Reliable Online Log Parsing using Large Language Models with Retrieval-Augmented Generation

Hansae Ju
Major in Bio Artificial Intelligence
Dept. of Applied Artificial Intelligence
Hanyang University, Ansan, Republic of Korea
sparky@hanyang.ac.kr

***Abstract*—Software logs play a crucial role in the development and maintenance of large-scale software systems, recording essential runtime information. Log parsing, which converts log messages into structured templates, is fundamental for various automated analysis tasks such as anomaly detection and root cause analysis. Unlike traditional log parsing methods that struggle with accurately identifying variables, recent Large Language Models (LLMs)-based approaches leverage their in-context learning capabilities to achieve more precise variable identification. Nonetheless, LLMs require significant computational resources and often struggle with unseen data, rasing concerns about their reliability in real-world online scenarios. This study introduces a novel approach inspired by Retrieval-Augmented Generation to enhance the reliability and efficiency of LLM-based log parsing. The proposed method involves simple validation of generated templates through regex matching, incorporating contextually similar examples from a continuously updated template database into the LLM prompt. Experimental results demonstrate significant improvements in parsing performance, robustness to unseen data, and cost efficiency.**

***Index Terms*—log parsing, large language models, reliability**

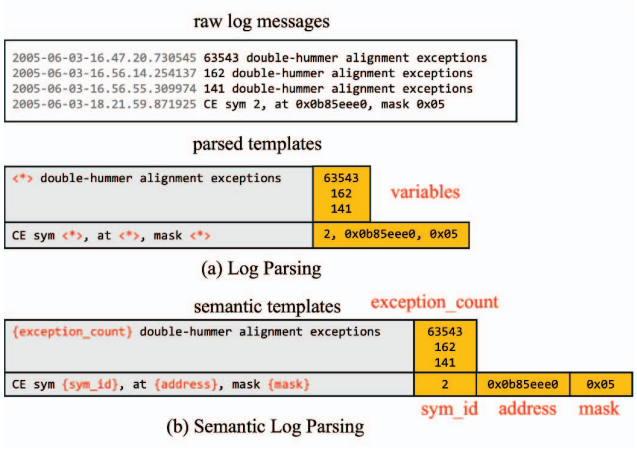

Fig. 1. An Example of Log Parsing and Semantic Log Parsing from BGL

I. Introduction

In modern large-scale software systems, where terabytes of logs are generated daily, automated log analysis is essential for system reliability [1]. Tasks such as anomaly detection, bug localization, and log compression depend on effective log parsing, which structures log messages into templates. Logs contain both structured text (e.g., timestamps, log levels) that can be identified through regular expression (regex) pattern matching, and unstructured text, including developer-written descriptions and dynamic runtime variables. The primary goal of log parsing is to separate static text from variable values, transforming raw log data into event templates. However, traditional approaches such as frequent pattern mining [2], clustering [3], and parsing trees [4] often struggle to accurately identify variables in complex log data.

Moreover, there is a growing need for semantic log parsing, which not only identifies variable positions but also understands and generates meaningful variable names. This deeper level of understanding can significantly enhance downstream tasks such as anomaly detection and root cause analysis [5], [6]. As shown in Fig. 1, semantic log parsing provides a more comprehensive interpretation of log events by distinguishing between static and dynamic components, in contrast to traditional log parsing methods that focus solely on identifying variable positions.

The recent emergence of Large Language Models (LLMs) with advanced contextual understanding presents new opportunities for log parsing [7]–[9]. While LLMs show promise in handling complex and unstructured log data, they still struggle with unseen templates and high computational costs, which limit their effectiveness in practice [8]. Despite their strengths in understanding context and generating text and code [10], LLMs have not yet been fully explored for semantic log parsing [11]. These challenges are particularly significant in online log parsing, leading to reliability issues that hinder their widespread adoption in the industry.

