Information and Software Technology 198 (2026) 108205

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

# Industrial log analysis revisited: A task-oriented evaluation of parsing and anomaly detection under real-world constraints

Yicheng Sun [a], Jacky Wai Keung [a], Xiaoxue Ma [b,*], Yihan Liao [a], Zhenyu Mao [a], Hi Kuen Yu [a]

[a] Department of Computer Science, City University of Hong Kong, Hong Kong Special Administrative Region of China
[b] Department of Electronic Engineering and Computer Science, Hong Kong Metropolitan University, Hong Kong Special Administrative Region of China

ARTICLE INFO

Dataset link: https://github.com/log-detection/industrial_logs

*Keywords:*
Industrial software
Log analysis
Log parsing
Log anomaly detection
Large language models

ABSTRACT

**Context:** Log analysis is a critical component for monitoring, diagnosis, and risk mitigation in software systems. However, most existing research evaluates log parsing and anomaly detection models on benchmark datasets derived from legacy or open-source systems, which fail to reflect the structural diversity, semantic density, and annotation constraints of real-world industrial logs. In industrial settings, detected anomalies can support early warning, root-cause localization, preventive maintenance, and operational risk control.
**Objective:** This study aims to systematically examine how log parsing accuracy, anomaly detection performance, and annotation availability interact in industrial settings, and to assess whether improvements in parsing quality translate into meaningful downstream gains for anomaly detection.
**Method:** In this work, we conduct a comprehensive empirical evaluation across four datasets, including one benchmark dataset and three large-scale industrial datasets collected from manufacturing, process control, and energy monitoring environments. Using a unified evaluation framework, we assess multiple state-of-the-art parsing and detection models under standardized conditions. We further analyze the impact of annotation scarcity, identify common industrial parsing errors, and perform controlled parsing corrections to isolate their downstream effects on anomaly detection.
**Results:** Our results show that models achieving strong performance on benchmark datasets suffer substantial degradation on industrial logs, particularly under limited supervision. Semi-supervised approaches demonstrate greater robustness in low-label regimes, while continued annotation yields sustained performance improvements. Crucially, we find that improving parsing accuracy benefits anomaly detection only when corrections preserve anomaly-critical semantic cues, whereas many parsing refinements provide little practical gain.
**Conclusion:** These findings highlight fundamental limitations of benchmark-centric evaluation and parser-centric optimization for industrial log analysis. We argue that log parsing should be treated as a task-aware semantic transformation, and that effective industrial monitoring systems must prioritize label efficiency, semantic fidelity, and downstream utility to achieve reliable performance in complex, real-world environments.

## 1. Introduction

Log data has become an essential asset in maintaining the safety, stability, and reliability of industrial software systems [1–3]. Generated automatically during system execution, logs provide valuable records of internal operations, offering insights into system behaviors, failures, and potential risks [4–6]. As software continues to increase in complexity and scale, the role of automated log analysis – especially log anomaly detection – has become increasingly critical for timely risk mitigation and operational resilience [7–9].

In industrial software systems, detected anomalies are not merely evaluation labels, but actionable signals that can support a wide range of downstream operational decisions [10,11]. For example, early anomaly alerts can help maintenance engineers identify incipient equipment faults before they escalate into production interruptions; abnormal event sequences can guide root-cause analysis by narrowing the search space to specific components, sensors, controllers, or communication modules; and recurring anomaly patterns can inform preventive maintenance, alarm prioritization, incident triage, and safety compliance auditing [12,13]. In safety–critical environments such as

* Corresponding author.
*E-mail addresses:* yicsun2-c@my.cityu.edu.hk (Y. Sun), Jacky.Keung@cityu.edu.hk (J.W. Keung), kxma@hkmu.edu.hk (X. Ma), yihanliao4-c@my.cityu.edu.hk (Y. Liao), zhenyumao2-c@my.cityu.edu.hk (Z. Mao), hikuenyu2-c@my.cityu.edu.hk (H.K. Yu).

https://doi.org/10.1016/j.infsof.2026.108205
Received 3 February 2026; Received in revised form 23 May 2026; Accepted 23 May 2026
Available online 30 May 2026

**Table 1**
Comparison of log statistics between LogHub datasets and industrial systems.

| System | Collection period | Avg. log length (tokens) | # Unique templates | Total log count | Source |
|---|---|---|---|---|---|
| BGL | Before 2010 | 197 | 1531 | 2,906,729 | LogHub |
| Thunderbird | Before 2010 | 205 | 2732 | 2,878,984 | LogHub |
| Industrial-1 | 2023–2024 | 342 | 4508 | 8,123,410 | Confidential |
| Industrial-2 | 2024 | 376 | 5121 | 9,504,822 | Confidential |
| Industrial-3 | 2024 | 428 | 6744 | 10,274,915 | Confidential |
| Industrial-4 | 2024–2025 | 393 | 5612 | 7,893,143 | Confidential |
| Industrial-5 | 2024–2025 | 411 | 6039 | 8,681,777 | Confidential |

manufacturing lines, process control systems, and SCADA infrastructures, timely and reliable anomaly detection can therefore reduce downtime, support faster fault recovery, and prevent local software or hardware failures from propagating into larger operational risks.

A significant body of research on log anomaly detection has relied on publicly available datasets, most notably those hosted on LogHub [14]. While these datasets have served as important benchmarks, they exhibit several limitations [15,16]. One significant limitation of LogHub is that it consists primarily of logs collected from systems over a decade old, many of which follow relatively consistent and uniform logging formats [17–19]. As a result, models that perform well on these datasets may not generalize to contemporary industrial software systems, which are increasingly complex, modularized, and deployed in heterogeneous environments.

To better understand the characteristics and challenges of modern industrial logs, we conducted in-depth, one-on-one interviews with seven software maintenance engineers from large-scale industrial organizations across sectors such as energy, automation, and manufacturing. The engineers reported that log data in modern industrial software systems tends to be significantly more diverse and complex than that of traditional open-source systems. Logs are often verbose, contain proprietary terminologies, and are deeply tied to both hardware and software subsystems. Moreover, logging practices vary widely across components, introducing inconsistencies in format, granularity, and semantic structure.

To quantify these observations, we compared logs from two relatively complex LogHub datasets – BGL and Thunderbird – with five anonymized industrial systems provided by our collaborators. For comparability, we extracted logs from a fixed observation window for each industrial system and computed the corresponding descriptive statistics. As shown in Table 1, BGL and Thunderbird have average log lengths of 197 and 205 tokens, respectively, and 1531 and 2732 unique log templates. In contrast, logs from the industrial systems exhibit substantially greater complexity, with average lengths ranging from 342 to 428 tokens and template counts exceeding 4000 in all cases. These differences highlight the unique structural and semantic challenges inherent in processing industrial logs.

Despite these evident challenges, the majority of existing studies still benchmark log analysis models solely on LogHub datasets [20,21], overlooking the more stringent and heterogeneous demands of industrial software environments. In contrast to cloud or web-based systems, industrial software is characterized by tightly coupled hardware–software interactions, domain-specific error codes, and highly context-dependent execution behaviors. These factors collectively exacerbate the difficulty of anomaly detection, necessitating advanced techniques that can handle both structural diversity and semantic ambiguity with greater robustness.

A further limitation of current research lies in the widespread treatment of log parsing and anomaly detection as largely independent tasks [22,23]. Parsing is often evaluated in isolation using parser-centric metrics, while anomaly detection models are assessed assuming a fixed and error-free parsing output [24,25]. This separation overlooks the fact that parsing acts as a semantic transformation that fundamentally shapes the input space of downstream detectors. As a result, improvements in parsing accuracy are frequently assumed to be universally beneficial, without examining whether they preserve or distort anomaly-critical information. Such a decoupled perspective obscures the causal relationship between parsing errors and detection failures, and limits our understanding of how end-to-end log analysis pipelines behave in realistic industrial settings. These practical uses make anomaly detection highly dependent on whether the upstream parsing process preserves operationally meaningful information, because distorted templates may hide the very signals needed for diagnosis, prioritization, and intervention.

To address this research gap, we collaborated with industrial partners to collect, preprocess, and anonymize logs from three large-scale industrial software systems. These logs were curated into benchmark-quality datasets while preserving the structural diversity, semantic richness, and operational complexity of real-world industrial environments. Building on these datasets, we conducted a comprehensive set of experiments that jointly examine log parsing and anomaly detection under realistic industrial constraints. Specifically, we evaluated multiple state-of-the-art parsers and detectors across three industrial datasets and one representative LogHub dataset, analyzed performance under standardized and label-scarce settings, characterized common industrial parsing errors, and performed controlled parsing corrections with expert review to assess their downstream impact on anomaly detection. Our primary contributions are summarized as follows:

- We construct and analyze four log datasets, including three large-scale real-world industrial datasets from manufacturing, process control, and energy monitoring systems, together with one representative LogHub benchmark.
- We conduct a unified empirical evaluation of log parsing and anomaly detection models, comparing rule-based, supervised, semi-supervised, and LLM-based approaches under standardized experimental settings.
- We design and perform controlled experiments to study annotation scarcity, systematically varying label budgets to assess model behavior under limited supervision in industrial environments.
- We carry out task-oriented analysis of parsing errors, correcting common industrial log parsing error types and re-evaluating downstream anomaly detection with expert review.

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 dataset construction, log parsing procedures, and the anomaly detection models considered 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–RQ5 and discusses their research and practical implications. Section 7 outlines potential threats to validity. Finally, Section 8 concludes the paper.

This paper is a substantial extension of our earlier APSEC 2025 paper [10]. Compared with the conference version, the journal manuscript broadens both the empirical scope and the analytical depth of the study. Specifically, we extend the parsing evaluation with a new efficiency analysis on the full datasets, add controlled label-budget experiments to examine annotation scarcity in industrial settings, and introduce a new task-oriented study of how correcting common parsing

errors affects downstream anomaly detection performance, including both recovery-based and global F1-based evaluation. We also substantially revise the related work, discussion, and methodological descriptions, and add representative industrial log entries and reproducibility materials.

## 2. Background and related work

### 2.1. Log collection

Logs are a primary source of runtime evidence for monitoring, diagnosing, and auditing software-intensive systems [1,3]. In industrial environments, however, log collection is substantially more challenging than in open-source or academic settings. Industrial logs are generated by heterogeneous software–hardware ecosystems, often under proprietary architectures, real-time constraints, and domain-specific operational requirements. As a result, they tend to be longer, semantically denser, and structurally more diverse than the logs commonly used in public benchmarks [26,27]. In practice, industrial logs often contain intertwined control states, sensor readings, operator actions, protocol traces, and system-specific error semantics. These properties make them not only harder to parse, but also more likely to expose weaknesses in downstream anomaly detection pipelines when parsing errors occur.

Existing public repositories such as LogHub [14] have played an important role in advancing log analysis research, but they are still dominated by logs from legacy server systems, distributed platforms, and open-source infrastructures. Recent empirical studies and benchmark-based evaluations have increasingly suggested that datasets such as LogHub, while highly valuable for methodological comparison, are becoming less representative of contemporary software systems [15,16]. Most LogHub datasets originate from relatively stable server-side or distributed computing environments and therefore capture only part of the logging characteristics of modern industrial software [17,18].

By contrast, newer industrial systems often produce logs under tighter hardware–software coupling, richer operational semantics, stronger context dependence, and more heterogeneous formatting conventions [19]. Consequently, conclusions derived solely from benchmark-style datasets may not fully generalize to real-world industrial settings. These limitations highlight the importance of evaluating log parsing and anomaly detection on industrial log data that better reflect the structural complexity and operational variability of contemporary software systems. This external-validity concern is particularly important for empirical studies that aim to understand how parsing quality affects downstream anomaly detection in real deployment-oriented settings.

### 2.2. Log analysis task

Log analysis generally involves two closely related tasks: log parsing and log anomaly detection [15]. Log parsing converts raw log messages into structured event templates and parameters, thereby making heterogeneous system logs amenable to downstream analysis [28]. Building on structured logs or log sequences, anomaly detection aims to identify runtime behaviors that deviate from expected system operation [29]. Together, these two tasks form the core sequential pipeline of automated log analysis. In this pipeline, the dependency is primarily upstream-to-downstream: parsing outputs shape the input representation used by anomaly detection, and parsing errors may distort event structures or anomaly-relevant semantic cues [30,31], thereby reducing the reliability of downstream anomaly detection.

Accurate log parsing remains challenging because modern logs often contain diverse parameter patterns, nested structures, and variable textual forms [9,32]. Prior work [33–36] has shown that parsing quality strongly affects downstream tasks such as anomaly detection and root cause analysis. Existing parsers can be broadly divided into rule-based methods, such as Drain [37] and Spell [38], and more recent LLM-based methods, such as LogPPT [35], DivLog [9], and LILAC [15]. While LLM-based parsers have shown strong performance on complex textual structures, their computational cost and deployment overhead still limit practical adoption in many industrial settings.

For anomaly detection, existing approaches [39–43] range from sequence-based and semantic-based supervised models to semi-supervised and self-supervised methods. Representative examples include DeepLog [44], LogRobust [40], PLELog [20,45], LogBERT [21], and SemiRALD [39]. These methods have substantially advanced log-based failure detection, yet most evaluations still rely heavily on public datasets, leaving open the question of how robust such pipelines remain when confronted with the structural and semantic complexity of industrial logs.

