A Comparative Analysis of Log Normalization Strategies on MTTD and MTTR.

Peter Adeyemo
College of Engineering
Carnegie Mellon University
Kigali, Rwanda
padeyemo@andrew.cmu.edu

Yacoba Oduro-Yeboah
College of Engineering
Carnegie Mellon University
Kigali, Rwanda
yoduroye@andrew.cmu.edu

Jean Gabriel Mpuhwezimana
College of Engineering
Carnegie Mellon University
Kigali, Rwanda
jmpuhwez@andrew.cmu.edu

I. Introduction

Logs are a record of all events in a system, which are used to monitor, detect, and investigate anomalies or incidents. Incident detection and monitoring systems, such as Security Information and Event Management (SIEM), generate logs from various sources, which vary widely in structure and syntax. These heterogeneous logs must first be normalized and parsed into a uniform structure before the analysis. Log normalization refers to the conversion of raw, unstructured logs into structured formats. Traditional rule-based parsers structure logs based on predefined templates and patterns which are explicitly coded, making traditional parsers unable to parse logs formats from evolving sources [3,5]. Security analysts would have to manually update the rules, which is a time-consuming process and produces inconsistent extractions. Inconsistent extractions hinder automated processes, increasing Minimum Time to Detect (MTTD) and Minimum Time to Respond (MTTR) in automated log analysis. MTTD measures the average time to identify a security incident to mitigate or minimize the impact of a breach, and MTTR measures the average time to resolve a security incident after detection. Prior work reports that human-written parsers achieved a precision of only 0.50 and recall of 0.48 across diverse security queries [1], thus the need for more robust and adaptive methods.

Research Question: To what extent does structured log normalization using hybrid parsing models improve log parsing precision, detection accuracy, and reduce MTTD and MTTR in automated SIEM pipelines compared to rule-based parsing approaches?

Hypothesis: Utilizing LLM-based log parsing in addition to rule-based parsing will significantly improve parsing precision (target 98%) and coverage, leading to faster ingestion and prioritization of logs for anomaly detection, thereby reducing MTTD and MTTR. Large language models(LLM) are deep learning models that understand and generate natural language by learning patterns in text. These LLM-based parsers are adaptive and extract patterns in log events, supplementing rule-based implementations, which are unable to capture all logs.

The study focuses on system logs from the Hadoop Distributed File System (HDFS) dataset [LogPai][7], as well as application logs such as Nginx and Apache access and error logs. Two normalization strategies will be compared: rule-based (Drain3)[8] and hybrid-based (LibreLog). The study assumes complete log availability and that threat or anomaly events occur within these datasets. Potential security implications of using LLMs, such as new attack surfaces, are outside the scope. The intended audience includes security analysts, SIEM developers, and researchers interested in automated log analysis.

II. Importance and Prior Work

Accurate and timely detection of security incidents is a critical goal for all modern security solutions. Automated log analysis is a key component in Security Information and Event Management (SIEM) pipelines, but heterogeneous log formats from multiple sources, including system logs (e.g., syslog, auth.log) and application logs (e.g., Nginx/Apache) pose significant challenges for automated processing. Traditional rule-based log parsing is labor-intensive, difficult to scale, and often inconsistent, requiring security analysts to maintain complex parsing rules. These limitations directly impact the mean time to detect (MTTD) and mean time to respond (MTTR), reducing the overall effectiveness of the system.

This problem matters to both security teams and SIEM engineers, who spend significant time managing and interpreting logs, and to researchers exploring better ways to automate log analysis.If parsing can be automated effectively, analysts can focus on real threats instead of maintaining rules, improving both efficiency and accuracy of detection.

Log parsing in automated log analysis has been extensively studied, and several approaches have been proposed. Based on the employed techniques, existing general log parsers can be categorized into semantic and statistical based methods[LogSHD]. Statistical methods, like n-gram models or clustering, look for common patterns but often struggle when logs contain subtle semantic differences or new formats. Semantic methods try to understand the meaning or context in logs, which allows for more flexible parsing. Recent advances in Large Language Models (LLMs) have exhibited superior

capabilities in understanding and generating text, making them particularly effective for parsing semi-structured log data [2]. For instance, LibreLog [2] uses the LLaMA3-8B model to convert unstructured logs into structured templates. Their research work was an LLM-based log parser designed to achieve accuracy, speed and privacy concerns. Their LLM-based model is being used as the baseline model for our research work, however their work did not highlight the impact of improved normalization on overall security process. Our research extends their work examining the impact of LLM-based parser on MTTD and MTTR in anomaly detection. Huang et al. [3] combined Locality-Sensitive hashing(LSH) with Dynamic Time warping to automatically parse logs and achieved better parsing accuracy than traditional statistical methods. Their method reduces the search space for pattern matching and speeds up the clustering process of diverse log dataset. Their research work did not capture the comparative analysis of their model with current implementations of hybrid and LLM-based parsers. Parsing is the first step in the automated log analysis pipeline, and their work failed to capture the effect of their parsing model on threat detection analysis. Liu et al. [4] proposed a unified log parser which used token encoders to break logs into tokens and encode using character-level embeddings. Then it applies context encoders to identify common patterns in tokens. Their model achieved higher parsing accuracy compared to other statistical methods and was able to process millions of logs in a few minutes. As emphasized in our research work, their model shows that AI model architectures are more effective than rule-based systems in parsing heterogeneous logs. Locke et al. [5] developed LogAssist, which uses Drain-based abstraction and n-gram modeling to efficiently identify common log event sequences, focusing on log summarization and pattern discovery using the Drain algorithm. However, their model fails to recover logs the Drain algorithm is unable to parse, making it inefficient for heterogeneous log datasets. Meng et al. [6] proposed Log Parse which has a cross-service adaptive feature that can transfer models between services, while also compressing logs to save space and remove redundancy. Yet their model is unable to generalize to unseen log formats as compared to hybrid and LLM-based models.

