. In: 2017 IEEE International Conference on Web Services. ICWS, IEEE, pp. 33–40.
- Hsiao, Shun-Wen, Tseng, Chih-Chung, 2026. Anomaly detection on system log with language modeling and reconstruction gate. *J. Inf. Sci. Eng.* 42 (1).
- Huang, Junjie, Jiang, Zhihan, Chen, Zhuangbin, Lyu, Michael, 2025. No more labelled examples? An unsupervised log parser with LLMs. *Proc. the ACM Softw. Eng.* 2 (FSE), 2406–2429.
- Huo, Yintong, Su, Yuxin, Lee, Cheryl, Lyu, Michael R, 2023. Semparser: A semantic parser for log analytics. In: 2023 IEEE/ACM 45th International Conference on Software Engineering. ICSE, IEEE, pp. 881–893.
- Jiang, Zhihan, Liu, Jinyang, Chen, Zhuangbin, Li, Yichen, Huang, Junjie, Huo, Yintong, He, Pinjia, Gu, Jiazhen, Lyu, Michael R, 2024. LILAC: Log parsing using LLMs with adaptive parsing cache. *Proc. the ACM Softw. Eng.* 1 (FSE), 137–160.
- Khan, Zanis Ali, Shin, Donghwan, Bianculli, Domenico, Briand, Lionel, 2022. Guidelines for assessing the accuracy of log message template identification techniques. In: Proceedings of the 44th International Conference on Software Engineering. pp. 1095–1106.
- Khan, Zanis Ali, Shin, Donghwan, Bianculli, Domenico, Briand, Lionel C, 2024. Impact of log parsing on deep learning-based anomaly detection. *Empir. Softw. Eng.* 29 (6), 139.
- Landauer, Max, Onder, Sebastian, Skopik, Florian, Wurzenberger, Markus, 2023. Deep learning for anomaly detection in log data: A survey. *Mach. Learn. Appl.* 12, 100470.
- Landauer, Max, Skopik, Florian, Wurzenberger, Markus, 2024. A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques. *Proc. the ACM Softw. Eng.* 1 (FSE), 1354–1375.
- Le, Van-Hoang, Zhang, Hongyu, 2021. Log-based anomaly detection without log parsing. In: 2021 36th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 492–504.
- Le, Van-Hoang, Zhang, Hongyu, 2023. Log parsing with prompt-based few-shot learning. In: 2023 IEEE/ACM 45th International Conference on Software Engineering. ICSE, IEEE, pp. 2438–2449.
- Li, Zhenhao, Luo, Chuan, Chen, Tse-Hsun, Shang, Weiyei, He, Shilin, Lin, Qingwei, Zhang, Dongmei, 2023. Did we miss something important? Studying and exploring variable-aware log abstraction. In: 2023 IEEE/ACM 45th International Conference on Software Engineering. ICSE, IEEE, pp. 830–842.

- Lin, Yijun, Chiang, Yao-Yi, 2022. A semi-supervised learning approach for abnormal event prediction on large network operation time-series data. In: 2022 IEEE International Conference on Big Data. Big Data, IEEE, pp. 1024–1033.
- Liu, Yudong, Zhang, Xu, He, Shilin, Zhang, Hongyu, Li, Liqun, Kang, Yu, Xu, Yong, Ma, Minghua, Lin, Qingwei, Dang, Yingnong, et al., 2022. Uniparser: A unified log parser for heterogeneous log data. In: Proceedings of the ACM Web Conference 2022. pp. 1893–1901.
- Lou, Jian-Guang, Fu, Qiang, Yang, Shenqi, Xu, Ye, Li, Jiang, 2010. Mining invariants from console logs for system problem detection. In: 2010 USENIX Annual Technical Conference. USENIX ATC 10.
- Ma, Zeyang, Chen, An Ran, Kim, Dong Jae, Chen, Tse-Hsun Peter, Wang, Shaowei, 2024a. LLMParser: An exploratory study on using large language models for log parsing.
