Grenze International Journal of Engineering and Technology, June Issue

LLMs for Log Analysis: A Review

Bhavesh Sawant[1] and M. K. Chavan[2]
[1-2]Department of CSE, Walchand College of Engineering, Maharashtra, India
Email: bhavesh.sawant, manik.chavan@walchandsangli.ac.in

***Abstract*— The rapid evolution of information and communication systems has resulted in the generation of massive amounts of log data, presenting unique opportunities for system monitoring and management. This review paper explores recent advancements in log analysis, with a particular focus on anomaly detection and system management techniques. The literature highlights a shift from traditional statistical approaches toward semantically-aware methods, driven largely by the integration of Large Language Models (LLMs). While log parsing remains essential for structuring raw log data, its inability to capture complex semantic patterns has led researchers to investigate alternative methods such as clustering, frequent n-gram mining, and one-to-one mapping. The unsupervised and semi-supervised techniques, along with deep learning methods have gained prominence for their ability to analyze sequential log data and detect anomalies. Additional models like MultiLog, which leverage pre-trained language models such as BERT, have shown promise in minimizing the need for labelled data. Although still in its early stages, the integration of LLMs into log-based forecasting and anomaly detection presents considerable potential. This review concludes by emphasizing the need for future research to refine LLM applications and develop robust semantic analysis techniques to tackle the growing complexity of modern log data.**

***Index Terms*— Log Analysis, Anomaly Detection, Large Language Models (LLMs), Log Parsing, Semantic analysis.**

I. INTRODUCTION

A. The Importance of Centralized Log Analysis in Modern Systems

Log analysis is crucial for managing complex systems, especially those with distributed microservice architectures [1][3]. These systems often generate large volumes of log data from various sources, making it difficult to get a unified view of system activities. Centralizing these logs provides several advantages for security, troubleshooting, and system optimization [3]. Centralized logging is vital for effective security monitoring in modern systems [4]. By collecting logs from diverse sources like firewalls, servers, and applications into a central repository, organizations gain comprehensive visibility into system activities and can effectively detect and analyses potential security threats.[4] For instance, analyzing firewall logs helps identify malicious activities such as unauthorized access attempts or data breaches [4]. This centralized view enables security professionals to: Identify patterns and anomalies: Correlating events across different logs helps detect suspicious activities that might go unnoticed when analyzing individual log sources [4]. Investigate incidents: Centralized logs provide a chronological trail of events leading up to and following a security incident, enabling thorough investigations and forensic analysis [4].

Grenze ID: 01.GIJET.11.2.602_22

Implement proactive security measures: By understanding attack vectors and vulnerabilities revealed through log analysis, organizations can strengthen their security posture and prevent future incidents [4]. With centralized logs, engineers can: Trace back errors: By following the flow of events across different system components, engineers can trace errors back to their origin and identify the source of the problem. [3] Identify faulty components: Analyzing log patterns and correlations helps isolate the specific component or service responsible for the issue, speeding up the debugging process.
Improve collaboration: A central log repository allows multiple teams to access and analyses the same data, facilitating collaboration and faster problem resolution. Analyzing centralized logs can provide valuable insights into system performance, bottlenecks, and resource utilization patterns.[3] This information is crucial for system administrators to optimize resource allocation, improve application performance, and plan for future capacity requirements [6].
For example, logs can reveal: Performance bottlenecks: Identifying which system components or services are experiencing high latency or slow response times. [6] Resource constraints: Determining which components are consuming excessive resources, such as CPU, memory, or network bandwidth. Usage patterns: Understanding how users interact with the system and identifying areas for optimization or improvement.[6]

