World of Logs: A Dataset of Logs from Online Documents

Xiaohui Wang
University of Waterloo
Waterloo, Ontario, Canada
x362wang@uwaterloo.ca

Kundi Yao
University of Waterloo
Waterloo, Ontario, Canada
kundi.yao@uwaterloo.ca

Lizhi Liao
University of Guelph
Guelph, Ontario, Canada
lizhi.liao@uoguelph.ca

Pengyu Nie
University of Waterloo
Waterloo, Ontario, Canada
pynie@uwaterloo.ca

Xuan Zhang
Yunnan University
Kunming, Yunnan, China
zhxuan@ynu.edu.cn

Weiyi Shang
University of Waterloo
Waterloo, Ontario, Canada
wshang@uwaterloo.ca

Abstract

Software logs serve as valuable resources for understanding system running and are extensively used in diverse software maintenance tasks. As software systems get more complex and log data grows, a good log dataset is fundamental for developing automated log analysis tools. However, current log datasets are limited in three aspects, i.e., narrow in scope, lacking context information, and outdated. To bridge this gap, in this paper, we aim to extract software logs from online resources (e.g., JIRA issue reports, GitHub repositories, and Stack Overflow discussions), which concern various types of software systems and provide context for logs, such as observed behaviors and expected behaviors. This work introduces **`WoL`**, a dataset comprising real-world logs along with their contextual information. WoL currently contains over 2.5 million log messages or logging statements from diverse online resources and is publicly available to facilitate reproducible research. WoL can be used for various log-related tasks, including understanding logging intent and quality, anomaly detection, and linking logs with software artifacts for contextual analysis. WoL is publicly available on Zenodo and will be continuously updated. Furthermore, based on WoL, we develop a search engine, **`LogSearch`**, to support user queries.

CCS Concepts

• **Software and its engineering → Software creation and management**;

Keywords

Software Logs, Datasets, Large Language Models

ACM Reference Format:
Xiaohui Wang, Kundi Yao, Lizhi Liao, Pengyu Nie, Xuan Zhang, and Weiyi Shang. 2026. World of Logs: A Dataset of Logs from Online Documents. In *Proceedings of Make sure to enter the correct conference title from your rights confirmation email (MSR '26).* ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

MSR '26, Rio de Janeiro, Brazil

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Software logs are fundamental artifacts for understanding system runtime behavior. Specifically, logs are widely used in software maintenance, including debugging [7, 30, 31], anomaly detection [8, 10, 13], failure diagnosis [5, 15, 33], performance diagnosis and improvement [4, 24, 25]. As software systems grow in complexity and scale, the volume of log data has expanded dramatically, driving the development of automated log analysis tools [12]. High-quality log datasets serve as the foundation for training and evaluating these tools, yet their availability and comprehensiveness remain significant challenges [21].

Current real-world log datasets are limited in three aspects. First, most datasets are **narrow in scope**, which poses a challenge to the generalization of tools developed based on them. For example, *Failure Dataset*[1] [6] only contains error logs from OpenStack[2], and *Computer Failure Data Repository (CFDR)*[3] primarily focuses on component failure logs. This limited scope hinders the generalizability of tools developed using these datasets [34]. Second, existing datasets usually **lack contextual information**. For example, *LogHub*[4] [34], the most widely used dataset, offers large volumes of raw logs collected from both production data and lab environments. However, only six project datasets in LogHub include basic labels (e.g., normal/abnormal), and these labels are incomplete [27]. This absence of contextual richness, such as the circumstances under which logs were generated, associated developer discussions, and the rationale behind logging decisions, limits deep analysis and practical applicability [2]. Third, many datasets are **outdated**. For instance, the Blue Gene/L (BGL) and Thunderbird datasets in LogHub, collected between 2004 and 2006, are still being used for anomaly detection research [19, 20, 22]. These outdated datasets limit their effectiveness for more recent systems, which have grown significantly larger in scale and more complex in structure.