III. Methodology

This study proposes a structured log processing pipeline that leverages a Large Language Model (LLM) to parse heterogeneous logs collected from multiple sources, including system logs (e.g., syslog, auth.log) and application logs (e.g., Nginx/Apache access and error logs). Raw logs are preprocessed to remove noise, standardized, and converted to tokenized content. Each entry is paired with a structured representation (event template + parameters) to form a dataset for LLM-based parsing. The hybrid model (LibreLog) learns semantic and contextual patterns from the data, enabling it to generalize to unseen log structures. This approach differs from deterministic parsers like Drain3 and Spell, which rely on predefined rules and heuristics. Hyper-parameters are tuned to optimize parsing precision and recall. The structured logs are fed into an anomaly detection and alerting pipeline, and the effect of normalization strategies on threat detection and incident response is evaluated. The methodology is divided into three sub-problems:

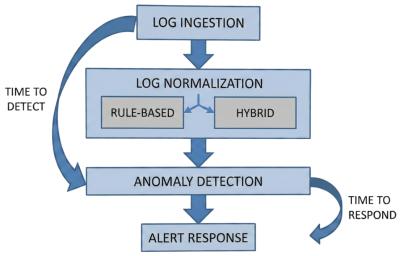

Fig. 1. Flow of log analysis

A. Sub-problem 1: Log Normalization

Goal: Compare the impact of two log normalization strategies, rule-based (Drain3) and a hybrid approach which combines Drain3 and LLM-based(LibreLog) on the quality of structured logs.
Data: System and application logs, including HDFS logs from LogPai, and Nginx/Apache access and error logs.
Method: Each normalization strategy is applied to the same set of raw logs. Rule-based parsing uses predefined templates (Drain3), and the Librelog approach first applies rule-based parsing, with LLM-based parsing handling any logs that remain unstructured.
Evaluation: We measure normalization coverage (the percentage of logs successfully structured) and processing latency to assess both accuracy and efficiency of each method.

B. Sub-problem 2: Anomaly Detection

Goal: Determine how the choice of normalization strategy affects the detection of anomalies, threats, and operational issues.
Method: Features are extracted from the normalized logs, such as template frequencies, status codes, and user/IP activity patterns. An unsupervised model, IsolationForest, is then applied to identify anomalous log entries across the two normalization outputs. This allows us to evaluate how well each normalization strategy supports downstream anomaly detection.
Evaluation: Detection performance is assessed using precision, recall, and F1-score. Detection latency is also recorded as a proxy for Mean Time to Detect (MTTD).

C. Sub-problem 3:Automated Alerting and Response

Goal: Evaluate how effectively the pipeline can reduce response time and support prioritization of incidents.
Method: For each detected anomaly, the system generates alerts and records the time elapsed from log ingestion to alert creation. This measures the responsiveness of the pipeline and

its suitability for near real-time SIEM operations.

Evaluation: Metrics include Mean Time to Respond (MTTR proxy) and effectiveness of alert prioritization, indicating how quickly and accurately the system can guide analysts to investigate critical events.

Comparison Approach: All experiments are conducted on the same dataset across three normalization strategies. Results are compared in terms of detection accuracy, normalization coverage, and latency (MTTD and MTTR proxies) to determine whether hybrid-based normalization improves threat detection and incident response performance over rule-based approaches.

IV. Analysis and Results

In this section, we measure the different evaluation metrics in the log normalization process and compare the results obtained from both Drain and Librelog. The datatset made up of 11 million log events was passed through both model implementations and evaluated for normalization efficiency. The key metrics for analysis were:

Unique templates: Templates capture the constant structures in logs into placeholders, grouping huge log datasets into meaningful patterns. Unique templates count the number of distinct logs the model identified in the log dataset. From our result, Librelog captured more logs compared to Drain, since drain uses a rule-based clustering model, and unable to capture novel logs whose structure is not predefined. The Librelog captured more unique templates, showing its ability to learn patterns and generate semantic meaning of unknown log formats.