B. Machine Learning: Enhancing Log Analysis Efficiency

The volume of log data generated by modern systems often makes manual analysis impractical [1][2][4]. Machine learning offers techniques to automate and enhance the efficiency of log analysis: Automated anomaly detection: Machine learning algorithms, especially deep learning models like LSTM networks, autoencoders, and transformer models, can learn normal system behavior patterns from historical logs and automatically detect deviations. [1] This enables proactive identification of security threats, performance bottlenecks, and system malfunctions. [5] Examples of such models include DeepLog, LogBERT, and LogFiT. [1] These models leverage the power of deep learning to identify anomalies by predicting the next log entry in a sequence, analyzing the semantic relationships between log events, and learning the linguistic patterns of normal system behavior. [1][3][5] Some models, like LogFiT, don't need log parsing, making them more adaptable to changes in log content [1]. Log parsing and grouping: Machine learning can automate log parsing, the process of structuring unstructured log messages, and log grouping, the organization of parsed logs into meaningful sequences [1]. These models can process large amounts of diagnostic information from various sources, including logs, traces, and metrics, to provide insights that help engineers resolve issues more effectively [6]. By combining centralized log analysis with machine learning, organizations can effectively manage the growing complexity and volume of log data generated by their systems. These technologies are crucial for maintaining system health, optimizing performance, and ensuring a secure and reliable operating environment.
Supervised Learning models require labelled data, where log events are pre-classified as normal or anomalous. Examples include LogSy, LogRobust, and HitAnomaly. However, obtaining labelled data is costly and limits scalability [1]. Unsupervised Learning models learn patterns from unlabeled data, making them more practical for large-scale log analysis. Examples include DeepLog, LogAnomaly, LogBERT, and LogFiT. These models often use techniques like clustering, forecasting, and reconstruction to detect anomalies [1]. Parsing-Based vs. Parsing-Free having, Parsing-based models rely on log parsers to extract templates, which can limit their adaptability to evolving log structures [1] and Parsing-free models, like LogSy, work directly with raw log data, offering greater flexibility [1]. Specific Architectures like: DeepLog: A pioneering model using LSTM to predict the next log event [1]. LogAnomaly: Similar to DeepLog but incorporates semantic information using template2vec [1]. LogBERT: Utilizes the BERT architecture, leveraging its ability to understand context and relationships within log sequences [1]. LogFiT: A more recent model that fine-tunes pretrained BERT-based models (like RoBERTa and Longformer) using a novel masked sentence prediction objective. It operates directly on raw logs, eliminating the need for parsing [1]. UniLog: An end-to-end logging framework that utilizes in-context learning (ICL). It determines logging positions, predicts verbosity levels, and generates log messages without model tuning and uses minimal training data [2].

II. Related Work

A. Theoretical Comparative Study

This review examines existing research in log analysis, focusing on the methodologies used and findings discovered. Additionally, this review will highlight how each cited work contributes to the overall understanding of log analysis. Early log analysis techniques relied on computationally intelligent techniques. These methods, while useful in their time, are hampered by the constantly updating nature of modern systems.

TABLE I. LITERATURE REVIEW

Ref.	Methodology	Key Findings/Strengths	Limitations
2	Experimental comparison of logging methods	UniLog demonstrates superior performance in automatic logging tasks.	Limited to specific datasets; may not generalize well.
3	Use of LSTM autoencoder for anomaly detection	Effective in unsupervised settings for distributed systems.	Lack of dataset specifics limits applicability.
4	Experimental comparison of classification algorithms	Achieved high MCC values indicating good performance on imbalanced datasets.	Limited dataset details; specific conditions of analysis unclear.
10	BERT-based anomaly detection system	High F1 scores indicate strong performance in detecting anomalies.	Comparison limited to a few models; broader context needed.
11	Evaluation of interpretability of LLMs	High user satisfaction in the interpretability of LLM-generated outputs.	Limited quantitative accuracy measures
13	Development of a semantic parsing approach	Demonstrates effective semantic understanding in log parsing.	Limited dataset details; potential overfitting issues.
17	Benchmarking of LLMs for security-focused log analysis	Fine-tuning LLMs improves performance in log analysis	Limited number of LLM architectures
18	Evaluation of LLMs for generating complete logging statements	LLMs show promise but require improvement for practical use. Broader programming contexts enhance performance	inconsistency in LLM performance across different logging ingredients
20	LLM-based log parsing framework (LILAC) with adaptive parsing cache and ICL	LILAC demonstrates superior accuracy and robustness compared to state-of-the-art log parsers	limited exploration of other LLM applications in log analysis
21	LLM-based log parsing framework (LLMParser) with adaptive parsing cache and ICL	LLMParser exhibits superior accuracy and robustness compared to state-of-the-art log parsers	limited exploration of other LLM applications in log analysis
24	LLM-based log parser (LogBatcher) with batch-prompting strategy	LogBatcher is effective and efficient, outperforming state-of-the-art baselines	limited exploration of the batch-prompting strategy for other log analysis tasks