Recent advances in natural language processing (NLP) and large language models (LLMs) have demonstrated significant potential for log analysis tasks [23, 29]. However, the lack of large-scale, diverse, and contextually rich log datasets constrains the development of these advanced techniques. To bridge this gap and mitigate limitations of existing datasets, we extract software logs from online developer communications, including JIRA issue reports, GitHub discussions, Stack Overflow posts, and broader web content, as a

[1] https://figshare.com/articles/Failure_dataset/7732268/2
[2] https://www.openstack.org/
[3] https://www.usenix.org/cfdr
[4] https://github.com/logpai/loghub/

rich, diverse, and continuously updated source of real-world logs. These sources naturally embed logs within discussions about troubleshooting and system behaviors, providing both log content and valuable context about their practical usage.

We present World of Logs (**WoL**), a dataset comprising over 2.5 million software logs from 6.3 million web pages, along with rich metadata (project names, timestamps, severity levels, discussions, URLs). To extract logs from unstructured text at scale, we develop a three-stage pipeline using fine-tuned language models. Our approach achieves an end-to-end F1-score of 0.70 for complete log extraction, balancing precision and recall across highly imbalanced data. Furthermore, we present **LogSearch**, a specialized log search engine built on WoL that currently indexes 542,749 log messages and logging statements, enabling developers to search, filter, and explore logs with their full contextual information. To the best of our knowledge, LogSearch is the first search engine specifically designed to support software log research. Developers can use LogSearch to find particular logs for diverse goals (e.g., troubleshooting, improving logging), thereby helping them to develop and maintain code.

The contributions of this paper are twofold: First, we published a **continuously updated dataset, WoL**[5], providing software logs extracted from diverse online resources and their contextual information to facilitate log analysis. Second, we present a **log search engine, LogSearch**[6], built on WoL to help developers search particular logs. WoL and LogSearch are updated approximately every four months, relying mainly on automated inference pipelines with lightweight manual curation during release.

2 Dataset Construction

In this section, we present how the dataset was generated, including its data sources and the techniques we used for data consolidation.

2.1 Data Source

We endeavor to collect log data as comprehensively as possible from diverse sources across the web. Thus, we integrate widely used data sources in the software engineering domain (i.e., JIRA, SO, and GitHub) to maintain focus and extend our coverage to the broader web.

2.1.1 JIRA. JIRA is a project management and issue-tracking tool that enables developers to record detailed information, including logs and contextual data, to track bugs, manage workflows, and improve collaboration across the software development lifecycle. We use the latest data dump (Version v7 2025-06-23)[7] provided by Montgomery et al. [26].

2.1.2 GitHub. GitHub is the largest collaborative platform for software development, offering diverse features that allow developers to record and share rich historical information. To center on the extraction of logs and their contextual information rather than code itself, we only involve artifacts such as commit messages, pull requests, and issue discussions in GitHub, which capture diverse development activities and interactions. Although GHTorrent is the most complete GitHub dump [17, 18], it lacks issue descriptions and comments. We thus rely on GitHub Archive, from which the BigCode project[8] has already extracted issue and pull request data that we adopt directly.

2.1.3 Stack Overflow. SO is the largest question-and-answer (Q&A) platform for programming, where developers share and discuss technical problems, solutions, and best practices. In our work, we rely on the official SO data dump (downloaded on July 22, 2025)[9], which provides access to posts (questions and answers), comments, and associated metadata.

2.1.4 Common Crawl. CC is a non-profit organization that provides large-scale, openly available web crawl data for research and innovation. Its regularly updated corpus, containing petabytes of web pages and metadata, is widely used in fields such as NLP [11] and security [1, 28]. For log analysis research, CC offers an opportunity to extend data collection beyond domain-specific platforms and capture real-world, diverse, and continuously evolving web contexts. In our work, we start with the most recent dump (CC-MAIN-2025-43)[10], considering only webpages written in English.

2.2 Log Extraction

To center on the extraction of logs and their contextual information rather than code itself, we extract logs from developer communication corpus, i.e., JIRA issue descriptions and comments, SO Q&A pairs, and GitHub discussions. We build a three-stage pipeline model for log extraction.

2.2.1 Three-Stage Pipeline Model. A pipeline design is adopted instead of an end-to-end approach to effectively handle extreme data imbalance and enable specialization at different text granularities. To mitigate error propagation between stages, we retain the largest uncertainty score (1 - maximum class probability) for each sample to support inspection and selective filtering. We suggest that samples with uncertainty above 0.05 should be treated with caution.