This study proposes an approach inspired by Retrieval-Augmented Generation (RAG) [12] for enabling reliable semantic log parsing with LLM parsers, even in online scenarios. The proposed method validates templates generated through regex matching and injects examples similar to the input logs from a matched template database into the prompt. This approach can be applied to any LLM with emergent capabilities, enhancing robustness against unseen data while significantly reducing the high cost of LLM calls. Additionally, it ensures the parsing of templates with consistent variable

2994-810X/24/$31.00 ©2024 IEEE
DOI 10.1109/ISSREW63542.2024.00055

semantics across log messages. To systematically evaluate the effectiveness of the proposed approach, we address the following research questions:

- RQ1: Do regex-validated examples similar to the input logs improve log parsing performance?
- RQ2: How robust is the proposed approach to unseen data compared to existing methods?
- RQ3: How much does the proposed approach improve cost efficiency?

Experimental results indicate that the proposed approach: (1) significantly enhances few-shot in-context learning ability with similar regex-validated examples; (2) demonstrates robustness on unseen data; and (3) achieves log parsing at less than 10% of the cost compared to existing methods. These findings support the potential for our approach to improve log parsing efficiency and accuracy, paving the way for scalable real-time log analysis in practice.

II. Related Works

A. Semantic Log Parsing

Recent advances in semantic log parsing have highlighted the importance of dynamic variables in log data. For instance, VALB [5] conducted a manual analysis of log datasets, identifying 10 distinct variable categories and introducing an LSTM-based framework to classify these categories at the word level. Similarly, SemParser [13] manually labeled the concepts associated with each dynamic value in the logs and proposed a Semantic Miner combining an LSTM model with a Joint Parser to utilize (concept, instance) pairs.

However, these methods rely heavily on manual labeling and do not fully address the challenges of online scenarios, where log data is continuously generated and evolves over time. In contrast, our study leverages LLMs for semantic template labeling, aiming to reduce manual intervention while ensuring that variable names are generated contextually and meaningfully.

B. LLMs for Log Parsing

Le and Zhang [7] developed LogPPT, a log parser based on a masked language model, which transforms log parsing into a token classification task, categorizing tokens as either variables or static text.While this approach improved accuracy over traditional algorithms, it required an offline setting where the model is trained on labeled datasets. They also explored using ChatGPT for log parsing, but it demonstrated lower accuracy than LogPPT [11]. The authors noted limitations of closed-source models, such as high costs, lack of fine-tuning, stability issues, and data privacy concerns.

More recently, LLMParser [8] demonstrated that fine-tuning open-source LLMs can lead to high performance in log parsing. However, this approach struggles with unseen template data, highlighting a challenge in model generalizability.

LogPrompt [6] investigates advanced prompting strategies to enhance LLM performance in log parsing and anomaly detection, but its robustness on unseen data remains unproven. LogDiv [9] uses k-nearest neighbors (kNN) for few-shot prompting but depends on a large labeled log set and hasn't been validated on unseen templates.

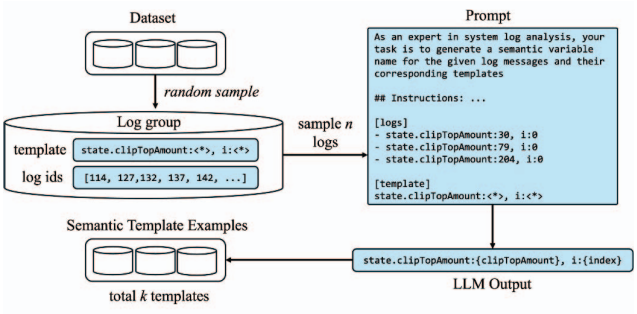

Fig. 2. Semantic Template Labeling via Prompting

While these studies make significant strides in applying LLMs to log parsing, they do not fully address key issues such as semantic log parsing, robustness to unseen data, and cost efficiency. This paper proposes a novel approach that leverages LLMs with vector search techniques to improve semantic understanding, enhance robustness, and reduce computational costs in log parsing.

III. Proposed Method

A. Semantic Template Labeling

Currently, there are no available semantic template datasets, and creating such datasets requires extensive manual labeling and domain knowledge. This study uses high-performance closed-source LLMs to generate semantic templates from ground-truth templates, as shown in Fig. 2.