Xu, J. et al. [2] introduces UniLog, an automatic logging framework that leverages large language models (LLMs) like GPT-3 to improve logging in code execution environments. A key strength of the paper is its focus on modern, state-of-the-art LLMs, but the study's limitation lies in its dataset specificity, which may reduce the generalizability of the findings to other environments or log types. Khudyakov, D. A. et al. [3] employs an LSTM autoencoder model to perform anomaly detection on distributed system logs. The unsupervised nature of the approach is a strength, especially for large, complex systems, but the lack of detailed dataset information and performance metrics limits the study's reproducibility and broader applicability. Efeoğlu, E. et al. [4] paper conducts a comparative study of various classification algorithms—Naïve Bayes, SMO, and Decision Trees—on firewall log data.

A key strength of this paper is its thorough use of multiple algorithms and its focus on the MCC metric, which is particularly relevant for imbalanced data. However, the paper provides limited information about the dataset itself, making it harder to assess whether the results are replicable in other environments. Lee, Y., Kim et al. [10] introduces LAnoBERT, a BERT-based model designed for log anomaly detection. The use of a pre-trained language model like BERT is a significant strength, as it allows for better context understanding in log analysis tasks. However, the comparison is limited to a few models, which reduces the scope of the findings. Liu, Y., Tao et al. [11] explores the use of large language models (LLMs) to perform interpretable log analysis. The focus on interpretability is an important contribution, particularly for practitioners who need more understandable models, but the lack of numerical performance measures limits a full evaluation of the approach's effectiveness.

Huo, Y. et al. [13] presents SemParser, a semantic parser designed to enhance log analytics by incorporating semantic understanding into traditional parsing techniques. However, the paper provides limited information about the datasets used, and potential overfitting issues may arise due to the model's strong performance on specific log types. Karlsen, E. et al. [17] research benchmarks five LLM architectures, including BERT, RoBERTa, and GPT-Neo, for security-centric log analysis tasks like anomaly detection. Fine-tuning these LLMs using sequence classification significantly enhances their performance. DistilRoBERTa emerges as the top-performing model, achieving an impressive average F1-score of 0.998. However, the study emphasizes the limited exploration of integrated application and system logs, highlighting a gap in current research. Despite its narrow architectural focus, the work underscores the transformative potential of fine-tuned LLMs in log analysis. Li, Y., et al. [18] investigates LLMs' effectiveness in generating logging statements using a specialized dataset, LogBench. Jiang, Z., et al. [20] introduces LILAC, a log parsing framework leveraging adaptive parsing caches and in-context learning (ICL) to address LLM inefficiencies. Evaluated on Loghub-2.0, LILAC achieves

superior accuracy compared to traditional parsers like AEL and Drain. Its innovations, particularly the adaptive parsing cache, make LILAC a robust solution for parsing challenges, emphasizing the need for broader applicability. Jiang, Z., et al. [21] LLMParser is presented as an efficient log parsing framework featuring adaptive parsing caches and candidate sampling algorithms. Benchmarked against state-of-the-art parsers like AEL and LogPPT, it demonstrates high accuracy and robustness.
Xiao, Y., et al. [24] introduces LogBatcher, a cost-effective LLM-based log parser that employs a batch-prompting strategy for enhanced accuracy. The study focuses exclusively on parsing, overlooking other log analysis tasks. Its contributions to efficient parsing make it a valuable addition to the field, with potential for expansion into other analysis areas.
Reviewed research highlights the growing potential of LLMs in the field of log analysis. While they demonstrate promising results in tasks like log parsing and anomaly detection, limitations remain in areas such as consistency, generalizability, and handling complex log data. Future research should focus on addressing these limitations and exploring the application of LLMs in a broader range of log analysis tasks.

B. Comparative Study based on Dataset

TABLE II. COMPARATIVE STUDY OF DATASET

Ref.	Dataset Used	Dataset Source/Description	Data Size
12	BGL, HDFS, Thunderbird, Zookeeper logs	Collected from open-source platforms like BGL, HDFS, Thunderbird, etc.	Mid-sized datasets from various systems
16	Loghub	Benchmark covering diverse systems like Windows & Linux OS, distributed systems, mobile systems, etc.	Each system dataset contains 2,000 manually labelled and raw log messages.
18	LogBench-O & LogBench-T	GitHub repositories. LogBench-T is transformed from LogBench-O.	LogBench-O: 3,870 methods with 6,849 logging statements
20	LogHub-2.0	A collection of large-scale datasets for log parsing.	14 datasets averaging 3.6 million log messages
21	LogPub	A collection of large-scale datasets for log parsing with ground-truth templates, built upon LogHub.	14 datasets averaging 3.6 million log messages each with ~3,500 unique templates.
23	Goby	An enterprise dataset containing information about different events.	1,187 tables.