① **Document-Level Classification.** A document-level binary classifier (based on `Qwen3-Embedding-0.6B`[11] [32]) is first applied to filter out documents that are unlikely to contain logs. Since the proportion of documents containing logs is relatively small, this stage serves as a crucial filtering step to remove irrelevant text and reduce downstream noise.

② **Chunk-Level Classification.** Documents predicted to contain logs are then segmented into chunks of ten lines with an overlap of three preceding lines to preserve contextual continuity. A chunk-level three-class classifier (also based on `Qwen3-Embedding-0.6B`) is subsequently applied to distinguish among three types of text segments: (a) non-log, (b) partially log, and (c) fully log — the last category commonly appears when developers paste entire blocks of logs. This stage enables the system to focus on finer-grained regions likely to contain log content while discarding purely natural-language chunks.

[5] https://zenodo.org/records/17569722
[6] https://search.logsearches.org
[7] https://zenodo.org/records/15719919
[8] https://huggingface.co/datasets/bigcode/the-stack-github-issues
[9] https://stackoverflow.com/help/data-dumps
[10] https://data.commoncrawl.org/crawl-data/CC-MAIN-2025-43/index.html
[11] https://huggingface.co/Qwen/Qwen3-Embedding-0.6B

③ **Token-Level Classification.** To perform token-level log classification, we first split each issue report into different word tokens based on whitespace. This helps us retain special pattern tokens in logs, such as date (e.g., "`2013-01-29`"), time (e.g., "`17:37:41,357`"), and module name (e.g., "`webproxy.WebAppProxy`"). Then, a token-level binary classifier (based on `CodeBERT`[12] [9]) is used to precisely extract log tokens embedded within natural-language text.

Post-Processing. Lastly, post-processing is applied to reassemble the previously divided chunks into complete documents, i.e., restoring them to their original form, and then extracting complete log blocks from the reports.

Table 1: Summary of three datasets involved in training.

Dataset	# Train	# Val	# Test	# Total Docs
InitialDataset	487	52 (Storm)	52 (AMQ)	591
MisPredDataset	284	53 (Storm)	76 (AMQ)	413
UncertDataset	907	113 (Mix)	114 (Mix)	1,134

2.2.2 Datasets and Annotation. Our training process involves three datasets, as summarized in Table 1. The development of these datasets is described in Section 2.2.3.

We manually annotate logs in each document, adding <log> and </log> tags around the logs. The first author first familiarized herself with the log content in the initial 591 issue reports and conducted the annotations. During this process, uncertain cases were discussed with the other authors to reach a consensus. Instances that remained ambiguous after discussion (e.g., thread information or command outputs) were left unannotated. We fine-tuned the base models using the manually annotated samples.

2.2.3 Training and Evaluation. We adopted a progressive fine-tuning framework consisting of iterative annotation and refinement cycles. We first located all 591 JIRA issue reports containing logs from an existing large JIRA issue dataset [3] and manually annotated their log content to fine-tune the base models. We then applied the fine-tuned models to 2,798 issue reports sampled from all eight projects in dataset [3] at a 95% confidence level, ensuring that the sample accurately reflects the overall distribution of issue reports. This process enabled us to retrieve more log-containing documents, which are relatively rare. From the model predictions, we manually annotated all positive predictions and a 95% confidence-level subset of predicted negatives. The misclassified samples from this stage, together with all data from the first step, were used for further fine-tuning. The updated models were subsequently applied to all JIRA Apache issues created in 2025, and cases with prediction uncertainty above 0.01 were manually annotated. Finally, we fine-tuned the base models using the union of initial data, mispredicted samples, and high-uncertainty cases.

For each embedding model, an independent classification head was added and trained during fine-tuning. Each classifier was optimized independently using the cross-entropy loss function. Low-Rank Adaptation (LORA) [14] based fine-tuning was applied to `Qwen3-Embedding-0.6B` and full fine-tuning to `CodeBERT`.