For each iteration, a single template is randomly selected from the dataset. This process is repeated k times, where k corresponds to the number of shot examples used in the log parsing prompt. In each iteration, to ensure diversity, the n most diverse logs—determined by the greatest cosine distance between log embeddings—are selected as input logs for the labeling prompt. These n logs and their corresponding template are then provided to the model with specific instructions. The algorithm for selecting diverse logs effectively reduces model confusion by ensuring that if all selected n logs in an iteration are identical, the template contains only static text; otherwise, it contains more than one variable. Note that this semantic template labeling process requires only k seen templates across the k iterations, meaning that all other logs correspond to unseen templates. In this experiment, we fixed $k = 4$ for 4-shot in-context learning and set $n = k$.

B. RAG-based Online Log Parsing

Fig. 3 provides an overview of the proposed RAG-based online log parsing pipeline. In this pipeline, all logs are converted into embedding vectors based on their similarity, with these vectors representing the meaning of the text as dense vectors. The process begins with the input log retrieving the most similar log groups from a matched log group vector database. Each log group contains multiple log message vectors for

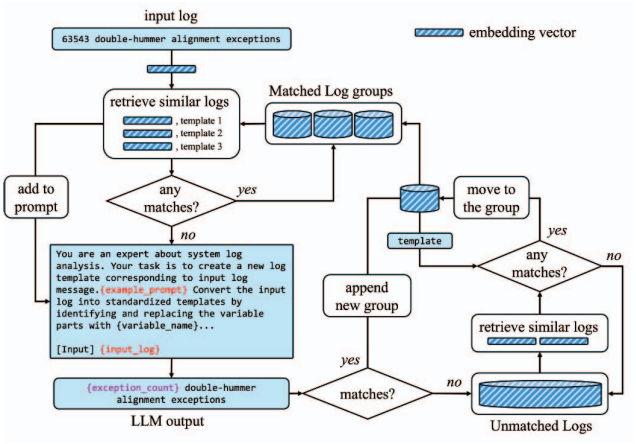

Fig. 3. Overview of Proposed RAG-based Online Log Parsing Pipeline

a single template, and the template with the highest cosine similarity to the input vector is selected.

A template is considered valid if it matches the input log when converted to a regex, and it is then directly added to the matched group. In online scenarios where a golden label is unavailable, additional manual effort is required to verify the validity of templates. However, since the LLM previously generated a template considered valid based on the most similar log to the input log, there is a high likelihood that this input log will generate the same template. Thus, if the template matches, an early match is made.

If no matching template exists, a series of similar log-template pairs (i.e., examples) obtained earlier are added to the prompt to perform few-shot in-context learning. Unlike typical few-shot prompting that uses fixed examples, retrieval augmentation provides the most similar examples to the model. Note that, unlike previous research that selected similar examples from a fixed set of candidates [9], this method selects examples from the matched log groups that are gradually built in an online setting. The generated template is then verified against the input log, and if a match is found, a new log group is created. Subsequently, similar logs from unmatched logs are retrieved to recheck if any logs match the new template, ensuring that logs that initially failed parsing due to insufficient examples are given another opportunity for matching.

IV. Research Design

To address RQ1, we compared performance by adjusting the ratio of fixed examples to matched log template examples obtained through similarity search. We used a base setting of 4-shot prompting and varied the maximum number of retrievals (MR) to 0, 1, 2, and 4 for comparison. For example, when MR is 0, all examples are fixed initial examples; when MR is greater than 0, the most similar templates are included up to the specified MR, with fixed initial examples filling the remainder, always using a total of 4 examples.

To address RQ2, we set the baseline as LLMParser [8]. LLMParser samples 50 templates during training to fine-tune open-source LLMs, testing an average of 103 templates and 201 logs. In comparison, our approach uses only 4 fixed examples, amounting to 66% and 11% of the data used by LLMParser, respectively. Additionally, while LLMParser focuses on generating templates, our study involves the more complex task of generating semantic templates. Despite this disadvantage in direct comparison, we chose $LLMParser_{Llama}$, the highest-performing LLM-based parser. The dataset used was from 7 projects in Loghub [14], [15], tested on unseen logs by LLMParser. To address RQ3, we recorded the number of LLM API calls and tokens used.