Mudgal, P. et al. [16] uses the Loghub dataset, a diverse benchmark covering logs from systems such as Windows and Linux OS, distributed systems, mobile systems, server applications, and standalone software. Each dataset contains approximately 2,000 manually labelled and raw log messages. This dataset is utilized for log parsing, anomaly detection, root cause analysis, and other log analytics tasks. Karlsen, E. et al. [17] employs six datasets: three web application logs (Apache Access, CSIC, PKDD) and three system logs (Thunderbird, BGL, Spirit). These datasets are described in detail in the study, including their structure and use cases. The datasets are used for benchmarking various large language models (LLMs) in log analysis, focusing on security and system interpretation.
Li, Y. et al. [18] introduces LogBench-O and LogBench-T, datasets derived from GitHub repositories and transformed code. LogBench-O consists of 3,870 methods and 6,849 logging statements extracted from 2,430 Java files, while LogBench-T provides semantically equivalent transformed code. These datasets are used to evaluate the effectiveness and generalizability of LLMs in generating logging statements.
Jiang, Z. et al. [20] leverages the LogHub-2.0 dataset, a collection of 14 datasets, each averaging 3.6 million log messages with approximately 3,500 unique templates. This large-scale dataset supports evaluating the LILAC system's performance in log parsing tasks. Its significant strengths include the provision of ground truth templates, which ensure precise evaluation, and the dataset's coverage of diverse systems, enhancing generalizability.
Jiang, Z. et al. [21] uses the LogPub dataset, which is an extension of LogHub, containing 14 datasets with an average of 3.6 million log messages and ~3,500 unique templates per dataset. These datasets are employed to evaluate the effectiveness of the LLMParser framework in log parsing. Strengths include its large size, ground truth templates for evaluation, and coverage of diverse systems. Wenz, F. et al. [23] MIT Data Warehouse uses dataset includes 99 tables from an enterprise data warehouse, combining information about MIT faculty, facilities, and students. It is used to evaluate the performance of LLMs in text-to-SQL tasks within an enterprise context. Its real-world nature provides valuable insights into LLMs' capabilities for enterprise applications.

C. Analysis of Comparative Current Methodologies

TABLE III. EXAMINING THE DIFFERENCES BETWEEN CURRENT MACHINE LEARNING APPROACHES

Ref.	Model/Technique Used	Key Findings/Strengths
1	Deep Learning (DL) models, Traditional ML	Advocates for unsupervised DL techniques due to labelled data cost.
2	UniLog vs. LANCE, GPT-3	UniLog demonstrates superior performance in automatic logging tasks.
3	LSTM autoencoder	Effective in unsupervised settings for distributed systems.
4	Naïve Bayes, SMO, Decision Trees	Achieved high MCC values indicating good performance on imbalanced datasets.
5	Unsupervised clustering and supervised classification	Offers a hybrid approach that leverages the strengths of both methods.
10	BERT	High F1 scores indicate strong performance in detecting anomalies.
11	Large Language Models (LLMs)	High user satisfaction in the interpretability of LLM-generated outputs.
13	Semantic parsing methods	Demonstrates effective semantic understanding in log parsing.
15	SVM, KNN	Shows that SVM outperforms KNN in log anomaly detection tasks.

Various methodologies are employed across studies, ranging from traditional machine learning (e.g., Naïve Bayes, SVM) to advanced deep learning techniques (e.g., LSTM, BERT, LLMs). Hybrid approaches that combine both supervised and unsupervised learning show promising results, as indicated in ref. [5]. Different studies employ various accuracy metrics, including F1 Score, MCC, and user satisfaction metrics, making direct comparisons challenging. Deep learning models, particularly those using pre-trained language models like BERT and LLMs, demonstrate high performance in complex tasks like log anomaly detection and semantic parsing. The adaptability and contextual understanding of LLMs are particularly noted in ref [11]. There is a growing trend toward using large-scale, real-time datasets and advanced neural networks, reflecting the complexity of modern systems. The incorporation of semantic understanding in log parsing is gaining traction, as seen in ref. [13], which aims to enhance traditional parsing techniques.