We evaluated the model at each classification stage using precision, recall, and F1-score, and further reported the overall extraction performance across the pipeline. For the end-to-end evaluation, only **completely extracted log messages** were counted as true positives to compute precision, recall, and F1-score. The evaluation results are shown in Table 2.

[12]https://huggingface.co/microsoft/codebert-base

Table 2: Evaluation results of the final models

Classifier	Class	Precision	Recall	F1-Score	Accuracy
Doc-Level	No	0.96	0.93	0.94	0.97
	Log	0.98	0.99	0.98	0.97
Chunk-Level	No	0.94	0.95	0.94	0.94
	Part	0.91	0.91	0.91	0.94
	All	0.95	0.95	0.95	0.94
Token-Level	No	0.95	0.97	0.96	0.97
	Log	0.98	0.96	0.97	0.97
Overall	Log	0.73	0.66	0.70	-

3 Dataset Overview

The WoL dataset is stored as a MongoDB dump, comprising 2,585,830 log messages and logging statements collected from the web by November 10, 2025. We have extracted software logs from 2,686,223 JIRA issues, 799,968 GitHub comments (i.e., issues and pull requests), and 2,759,000 SO questions, and are continuously updating our dataset. Table 3 reports the number of webpages containing logs and the total number of webpages inferred. SO has the highest ratio of containing logs (0.21), while CC has the lowest ratio (0.01). Table 4 presents different log types across different data sources.

Table 3: Distribution of webpages containing logs across data sources (Data updated to Nov 05, 2025).

Sources	# Webpages with logs	# Webpages inferred	Ratio
JIRA	360,778	2,686,223	0.1426
GitHub	114,679	799,968	0.1434
SO	585,304	2,759,000	0.2121
CC	773	51,900	0.0149
Total	1,061,534	6,297,091	-

Table 4: Distribution of log types across data sources (Logging represents logging statements).

Source	# Build logs	# Logging	# Exception	# Runtime logs	# Test logs	# Other
JIRA	68,685	11,091	281,508	963,228	71,922	194,697
GitHub	26,099	9,121	47,952	149,061	8,950	78,482
SO	52,734	319,452	165,347	496,204	19,109	241,716
CC	104	25	121	650	13	756
Total	147,622	339,689	494,928	1,609,143	99,994	515,651

Table 5 presents the representative fields in WoL. JIRA, GitHub, and SO offer common fields: `unique_url`, `description`, `created`, `updated`, `log_msgs`, and `pred_uncertainty`. JIRA additionally provides `project`, `components`, `issuetype`, `priority`, etc. GitHub provides `type`, `repo`, `comment_count`, and `comments`. SO provides `score`, `tags`, `answers`, `answer_count`, and `is_accepted`. Due to the high diversity of webpages collected from CC, the only representative metadata we can reliably provide is `unique_url`.

4 Data Application

WoL can be used for a variety of software log-related tasks because of its richness of information as well as its large-scale size.

Understanding logging intent and quality. WoL provides rich contextual information for logs, including developer discussion text

Table 5: Representative fields in WoL.

Fields	Descriptions	Data Sources			
		JIRA	GitHub	SO	CC
unique_url	Unique URL of the data source	✓	✓	✓	✓
description	Main textual content, e.g., issue descriptions, question bodies	✓	✓	✓	✓
created	Original creation timestamp shown on the webpage	✓	✓	✓	
updated	Last updated timestamp displayed on the webpage	✓	✓	✓	
closed	Closing timestamp displayed on the webpage		✓		
project	Name of the associated project	✓			
components	Name of the component within the project	✓			
issuetype	Type of the issue, e.g., Bug, Improvement	✓			
priority	Reported priority level of the issue	✓			
environment	Runtime or deployment environment information	✓			
watchCount	Number of watchers subscribed to the issue	✓			
comments	Array of associated comments	✓	✓		
type	Record type (Issue or Pull Request)		✓		
repo	Repository name		✓		
comment_count	Total number of comments		✓		
score	Net vote count, i.e., upvotes - downvotes			✓	
tags	Tags assigned by reporters			✓	
answers	Array of answers associated with the question			✓	
answer_count	Total number of answers			✓	
is_accepted	Indicates whether the answer is accepted			✓	
log_msgs	Array of extracted log messages	✓	✓	✓	✓
pred_uncertainty	Uncertainty score of model predictions in log extraction	✓	✓	✓	✓