- Ma, Xiaoxue, Keung, Jacky, He, Pinjia, Xiao, Yan, Yu, Xiao, Li, Yishu, 2023. A semi-supervised approach for industrial anomaly detection via self-adaptive clustering. IEEE Trans. Ind. Informatics 20 (2), 1687–1697.
- Ma, Zeyang, Kim, Dong Jae, Chen, Tse-Hsun, 2024b. Librelog: Accurate and efficient unsupervised log parsing using open-source large language models. arXiv preprint arXiv:2408.01585.
- Ma, Xiaoxue, Li, Yishu, Keung, Jacky, Yu, Xiao, Zou, Huiqi, Yang, Zhen, Sarro, Federica, Barr, Earl T, 2025. Practitioners' expectations on log anomaly detection. IEEE Trans. Softw. Eng.
- Ma, Xiaoxue, Zou, Huiqi, He, Pinjia, Keung, Jacky, Li, Yishu, Yu, Xiao, Sarro, Federica, 2024. On the influence of data resampling for deep learning-based log anomaly detection: Insights and recommendations. IEEE Trans. Softw. Eng.
- Nipa, Nadira Anjum, Bouguila, Nizar, Patterson, Zachary, 2025. A comparative study of log-based anomaly detection methods in real-world system logs. In: IoTBDS. pp. 141–152.
- Pang, Yue, Zhang, Min, Liu, Yanli, Li, Xiangbin, Wang, Yidi, Huan, Yahang, Liu, Zhuo, Li, Jin, Wang, Danshi, 2024. Large language model-based optical network log analysis using LLaMA2 with instruction tuning. J. Opt. Commun. Netw. 16 (11), 1116–1132.
- Pirelli, Solal, 2024. Scalable teaching of software engineering theory and practice: An experience report. In: Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training. pp. 286–296.
- Sun, Yicheng, Keung, Jacky, Liao, Yihan, Yu, Hi Kuen, 2025a. Understanding industrial log analysis: A multi-dataset evaluation of parsing and anomaly detection. In: 2025 32nd Asia-Pacific Software Engineering Conference. APSEC, IEEE, pp. 491–502.
- Sun, Yicheng, Keung, Jacky Wai, Ma, Xiaoxue, Liao, Yihan, Mao, Zhenyu, Yu, Hi Kuen, 2026a. Industrial log analysis revisited: A task-oriented evaluation of parsing and anomaly detection under real-world constraints. Inf. Softw. Technol. 108205.
- Sun, Yicheng, Keung, Jacky, Ma, Xiaoxue, Liao, Yihan, Yu, Hi Kuen, Li, Yishu, 2026b. When parsing goes wrong: An empirical study of error propagation and data augmentation in log anomaly detection. In: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering. EASE 2026, IEEE.
- Sun, Yicheng, Keung, Jacky Wai, Yang, Zhen, Liu, Shuo, Liao, Yihan, 2025b. SemiS-MAC: A semi-supervised framework for log anomaly detection with automated hyperparameter tuning. Inf. Softw. Technol. 107869.
- Sun, Yicheng, Keung, Jacky Wai, Yang, Zhen, Liu, Shuo, Yu, Hi Kuen, 2025c. SemiRALD: A semi-supervised hybrid language model for robust anomalous log detection. Inf. Softw. Technol. 107743.
- Sun, Yicheng, Keung, Jacky, Yang, Zhen, Liu, Shuo, Yu, Hi Kuen, 2026c. Improving anomaly detection in software logs through hybrid language modeling and reduced reliance on parser. Autom. Softw. Eng. 33 (1), 12.
- Sun, Yicheng, Keung, Jacky, Yu, Hi Kuen, Liu, Shuo, Liao, Yihan, Zhang, Jingyu, 2025d. Beyond log parsers: A scalable AI-driven framework for efficient log anomaly detection in software engineering. In: 2025 IEEE 49th Annual Computers, Software, and Applications Conference. COMPSAC, IEEE, pp. 1382–1387.
- Sun, Yicheng, Keung, Jacky Wai, Yu, Hi Kuen, Luo, Wenqiang, 2026d. LogMeta: A few-shot model-agnostic meta-learning framework for robust and adaptive log anomaly detection. J. Syst. Softw. 112781.