Mudgal, P. et al. [16] explores prompt engineering with ChatGPT, focusing on its application in log parsing and analysis. ChatGPT demonstrated excellent performance in log parsing tasks, showcasing its ability to interpret and analyze structured log data effectively. However, its limitations became apparent in tasks like anomaly detection and summarization, where the performance was less robust. The findings highlight the potential of prompt engineering to improve task-specific LLM capabilities while underscoring the need for further enhancement in areas requiring deeper semantic understanding. Karlsen, E. et al. [17] research evaluates fine-tuned LLMs, including BERT, RoBERTa, DistilRoBERTa, GPT-2, and GPT-Neo, specifically for anomaly detection. Fine-tuning significantly improved the models' performance, with DistilRoBERTa achieving state-of-the-art results. The study emphasizes the impact of fine-tuning on tailoring LLMs for specific log analysis tasks. The lightweight and efficient design of DistilRoBERTa further contributes to its suitability for real-world applications requiring high performance and scalability. Li, Y. et al. [18] evaluates various LLMs on LogBench-O and LogBench-T datasets, comparing their performance to traditional tools. It provides a comprehensive analysis, emphasizing the importance of input instructions and demonstrations in achieving optimal results. The findings underscore the strengths and weaknesses of different LLM architectures in handling real-world logging tasks. The research also highlights the critical role of dataset design and task-specific instructions in maximizing the utility of LLMs for log analysis. Jiang, Z. et al. [20] research introduces an ICL-enhanced parser with an adaptive parsing cache, aimed at improving log parsing accuracy and efficiency. The parser effectively addresses inconsistencies often observed in LLM-generated outputs. By leveraging in-context learning (ICL) and adaptive caching, the system demonstrates state-of-the-art accuracy and significantly enhanced efficiency in parsing tasks. The study's methodology ensures reliability and scalability for large-scale log analysis applications.

Jiang, Z. et al. [21] The LLMParser framework combines an ICL-enhanced parser with adaptive parsing cache and hierarchical candidate sampling to achieve state-of-the-art log parsing accuracy. The approach excels in efficiently handling large-scale log data, demonstrating robust performance in diverse scenarios. The hierarchical candidate sampling method ensures comprehensive coverage of log templates, while adaptive caching optimizes parsing efficiency. This combination makes LLMParser a highly effective tool for modern log analysis. Cui, T. et al. [22] The LogEval benchmark suite evaluates multiple LLMs using a diverse dataset and multidimensional evaluation metrics across tasks like log parsing, anomaly detection, fault diagnosis, and summarization. It provides a comprehensive platform for assessing LLM capabilities, offering valuable insights into their strengths and weaknesses. By incorporating diverse datasets and task-specific benchmarks, LogEval ensures a robust and holistic evaluation process. The study highlights key areas for improvement in LLM design for log analysis.Xiao, Y. et al. [24] employs clustering, cache matching, and batch prompting techniques to enhance log parsing with LLMs. The proposed approach achieves high accuracy and efficiency, offering a demonstration-free

solution to log analysis. By combining these innovative methods, the system reduces computational overhead while maintaining parsing precision. The findings emphasize the potential of lightweight, scalable frameworks for real-world log parsing applications, providing a cost-effective alternative to traditional methods. Xu, A. & Gau, A. [25] employs hierarchical clustering with OpenAI embeddings and a neural network encoder, alongside LLM-based parsing using Claude-3.5-Sonnet. The approach achieved high accuracy in log parsing and demonstrated online parsing capabilities, making it suitable for real-time applications. The use of a word count feature added an additional layer of precision, enhancing the overall effectiveness of the model. The study highlights the versatility and adaptability of combining clustering techniques with LLMs for log data analysis.
The rapid evolution of LLM technologies is transforming the landscape of log analysis. Researchers are exploring diverse techniques like fine-tuning, ICL, prompting strategies, and caching mechanisms to leverage the power of LLMs for tasks ranging from parsing to anomaly detection and summarization. While the field is still relatively young, the progress showcased in these papers demonstrates the significant potential of LLMs in revolutionizing log analysis practices and enabling more efficient and insightful analysis of large-scale log data.