Performance was evaluated using two metrics: Grouping Accuracy (GA) [16] and Parsing Accuracy (PA) [17]. GA measures the proportion of correctly grouped log messages, while PA assesses the accuracy of token classification within log messages. Both metrics provide insights into the log parser's overall effectiveness and precision.

A. Implementation Details

For proactive experimentation, we used OpenAI's latest and highly efficient model, gpt4o-mini[1]. Since relying on third-party APIs can cause reliability issues in the industry, achieving similar performance with open-source models is a top priority for future research. The embedding model used was bge-small-en [18], an open-source model that embeds sentence-level text into 384-dimensional vectors, as only log messages are converted to embedding vectors. To efficiently manage these vectors, we used Qdrant[2], a Rust-based vector database that supports snapshots and vector similarity search.

B. Experimental Results

Table IV presents the performance indicators by project and MR count, with the highest performance in each project highlighted in bold.

For RQ1, in most projects, GA improved by an average of 36.2% and PA by 58.7% when MR was 4 compared to 0. Additionally, there is a general trend of performance improvement as MR increases; however, slight increases in GA and decreases in PA at MR 1 and 2 suggest that further experiments are needed to confirm the linear relationship between performance and MR. Overall, the results indicate that using similar regex-validated examples significantly improves performance compared to using only fixed examples.

For RQ2, except for the Linux project, which exhibited negligible average PA differences and very low performance, most projects showed better performance. Despite our experiments involving a more complex semantic log parsing task with 9 times more unseen data than the baseline, our approach demonstrated equal or better performance, confirming its robustness to unseen data.

For RQ3, compared to existing approaches that perform LLM calls for every log message, our method reduced the

[1]https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence
[2]https://qdrant.tech

TABLE I
COMPARISONS WITH DIFFERENT RETRIEVALS WITHIN 4-SHOTS PROMPTING IN ONLINE SCENARIO

Project	Unseen Template	Unseen Log	No Retrievals			Max 1 Retrievals			Max 2 Retrievals			Max 4 Retrievals			Baseline
			GA	PA	LC	GA	PA	LC	GA	PA	LC	GA	PA	LC	PA
Android	154	1981	0.765	0.467	85	0.957	0.642	158	**0.964**	0.608	150	0.958	**0.696**	154	0.701
BGL	116	1269	0.295	0.269	37	0.887	0.704	107	0.759	0.445	117	**0.963**	**0.920**	112	0.771
Hadoop	110	1983	0.387	0.116	69	**0.957**	0.372	151	0.624	0.371	202	**0.957**	**0.612**	113	0.516
HealthApp	71	1704	0.881	0.701	63	0.904	0.718	71	0.902	0.732	68	**0.999**	**0.829**	79	0.800
Linux	112	1874	0.605	0.100	27	0.729	0.136	107	0.703	0.144	332	**0.734**	**0.159**	117	0.818
Mac	337	1968	0.730	0.295	432	0.242	0.032	167	0.711	**0.399**	429	**0.735**	0.396	376	0.376
Thunderbird	145	1996	**0.958**	0.902	135	0.927	0.852	120	0.955	0.466	143	0.948	**0.911**	174	0.573
Average	149	1825	0.660	0.407	121	0.800	0.498	125	0.807	0.452	191	**0.899**	**0.646**	160	0.651

number of LLM calls (LC) by 91.3% to 93.4%. The total cost of API calls for all experiments was approximately $1.

V. CONCLUSION AND FUTURE WORKS

This paper introduced a novel approach to improving LLM-based log parsers by addressing key challenges such as high computational costs, limited semantic awareness, and low robustness to unseen data. By leveraging semantic template labeling, regex validation, and similarity search, we developed an efficient online log parsing pipeline. Our experimental results demonstrated significant enhancements in parsing performance, robustness to unseen data, and cost efficiency, validating the effectiveness of our approach.

In future work, we plan to expand this research by experimenting with open-source LLM models and applying the methodology to a broader range of datasets. Additionally, we aim to explore the parallelization of log parsing processes and the handling of large-scale log data to improve the system's applicability in real-world scenarios. Ensuring consistent variable semantics across diverse environments will also be a critical focus, contributing to greater scalability and generalizability of LLM-based log parsing systems.