### 2.3. Empirical studies on parsing–detection interaction

Beyond standalone parser or detector comparisons, a growing line of work has begun to examine the interaction between log parsing and downstream anomaly detection [30,31]. Fu et al. [30] conducted one of the earliest systematic empirical studies on how log parsers affect log-based anomaly detection performance. Using multiple state-of-the-art parsers and anomaly detection methods on public datasets, they showed that higher parsing accuracy does not necessarily lead to better downstream detection performance, and that the impact of parsing varies across datasets and detection models. Khan et al. [31] extended this line of inquiry through a larger-scale empirical study covering 13 log parsing techniques and seven anomaly detection methods on three publicly available datasets. Their results further showed that log parsing accuracy is not strongly correlated with anomaly detection accuracy and highlighted *distinguishability* in parsing results, rather than parsing accuracy alone, as a more important property for effective anomaly detection.

These studies provide an important foundation, but they leave several questions insufficiently explored. First, their empirical evidence is still centered primarily on public datasets, which do not adequately capture the heterogeneity, semantic density, and context dependence of industrial logs. As a result, it remains unclear whether conclusions derived from benchmark-style datasets generalize well to industrial software systems. Second, prior work mainly examines the parsing–detection relationship at an aggregate level, focusing on overall parser comparison and end-to-end performance differences, while offering limited insight into how different categories of parsing errors propagate through anomaly detection pipelines in practice. Third, existing studies pay relatively little attention to software engineering constraints that are especially salient in industry, such as operational variability, limited labeled data, and cross-dataset mismatch between public and industrial environments.

From a task-oriented industrial perspective, these gaps motivate a broader evaluation of how parsing errors and data conditions jointly shape downstream anomaly detection in real-world settings. In particular, this study combines public and industrial datasets to analyze: (1) the characteristics of common parsing errors in industrial logs, (2) how such errors affect downstream anomaly detection across multiple models, (3) whether public benchmark data can effectively support industrial detection tasks, and (4) how performance changes under different levels of industrial label availability. This broader design moves beyond parser comparison alone toward a more deployment-relevant understanding of log analysis in industrial software systems.

### 2.4. Research gap

Although prior empirical studies have established that log parsing and downstream anomaly detection are related, important gaps remain in current understanding. Existing evidence is still derived primarily from publicly available benchmark datasets, which do not

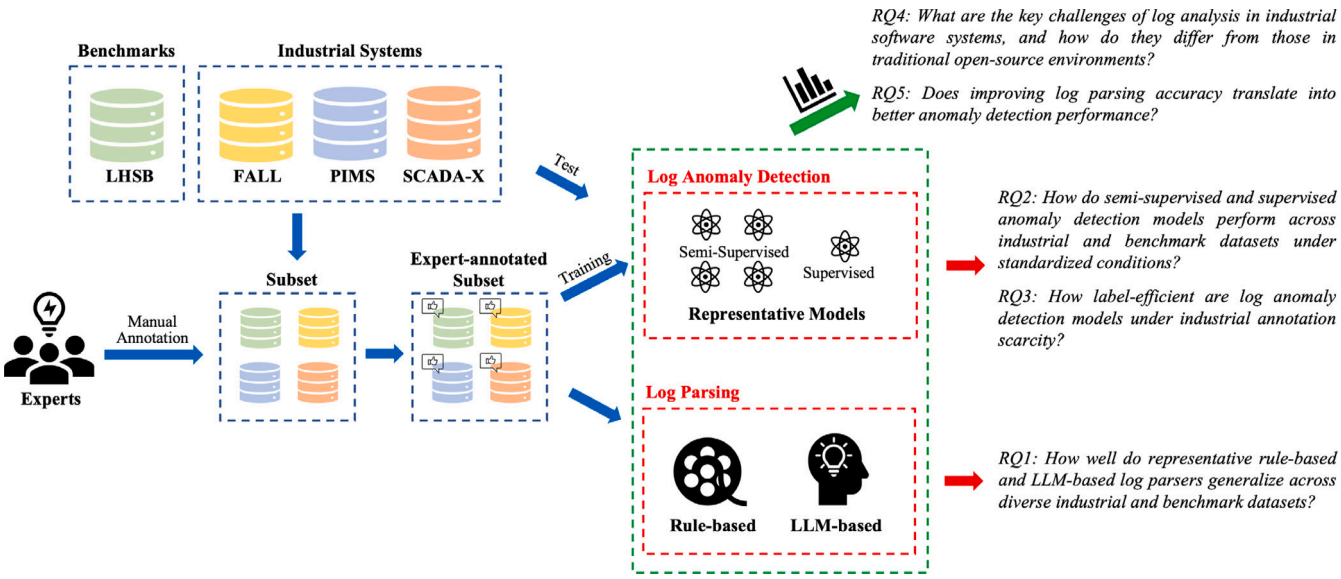


**Fig. 1.** The workflow of steps for our study.

adequately reflect the structural heterogeneity, semantic density, and context dependence of industrial logs. As a result, it remains unclear whether conclusions about parsing–detection interaction drawn from benchmark-style data generalize to industrial software systems. In addition, prior work has mostly examined this relationship at an aggregate level, emphasizing overall parser comparison or end-to-end performance variation, while offering limited insight into how specific categories of parsing errors affect downstream anomaly detection in practice.

A further gap concerns the practical constraints under which industrial log analysis is deployed. Real-world industrial environments often involve limited labeled data, evolving operational conditions, and substantial mismatch between public benchmarks and proprietary production systems. These factors are rarely incorporated into existing empirical evaluations, yet they directly shape the robustness and usefulness of anomaly detection pipelines in practice. Taken together, the literature still lacks a task-oriented understanding of how parsing errors, industrial data characteristics, and label availability jointly influence downstream anomaly detection under realistic software engineering constraints.

## 3. Methodology

### 3.1. Overview

This section outlines the end-to-end methodological pipeline used in this study, as illustrated in Fig. 1, which depicts the complete workflow from dataset construction and log parsing to anomaly detection and expert validation. The workflow begins with the construction of four representative log datasets, including one benchmark dataset and three real-world industrial datasets. Raw logs are collected, anonymized, and preprocessed, after which structurally representative sub-datasets are curated to support both log parsing evaluation and downstream anomaly detection under consistent conditions. These sub-datasets serve as the common input for all subsequent experiments.

Following dataset preparation, we evaluate log parsing using both rule-based and LLM-based parsers, producing structured templates for each dataset. The parsed logs are then fed into supervised and semi-supervised anomaly detection models, all trained and evaluated under a unified protocol with controlled label availability. To ensure experimental reliability, anomaly labels and parsing ground truths are validated through a multi-stage expert review process. This stepwise pipeline – dataset construction, parsing, detection, and expert validation – enables controlled analysis of how parsing behavior and annotation scarcity influence downstream anomaly detection performance in industrial software systems.

### 3.2. Scenarios and dataset construction

To ensure a realistic and comprehensive evaluation of log parsing and anomaly detection models in industrial software environments, we constructed four datasets that reflect diverse system architectures, deployment contexts, and logging practices. These datasets cover both well-established benchmarks and real-world industrial systems from multiple sectors, enabling the assessment of model generalizability across traditional and modern log sources. In particular, the industrial datasets were collected in collaboration with domain experts and capture a wide range of operational characteristics, such as hardware-software interaction, dynamic logging templates, and domain-specific terminologies. This diversity is critical for understanding how existing techniques perform under the complexities of industrial log data.

- **LHSB**: A curated subset of log data selected from the LogHub platform [14], widely recognized as a benchmark for log analysis research. This dataset serves as a controlled baseline for evaluating model performance under standard conditions.
- **FALL**: Derived from a low-resource industrial assembly line system, this dataset captures logging behaviors in high-precision manufacturing environments. It is characterized by relatively sparse log frequency, irregular message formatting, and high variability—conditions common in embedded or cost-constrained deployments.
- **PIMS**: Collected from a modern process industry management system in the chemical sector, this dataset contains high-volume logs with deeply nested event sequences and frequent interactions between software modules and physical sensors. These logs exhibit rich temporal dynamics and highly domain-specific terminologies.
- **SCADA-X**: Obtained from a supervisory control and data acquisition (SCADA) system used in energy infrastructure. This dataset includes logs from distributed control nodes and illustrates the complexity of multi-source, real-time system monitoring in safety–critical environments. Logs are heterogeneous in structure and often contain timestamp synchronization issues and context-sensitive event patterns.

**Table 2**
Summary of constructed datasets.

| Dataset | Abbr. | Type | Total logs | Event template | Sub. logs | Anom. | Avg. Len (Chars) |
|---|---|---|---|---|---|---|---|
| LogHub Subset Benchmark | **LHSB** | Benchmark | **1,437,705** (6.87% anom.) | 2472 | 9040 | 964 (10.7%) | 172.4 |
| Factory Assembly Line Logs | **FALL** | Industrial | **32,615** (9.56% anom.) | 2290 | 7190 | 992 (13.8%) | 227.5 |
| Process Industry Management System Logs | **PIMS** | Industrial | **871,243** (9.93% anom.) | 4508 | 14,382 | 1249 (8.7%) | 312.6 |
| Supervisory Control and Data Acquisition System Logs | **SCADA-X** | Industrial | **902,377** (12.37% anom.) | 6039 | 15,110 | 1013 (6.7%) | 351.1 |

Table 2 summarizes the key characteristics of each dataset, including log volume, average sequence length, template diversity, and domain application. For each dataset, we also constructed a structurally representative subset, which serves two primary purposes: (1) facilitating efficient log parsing by reducing redundant or low-information entries, and (2) providing a unified and balanced training set for subsequent anomaly detection models. These subsets are curated to preserve the structural and semantic diversity of the full datasets while enabling fair and consistent evaluation across different log analysis tasks.

To further improve transparency and provide concrete evidence of the industrial log characteristics analyzed in this study, Table 3 presents representative anonymized log entries from the three industrial datasets. For each dataset, we include multiple normal and anomalous examples covering different log types, such as motion control, operator actions, sensor monitoring, protocol communication, and alarm escalation. These entries were selected to preserve key structural and semantic properties of the original logs while avoiding disclosure of sensitive operational details.

As shown in Table 3, the three industrial datasets differ not only in application context but also in message organization, field composition, and anomaly expression. Compared with benchmark-style logs, these industrial logs are typically longer, contain richer parameter structures, and more tightly couple software events with physical operations. Moreover, several anomalous cases differ from normal logs only in subtle but operationally critical details, such as sensor drift, checksum mismatch, repeated retries, or scheduling conflict. These characteristics help explain why both parsing and anomaly detection are substantially more challenging in industrial environments than benchmark-based evaluation alone would suggest.

#### 3.2.1. LHSB dataset

For the LHSB dataset, our objective was to construct a raw log corpus that accurately reflects the structural complexity and event distribution characteristic of real-world system operations. While the LogHub platform provides 2000 manually curated log entries per dataset, these subsets often exclude rare anomalies, particularly those occurring within short intervals or under limited conditions. To overcome this limitation, we selected five LogHub [14] datasets – HDFS [46], Hadoop [47], OpenStack [44], BGL [48], and Thunderbird [48] – all of which include full log traces accompanied by ground-truth anomaly labels. To facilitate anomaly detection and log parsing evaluation under realistic conditions, we constructed a representative anomaly-centric subset through a two-stage strategy: (1) anomaly interval-based extraction and (2) dataset integration with a controlled anomaly ratio.

In the first stage, we extracted all log entries that occurred within the labeled anomalous intervals provided in the LogHub benchmark. For each dataset, the associated metadata files were parsed to identify the start and end timestamps of each abnormal window. We selected all log entries that occurred within a 90-second window before and after each annotated anomaly event. All log lines falling within these intervals were retained in their original format, ensuring comprehensive coverage of both frequent and infrequent anomalies that may be underrepresented in the original benchmark subsets. This procedure resulted in a total of 1,437,705 raw log entries, which serve as the test set for anomaly detection. Furthermore, we separately extracted the specific anomalous entries within these intervals to be used in the second stage.

In the second stage, we constructed a representative dataset by combining previously extracted anomaly logs with 10,000 randomly sampled entries from five selected LogHub benchmark subsets. To reflect the class imbalance typical in real-world environments, we reduced redundant normal logs and oversampled rare anomalies, resulting in an approximate 9:1 ratio of normal to anomalous entries. Due to missing anomaly labels, the remaining LogHub datasets were excluded. The final dataset contains 6940 raw logs, selected without any template-based parsing or post-processing. This subset serves both as a challenging benchmark for log parsers and a unified training set for downstream anomaly detection models.

#### 3.2.2. FALL: Low-resource industrial logs from assembly line systems

The FALL dataset represents a class of low-resource industrial environments that are often underrepresented in mainstream log analysis research. In such systems, log volumes are relatively limited compared to large-scale cloud infrastructure, yet the structural and semantic complexity of the logs is significantly higher due to tight integration between hardware components and real-time control mechanisms. FALL was collected from a real-world robotic assembly line operating in a high-precision manufacturing setting. The system includes multiple programmable logic controllers (PLCs), robotic arms, sensor arrays, and embedded software modules. These components collaboratively execute coordinated tasks such as material transport, positioning, and quality inspection. Logs are generated from both software control layers and physical hardware events, encompassing robotic joint positions, motion feedback, execution delays, fault alerts, and sensor threshold violations.