(covering topics such as troubleshooting, feature requests, and logging improvements) and metadata (e.g., issue type, timestamp, and target component). These structured and unstructured data enable a deep exploration of how and why developers create and modify logging statements. By leveraging advanced NLP techniques (e.g., contextual embeddings, intent classification, and temporal analysis), researchers can uncover the intent behind different logging practices, distinguishing between logs written for debugging, maintenance, or auditing. Moreover, researchers can trace the evolution of log statements over time , revealing how developers adjust verbosity levels, parameter usage, or message phrasing to improve clarity and maintainability. Through this process, researchers can identify recurring patterns that correlate with high-quality logging, such as consistent use of error levels, informative variable naming, and alignment with issue type or component context. Ultimately, this analysis helps formalize best practices for log design, enabling automated suggestions and quality assessment tools that enhance consistency and reduce technical debt in software projects.

Supporting anomaly detection and root cause analysis. Logs serve as the primary evidence of system behavior in increasingly large and complex software, and their contextual information is crucial for accurate anomaly detection and diagnosis. By integrating log content with related developer discussions, stack traces, and tool reports, WoL pairs raw system outputs with human interpretation. This integration not only improves the precision of anomaly detection models but also allows researchers to capture semantic cues about potential causes. For example, whether an exception results from configuration errors, dependency updates, or concurrency issues. Moreover, aligning logs with developer discussions provides insights into how anomalies are perceived and resolved, allowing researchers to study the human-in-the-loop dimension of software maintenance. Such cross-linked analysis helps move from merely detecting anomalies to diagnosing their underlying causes and enhancing system reliability.

Linking logs with software artifacts for contextual analysis. WoL enables the integration of logs with other development artifacts to support contextual analysis across the software lifecycle. The occurrence of logs within developer communications offers valuable contextual links, e.g., connecting log snippets with issue metadata (e.g., project, component, and timestamp) and textual cues from developer comments. Leveraging techniques such as graph and semantic embedding, logs can be automatically matched to semantically related artifacts even across heterogeneous sources. Such linkage transforms scattered log mentions into a connected graph of software evolution evidence, enabling researchers to study the relationships between logging practices, problem resolution, and knowledge sharing throughout the development process.

5 Related Work

Several open-source log datasets have been released to support research on system monitoring and failure analysis. To the best of our knowledge, Loghub [16, 34] provides the most comprehensive public log collections, including both production data made available from prior studies and logs generated in controlled laboratory settings. In another study[13] [6], error logs were collected by deliberately injecting failures into the OpenStack cloud management system. The Computer Failure Data Repository (CFDR)[14] focuses primarily on component-level failure logs from a wide range of large-scale production systems. Beyond system and failure logs, other log repositories target more domain-specific purposes. For example, SecRepo[15] curates datasets related to security incidents, including malware traces and system logs. Similarly, EDGAR[16] contains Apache web server logs capturing user access statistics from internet search activities.

While these resources have contributed significantly to the field, current log data remains limited in both breadth and diversity. Most existing datasets are narrow in scope, restricted to specific domains, or outdated. By contrast, our dataset is comprehensive (i.e., we try to collect log data from across the web), grounded in real-world environments, continuously updated, and enriched with rich contextual information. These features make WoL more useful for advancing log analysis research.

6 Conclusion

We release WoL, a continuously updated dataset that provides software logs extracted from diverse web sources and enriches them with contextual information, including the source text from which the logs are extracted and metadata such as project name, bug severity, and creation date. By the time the paper was submitted, WoL contained 2.5 million logs extracted from 6.3 million web pages. WoL aims to facilitate various log-related tasks, including understanding logging intent and quality, anomaly detection, and jointly mining logs with other software artifacts. Furthermore, we provide LogSearch, a search engine built on top of WoL to help developers search, compare, and analyze log messages efficiently. The released dataset contain inherent extraction errors due to the pipeline's design, which prioritizes coverage over precision. We document this limitation and provide uncertainty scores to support downstream use. Future work will include quality audits for the released WoL dataset and example analyses demonstrating the benefits of WoL's contextual information.