ACKNOWLEDGMENT

I would like to thank Prof. Scott Uk-Jin Lee for his invaluable guidance and supervision throughout my Ph.D. research. This research was partly supported by the MSIT(Ministry of Science and ICT), Korea, under the Convergence security core talent training business support program(IITP-2024-RS-2024-00423071) supervised by the IITP(Institute of Information & Communications Technology Planning & Evaluation) and the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2023R1A2C1006390).

REFERENCES

[1] J. Liu, J. Zhu, S. He, P. He, Z. Zheng, and M. R. Lyu, "Logzip: Extracting Hidden Structures via Iterative Clustering for Log Compression," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. San Diego, CA, USA: IEEE, Nov. 2019, pp. 863–873.

[2] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, "Logram: Efficient Log Parsing Using nn-Gram Dictionaries," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 879–892, Mar. 2022.

[3] K. Shima, "Length Matters: Clustering System Log Messages using Length of Words," Nov. 2016.

[4] P. He, J. Zhu, Z. Zheng, and M. R. Lyu, "Drain: An Online Log Parsing Approach with Fixed Depth Tree," in *2017 IEEE International Conference on Web Services (ICWS)*, Jun. 2017, pp. 33–40.

[5] Z. Li, C. Luo, T.-H. Chen, W. Shang, S. He, Q. Lin, and D. Zhang, "Did We Miss Something Important? Studying and Exploring Variable-Aware Log Abstraction," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. Melbourne, Australia: IEEE, May 2023, pp. 830–842.

[6] Y. Liu, S. Tao, W. Meng, J. Wang, W. Ma, Y. Chen, Y. Zhao, H. Yang, and Y. Jiang, "Interpretable Online Log Analysis Using Large Language Models with Prompt Strategies," in *Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension*. Lisbon Portugal: ACM, Apr. 2024, pp. 35–46.

[7] V.-H. Le and H. Zhang, "Log Parsing with Prompt-based Few-shot Learning," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. Melbourne, Australia: IEEE, May 2023, pp. 2438–2449.

[8] Z. Ma, A. R. Chen, D. J. Kim, T.-H. Chen, and S. Wang, "LLMParser: An Exploratory Study on Using Large Language Models for Log Parsing," in *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, Apr. 2024, pp. 1–13.

[9] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Prompting for Automatic Log Template Extraction," Jul. 2023.

[10] X. Liu, Y. Zheng, Z. Du, M. Ding, Y. Qian, Z. Yang, and J. Tang, "GPT Understands, Too," *arXiv*, Mar. 2021.

[11] V.-H. Le and H. Zhang, "Log Parsing: How Far Can ChatGPT Go?" Aug. 2023.

[12] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-t. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Apr. 2021.

[13] Y. Huo, Y. Su, C. Lee, and M. R. Lyu, "SemParser: A Semantic Parser for Log Analytics," in *2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE)*. Melbourne, Australia: IEEE, May 2023, pp. 881–893.

[14] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A Large Collection of System Log Datasets towards Automated Log Analytics," Aug. 2020.

[15] Z. A. Khan, D. Shin, D. Bianculli, and L. Briand, "Guidelines for assessing the accuracy of log message template identification techniques," in *Proceedings of the 44th International Conference on Software Engineering*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, Jul. 2022, pp. 1095–1106.

[16] J. Zhu, S. He, J. Liu, P. He, Q. Xie, Z. Zheng, and M. R. Lyu, "Tools and Benchmarks for Automated Log Parsing," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Montreal, QC, Canada: IEEE, May 2019, pp. 121–130.

[17] Y. Liu, X. Zhang, S. He, H. Zhang, L. Li, Y. Kang, Y. Xu, M. Ma, Q. Lin, Y. Dang, S. Rajmohan, and D. Zhang, "UniParser: A Unified Log Parser for Heterogeneous Log Data," in *Proceedings of the ACM Web Conference 2022*, ser. WWW '22. New York, NY, USA: Association for Computing Machinery, 2022, pp. 1893–1901.

[18] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-Pack: Packaged Resources To Advance General Chinese Embedding," 2023.