Over a continuous 20-day production period in October 2024, we collected a total of 32,615 raw log entries. Due to the inherent redundancy in system polling and status reports, we applied frequency-aware filtering to remove repetitive low-entropy entries, resulting in a curated subset of 7190 log messages. These retained entries exhibit high diversity in structure, including unstructured textual errors, parameterized control signals, and hybrid-format sensor streams. Such variability poses significant challenges for conventional log parsers and anomaly detection models, particularly those trained on more uniform datasets like LogHub.

Importantly, FALL serves as a representative dataset for evaluating log analysis techniques in resource-constrained industrial deployments, such as early-stage automation systems, edge computing nodes, or embedded industrial controllers. Its inclusion allows us to explore how current approaches generalize beyond high-volume cloud logs to realistic, small-scale, heterogeneous industrial settings.

#### 3.2.3. PIMS: Logs from process industry management systems

The PIMS dataset was collected from a large-scale industrial automation platform used in the chemical manufacturing sector. These systems typically operate continuously and are responsible for orchestrating complex process flows involving temperature regulation, chemical mixing, and pressure control, often under strict safety and quality constraints. The PIMS environment consists of a distributed architecture integrating industrial control software, batch process schedulers, human–machine interface (HMI) modules, and field-level sensor networks. Logs are generated from various layers, including control

**Table 3**
Representative anonymized log entries from the three industrial datasets. Each dataset includes both normal and anomalous logs and reflects distinct formatting conventions, operational contexts, and anomaly patterns in real-world industrial software systems.

| Dataset | Label | Log type | Representative log sample |
|---|---|---|---|
| FALL | Normal | Motion control | *PLC-A7\|CELL=ASM-03\|ROBOT=R2\|STEP=PickAndPlace\|joint_temp=41.2 C\|arm_pos=(0.22,0.98,1.14)\|vacuum=ON \|grip_force=18.4N\|part_id=B84219\|cycle_status=completed\|latency=34 ms* |
| | Normal | Vision inspection | *EDGE-CAM2::station=vision_inspect::batch=842::frame=28841::surface_score=0.993::defect_count=0:: lighting=stable::trigger=auto::result=PASS::forward_to=packing_queue* |
| | Normal | Conveyor status | *conv_ctl@line04 module=belt_sync zone=Z2 speed=0.84 m/s load=12.8 kg sensor_A=clear sensor_B=clear gate=OPEN queue=normal dispatch=ready note=no timing drift detected* |
| | Normal | Operator command | *HMI-TERM-03 cmd=resume_cell user=op_shift_b target=ASM-03 prereq_check=passed safety_door=locked est_cycle=42 s downstream=packer-2 status=accepted audit_trail=written* |
| | Anomalous | Sensor deviation | *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 operator_override=false action=halt_cell reason=torque deviation exceeds limit* |
| | Anomalous | Scheduling conflict | *PLC-B2\|CELL=ASM-05\|task=drill_mount_301\|preempted_by=task_299\|fixture=station-12\|expected_window=07:58:00-07:58:20\|actual_start=missed\|retry=5\|status=true\|alarm=raised\|reason=scheduling conflict under peak batch load* |
| | Anomalous | Fault recovery misfire | *servo_node:333:error=trigger caught and handled internally; controller=joint_driver_7; fault_class=overcurrent; auto_recovery=invoked; settle_time=812ms; final_state=unstable; downstream_signal=delayed* |
| | Anomalous | Request path failure | *MES-GW/request=GET /machine_status?line=3&batch=842&station=vision_inspect_02/result=502 Gateway Timeout/retry=3/cache_bypass=true/upstream=cell-monitor/impact=inspection queue blocked* |
| PIMS | Normal | Process state update | *PROC\|unit=Reactor-2\|phase=mixing\|agitator_rpm=180\|temp=82.6 C\|pressure=3.42bar\|feed_valve=V12: OPEN\|cooling_loop=stable\|batch_id=CHM-241108-07\|interlock=clear\|status=within_control_limits* |
| | Normal | Operator audit | *HMI-AUDIT operator=OP_17 action=confirm_recipe recipe=PX-44 target_ph=6.80 target_temp=78.0 C workstation=blend-console-02 approval=granted downstream_scheduler=updated note=manual review completed without deviation* |
| | Normal | Flow regulation | *PIPEMON::segment=distill-A::flow=18.1L/min::pressure=5.02 bar::valve_group=VG-04::controller=PID-04::setpoint=18.0::delta=0.1::alarm_state=clear::checksum=OK::replication=historian_committed* |
| | Normal | Safety loop check | *SIF-CHK unit=neutralization-3 loop=shutdown_guard_B sensor_state=nominal relay_path=verified override_switch=OFF audit=passed by=svc_robot_2 response_time=46ms maintenance_window=none* |
| | Anomalous | Integrity failure | *ALARM::area=distillation::controller=PID-04::output_stream=R2_buffer::checksum=FAILED::expected? =7F3A9C::received=7F3A1C::flow=18.2 L/min::pressure=5.08 bar::mode=AUTO::safety_override=engaged:: event=output mismatch detected* |
| | Anomalous | Numerical drift | *reactor-log@ vessel=MX-09 temp=91.4 C target=88.0C delta=3.4C ph=6.52 target_ph=6.80 feed_ratio=1.28 stability=degrading warning=thermal rise sustained for 240 s action=request cooling escalation* |
| | Anomalous | Retry overlooked | *BATCH-SCHED job=blend_sequence_884 subsystem=feed_scheduler retries=5 status=true queue=backlog_high assigned_node=sched-03 commit=deferred alarm_text=dispatch acknowledged but execution not confirmed* |
| | Anomalous | Authorization failure | *AUTH: user=alice, src=192.168.0.15, method=SSH, target=reactor-gw-02, result=failed, policy=restricted-maintenance-window, escalation=security_notify, followup=access unauthorized after repeated credential mismatch* |
| SCADA-X | Normal | Telemetry sync | *RTU_NORTH/substation=E14 feeder=FD-22 voltage=109.8 kV current=241 A breaker=CB-22 state=CLOSED sync=OK historian=written poll_seq=388221 remote_cmd=none alarm_state=clear* |
| | Normal | Access Session | *GW-SEC-11 :: session=93AF12 :: source=dispatch-console-04 :: user=grid_operator_7 :: auth=success :: scope=substation_read :: target=E14/FD-22 :: command=read_status :: response=completed :: duration=118 ms* |
| | Normal | Alarm clearance | *ALM-HUB region=west-grid source=substation_E14 class=voltage_warning severity=low previous_state=active new_state=cleared clear_cause=load_normalized acknowledgment=received archive=replicated* |
| | Normal | Protocol handshake | *PROTO-DNP3 19.401Z master=ctrl-center-2 rtu=E14-R12 seq=5512 op=integrity-poll link=healthy latency=23 ms confirm=received object-group=30 variation=1 status=nominal no retransmission required* |
| | Anomalous | Scheduling conflict | *EMS-SCHED region=west-grid task=load_rebalance_301 preempted_by=task_299 target=substation_E14 feeder=FD-22 expected_window=03:25:00-03:26:00 actual_dispatch=missed retry=5 breaker_command=deferred status=true escalation=raised reason=scheduling conflict under peak load* |
| | Anomalous | Unauthorized access | *SEC-EVT::node=gw-sec-11::user=alice::source=10.18.4.33::scope=relay_write::auth=denied::policy=night-shift-lockdown::attempt=4::target=substation_E21/CB-04::event=Access unauthorized* |
| | Anomalous | Network field failure | *NETTRACE path=10.0.0.5:8080->172.16.4.10:443 route=west-backbone packet_loss=3.8% rtt=412 ms handshake=partial mirror_state=inconsistent failover=not triggered consequence=telemetry delay across historian replication* |
| | Anomalous | Failure escalation | *OPS-ALERT station=E32 transformer=T09 event=output mismatch: checksum failure expected=AA73FC actual=AA71FC relay_trip=not issued operator_notice=sent severity=high cascade_risk=possible downstream_impact=regional balancing instability* |

logic modules, sensor event streams, alert management subsystems, and operator interaction logs. These logs capture critical operational states such as valve openings, reactor conditions, control loop anomalies, and overridden safety procedures.

Over a one-month production window in November 2024, we collected 871,243 raw log entries from the active system. After removing low-informative periodic messages and normalizing timestamp formats, we constructed a refined dataset of 14,382 structurally representative log entries. The dataset features highly nested parameter structures, multi-source event interleaving, and extensive use of domain-specific codes, many of which are not documented in public manuals. These characteristics present unique challenges to existing parsing tools, particularly with respect to template extraction and semantic segmentation.

The PIMS dataset represents a high-throughput, mission-critical industrial scenario in which accurate log analysis is essential for early fault detection and regulatory compliance. Its inclusion in our benchmark enables evaluation of log analysis models under conditions of high event density, overlapping system activities, and rich domain semantics.

#### 3.2.4. SCADA-X: Logs from supervisory control and data acquisition systems

The SCADA-X dataset originates from a Supervisory Control and Data Acquisition (SCADA) system deployed in a national-scale energy distribution network. SCADA systems are foundational to critical infrastructure sectors such as electricity, water, and gas, enabling real-time monitoring and control of geographically distributed assets, including substations, transformers, and load balancers. This particular SCADA system integrates multiple subsystems, including remote terminal units (RTUs), programmable automation controllers, historian servers, and alarm processing engines. Logs are produced through continuous telemetry, status polling, fault propagation tracking, and remote operator commands. These logs capture a diverse set of operational signals, ranging from analog voltage readings and protocol handshakes to access control violations and alarm escalations.

Between September and October 2024, we collected 902,377 raw logs during routine operation and scheduled stress testing. Due to the presence of timestamp drift, inconsistent message granularities, and asynchronous log bursts, we applied time-aligned windowing and event deduplication to extract 15,110 high-fidelity entries for downstream analysis. These logs are notable for their structural heterogeneity and temporal noise, with many sequences combining short diagnostic flags and verbose alarm reports within the same temporal window.

SCADA-X presents a highly safety–critical, real-time control environment where even small anomalies can indicate cascading risks. It is therefore well-suited for evaluating the robustness of anomaly detection models under high-noise, time-sensitive, and regulation-bound industrial contexts.

### 3.3. Log parsing evaluation on industrial datasets

Effective log parsing is a prerequisite for downstream log analysis tasks such as anomaly detection, root cause localization, and automated alert generation. To evaluate the capability of existing parsers to handle structurally diverse and domain-specific log data, we conducted a systematic parsing experiment on the curated subsets of our four datasets introduced in Table 2. These subsets serve as a common ground for parser comparison, ensuring consistency across industrial and benchmark data.

Our evaluation covers both traditional rule-based log parsers and recent LLM-based approaches. Rule-based parsers typically rely on regular expression patterns or heuristic clustering strategies, offering fast and lightweight solutions but often struggling with log diversity and format drift. In contrast, LLM-based parsers leverage pretrained language models to identify and extract templates, enabling more adaptive and semantically informed parsing in complex or evolving logging environments.

**Rule-based Parsers.** We selected four widely used rule-based log parsers: **Drain** [37], **Spell** [38], **AEL** [49], and **IPLoM** [50]. These parsers represent distinct classes of traditional techniques. Drain utilizes a fixed-depth tree to match token sequences; Spell employs Longest Common Subsequence (LCS) for grouping; AEL is frequency-based and tuned for accuracy in structured systems; and IPLoM applies multi-stage partitioning with statistical insights. These tools have been extensively validated on LogHub datasets, making them strong baselines for testing their generalization to industrial logs.

**LLM-based Parsers.** To examine the capabilities of large language models in log parsing, we evaluated three representative LLM-based approaches: **DivLog** [51], **LILAC** [15], and **LUNAR** [52]. These methods leverage the contextual understanding and few-shot generalization capabilities of pretrained LLMs (e.g., GPT-style models) to extract log templates without handcrafted rules. DivLog introduces a provenance-guided prompt formulation strategy, LILAC combines clustering with in-context learning to reduce reliance on large annotated corpora, and LUNAR proposes a contrastive grouping mechanism to enhance structural pattern abstraction in high-variance log streams. We selected these models for their scalability and promising performance in recent literature.

Each parser was applied to the subset of each dataset, and its output was evaluated using standard parsing metrics. By comparing both rule-based and LLM-based approaches across industrial and benchmark logs, our experiment aims to uncover the strengths and limitations of current parsing strategies when confronted with real-world industrial log complexity.

### 3.4. Log anomaly detection evaluation in industrial settings

Anomaly detection serves as a cornerstone of intelligent monitoring systems in industrial software, providing early warnings for faults, misconfigurations, and runtime anomalies that may elude traditional rule-based diagnostic methods. To rigorously evaluate the generalization capabilities of existing detection models in the context of real-world industrial applications, we conducted a comprehensive comparison across four structurally diverse datasets introduced in Table 2. These datasets span multiple industrial scenarios – including low-resource assembly lines, process control systems, and large-scale SCADA platforms – offering a realistic foundation for testing model robustness under heterogeneous log structures, varying anomaly densities, and domain-specific noise characteristics.