Parsing time and Parsing Speed: This metric measured the time taken to parse unstructured logs into a structured format. From the result, Librelog took more time to parse compared to the Drain implementation.This happens because LibreLog uses both Drain and an LLM: logs are first parsed by Drain, and the unrecognized ones are then forwarded to LibreLog. This also explains why the speed of Drain was higher than Librelog.

The normalized logs are then fed to an anomaly detection model built using the Isolation Forest Algorithm. The Isolation Forest Algorithm performed better when trained on normalized logs from Librelog than it did for Drain. As stated in the hypothesis, the adaptive hybrid model achieved greater precision and recall. The Librelog-based anomaly detection found more real threats because it organized logs in a more detailed way, making it easier for the Isolation Forest Algorithm to differentiate between normal and abnormal logs. The high precision rate obtained from the parsing based on Librelog indicates a lower false positive rates,

A. Performance Comparison

TABLE I
SUMMARY OF NORMALIZATION AND ANOMALY DETECTION PERFORMANCE FOR LIBRELOG AND DRAIN.

Normalization Performance			
Category	**LibreLog**	**Drain**	**Best Performer**
Parsing Speed (logs/s)	16,779.07	17,119.37	Drain
Parsing Time (seconds)	666.046	652.806	Drain
Template Richness	55 templates	45 templates	LibreLog
Inference Timing (Normalization Output → Anomaly Model Input)			
Total Inference Time (seconds)	0.2406	0.2512	LibreLog
Avg Latency per Block (ms)	0.0021	0.0022	LibreLog
Anomaly Detection (Isolation Forest)			
Precision	0.9953	0.7318	LibreLog
Recall	0.8180	0.7992	LibreLog (slightly)
F1 Score	0.8980	0.7640	LibreLog

V. Conclusion

This project has explored the potential of combining rule-based and LLM-based log parsing to improve automated log analysis in SIEM pipelines, hoping to increase speed and accuracy. We explored Drain3 and LibreLog respectively in preparing logs for anomaly detection.

Based on our results, the hybrid strategy provided a significant gain in accuracy with almost no downside in speed. LibreLog found 55 unique log templates, whereas Drain only found 45. This proves that LibreLog captured complex logs and anomalies that Drain missed entirely. For 11 million logs, it took about 666 seconds (compared to Drain's 652 seconds). This is a negligible difference when compared per log. Furthermore, the time it took to predict an anomaly differed by less than 0.001 milliseconds.However, the answer is slightly nuanced because we haven't fully unlocked the power of those extra templates yet. We used Isolation Forest for detection, which works, but it treats log messages largely as isolated events. It likely didn't take full advantage of the richer data LibreLog provided.

To fully realize the benefits of this hybrid parsing, future work should implement DeepLog[]. Unlike Isolation Forest, DeepLog looks at the sequence of logs. Since we know LibreLog captures data that Drain misses, DeepLog would be able to use those missing logs to predict attacks much earlier. This would likely prove that while LibreLog has a tiny processing delay, it significantly reduces the overall Mean Time to Detect (MTTD) and Mean Time to Respond (MTTR) by catching errors the first time they appear.

References

[1] S. Huang and Z. Luan, "Semantic-aware log understanding and analysis," in *Proc. 33rd Int. Symp. High-Performance Parallel and Distributed Computing (HPDC)*, ACM, pp. 413–416, Aug. 2024.

[2] Z. Ma, D. J. Kim, and T.-H. P. Chen, "LibreLog: Accurate and efficient unsupervised log parsing using open-source large language models," in *Proc. 2025 IEEE/ACM 47th Int. Conf. Software Engineering (ICSE)*, IEEE Comput. Soc., pp. 924–936, Apr. 2025.

[3] S.-W. Huang, X. Wu, and H. Li, "LogLSHD: Fast log parsing with locality-sensitive hashing and dynamic time warping," in *Proc. 21st Int. Conf. Predictive Models and Data Analytics in Software Engineering*, ACM, pp. 11–20, Jun. 2025.

[4] Y. Liu *et al.*, "UniParser: A unified log parser for heterogeneous log data," in *Proc. ACM Web Conf. (WWW '22)*, New York, NY, USA: ACM, pp. 1893–1901, Apr. 2022.
[5] S. Locke, H. Li, T.-H. P. Chen, W. Shang, and W. Liu, "LogAssist: Assisting log analysis through log summarization," *IEEE Trans. Softw. Eng.*, vol. 48, no. 9, pp. 3227–3241, Sept. 2022.
[6] W. Meng *et al.*, "LogParse: Making log parsing adaptive through word classification," in *Proc. 29th Int. Conf. Comput. Commun. Netw. (ICCCN)*, pp. 1–9, Aug. 2020.
[7] logpai/logparser, Python, LOGPAI, Nov. 13, 2025. Accessed: Nov. 13, 2025. [Online]. Available: https://github.com/logpai/logparser
[8] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An online log parsing approach with fixed depth tree," in *Proc. IEEE Int. Conf. Web Services (ICWS)*, pp. 33–40, Jun. 2017.
[9] "GitHub - zeyang919/LibreLog." Accessed: Nov. 13, 2025. [Online]. Available: https://github.com/zeyang919/LibreLog