[13] https://figshare.com/articles/Failure_dataset/7732268/2
[14] https://www.usenix.org/cfdr
[15] http://www.secrepo.com
[16] https://www.sec.gov/data-research/sec-markets-data/edgar-log-file-data-sets

References

[1] Tim Allison, Wayne Burke, Valentino Constantinou, Edwin Goh, Chris Mattmann, Anastasija Mensikova, Philip Southam, Ryan Stonebraker, and Virisha Timmaraju. 2020. Building a wide reach corpus for secure parser development. In *2020 IEEE Security and Privacy Workshops (SPW)*. 318–326.

[2] Mohamed Amine Batoun, Mohammed Sayagh, Roozbeh Aghili, Ali Ouni, and Heng Li. 2024. A literature review and existing challenges on software logging practices: From the creation to the analysis of software logs. *Empirical Software Engineering* 29, 4 (2024), 103.

[3] An Ran Chen, Tse-Hsun Chen, and Shaowei Wang. 2021. Demystifying the challenges and benefits of analyzing user-reported logs in bug reports. *Empirical Software Engineering* 26 (2021), 1–30.

[4] Jinfu Chen, Weiyi Shang, Ahmed E. Hassan, Yong Wang, and Jiangbin Lin. 2019. An experience report of generating load tests using log-recovered workloads at varying granularities of user behaviour. In *34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 669–681.

[5] Edward Chuah, Shyh-hao Kuo, Paul Hiew, William-Chandra Tjhi, Gary Lee, John Hammond, Marek T Michalewicz, Terence Hung, and James C Browne. 2010. Diagnosing the root-causes of failures from cluster log files. In *2010 International Conference on High Performance Computing*. 1–10.

[6] Domenico Cotroneo, Luigi De Simone, Pietro Liguori, Roberto Natella, and Nematollah Bidokhti. 2019. How bad can a bug get? an empirical analysis of software failures in the OpenStack cloud computing platform. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE)*. 200–211.

[7] Di Cui, Ting Liu, Yuanfang Cai, Qinghua Zheng, Qiong Feng, Wuxia Jin, Jiaqi Guo, and Yu Qu. 2019. Investigating the impact of multiple dependency structures on software defects. In *Proceedings of the 41st International Conference on Software Engineering (ICSE)*. 584–595.

[8] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 1285–1298.

[9] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. 2020. CodeBERT: A pre-trained model for programming and natural languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1536–1547.

[10] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution anomaly detection in distributed systems through unstructured log analysis. In *The Ninth IEEE International Conference on Data Mining (ICDM)*. 149–158.

[11] Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. *Learning word vectors for 157 languages.* arXiv:1802.06893 doi:10.48550/arXiv.1802.06893

[12] Shilin He, Pinjia He, Zhuangbin Chen, Tianyi Yang, Yuxin Su, and Michael R Lyu. 2021. A survey on automated log analysis for reliability engineering. *ACM computing surveys (CSUR)* 54, 6 (2021), 1–37.

[13] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R. Lyu, and Dongmei Zhang. 2018. Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/SIGSOFT FSE), 2018*. 60–70.

[14] Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations (ICLR)*.

[15] He Jiang, Xiaochen Li, Zijiang Yang, and Jifeng Xuan. 2017. What causes my test alarm? Automatic cause analysis for test alarms in system and integration testing. In *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. 712–723.

[16] Zhihan Jiang, Jinyang Liu, Junjie Huang, Yichen Li, Yintong Huo, Jiazhen Gu, Zhuangbin Chen, Jieming Zhu, and Michael R. Lyu. 2024. A Large-Scale Evaluation for Log Parsing Techniques: How Far Are We?. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*. 223–234.

[17] Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. 2016. Using dynamic and contextual features to predict issue lifetime in GitHub projects. In *Proceedings of the 13th International Conference on Mining Software Repositories, MSR*. 291–302.

[18] Zoe Kotti, Konstantinos Kravvaritis, Konstantina Dritsa, and Diomidis Spinellis. 2020. Standing on shoulders or feet? An extended study on the usage of the MSR data papers. *Empirical Software Engineering* 25, 5 (2020), 3288–3322.