Our evaluation includes five representative detection models: four semi-supervised methods – **PLELog** [20], **LogBERT** [21], **SemiRALD** [17], and **SemiSMAC** [53] – and one supervised method—**LogRobust** [40]. These models were selected to cover a spectrum of design philosophies and data dependencies.

**PLELog** [20,45] introduces a semi-supervised log-based anomaly detection technique that employs probabilistic label estimation to efficiently handle unlabeled log data. This approach integrates semantic information from log events into fixed-length vectors, clusters these vectors using HDBSCAN, and refines the detection process with an attention-based GRU network. Empirical evaluations on datasets such as HDFS and BGL demonstrate PLELog's robust anomaly detection capabilities, reducing the need for extensive manual labeling.

**LogBERT** [21] is a BERT-based model for anomaly detection, which employs MLM and DeepSVDD losses during training. It processes log sequences refined by a Drain parser to learn normal log patterns through tasks such as masked log key prediction and hypersphere minimization. This strategy enables LogBERT to capture deep contextual relationships within log data, establishing a robust framework for anomaly detection based on deviations from learned log patterns.

**SemiRALD** [17] is a novel semi-supervised learning-based approach for log anomaly detection that integrates a hybrid language model combining RoBERTa with an attention-based LSTM network. This framework is specifically designed to robustly analyze log data. The system utilizes ChatGPT for efficient log parsing, ensuring the integrity of the parsed log information, and employs a semi-supervised learning paradigm to address the challenge of data scarcity. By learning the typical patterns of system behavior, SemiRALD effectively detects anomalies, making it particularly suitable for environments with limited labeled data. Extensive experiments have demonstrated its promising performance in log anomaly detection.

**SemiSMAC** [53] is a semi-supervised framework, combining the power of LLMs and Sequential Model-based Algorithm Configuration (SMAC) for enhanced performance. The model utilizes ChatGPT for efficient log parsing and a new log grouping approach, which requires only a small number of labeled samples to generate pseudo-labels for the rest of the data. This technique expands the training set and enables better anomaly detection. SMAC is then used to optimize the hyperparameters of the embedded models, further improving the framework's performance. The SemiSMAC-LSTM variant, using an LSTM backbone, demonstrates superior results in experiments on multiple benchmark datasets, particularly excelling in low-resource scenarios. This makes SemiSMAC a promising solution for scalable and automated anomaly detection, even in challenging real-world applications.

**LogRobust** [40] is a supervised model built on an attention-based Bi-LSTM architecture. It employs a specialized log parser to preprocess logs, generating TF-IDF and word semantic vectors to extract log features. Both normal and abnormal logs contribute to the training process. The inclusion of attention mechanisms strengthens the model's ability to pinpoint significant anomalies in log data, thereby improving detection accuracy and reliability across various operational contexts.

A key challenge in deploying anomaly detection in industrial environments lies in the scarcity of labeled anomaly data. Many real-world anomalies are rare, subtle, and domain-specific—requiring domain experts to interpret complex temporal patterns across multiple components. As a result, the manual annotation of logs is costly, time-consuming, and often inconsistent across systems. This makes models that require large volumes of labeled training data, such as LogRobust, difficult to apply in practice. Semi-supervised methods offer a more viable path forward, yet their performance and data requirements vary significantly. For example, PLELog assumes access to 50% of normal logs for initialization, while SemiRALD demonstrates promising results using as little as 0.1% of total logs.

To ensure a fair, reproducible, and interpretable evaluation across models and datasets, we constructed dedicated sub-training sets for each of the four datasets. These subsets were carefully curated following a unified sampling protocol, while preserving the statistical properties of the full dataset. Labeling was guided by domain-specific heuristics and expert validation, particularly for industrial logs where anomalies often exhibit subtle, system-dependent behaviors.

All models were trained and evaluated using this standardized setup, which controls for data-related confounding variables. By aligning the training data structure across models, we ensure that observed performance differences are attributable to the models' inherent detection capabilities, rather than artifacts introduced by inconsistent data volume or labeling coverage. This evaluation design allows us to fairly compare supervised and semi-supervised methods under realistic industrial conditions.

### 3.5. Sub-dataset validation and processing

To ensure the reliability and experimental validity of the constructed sub-datasets, we conducted a structured multi-stage validation process involving expert review and iterative consensus-building.

Each of the four sub-datasets underwent manual annotation by four domain experts with over five years of experience in software development and system operations. These experts were divided into two independent teams, and a two-round cross-validation protocol was implemented. In the first round, each team independently labeled log entries as either *normal* or *anomalous*, and determined the correct parsing results—i.e., the *log parsing ground-truth templates*. In the second round, each team cross-validated both the anomaly labels and the parsing ground truth produced by the other team, focusing on discrepancies in either classification or structural interpretation. Entries for which the two teams could not reach agreement in either labeling or parsing output were flagged for arbitration. A round-table discussion was subsequently held involving all four experts, during which final decisions were made collaboratively. These sessions allowed for the resolution of ambiguity in both anomaly detection and log parsing, resulting in finalized and consensus-based labels and template structures for each disputed entry.

During the round-table meetings, the experts were also asked to assess the representativeness and balance of each sub-dataset. Representativeness was defined as the extent to which the sub-dataset captured the diversity of the full dataset, including typical variations in system states, workload conditions, and both common and rare anomaly patterns. The goal was to ensure that the selected logs could adequately reflect realistic operational contexts. Balance, on the other hand, referred to the proportional distribution of normal and anomalous samples, and the reduction of excessive repetitions of near-identical logs (e.g., differing only in dynamic fields). The experts were instructed to confirm that no single log template was overrepresented to the point of inflating evaluation performance. Only when all four experts agreed that both criteria were satisfactorily met was the sub-dataset approved for use in subsequent experiments.

In total, 527 log entries across the four sub-datasets required arbitration and were resolved through expert discussion. The final labels and parsing ground truths were established for these entries through consensus. Moreover, the expert panel unanimously rated the sub-datasets as highly representative and well-balanced. The logs spanned all original datasets and no log format appeared more than six times within any sub-dataset. These validated sub-datasets were thus confirmed to be suitable for use in both log parsing evaluation and as training data for semi-supervised anomaly detection models.

## 4. Experimental setup

### 4.1. Research questions

In this section, we present the core research questions and describe the corresponding experimental designs. Each question targets a specific aspect of log analysis in industrial settings, aiming to provide insight into parser robustness, anomaly detection generalization, and the practical implications of industrial log diversity.

*RQ1: How well do representative rule-based and LLM-based log parsers generalize across diverse industrial and benchmark datasets?*

In this study, we evaluated seven log parsers – Drain, Spell, AEL, IPLoM (rule-based), and DivLog, LILAC, LUNAR (LLM-based) – across curated subsets of four datasets, including one benchmark dataset (LHSB) and three industrial datasets (FALL, PIMS, and SCADA-X). Each parser was applied independently to extract templates from raw logs, and the results were assessed using standard parsing metrics. This subset-based evaluation was designed to measure how well different parsers generalize across datasets with varying levels of structural regularity, semantic density, and operational complexity.

By comparing parser performance across benchmark and industrial subsets, we aimed to quantify the extent to which each method adapts to format variation, template diversity, and domain-specific semantics. This is particularly important in industrial logs, where messages are often less repetitive, more weakly standardized, and more tightly coupled with control states, sensor values, protocol traces, and

hardware-specific information. Through this comparison, we examine whether recent LLM-based parsers provide more robust parsing behavior than traditional rule-based methods under realistic industrial conditions.

To complement the accuracy-oriented evaluation, we further conducted an efficiency analysis by measuring the time required for the same seven parsers to process the complete logs from all four full datasets, comprising more than 3.24 million log messages in total. All runtime measurements were conducted on the same Ubuntu 22.04 server with a 32-core Intel Xeon CPU, 128 GB RAM, and Python 3.10, using CPU-only execution for local parsers and end-to-end wall-clock timing for API-based parsers. For the LLM-based parsers, a unified API setting was adopted throughout all experiments to ensure implementation consistency and fair comparison across methods. Specifically, DivLog, LILAC, and LUNAR were all implemented using the official `gpt-3.5-turbo-0613` API with deterministic decoding. For LUNAR, both *serial* and *parallel* modes were reported to reflect different deployment settings. This additional experiment was intended to assess the practical cost of parser deployment at scale, since industrial log parsing is often part of a larger operational pipeline involving continuous monitoring and downstream anomaly detection. Together, the two experiments provide a more complete evaluation of parser generalization from both effectiveness and efficiency perspectives.

*RQ2: How do semi-supervised and supervised anomaly detection models perform across industrial and benchmark datasets under standardized conditions?*

In this study, we investigated the performance and robustness of five state-of-the-art log anomaly detection models: PLELog, LogBERT, SemiRALD, SemiSMAC (semi-supervised), and LogRobust (supervised). Each model was trained and evaluated using the standardized, labeled subsets we created for the four datasets. These subsets ensure fair comparison by aligning data quantity, label coverage, and structural balance.

Through this controlled experiment, we aim to evaluate model generalization to complex industrial log patterns, especially under low-label or sparse-anomaly conditions. We further analyze the trade-offs between annotation cost and detection performance, identifying which models are most practical for deployment in real-world industrial monitoring systems with limited human supervision.

*RQ3: How label-efficient are log anomaly detection models under industrial annotation scarcity?*

While RQ2 evaluates model performance under a fixed, standardized labeling setup, real-world industrial deployment often faces severe annotation scarcity. In practice, obtaining labeled logs requires experienced engineers to manually inspect, interpret, and verify large volumes of system events, a process that is both time-consuming and labor-intensive. As a result, annotation quickly becomes a bottleneck, leading to substantial human cost and limiting the scalability of anomaly detection solutions.

In this section, we extend the controlled protocol of RQ2 by explicitly varying the available labeled budget, with the goal of quantifying the *annotation cost-performance* trade-off and identifying the practical operating range of different detection models under realistic industrial constraints. We retained the *same* standardized train/validation/test splits as in RQ2 to ensure comparability across research questions. The only factor changed in this section is the size of the labeled subset available during training. For each dataset, we constructed multiple training variants by sampling labeled instances at the following label ratios: 10%, 20%, 50%, 150%, 200%.

For the low-label regimes (10%, 20%, and 50%), labeled instances were selected using the same stratified sampling strategy as in RQ2. This procedure preserves structural balance and avoids bias toward dominant log templates or system modules by maintaining comparable distributions over log structures, temporal segments, and anomaly categories. For the extended-label regimes (150% and 200%), we followed the same principles used throughout Section 3, but enlarged the labeled pool by incorporating additional annotated logs beyond the original standardized subset. These additional samples were selected using the same subset construction rules and underwent the identical pipeline as described in Section 3.5. This includes consistency checks, noise filtering, and structural normalization, ensuring that the expanded labeled sets remain compatible with the original training data while providing increased sample diversity.

The inclusion of label ratios above 100% is intended to simulate scenarios in which industrial teams gradually accumulate annotations over extended system operation periods. This design allows us to analyze not only how models behave under extreme label scarcity, but also how effectively they continue to benefit from additional expert effort once the initial labeled core has been established.

*RQ4: What are the dominant parsing and detection error patterns in industrial logs?*

To address this question, we conducted a comprehensive analysis of structural and semantic characteristics across the four datasets. Unlike traditional benchmark datasets such as LHSB, which are relatively uniform in structure and anomaly distribution, industrial logs (FALL, PIMS, SCADA-X) exhibit high variability in message format, frequent use of domain-specific error codes, nested parameter structures, and a mixture of control-layer and hardware-level events. We measured key attributes including average log length, template diversity, log entropy, and anomaly sparsity to quantify this heterogeneity. These indicators reflect real-world challenges in industrial contexts, such as noisy sensor streams, interleaved software–hardware events, and log evolution over time due to firmware or configuration changes.

To understand how these characteristics affect model performance, we performed both quantitative and qualitative evaluations. Quantitatively, we compared the performance degradation of log parsers and anomaly detection models from LHSB to industrial datasets, observing significant drops in F1-scores and parsing accuracy. Qualitatively, we conducted a case study analyzing common error types in parsing and detection outputs. We also conducted interviews with industrial maintenance engineers to validate our observations and gain insights into critical but frequently overlooked anomaly patterns—such as subtle numerical deviations, failed overrides, or domain-specific fault signatures. These findings help surface the practical limitations of current log analysis systems and motivate the development of models tailored to industrial software scenarios.

*RQ5: Does improving log parsing accuracy translate into better anomaly detection performance?*

Although improving log parsing accuracy is often treated as an independent objective, its ultimate purpose in industrial monitoring systems is to enhance downstream tasks, most notably anomaly detection. As summarized in RQ4, industrial log parsers are prone to a set of recurrent error types, yet it remains unclear to what extent correcting these parsing errors actually benefits anomaly detection models. Therefore, in this section, we investigate the subtle and non-trivial relationship between log parsing quality and downstream anomaly detection performance, moving beyond parser-centric metrics to task-oriented evaluation.

Based on the common industrial parsing errors identified in RQ4, we considered eight representative error categories. For each category, we constructed a controlled intervention by replacing incorrectly parsed log messages with their corresponding *correct* templates, while keeping all other logs unchanged. This procedure allows us to isolate the impact of specific parsing errors on anomaly detection without altering the overall data distribution or model configuration. The corrected logs were then fed into the anomaly detection pipeline, using the same experimental settings as in RQ2.