III. Conclusions

reveal a notable evolution in the field of log analysis, particularly regarding anomaly detection. Traditional methods, often reliant on potentially brittle and inflexible log parsing techniques, are being surpassed by the capabilities of large language models (LLMs). The ability of LLMs to discern semantic information from unstructured data proves highly advantageous in capturing the complexities of system behaviors recorded in logs. Methods like LogFiT demonstrate the effectiveness of fine-tuning pre-trained LMs on log data for anomaly detection, leading to more robust and adaptable solutions [1]. UniLog showcases the power of LLMs in automating the generation of logging statements, improving accuracy and efficiency compared to conventional approaches [2]. LogPrompt emphasizes the importance of interpretability in LLM-based log analysis. Its ability to provide justifications for predictions enhances trust and practical applicability [11]. The studies underscore the significance of analyzing log sequences holistically to understand system behavior, moving beyond the analysis of individual log messages [12]. Despite these advancements, open challenges and future research directions remain. For instance, ensuring generalizability and adaptability across diverse and constantly evolving log data is crucial. Further investigation into efficiently leveraging spatiotemporal features in logs and developing robust evaluation metrics tailored for LLM-based log analysis are also essential. By addressing existing challenges and continuing to explore the capabilities of LLMs, the field is poised for significant advancements that will benefit both researchers and practitioners in developing reliable and insightful log analysis techniques.

References

[1] Almodovar, C., Sabrina, F., Karimi, S., & Azad, S. (2024). LogFiT: Log anomaly detection using fine-tuned language models. IEEE Transactions on Network and Service Management.

[2] Xu, J., Cui, Z., Zhao, Y., Zhang, X., He, S., He, P., ... & Zhang, D. (2024, February). UniLog: Automatic Logging via LLM and In-Context Learning. In Proceedings of the 46th IEEE/ACM International Conference on Software Engineering (pp. 1-12).

[3] Khudyakov, D. A., & Yakhyaeva, G. E. (2024, June). Unsupervised Anomaly Detection on Distributed Log Tracing through Deep Learning. In 2024 IEEE 25th International Conference of Young Professionals in Electron Devices and Materials (EDM) (pp. 1830-1833). IEEE.

[4] Efeoğlu, E., & Tuna, G. (2024). Classification of Firewall Log Files with Different Algorithms and Performance Analysis of These Algorithms. Journal of Web Engineering, 23(4), 561-593.

[5] Zhu, Y., Min, G., Wu, Y., & Wang, H. (2023, December). Automatic Anomaly Detection Using Unlabelled Log Data. In 2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys) (pp. 507-514). IEEE.

[6] Chen, Y., Xie, H., Ma, M., Kang, Y., Gao, X., Shi, L., ... & Xu, T. (2024, April). Automatic root cause analysis via large language models for cloud incidents. In Proceedings of the Nineteenth European Conference on Computer Systems (pp. 674-688).

[7] Hussein, S. A., & Répás, S. R. (2024). Anomaly Detection in Log Files Based on Machine Learning Techniques. Journal of Electrical Systems, 20(3s), 1299-1311.

[8] Ihalage, A., Taheri, S. M., Muhammad, F., & Al-Raweshidy, H. (2024). Convolutional vs Large Language Models for Software Log Classification in Edge-Deployable Cellular Network Testing. arXiv preprint arXiv:2407.03759.

[9] Xie, Y., Pan, Z., Ma, J., Jie, L., & Mei, Q. (2023, April). A prompt log analysis of text-to-image generation systems. In Proceedings of the ACM Web Conference 2023 (pp. 3892-3902).