[19] Max Landauer, Florian Skopik, and Markus Wurzenberger. 2024. A critical review of common log data sets used for evaluation of sequence-based anomaly detection techniques. *Proceedings of the ACM on Software Engineering* 1, FSE (2024), 1354–1375.

[20] Van-Hoang Le and Hongyu Zhang. 2021. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 492–504.

[21] Van-Hoang Le and Hongyu Zhang. 2022. Log-based anomaly detection with deep learning: How far are we?. In *Proceedings of the 44th international conference on software engineering (ICSE)*. 1356–1367.

[22] Xiaoyun Li, Pengfei Chen, Linxiao Jing, Zilong He, and Guangba Yu. 2020. Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults. In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. 92–103.

[23] Yichen Li, Yintong Huo, Zhihan Jiang, Renyi Zhong, Pinjia He, Yuxin Su, Lionel C. Briand, and Michael R. Lyu. 2024. Exploring the Effectiveness of LLMs in Automated Logging Statement Generation: An Empirical Study. *IEEE Transactions on Software Engineering* 50, 12 (2024), 3188–3207.

[24] Lizhi Liao, Jinfu Chen, Heng Li, Yi Zeng, Weiyi Shang, Jianmei Guo, Catalin Sporea, Andrei Toma, and Sarah Sajedi. 2020. Using black-box performance models to detect performance regressions under varying workloads: an empirical study. *Empirical Software Engineering* 25, 5 (2020), 4130–4160.

[25] Lizhi Liao, Jinfu Chen, Heng Li, Yi Zeng, Weiyi Shang, Catalin Sporea, Andrei Toma, and Sarah Sajedi. 2022. Locating performance regression root causes in the field operations of Web-Based Systems: An Experience Report. *IEEE Transactions on Software Engineering* 48, 12 (2022), 4986–5006.

[26] Lloyd Montgomery, Clara Marie Lüders, and Walid Maalej. 2022. An Alternative Issue Tracking Dataset of Public Jira Repositories. In *19th IEEE/ACM International Conference on Mining Software Repositories (MSR)*. 73–77.

[27] Adam Oliner and Jon Stearley. 2007. What supercomputers say: A study of five system logs. In *37th annual IEEE/IFIP international conference on dependable systems and networks (DSN)*. 575–584.

[28] Mukund Srinath, Soundarya Nurani Sundareswara, C Lee Giles, and Shomir Wilson. 2021. PrivaSeer: A Privacy Policy Search Engine. In *International conference on web engineering*. 286–301.

[29] Junjielong Xu, Ziang Cui, Yuan Zhao, Xu Zhang, Shilin He, Pinjia He, Liqun Li, Yu Kang, Qingwei Lin, Yingnong Dang, Saravan Rajmohan, and Dongmei Zhang. 2024. UniLog: Automatic Logging via LLM and In-Context Learning. In *Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (ICSE)*. 14:1–14:12.

[30] Ding Yuan, Haohui Mai, Weiwei Xiong, Lin Tan, Yuanyuan Zhou, and Shankar Pasupathy. 2010. SherLog: error diagnosis by connecting clues from run-time logs. In *Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 143–154.

[31] Ding Yuan, Jing Zheng, Soyeon Park, Yuanyuan Zhou, and Stefan Savage. 2012. Improving software diagnosability via log enhancement. *ACM Transactions on Computer Systems* 30, 1 (2012), 4:1–4:28.

[32] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025. *Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models.* arXiv:2506.05176 doi:10.48550/arXiv.2506.05176

[33] Xiang Zhou, Xin Peng, Tao Xie, Jun Sun, Chao Ji, Dewei Liu, Qilin Xiang, and Chuan He. 2019. Latent error prediction and fault localization for microservice applications by learning from system trace logs. In *Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering (ESEC/SIGSOFT FSE)*. 683–694.

[34] Jieming Zhu, Shilin He, Pinjia He, Jinyang Liu, and Michael R Lyu. 2023. Loghub: A large collection of system log datasets for ai-driven log analytics. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*. 355–366.