We re-ran anomaly detection using the five models evaluated throughout the study under the corrected-parsing setting. All training, validation, and testing protocols were kept identical to RQ2, ensuring that any observed performance change could be attributed solely to parsing correction. To complement automatic evaluation, five experienced maintenance engineers independently reviewed the anomaly detection outputs, focusing on whether previously misclassified log instances were correctly detected after parsing correction.

For each parsing error category and each model, we quantified the downstream benefit of parsing correction as the percentage change in correctly detected anomalies among previously mispredicted logs. Specifically, we report:

$$\Delta = \frac{N_{\text{correct after correction}}}{N_{\text{incorrect before correction}}} \times 100\%$$

where $N_{\text{incorrect before correction}}$ denotes the number of log instances incorrectly classified before parsing correction, and $N_{\text{correct after correction}}$ denotes how many of these instances were correctly detected after the erroneous templates were replaced. This metric directly captures the extent to which improving parsing accuracy translates into meaningful gains for downstream anomaly detection.

To complement this recovery-oriented metric, we additionally report the average global F1 improvement after parsing correction, denoted as Avg. $\Delta$F1. Unlike $\Delta$, which is computed only over previously misclassified instances, Avg. $\Delta$F1 measures the overall change in anomaly detection performance on the full test set. Specifically, for each parsing error type, we first compute the F1-score before correction and after correction on the complete test data, and then average the difference across all evaluated settings:

$$\Delta F1 = F1_{\text{after correction}} - F1_{\text{before correction}}$$

$$\text{Avg. } \Delta F1 = \frac{1}{M} \sum_{m=1}^{M} \Delta F1_m$$

where $M$ denotes the number of evaluated settings. This metric provides a system-level view of the practical impact of parsing correction and complements $\Delta$ by showing whether a locally effective correction also leads to a meaningful improvement in overall detection performance.

### 4.2. Evaluation metrics

#### 4.2.1. Evaluation metrics for log parsing

To evaluate the performance of the ten open-source LLM-based parsers in log parsing, we follow recent studies [15,27] and use Parsing Accuracy (PA) and F1-score of Template Accuracy (FTA) as the evaluation metrics. PA measures the ability of the parser to accurately extract templates from the log entries, which is essential for downstream tasks such as anomaly detection and log analysis. PA is defined as the proportion of correctly parsed log messages to the total number of log messages. A log message is considered correctly parsed if, and only if, all tokens of the templates and variables are correctly identified.

$$PA = \frac{\text{Number of correctly parsed log entries}}{\text{Total number of log entries}}$$

where the "correctly parsed log" means that all dynamic variables are accurately identified and replaced.

FTA is a template-level metric that evaluates the correctness of identified templates in parsed logs. It is based on the precision and recall of the templates extracted from log messages. The FTA is calculated as the harmonic mean of Precision and Recall, which are computed specifically for the templates. A template is considered correctly identified if and only if the parsed log shares the same template as the ground-truth template, and all tokens of the parsed template match exactly with those of the ground-truth template.

$$Precision = \frac{\text{Number of correctly identified templates}}{\text{Number of parsed templates}}$$

$$Recall = \frac{\text{Number of correctly identified templates}}{\text{Number of ground-truth templates}}$$

$$FTA = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

where the FTA provides a balanced measure of template identification accuracy, considering both the ability to identify templates correctly (Precision) and the ability to capture all ground-truth templates (Recall).

*Handling of unparseable logs.* In practical deployments, log parsers may occasionally encounter inputs that are difficult to process due to irregular structures, unseen patterns, or highly dynamic content. In this study, however, all evaluated parsers (both rule-based and LLM-based) are designed to always return a parsing result for each input log message, rather than producing null outputs or discarding entries. As a result, there are no missing parsed outputs in our pipeline, and all log messages are preserved for downstream analysis.

In this context, what may be considered "unparseable" logs in other settings are treated here as cases of incorrect or degraded parsing. Specifically, when a parser fails to correctly identify dynamic variables or preserves insufficient structural information, the resulting template is still generated but does not match the ground-truth template. Such cases are therefore counted as incorrectly parsed logs and are fully reflected in the PA metric. For example, logs containing rare event patterns or complex parameter structures may be over-generalized (e.g., excessive wildcard abstraction) or under-generalized (e.g., dynamic tokens left unresolved), both of which lead to PA = 0 for that log entry. This design aligns with the operational behavior of industrial log analysis pipelines, where logs are typically not dropped but instead processed into potentially imperfect templates. It also ensures that all inputs – including structurally unusual or rare logs – are propagated to downstream anomaly detection models.

#### 4.2.2. Evaluation metrics for log anomaly detection

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. Study results

In this section, we present the experimental results and provide an analysis to address each research question.

In this study, we evaluated seven log parsers – Drain, Spell, AEL, IPLoM (rule-based), and DivLog, LILAC, LUNAR (LLM-based) – across the curated subsets of four datasets, including both benchmark (LHSB) and industrial systems (FALL, PIMS, SCADA-X). Each parser was applied independently to extract templates from raw logs, and the results were assessed using standard parsing metrics.

**Table 4**
The PA and FTA of seven representative log parsers evaluated across four datasets.

| | Drain | | Spell | | AEL | | IPLoM | | DivLog | | LILAC | | LUNAR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | PA | FTA | PA | FTA | PA | FTA | PA | FTA | PA | FTA | PA | FTA | PA | FTA |
| LHSB | 0.815 | 0.571 | 0.769 | 0.492 | 0.794 | 0.502 | 0.628 | 0.401 | 0.892 | 0.604 | 0.871 | 0.682 | 0.948 | 0.828 |
| FALL | 0.754 | 0.548 | 0.702 | 0.473 | 0.706 | 0.468 | 0.603 | 0.387 | 0.858 | 0.583 | 0.802 | 0.641 | 0.886 | 0.797 |
| PIMS | 0.737 | 0.503 | 0.674 | 0.468 | 0.682 | 0.481 | 0.573 | 0.362 | 0.863 | 0.571 | 0.785 | 0.618 | 0.852 | 0.774 |
| SCADA-X | 0.692 | 0.474 | 0.642 | 0.434 | 0.649 | 0.436 | 0.557 | 0.329 | 0.805 | 0.588 | 0.849 | 0.566 | 0.808 | 0.719 |

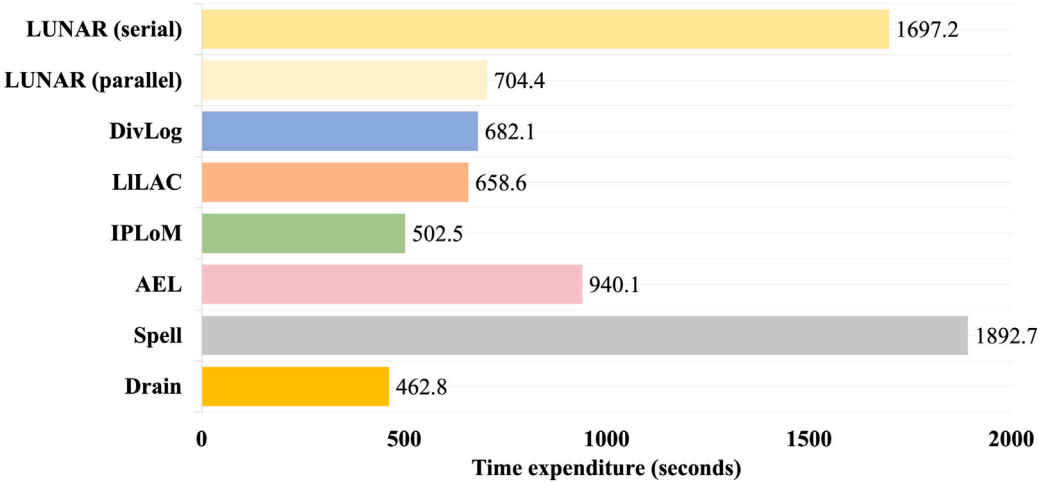


**Fig. 2.** Time expenditure of seven parsers on four full datasets.

### 5.1. Effectiveness of log parsers (RQ1)

Table 4 presents the PA and FTA of seven representative log parsers evaluated across four datasets. A consistent trend is observed: while most parsers perform reasonably well on the benchmark dataset (LHSB), their effectiveness significantly degrades on the industrial datasets, especially on SCADA-X and PIMS. For instance, Drain achieves a PA of 0.815 and FTA of 0.571 on LHSB, but its FTA drops to 0.474 on SCADA-X and 0.503 on PIMS, reflecting the difficulty of generalizing rule-based strategies to structurally complex logs.

LLM-based parsers generally outperform traditional methods across all datasets. LUNAR achieves the highest parsing accuracy overall, with FTA values of 0.828 on LHSB and 0.797 on FALL. Even on SCADA-X – arguably the most challenging dataset – LUNAR maintains a strong FTA of 0.719, surpassing all other models. In contrast, rule-based methods like IPLoM perform poorly on industrial datasets, with FTA dropping as low as 0.329 on SCADA-X and 0.362 on PIMS. These results suggest that conventional parsers are sensitive to format variability and fail to adapt to evolving logging schemes.

From a vertical perspective, we observe that every model's performance consistently declines when moving from LHSB to the industrial datasets. For example, DivLog achieves an FTA of 0.604 on LHSB but only 0.571 on PIMS and 0.588 on SCADA-X. Similarly, LILAC drops from 0.682 on LHSB to 0.566 on SCADA-X. This pattern highlights the limitations of training or evaluating log parsers solely on benchmark datasets, which may underestimate the challenges posed by real-world industrial systems with heterogeneous log formats, embedded noise, and domain-specific semantics.

To complement the accuracy-oriented comparison in RQ1, we further evaluated the practical efficiency of the seven parsers by measuring the time required to parse the complete logs from all four full datasets, comprising more than 3.24 million log messages in total. For LUNAR, we report both *serial* and *parallel* modes to reflect its different deployment settings. As shown in Fig. 2, the rule-based parsers remain substantially more efficient than most LLM-based alternatives. In particular, Drain and IPLoM required only 462.8 and 502.5 s, respectively, making them the two fastest parsers in the comparison. Among the LLM-based methods, DivLog and LILAC showed moderate time costs (682.1 and 658.6 s), whereas LUNAR incurred the highest computational cost. Its parallel mode required 704.4 s, while its serial mode reached 1697.2 s, which was markedly higher than all other parsers except Spell. These results indicate that although LLM-based parsers may provide greater flexibility and robustness for structurally complex industrial logs, their computational overhead remains a non-negligible practical concern.

This efficiency comparison provides an important complement to the main findings of RQ1. Overall, the evaluation indicates that while LLM-based parsers provide improved robustness in industrial settings, there remains a notable performance gap compared to their behavior on benchmark logs, where parsing conditions are generally more regular and computational trade-offs are less pronounced. At the same time, time expenditure must be treated as an important decision dimension when selecting parsers for industrial deployment. In real-world software systems, log parsing is often part of a larger operational pipeline involving continuous monitoring, anomaly detection, and fault diagnosis, meaning that excessively slow parsing can reduce the responsiveness of downstream analysis. From this perspective, the results suggest a clear trade-off: rule-based parsers such as Drain and IPLoM offer strong efficiency advantages, whereas LLM-based parsers, especially LUNAR in serial mode, require substantially higher processing time despite their stronger adaptability to complex industrial log structures.

**Answer to RQ1:** The results reveal a clear gap between benchmark and industrial datasets: all parsers, especially rule-based methods, experience noticeable accuracy degradation on structurally diverse industrial logs, although LLM-based parsers remain comparatively more robust. At the same time, efficiency results show that this robustness comes with an additional time cost, highlighting a practical trade-off between parsing effectiveness and computational overhead in industrial deployment.

**Table 5**
The performance of different models on four datasets.

| Model | LHSB | | | FALL | | | PIMS | | | SCADA-X | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | P | R | F1 | P | R | F1 | P | R | F1 | P | R | F1 |
| PLELog | 0.905 | 0.710 | 0.796 | 0.755 | 0.688 | 0.720 | 0.545 | 0.762 | 0.634 | 0.470 | 0.890 | 0.615 |
| LogBERT | **0.990** | 0.970 | 0.980 | 0.810 | 0.760 | 0.784 | 0.790 | 0.695 | 0.739 | 0.805 | 0.710 | 0.755 |
| SemiRALD | 0.975 | **0.995** | **0.985** | **0.910** | **0.930** | **0.920** | **0.905** | **0.938** | **0.921** | **0.900** | **0.925** | **0.912** |
| SemiSMAC | 0.935 | 0.940 | 0.937 | 0.720 | 0.795 | 0.755 | 0.675 | 0.780 | 0.724 | 0.885 | 0.818 | 0.850 |
| LogRobust | 0.950 | 0.963 | 0.956 | 0.765 | 0.890 | 0.823 | 0.750 | 0.802 | 0.775 | 0.795 | 0.780 | 0.787 |

### 5.2. Anomaly detection performance (RQ2)

As shown in Table 5, detection models achieve significantly higher performance on the benchmark dataset (LHSB) compared to industrial datasets, confirming that traditional evaluation settings may overestimate model generalizability. For example, PLELog achieves an F1-score of 0.796 on LHSB but drops to 0.615 on SCADA-X and 0.634 on PIMS. Similar patterns are observed for LogBERT (F1: 0.980 on LHSB vs. 0.755 on SCADA-X) and LogRobust (F1: 0.956 on LHSB vs. 0.775 on PIMS). These results suggest that anomaly detection in industrial environments is substantially more challenging due to noisy logs, diverse templates, and subtle anomaly characteristics.