- Tu, Tongwei, 2025. Log2Learn: Intelligent log analysis for real-time network optimization. In: 2025 4th International Conference on Artificial Intelligence, Internet and Digital Economy. ICAID, IEEE, pp. 162–166.
- Tufek, Alper, Aktas, Mehmet S., 2023. On the provenance extraction techniques from large scale log files. Concurr. Comput.: Pract. Exp. 35 (15), e6559.
- Wang, Jialai, Lu, Juncheng, Yang, Jie, Wang, Junjie, Gao, Zeyu, Zhang, Chao, Liang, Zhenkai, Chang, Ee-Chien, 2025. Improving LLM-based log parsing by learning from errors in reasoning traces. In: 2025 40th IEEE/ACM International Conference on Automated Software Engineering. ASE, IEEE, pp. 726–738.
- Wang, Zumin, Tian, Jiayu, Fang, Hui, Chen, Liming, Qin, Jing, 2022. LightLog: A lightweight temporal convolutional network for log anomaly detection on the edge. Comput. Netw. 203, 108616.
- Wu, Xingfang, Li, Heng, Khomh, Foutse, 2023. On the effectiveness of log representation for log-based anomaly detection. Empir. Softw. Eng. 28 (6), 137.
- Xie, Xiaoda, Jian, Songlei, Huang, Chenlin, Yu, Fengyuan, Deng, Yunjia, 2023. LogRep: Log-based anomaly detection by representing both semantic and numeric information in raw messages. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering. ISSRE, IEEE, pp. 194–206.
- Xu, Junjielong, Yang, Ruichun, Huo, Yintong, Zhang, Chengyu, He, Pinjia, 2024. DivLog: Log parsing with prompt enhanced in-context learning. In: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering. pp. 1–12.
- Yang, Lin, Chen, Junjie, Wang, Zan, Wang, Weijing, Jiang, Jiajun, Dong, Xuyuan, Zhang, Wenbin, 2021. Semi-supervised log-based anomaly detection via probabilistic label estimation. In: 2021 IEEE/ACM 43rd International Conference on Software Engineering. ICSE, IEEE, pp. 1448–1460.
- Yu, Xiao, Liu, Lei, Hu, Xing, Keung, Jacky Wai, Liu, Jin, Xia, Xin, 2024. Fight fire with fire: How much can we trust chatgpt on source code-related tasks? IEEE Trans. Softw. Eng. 50 (12), 3435–3453.
- Yu, Boxi, Yao, Jiayi, Fu, Qiui, Zhong, Zhiqing, Xie, Haotian, Wu, Yaoliang, Ma, Yuchi, He, Pinjia, 2024. Deep learning or classical machine learning? An empirical study on log-based anomaly detection. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering. pp. 1–13.
- Zhang, Jing, Luo, Chao, Shao, Rui, 2026. MTG-GAN: A masked temporal graph generative adversarial network for cross-domain system log anomaly detection. IEEE Trans. Netw. Serv. Manag.
- Zhang, Tianzhu, Qiu, Han, Castellano, Gabriele, Rifai, Myriana, Chen, Chung Shue, Pianese, Fabio, 2023. System log parsing: A survey. IEEE Trans. Knowl. Data Eng. 35 (8), 8596–8614.
- Zhang, Xu, Xu, Yong, Lin, Qingwei, Qiao, Bo, Zhang, Hongyu, Dang, Yingnong, Xie, Chunyu, Yang, Xincheng, Cheng, Qian, Li, Ze, et al., 2019. Robust log-based anomaly detection on unstable log data. In: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. pp. 807–817.
- Zhu, Jieming, He, Shilin, He, Pinjia, Liu, Jinyang, Lyu, Michael R, 2023. Loghub: A large collection of system log datasets for ai-driven log analytics. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering. ISSRE, IEEE, pp. 355–366.
- Zhu, Jieming, He, Shilin, Liu, Jinyang, He, Pinjia, Xie, Qi, Zheng, Zhibin, Lyu, Michael R, 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, pp. 121–130.