[10] Lee, Y., Kim, J., & Kang, P. (2023). Lanobert: System log anomaly detection based on bert masked language model. Applied Soft Computing, 146, 110689.
[11] Liu, Y., Tao, S., Meng, W., Wang, J., Ma, W., Chen, Y., ... & Jiang, Y. (2024, April). Interpretable online log analysis using large language models with prompt strategies. In Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension (pp. 35-46).
[12] Liao, L., Zhu, K., Luo, J., & Cai, J. (2023). LogBASA: Log Anomaly Detection Based on System Behavior Analysis and Global Semantic Awareness. International Journal of Intelligent Systems, 2023(1), 3777826.
[13] Huo, Y., Su, Y., Lee, C., & Lyu, M. R. (2023, May). Semparser: A semantic parser for log analytics. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 881-893). IEEE.
[14] Chang, Y., Luktarhan, N., Liu, J., & Chen, Q. (2023). ETCNLog: A System Log Anomaly Detection Method Based on Efficient Channel Attention and Temporal Convolutional Network. Electronics, 12(8), 1877.
[15] Alaca, Y., Basaran, E., & Çelik, Y. (2024). Enhancing Anomaly Detection in Large-Scale Log Data Using Machine Learning: A Comparative Study of SVM and KNN Algorithms with HDFS Dataset. ADBA Computer Science, 1(1), 14-18.
[16] Mudgal, P., & Wouhaybi, R. (2023, August). An assessment of ChatGPT on log data. In International Conference on AI-generated Content (pp. 148-169). Singapore: Springer Nature Singapore.
[17] Karlsen, E., Luo, X., Zincir-Heywood, N., & Heywood, M. (2024). Benchmarking Large Language Models for Log Analysis, Security, and Interpretation. Journal of Network and Systems Management, 32(3), 59.
[18] Li, Y., Huo, Y., Jiang, Z., Zhong, R., He, P., Su, Y., ... & Lyu, M. R. (2023). Exploring the effectiveness of llms in automated logging generation: An empirical study. arXiv preprint arXiv:2307.05950.
[19] Balasubramanian, P., Seby, J., & Kostakos, P. (2023, December). Transformer-based llms in cybersecurity: An in-depth study on log anomaly detection and conversational defense mechanisms. In 2023 IEEE International Conference on Big Data (BigData) (pp. 3590-3599). IEEE.
[20] Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., ... & Lyu, M. R. (2024). Lilac: Log parsing using llms with adaptive parsing cache. Proceedings of the ACM on Software Engineering, 1(FSE), 137-160.
[21] Jiang, Z., Liu, J., Chen, Z., Li, Y., Huang, J., Huo, Y., ... & Lyu, M. R. (2023). Llmparser: A llm-based log parsing framework. arXiv preprint arXiv:2310.01796.
[22] Cui, T., Ma, S., Chen, Z., Xiao, T., Tao, S., Liu, Y., ... & Pei, D. (2024). Logeval: A comprehensive benchmark suite for large language models in log analysis. arXiv preprint arXiv:2407.01896.
[23] Demiralp, Ç., Wenz, F., Chen, P. B., Kayali, M., Tatbul, N., & Stonebraker, M. (2024). Making LLMs Work for Enterprise Data Tasks. arXiv preprint arXiv:2407.20256.
[24] Xiao, Y., Le, V. H., & Zhang, H. (2024). Stronger, Faster, and Cheaper Log Parsing with LLMs. arXiv preprint arXiv:2406.06156.
[25] Xu, A., & Gau, A. (2024). HELP: Hierarchical Embeddings-based Log Parsing. arXiv preprint arXiv:2408.08300.
[26] Pham, T. A., & Lee, J. H. (2024). Lightweight Multi-task Learning Method for System Log Anomaly Detection. IEEE Access.
[27] Ma, Z., Chen, A. R., Kim, D. J., Chen, T. H., & Wang, S. (2024, April). Llmparser: An exploratory study on using large language models for log parsing. In Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (pp. 1-13).
[28] Habibur Rahman, M., Islam, T., Masum Rana, M., Tasnim, R., Rahman Mona, T., & Mamun Sakib, M. (2023). Machine Learning Approach on Multiclass Classification of Internet Firewall Log Files. arXiv e-prints, arXiv-2306.
[29] Mäntylä, M. V., Wang, Y., & Nyyssölä, J. (2024, March). LogLead-Fast and Integrated Log Loader, Enhancer, and Anomaly Detector. In 2024 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER) (pp. 395-399). IEEE.
[30] Le, V. H., & Zhang, H. (2023, May). Log parsing with prompt-based few-shot learning. In 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE) (pp. 2438-2449). IEEE.
[31] Zhu, J., He, S., He, P., Liu, J., & Lyu, M. R. (2023, October). Loghub: A large collection of system log datasets for ai-driven log analytics. In 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE) (pp. 355-366). IEEE.