From a horizontal comparison across models, **SemiRALD consistently outperforms all baselines across all datasets**, achieving F1-scores of 0.985 (LHSB), 0.920 (FALL), 0.921 (PIMS), and 0.912 (SCADA-X). These results indicate that SemiRALD is highly robust to industrial log variability and can generalize well even in low-resource or heterogeneous settings. Notably, even in SCADA-X – arguably the most complex dataset – SemiRALD maintains its leading performance, whereas other models such as PLELog and LogBERT experience a sharp decline.

Vertically, we observe that all models exhibit performance degradation when transitioning from LHSB to industrial datasets. The average F1-score drop from LHSB to SCADA-X is approximately 20%–25% for models like LogBERT and LogRobust, demonstrating that benchmark-trained models may not be reliable under industrial deployment conditions. This reinforces the necessity of evaluating detection models under realistic, diverse, and domain-specific log environments.

Overall, these results highlight two key insights: (1) industrial anomaly detection poses unique structural and semantic challenges that compromise the effectiveness of traditional models, and (2) SemiRALD demonstrates the best balance between detection accuracy and generalization, making it a promising candidate for real-world industrial log analysis systems.

**Answer to RQ2:** The results demonstrate that anomaly detection models perform significantly better on benchmark datasets than on real-world industrial logs, highlighting the limitations of conventional evaluation practices. Among all methods, SemiRALD exhibits the most consistent and robust performance across diverse industrial scenarios.

### 5.3. Impact of annotation scarcity in industrial settings (RQ3)

Table 6 reveals a consistent and pronounced performance degradation when the labeled budget is reduced to 10%–50%. Across all datasets, the 10% setting results in the largest F1-score drop, indicating that extremely scarce supervision substantially limits the models' ability to learn stable anomaly decision boundaries under heterogeneous log patterns. This effect is particularly evident on the more complex industrial datasets (FALL, PIMS, and SCADA-X), where all models experience steeper declines than on LHSB. These results suggest that sparse labeling is often insufficient to cover long-tail templates, evolving message variants, and rare anomaly signatures that are prevalent in real-world industrial systems.

Under severe annotation scarcity (10% and 20%), the semi-supervised approaches consistently outperform the purely supervised baseline (LogRobust). This observation is consistent with the findings of RQ2 and confirms the advantage of semi-supervised learning in leveraging large volumes of unlabeled logs to regularize representation learning and mitigate overfitting when labeled data are extremely limited.

When the labeled budget exceeds the standardized baseline (150% and 200%), F1-scores increase across most models and datasets, demonstrating that additional annotations continue to translate into measurable detection improvements. Notably, PLELog and LogRobust exhibit the most pronounced gains beyond 100%, particularly on the industrial datasets. This suggests that PLELog benefits strongly from improved coverage of rare structures and long-tail anomalies, likely due to its ability to exploit higher-quality labeled anchors within a semi-supervised learning paradigm. In contrast, LogRobust, as a fully supervised method, is more sensitive to label quantity and thus improves substantially when additional high-quality annotations reduce class imbalance and expand the diversity of anomaly exemplars.

Taken together, these results provide deployment-oriented guidance for industrial log monitoring systems. When only a very limited annotation budget is available, semi-supervised models are generally preferable due to their superior stability and higher detection accuracy under label scarcity. As the labeled budget increases, performance improves steadily across all methods, highlighting the critical role of training data volume and diversity.

From an industrial perspective, this finding has important implications: production systems such as manufacturing pipelines and large-scale monitoring infrastructures are inherently dynamic, with log patterns evolving over time. Periodic annotation updates and incremental dataset expansion therefore offer a practical and effective strategy for sustaining anomaly detection performance in long-running industrial deployments.

**Answer to RQ3:** The findings indicate that industrial anomaly detection systems must be designed for label efficiency, as semi-supervised models better tolerate annotation scarcity and incremental labeling remains essential for maintaining performance in evolving production environments.

### 5.4. Challenges in industrial log analysis (RQ4)

To better understand the limitations of current models, we conducted a case study by analyzing misclassified log entries from both the log parsing and anomaly detection tasks. Table 7 summarizes the most frequent error types and provides representative examples. These errors were primarily identified from the three industrial datasets (FALL, PIMS, SCADA-X), which exhibited greater diversity and irregularity compared to the benchmark dataset (LHSB). In addition, we consulted with five experienced maintenance engineers who reviewed a subset of the parsed and detected results and provided feedback on practical concerns often overlooked by automated systems.

One of the most prevalent parsing issues in industrial logs was the removal of critical semantic content, such as status keywords (e.g., *fail*, *unauthorized*) and system-specific error codes. Engineers emphasized

**Table 6**
The F1-score (%) under different label budgets on four datasets. Label ratio denotes the proportion of labeled training data relative to the standardized labeled subset used in RQ2 (100%). Higher ratios (150%/200%) represent extended labeled pools constructed via the same sub-dataset validation and processing pipeline.

| Model | LHSB | | | | | | FALL | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | 10% | 20% | 50% | **100%** | 150% | 200% | 10% | 20% | 50% | **100%** | 150% | 200% |
| PLELog | 0.413 | 0.446 | 0.558 | 0.796 | 0.852 | 0.871 | 0.429 | 0.504 | 0.683 | 0.720 | 0.748 | 0.823 |
| LogBERT | 0.627 | 0.616 | 0.746 | 0.980 | 0.982 | 0.986 | 0.408 | 0.477 | 0.572 | 0.784 | 0.802 | 0.837 |
| SemiRALD | 0.694 | 0.728 | 0.802 | 0.985 | 0.991 | 0.986 | 0.625 | 0.703 | 0.848 | 0.920 | 0.928 | 0.940 |
| SemiSMAC | 0.518 | 0.571 | 0.703 | 0.937 | 0.948 | 0.960 | 0.402 | 0.468 | 0.583 | 0.755 | 0.793 | 0.818 |
| LogRobust | 0.329 | 0.420 | 0.692 | 0.956 | 0.963 | 0.971 | 0.297 | 0.418 | 0.665 | 0.823 | 0.869 | 0.884 |
| Model | PIMS | | | | | | SCADA-X | | | | | |
| | 10% | 20% | 50% | **100%** | 150% | 200% | 10% | 20% | 50% | **100%** | 150% | 200% |
| PLELog | 0.337 | 0.376 | 0.485 | 0.634 | 0.718 | 0.785 | 0.385 | 0.427 | 0.502 | 0.615 | 0.737 | 0.824 |
| LogBERT | 0.429 | 0.471 | 0.620 | 0.739 | 0.759 | 0.816 | 0.478 | 0.515 | 0.684 | 0.775 | 0.786 | 0.830 |
| SemiRALD | 0.586 | 0.703 | 0.852 | 0.921 | 0.924 | 0.928 | 0.692 | 0.795 | 0.878 | 0.912 | 0.931 | 0.940 |
| SemiSMAC | 0.501 | 0.610 | 0.673 | 0.724 | 0.762 | 0.826 | 0.547 | 0.682 | 0.770 | 0.850 | 0.879 | 0.904 |
| LogRobust | 0.319 | 0.486 | 0.607 | 0.775 | 0.813 | 0.846 | 0.326 | 0.540 | 0.674 | 0.787 | 0.834 | 0.881 |

**Table 7**
Common errors in log parsing and anomaly detection observed in industrial software logs.

(a) Log parsing (Industrial logs)

| Error type | Freq. | Example (Raw log → Parsed template) |
|---|---|---|
| Status Keyword Dropped | 14.52% | *Reason=false, Service status: enabled → Reason=<*>, Service status: <*>* |
| User Identity Obscured | 14.08% | *Login failed for user alice on host backend-prod-01 → Login failed for user <*> on host <*>* |
| Error Code Removed | 13.36% | *ERROR CODE 502: Gateway Timeout at /api/process → ERROR CODE <*>: Gateway Timeout at <*>* |
| Incomplete Parameter Parsing | 12.95% | *Auth: user=alice, src=192.168.0.15, method=SSH, result=failed → Auth: user=alice, src=<*>, method=SSH, result=<*>* |
| Truncated Request Path | 12.34% | *GET /machine_status?line=3&batch=842 → GET /<*>* |
| Sensor Value Lost | 10.87% | *Sensor reading: arm_pos=(0.22, 0.98) vs. expected=(0.20, 1.00) → Sensor reading: arm_pos=<*>, expected=<*>* |
| Network Field Merged | 7.86% | *Connect from 10.0.0.5:8080 to 172.16.4.10:443 → Connect from <*> to <*>* |
| Sentence Boundary Lost | 7.01% | *Check at 12:00. Last OK: 11:30. Errors: 0 → Check at 12:00. Last OK: <*>* |

(b) Log anomaly detection (Industrial logs)

| Error type | Freq. | Description and example |
|---|---|---|
| Handled Fault Misclassified (FP) | 16.38% | Control-layer errors caught by internal recovery mechanisms were flagged as anomalies. (*Trigger 'error' caught and handled internally*) |
| Alert Verbality Misinterpreted (FP) | 15.72% | Benign logs with "alert" or "warn" were falsely flagged. (*Alert: pressure cycle initialized*) |
| Missing Failure Indicator (FN) | 14.65% | Critical keywords dropped in templates. (*Access unauthorized → Access <*>*) |
| Numerical Deviation Ignored (FN) | 14.18% | Slight numerical changes overlooked. (*torque deviation Δ=0.012 exceeds limit*) |
| Integrity Check Missed (FN) | 11.53% | Output mismatch due to checksum errors not flagged. (*output mismatch: checksum failure*) |
| Template Ambiguity (FN) | 10.04% | Faulty logs matched to success templates. (*Job /welding returned status 503*) |
| Retry Overlooked (FN) | 9.20% | Repeated retries not treated as degradation. (*retries=5, status=true*) |
| Scheduling Conflict Missed (FN) | 8.37% | Overridden or preempted tasks not flagged. (*Task 301 preempted by Task 299*) |

that in industrial environments, such keywords carry diagnostic meaning and are often used in standard operating procedures or linked to incident reports. For instance, in PIMS, the absence of an "unauthorized" label in the parsed template led to false negatives in access control monitoring. Another common error involved the failure to isolate meaningful variables such as usernames, IP addresses, and sensor values—especially when they appeared in compound expressions or nested structures. These omissions can significantly hinder root cause analysis and automated alerting.

In the anomaly detection task, we observed that models frequently misclassified benign logs due to superficial keyword matching. For example, control logs that contain terms like "error" or "alert" for traceability purposes were often labeled as anomalous, even though they represent expected system responses. Conversely, subtle yet critical faults – such as numerical drifts in torque, temperature, or position – were routinely ignored. Engineers pointed out that many industrial anomalies manifest through small parameter shifts rather than abrupt semantic deviations, which current models are not well-equipped to detect without context-aware mechanisms.

These findings underscore several key differences between traditional and industrial log environments. First, industrial logs are often multimodal and domain-specific, embedding physical state changes alongside software events. Second, formatting and template structures vary not only across systems but also over time within the same system, particularly in scenarios involving firmware updates or vendor component changes. Finally, industrial logs often exhibit low anomaly frequency and high class imbalance, which makes them ill-suited for models trained on homogeneous benchmark datasets. Future log analysis systems must incorporate richer semantic understanding, adaptive parsing techniques, and domain-specific heuristics to address these challenges effectively.

**Answer to RQ4:** Our analysis reveals that industrial log analysis presents unique challenges – such as semantic variability, subtle numerical anomalies, and inconsistent formatting – that are not adequately addressed by existing parsing or detection models. These findings highlight the need for domain-adaptive, context-aware approaches to achieve reliable performance in real-world industrial environments.

### 5.5. Downstream benefits of correcting parsing errors (RQ5)

Table 8 shows that correcting different parsing error types leads to markedly different levels of improvement in downstream anomaly detection. While some corrections result in substantial recovery of previously misclassified anomalies, others produce only marginal gains. This confirms that improvements in parsing accuracy do not uniformly translate into better anomaly detection performance; instead, the downstream effect is strongly dependent on the semantic nature of the corrected fields.

**Table 8**
Downstream impact of correcting different parsing error types. $\Delta$ (%) denotes the percentage of previously misclassified instances that become correctly detected after replacing erroneous templates with correct ones, aggregated across all five models under the RQ2 protocol. Avg. $\Delta$F1 denotes the average global F1 improvement computed on the full test set after parsing correction.

| Parsing error type | Imp. $\Delta$ (%) | Avg. $\Delta$F1 |
|---|---|---|
| Sensor Value Lost | 72.8 | +0.019 |
| Error Code Removed | 86.9 | +0.023 |
| Status Keyword Dropped | 59.4 | +0.012 |
| Truncated Request Path | 13.6 | +0.007 |
| Network Field Merged | 6.2 | +0.002 |
| Incomplete Parameter Parsing | 8.4 | +0.001 |
| Sentence Boundary Lost | 12.1 | +0.002 |
| User Identity Obscured | 2.5 | +0.0005 |

The largest gains are observed when correcting errors that remove or obscure explicit anomaly indicators. In particular, fixing *Error Code Removed* and *Sensor Value Lost* yields the highest improvements, with recovery rates of 86.9% and 72.8%, respectively. These fields often encode discrete failure states or numeric deviations that directly signal abnormal system behavior. When such information is preserved in the parsed templates, anomaly detectors are able to recover a large fraction of cases that were previously misclassified.

Correcting *Status Keyword Dropped* also leads to a substantial improvement (59.4%), indicating that status-related tokens play an important role in distinguishing normal from abnormal execution states. By contrast, errors related to structural or contextual information – such as *Truncated Request Path* (13.6%) and *Sentence Boundary Lost* (12.1%) – yield only moderate gains. This suggests that while such information can contribute to anomaly detection, it is generally less decisive than explicit error codes or sensor-level signals.

Errors involving *User Identity Obscured* and *Network Field Merged* result in minimal improvements, with gains below 10%. This indicates that identity-related attributes and network endpoint details are weak predictors of anomalous behavior in the evaluated industrial scenarios, or are already treated as high-variance noise by detection models. Correcting these fields therefore offers limited benefit for anomaly detection, despite potentially improving parser-level accuracy.

Notably, correcting different parsing error types leads to different levels of downstream improvement. While some corrections substantially recover previously misclassified anomalies, others yield only limited gains. To provide a more complete view, we also report the average global F1 improvement (Avg. $\Delta$F1) after correction. As expected, the global gains are smaller than the corresponding recovery rates. For example, correcting *Error Code Removed* and *Sensor Value Lost* produces clear benefits both locally and globally, whereas corrections to less important parsing errors, such as *Network Field Merged* and *User Identity Obscured*, result in only minor changes in overall F1. These results indicate that parsing correction does improve anomaly detection, but its system-level impact depends largely on whether the corrected error preserves or restores anomaly-critical semantic information.

Taken together, the results highlight that parser improvement efforts should be guided by downstream utility rather than parser-centric metrics alone. In resource-constrained industrial settings, prioritizing the correction of parsing errors that preserve explicit anomaly-carrying signals – such as error codes, sensor values, and status keywords – is likely to deliver the highest return for anomaly detection performance. Conversely, aggressively refining identity or formatting-related fields may offer limited practical benefit unless required for other operational objectives.

**Answer to RQ5:** The results indicate that parser improvements should prioritize preserving anomaly-critical semantic cues, as only such corrections translate into meaningful gains in industrial anomaly detection.

## 6. Discussion

This study set out to systematically examine log parsing and anomaly detection in realistic industrial settings, moving beyond benchmark-centric evaluation toward task-oriented and deployment-relevant analysis. By investigating five research questions (RQ1–RQ5), we provide a holistic view of how parsing accuracy, annotation availability, and error characteristics jointly shape anomaly detection performance in practice.

RQ1 and RQ2 establish a foundational observation: performance trends observed on benchmark datasets do not reliably generalize to industrial logs. While most models achieve strong results on LHSB, their performance degrades substantially on more complex datasets such as FALL, PIMS, and SCADA-X. These gaps persist even under standardized experimental conditions, indicating that industrial logs introduce additional challenges – such as higher template diversity, evolving message structures, and subtle anomaly signatures – that are not sufficiently captured by existing benchmarks. This finding underscores the risk of overestimating real-world readiness based solely on benchmark performance.

RQ3 extends the analysis by explicitly modeling annotation scarcity, a defining constraint in industrial environments. The results demonstrate that detection performance is highly sensitive to label availability, particularly on complex industrial datasets. Severe label scarcity leads to sharp performance degradation, whereas semi-supervised models exhibit greater robustness by leveraging unlabeled data. Moreover, continued performance gains beyond the standardized labeling baseline indicate that annotation remains valuable even after an initial labeled core is established. This suggests that label efficiency should be treated as a first-class design concern in industrial anomaly detection systems, rather than as a fixed experimental assumption.

RQ4 further reveals that these performance gaps are closely tied to systematic parsing failures that recur across industrial datasets. Rather than being random or dataset-specific, many parsing errors follow consistent patterns, including the loss of status keywords, error codes, numeric sensor values, and structured request paths. Importantly, this analysis reframes parsing errors not merely as technical imperfections, but as sources of semantic distortion that directly affect how downstream models perceive system behavior.

RQ5 provides a critical task-oriented perspective by examining whether improving parsing accuracy actually benefits anomaly detection. The results show that parsing improvements yield highly uneven downstream gains: correcting errors that erase explicit anomaly-carrying signals – such as error codes, sensor values, and status keywords – leads to substantial recovery in detection accuracy, while fixing identity-related or formatting-related errors offers little benefit. This finding highlights a key insight: higher parsing accuracy does not automatically imply better anomaly detection, and parser optimization should be guided by downstream utility rather than parser-centric metrics alone.

These findings point to a nuanced relationship between parsing and detection. Parsing should not be viewed as an isolated preprocessing step, but as a semantic transformation whose quality must be evaluated in terms of its impact on downstream decision-making. In industrial settings, overly aggressive generalization that removes anomaly-critical cues can be more harmful than imperfect but semantically faithful parsing. This challenges the common assumption that more abstraction necessarily leads to better generalization in log analysis pipelines.

In addition, we examined whether the observed template diversity and model degradation could be partially attributed to within-collection temporal drift. During the data processing phase, we discussed this issue with the engineers who provided the datasets. They emphasized that, although industrial software evolves over time due to maintenance cycles, feature updates, or the introduction of new production lines, logging practices in these systems are typically governed by predefined templates embedded in the codebase. In most cases,

new functionality reuses existing logging patterns, while variations are primarily introduced through dynamic fields (e.g., identifiers, sensor values, and context parameters). As a result, short- to medium-term drift within a fixed collection window is more likely to manifest as increased variability in parameter values rather than frequent changes in template structures.

A simplified example illustrates this practice. In many systems, logs are generated through fixed statements such as the following:

```
log.info("Task {} completed with status {} at node {}",
         taskId, status, nodeId);

try {
    executeTask(taskId);
} catch (Exception e) {
    log.error("Task {} failed due to {} (code={})",
              taskId, e.getMessage(), errorCode);
}
```

Here, the static message structure (the template) remains stable, while the dynamic variables (e.g., `taskId`, `status`, `nodeId`, `errorCode`) evolve with system state and operational context. According to the engineers, introducing entirely new log templates typically requires explicit code changes and is relatively infrequent compared to updates in runtime parameters.

This observation suggests that, within the time spans considered in this study, the impact of temporal drift on template structure may be limited, and the primary challenge for log analysis lies in handling increasing variability and complexity in dynamic fields. While we cannot rule out longer-term evolution effects beyond the collection windows, this perspective provides useful context for interpreting the representativeness of our datasets and the robustness of the evaluated models. We therefore view temporal drift as an important direction for future work, particularly in settings involving continuous deployment and long-term system evolution.

From a system design perspective, our results suggest several practical implications. First, anomaly detection pipelines should prioritize preserving semantically meaningful fields during parsing, especially those that encode failure states or quantitative deviations. Second, limited engineering resources should be allocated strategically: fixing high-impact parsing errors yields far greater returns than uniformly improving parser accuracy. Third, given the dynamic nature of industrial systems, incremental annotation and periodic pipeline updates represent a practical strategy for sustaining detection performance over time.

Finally, this work opens several directions for future research. One promising avenue is the development of task-aware parsers that explicitly optimize for downstream anomaly detection objectives. Another is adaptive parsing strategies that selectively retain or abstract fields based on their estimated anomaly relevance. More broadly, our findings call for tighter integration between parsing, detection, and human-in-the-loop processes, shifting the focus from isolated component optimization to end-to-end, utility-driven system design for industrial monitoring.

### 6.1. Implications

Our findings carry distinct implications for both future research and practical deployment of log analysis systems in industrial software environments. Rather than reiterating performance comparisons, these implications focus on how industrial characteristics reshape research assumptions and engineering priorities.

#### 6.1.1. Implications for research

From a research perspective, our results suggest that log analysis should be reframed as an end-to-end, task-oriented problem rather than a collection of isolated components. Traditional research often optimizes log parsing accuracy or anomaly detection performance independently, using benchmark datasets as proxies for real-world complexity. However, our findings indicate that higher parser accuracy does not necessarily translate into better anomaly detection, and that only certain types of parsing improvements yield meaningful downstream benefits. This calls for new evaluation paradigms that explicitly measure downstream utility, such as task-aware parsing metrics and causally grounded analyses of parser–detector interactions.

Moreover, the pronounced sensitivity to annotation scarcity observed in industrial datasets highlights the need for research on label-efficient and adaptive learning strategies. Future work should move beyond fixed-label experimental settings and instead model annotation as a dynamic and evolving resource. This includes studying incremental labeling, continual learning, and human-in-the-loop feedback mechanisms that better reflect how industrial systems are monitored and maintained over time.

#### 6.1.2. Implications for practice

From a practical standpoint, our findings suggest that industrial log analysis pipelines should prioritize robustness and semantic fidelity over maximal abstraction. In particular, parsing components should be designed to preserve anomaly-critical information – such as sensor values, error codes, and status indicators – even at the cost of reduced template compactness. Our results show that correcting errors that remove such signals delivers substantially higher returns than uniformly improving parser accuracy, providing a clear basis for prioritizing engineering effort under resource constraints.

In addition, the strong dependence of detection performance on label availability implies that industrial deployments should treat data annotation as an ongoing operational process rather than a one-time setup cost. Periodic annotation updates, targeted labeling of high-risk log segments, and selective correction of high-impact parsing errors can together offer a cost-effective strategy for maintaining detection performance in evolving production environments.

Finally, industrial log analysis systems must be designed with operational usability in mind. Logs serve not only as inputs for automated models, but also as artifacts for diagnosis, compliance, and auditing. This underscores the importance of explainable outputs, traceable decision paths, and interfaces that support collaboration between automated systems and human operators. Embedding anomaly detection within a human-centered workflow – rather than treating it as a fully autonomous decision-maker – is essential for safe and effective deployment in high-stakes industrial contexts.

## 7. Threats to validity

### 7.1. External validity

A primary threat to external validity concerns the representativeness of the industrial datasets used in this study. Although we collaborated with multiple industrial partners and selected systems from distinct domains (e.g., assembly lines, process control, and energy monitoring), the evaluated datasets still constitute a limited subset of possible industrial logging environments. Industrial software varies widely in logging practices, ranging from highly structured control-system logs to proprietary or resource-constrained embedded formats. Consequently, our findings may not fully generalize to industrial systems with substantially different logging conventions or operational constraints. In addition, certain log fields were anonymized to satisfy confidentiality requirements, which may reduce the availability of fine-grained semantic cues and affect both parsing and anomaly detection behavior. Nevertheless, anonymization was applied consistently across all models and experimental settings, ensuring that relative comparisons remain valid.

### 7.2. Internal validity

Internal validity may be affected by differences in model sensitivity to label availability, data volume, and log structure. Despite our efforts to construct standardized training subsets and adopt a unified evaluation protocol, some models are inherently more dependent on dense supervision (e.g., supervised baselines), while others rely more heavily on unlabeled data or structural regularities. These differences could influence absolute performance levels and relative rankings under certain conditions. To mitigate this threat, we fixed data splits, evaluation metrics, and training protocols across experiments, and avoided dataset-specific tuning. Nonetheless, subtle interactions between model assumptions and dataset characteristics may still contribute to observed performance variations.

### 7.3. Construct validity

A further threat relates to how parsing quality and downstream impact are operationalized. In RQ5, we assessed the effect of parsing improvements by selectively correcting specific error types and measuring downstream detection recovery on previously misclassified instances. While this design enables a controlled, task-oriented analysis, it does not capture all possible forms of parsing errors or interactions between multiple error types occurring simultaneously. Moreover, our improvement metric focuses on recovery among previously incorrect predictions rather than global performance changes, which may emphasize localized benefits over system-wide effects. However, this choice aligns with our research objective of understanding whether and when parsing corrections meaningfully translate into downstream gains.

Another threat to construct validity concerns the expert-based annotation and validation of anomaly labels and parsing ground truths. Because the industrial datasets span manufacturing, process control, and energy monitoring scenarios, incorrect or incomplete domain interpretation by annotators could affect the validity of the resulting labels and templates. To mitigate this threat, we selected four experts with complementary backgrounds in industrial software maintenance, system operations, and log-based fault diagnosis, and adopted a two-team, two-round cross-validation protocol to resolve all disputed cases through joint discussion. Nevertheless, expert judgment may still introduce subjectivity, especially for subtle anomalies whose interpretation depends on local operational context. Therefore, the resulting labels should be understood as consensus-based expert annotations rather than absolute ground truth.

## 8. Conclusion

This study systematically investigates log parsing and anomaly detection in industrial software systems and demonstrates that performance achieved on benchmark datasets does not reliably transfer to real-world industrial environments. Our results show that industrial logs pose distinct challenges due to their structural diversity, semantic density, and annotation scarcity, leading to substantial performance degradation under standardized evaluation. Importantly, we find that improving parsing accuracy benefits downstream anomaly detection only when corrections preserve anomaly-critical semantic cues, while many parsing refinements yield little practical gain. In addition, semi-supervised models exhibit greater robustness under limited supervision, and continued annotation provides sustained improvements over time. Together, these findings highlight the need for task-aware, label-efficient, and semantically grounded log analysis systems to support reliable monitoring and risk mitigation in complex industrial settings.

## CRediT authorship contribution statement

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

## 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 supported in part by the General Research Fund of the Research Grants Council of Hong Kong under No. 6000796, the Research Funds of the City University of Hong Kong under Nos. 9229109, 9229098, 9220103, and 9229029, the Hong Kong Metropolitan University Research Grant under No. RD/2025/1.24, and the Research Grant Council (RGC) of the Hong Kong Special Administrative Region, China under No. UGC/FDS16/E27/25. The authors gratefully acknowledge the financial and academic support provided by these institutions.

## Data availability

The representative data samples and source code used in this study are publicly available at https://github.com/log-detection/industrial_logs.

## References

[1] Solal Pirelli, 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, 2024, pp. 286–296.

[2] Shilin He, Jieming Zhu, Pinjia He, Michael R. Lyu, Experience report: System log analysis for anomaly detection, in: 2016 IEEE 27th International Symposium on Software Reliability Engineering, ISSRE, IEEE, 2016, pp. 207–218.

[3] Jian-Guang Lou, Qiang Fu, Shenqi Yang, Ye Xu, Jiang Li, Mining invariants from console logs for system problem detection, in: 2010 USENIX Annual Technical Conference, USENIX ATC 10, 2010.

[4] Vahid Garousi, Görkem Giray, Eray Tüzün, Cagatay Catal, Michael Felderer, Aligning software engineering education with industrial needs: A meta-analysis, J. Syst. Softw. 156 (2019) 65–83.

[5] Xingfang Wu, Heng Li, Foutse Khomh, On the effectiveness of log representation for log-based anomaly detection, Empir. Softw. Eng. 28 (6) (2023) 137.

[6] Xiaoxue Ma, Jacky Keung, Pinjia He, Yan Xiao, Xiao Yu, Yishu Li, A semisupervised approach for industrial anomaly detection via self-adaptive clustering, IEEE Trans. Ind. Inform. 20 (2) (2023) 1687–1697.

[7] Hetong Dai, Heng Li, Che-Shao Chen, Weiyi Shang, Tse-Hsun Chen, Logram: Efficient log parsing using *n* n-gram dictionaries, IEEE Trans. Softw. Eng. 48 (3) (2020) 879–892.

[8] Qiang Fu, Jian-Guang Lou, Yi Wang, Jiang Li, Execution anomaly detection in distributed systems through unstructured log analysis, in: 2009 Ninth IEEE International Conference on Data Mining, IEEE, 2009, pp. 149–158.

[9] Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, Pinjia He, DivLog: Log parsing with prompt enhanced in-context learning, in: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, 2024, pp. 1–12.

[10] Yicheng Sun, Jacky Keung, Yihan Liao, Hi Kuen Yu, Understanding industrial log analysis: A multi-dataset evaluation of parsing and anomaly detection, in: 2025 32nd Asia-Pacific Software Engineering Conference, APSEC, IEEE, 2025, pp. 491–502.

[11] Sania Partovian, Alessio Bucaioni, Francesco Flammini, Johan Thornadtsson, Analysis of log files to enable smart-troubleshooting in industry 4.0: a systematic mapping study, IEEE Access 12 (2023) 147640–147658.

[12] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Hongyi Liu, Ying Li, Scalalog: Scalable log-based failure diagnosis using llm, in: ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, IEEE, 2025, pp. 1–5.
[13] Yilun Liu, Yuhe Ji, Shimin Tao, Minggui He, Weibin Meng, Shenglin Zhang, Yongqian Sun, Yuming Xie, Boxing Chen, Hao Yang, Loglm: From task-based to instruction-based automated log analysis, in: 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP, IEEE, 2025, pp. 401–412.
[14] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, Michael R. Lyu, 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, 2023, pp. 355–366.
[15] Zhihan Jiang, Jinyang Liu, Zhuangbin Chen, Yichen Li, Junjie Huang, Yintong Huo, Pinjia He, Jiazhen Gu, Michael R. Lyu, LILAC: Log parsing using LLMs with adaptive parsing cache, Proc. ACM Softw. Eng. 1 (FSE) (2024) 137–160.
[16] Yicheng Sun, Jacky Wai Keung, Hi Kuen Yu, Wenqiang Luo, LogMeta: A few-shot model-agnostic meta-learning framework for robust and adaptive log anomaly detection, J. Syst. Softw. (2026) 112781.
[17] Yicheng Sun, Jacky Wai Keung, Zhen Yang, Shuo Liu, Hi Kuen Yu, SemiRALD: A semi-supervised hybrid language model for robust anomalous log detection, Inf. Softw. Technol. (2025) 107743.
[18] Boxi Yu, Jiayi Yao, Qiuai Fu, Zhiqing Zhong, Haotian Xie, Yaoliang Wu, Yuchi Ma, Pinjia He, 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, 2024, pp. 1–13.
[19] Yicheng Sun, Jacky Keung, Zhen Yang, Shuo Liu, Hi Kuen Yu, Improving anomaly detection in software logs through hybrid language modeling and reduced reliance on parser, Autom. Softw. Eng. 33 (1) (2026) 12.
[20] Yijun Lin, Yao-Yi Chiang, 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, 2022, pp. 1024–1033.
[21] Haixuan Guo, Shuhan Yuan, Xintao Wu, Logbert: Log anomaly detection via bert, in: 2021 International Joint Conference on Neural Networks, IJCNN, IEEE, 2021, pp. 1–8.
[22] Siyu Yu, Yifan Wu, Zhijing Li, Pinjia He, Ningjiang Chen, Changjian Liu, Log parsing with generalization ability under new log types, in: Proceedings of the 31st Acm Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2023, pp. 425–437.
[23] Tianzhu Zhang, Han Qiu, Gabriele Castellano, Myriana Rifai, Chung Shue Chen, Fabio Pianese, System log parsing: A survey, IEEE Trans. Knowl. Data Eng. 35 (8) (2023) 8596–8614.
[24] Zhiwei Zhang, Saifei Li, Lijie Zhang, Jianbin Ye, Chunduo Hu, Lianshan Yan, LLM-LADE: Large language model-based log anomaly detection with explanation, Knowl.-Based Syst. 326 (2025) 114064.
[25] Keyuan Qiu, Yingjie Zhang, Yiqiang Feng, Feng Chen, LogAnomEX: An unsupervised log anomaly detection method based on Electra-DP and gated bilinear neural networks, J. Netw. Syst. Manage. 33 (2) (2025) 1–29.
[26] Zhenhao Li, Chuan Luo, Tse-Hsun Chen, Weiyi Shang, Shilin He, Qingwei Lin, Dongmei Zhang, Did we miss something important? Studying and exploring variable-aware log abstraction, in: 2023 IEEE/ACM 45th International Conference on Software Engineering, ICSE, IEEE, 2023, pp. 830–842.
[27] Zeyang Ma, An Ran Chen, Dong Jae Kim, Tse-Hsun Peter Chen, Shaowei Wang, LLMParser: An exploratory study on using large language models for log parsing, 2024.
[28] Feng Fu, Hong Ding, Yong Qin, Jian Yu, Dechao Xu, Leveraging multi-agent framework for root cause analysis, Complex Intell. Syst. 12 (1) (2026) 4.
[29] Lingzhe Zhang, Tong Jia, Mengxi Jia, Yifan Wu, Hongyi Liu, Ying Li, XRAGLog: A resource-efficient and context-aware log-based anomaly detection method using retrieval-augmented generation, in: AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation, PDLM, 2025.
[30] Ying Fu, Meng Yan, Zhou Xu, Xin Xia, Xiaohong Zhang, Dan Yang, An empirical study of the impact of log parsers on the performance of log-based anomaly detection, Empir. Softw. Eng. 28 (1) (2023) 6.
[31] Zanis Ali Khan, Donghwan Shin, Domenico Bianculli, Lionel C. Briand, Impact of log parsing on deep learning-based anomaly detection, Empir. Softw. Eng. 29 (6) (2024) 139.
[32] Xiaoda Xie, Songlei Jian, Chenlin Huang, Fengyuan Yu, Yunjia Deng, 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, 2023, pp. 194–206.
[33] Zanis Ali Khan, Donghwan Shin, Domenico Bianculli, Lionel Briand, Guidelines for assessing the accuracy of log message template identification techniques, in: Proceedings of the 44th International Conference on Software Engineering, 2022, pp. 1095–1106.
[34] Yintong Huo, Yuxin Su, Cheryl Lee, Michael R. Lyu, Semparser: A semantic parser for log analytics, in: 2023 IEEE/ACM 45th International Conference on Software Engineering, ICSE, IEEE, 2023, pp. 881–893.
[35] Van-Hoang Le, Hongyu Zhang, Log parsing with prompt-based few-shot learning, in: 2023 IEEE/ACM 45th International Conference on Software Engineering, ICSE, IEEE, 2023, pp. 2438–2449.
[36] Yudong Liu, Xu Zhang, Shilin He, Hongyu Zhang, Liqun Li, Yu Kang, Yong Xu, Minghua Ma, Qingwei Lin, Yingnong Dang, et al., Uniparser: A unified log parser for heterogeneous log data, in: Proceedings of the ACM Web Conference 2022, 2022, pp. 1893–1901.
[37] Pinjia He, Jieming Zhu, Zibin Zheng, Michael R. Lyu, Drain: An online log parsing approach with fixed depth tree, in: 2017 IEEE International Conference on Web Services, ICWS, IEEE, 2017, pp. 33–40.
[38] Min Du, Feifei Li, Spell: Streaming parsing of system event logs, in: 2016 IEEE 16th International Conference on Data Mining, ICDM, IEEE, 2016, pp. 859–864.
[39] Yicheng Sun, Jacky Keung, Zhen Yang, Shuo Liu, Hi Kuen Yu, Semirald: A semi-supervised hybrid language model for robust anomalous log detection, 2024, Available at SSRN 4927951.
[40] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al., Robust log-based anomaly detection on unstable log data, in: Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, 2019, pp. 807–817.
[41] Xiaoxue Ma, Huiqi Zou, Pinjia He, Jacky Keung, Yishu Li, Xiao Yu, Federica Sarro, On the influence of data resampling for deep learning-based log anomaly detection: Insights and recommendations, IEEE Trans. Softw. Eng. (2024).
[42] Xiaoxue Ma, Yishu Li, Jacky Keung, Xiao Yu, Huiqi Zou, Zhen Yang, Federica Sarro, Earl T. Barr, Practitioners' expectations on log anomaly detection, IEEE Trans. Softw. Eng. (2025).
[43] Yicheng Sun, Jacky Keung, Hi Kuen Yu, Shuo Liu, Yihan Liao, Jingyu Zhang, 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, 2025, pp. 1382–1387.
[44] Min Du, Feifei Li, Guineng Zheng, Vivek Srikumar, Deeplog: Anomaly detection and diagnosis from system logs through deep learning, in: Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, 2017, pp. 1285–1298.
[45] Lin Yang, Junjie Chen, Zan Wang, Weijing Wang, Jiajun Jiang, Xuyuan Dong, Wenbin Zhang, Semi-supervised log-based anomaly detection via probabilistic label estimation, in: 2021 IEEE/ACM 43rd International Conference on Software Engineering, ICSE, IEEE, 2021, pp. 1448–1460.
[46] Wei Xu, Ling Huang, Armando Fox, David Patterson, Michael I. Jordan, Detecting large-scale system problems by mining console logs, in: Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles, 2009, pp. 117–132.
[47] Qingwei Lin, Hongyu Zhang, Jian-Guang Lou, Yu Zhang, Xuewei Chen, Log clustering based problem identification for online service systems, in: Proceedings of the 38th International Conference on Software Engineering Companion, 2016, pp. 102–111.
[48] Adam Oliner, Jon Stearley, What supercomputers say: A study of five system logs, in: 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN'07, IEEE, 2007, pp. 575–584.
[49] Zhen Ming Jiang, Ahmed E. Hassan, Gilbert Hamann, Parminder Flora, An automated approach for abstracting execution logs to execution events, J. Softw. Maint. Evol.: Res. Pr. 20 (4) (2008) 249–267.
[50] Adetokunbo A.O. Makanju, A. Nur Zincir-Heywood, Evangelos E. Milios, Clustering event logs using iterative partitioning, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 1255–1264.
[51] Alper Tufek, Mehmet S. Aktas, On the provenance extraction techniques from large scale log files, Concurr. Comput.: Pr. Exp. 35 (15) (2023) e6559.
[52] Junjie Huang, Zhihan Jiang, Zhuangbin Chen, Michael Lyu, No more labelled examples? An unsupervised log parser with LLMs, Proc. ACM Softw. Eng. 2 (FSE) (2025) 2406–2429.
[53] Yicheng Sun, Jacky Wai Keung, Zhen Yang, Shuo Liu, Yihan Liao, SemiSMAC: A semi-supervised framework for log anomaly detection with automated hyperparameter tuning, Inf. Softw. Technol. (2025) 107869.