Enhancing Log Abstraction with Semantic Variable Naming via Large Language Models

Hansae Ju [1] , Joonwoo Lee [1], Gichan Lee [2] and Scott Uk-Jin Lee [2*]
[1] Major in Bio Artificial Intelligence, Dept. of Applied Artificial Intelligence, Hanyang University, Ansan, Korea
[2] Dept. of Computer Science & Engineering, Hanyang University, Ansan, Korea
Email: {*sparky, joonywlee, fantasyopy, scottlee*}*@hanyang.ac.kr*

***Abstract*— Log abstraction is a crucial process in log analysis, aiming to extract dynamic variables and create log templates composed of static words. However, recent studies underline the significance of understanding the semantics of dynamic variables. A major limitation in contemporary semantic log abstraction research is the manual labeling of dynamic variables' semantics. To overcome this challenge, we introduce a novel framework that capitalizes on developers' inherent habit of assigning meanings to variables during code writing. In our framework, variable names within logging statements serve as semantic labels for dynamic variables. Given the unavailability of the source code for a system under analysis, our framework transforms log templates into Python logging statements and uses them as prompts for Large Language Models (LLM) to generate semantically meaningful variable names. This approach greatly improves the effectiveness of semantic log analysis and contributes significantly to the progress of automated log analysis.**

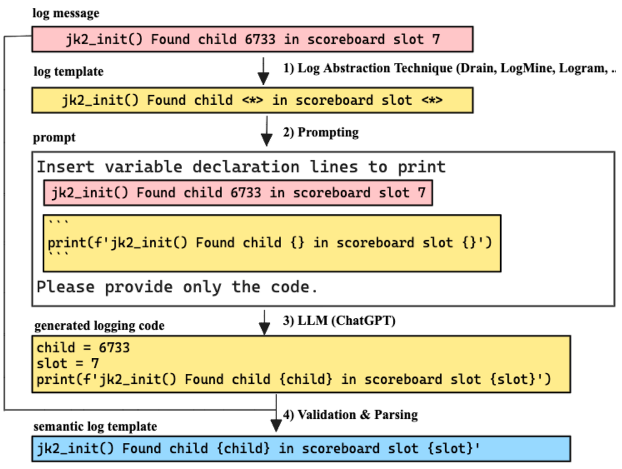
Keywords*— Log Abstraction, Large Language Models, Log Analysis

I. Introduction

Logging is a critical process in software maintenance and debugging, where developers record messages that describe specific events that occur during a program's execution. These logs, comprising a combination of static words and dynamic variables, are generated in large volumes, thereby necessitating their organization and abstraction for more efficient analysis [1].

Log abstraction refers to the process of abstracting dynamic variables from unstructured log messages to generate log templates composed of static words, which serve as a more organized and structured format for log analysis. This step is essential for many automated log analysis tasks, such as anomaly detection, log clustering, and root cause analysis [2]. Recent studies emphasize the importance of semantics in log abstraction, arguing that understanding the meaning of dynamic variables is essential to analyzing logs more effectively.

However, recent studies require manual labeling semantics of dynamic, which is not only laborious but also inconsistent across different studies. To address this issue, we propose a novel framework that leverages the inherent trait of developers to assign semantic meaning to variables when writing code. This framework treats the variable names within logging statements as semantic labels for dynamic variables, thereby providing a source of semantically meaningful labels without requiring manual effort. Typically, the source code of a system under log analysis is unobtainable, but our approach converts log templates parsed through existing log abstraction



```
jk2_init() Found child 6733 in scoreboard slot 7
```

```
jk2_init() Found child <*> in scoreboard slot <*>
```

```
Insert variable declaration lines to print
jk2_init() Found child 6733 in scoreboard slot 7
'''
print(f'jk2_init() Found child {} in scoreboard slot {}')
'''
Please provide only the code.
```

```
child = 6733
slot = 7
print(f'jk2_init() Found child {child} in scoreboard slot {slot}')
```

```
jk2_init() Found child {child} in scoreboard slot {slot}'
```

Fig. 1. Illustrative example of our apporach using logs from Apache

techniques into Python logging code and uses LLM to generate appropriate variable names at the variable positions in the converted logging code. Thanks to the high understanding and generation ability of LLM for natural language and code, we can obtain natural and semantic variable names considering the context of log templates and actual log messages.

II. Methodology

Our method involves using a Large Language Model (LLM), specifically ChatGPT, and a specialized prompting strategy[3]. The underlying strength of the LLM is its exceptional ability to understand and generate both natural language and code[4], an attribute we leverage to derive contextually relevant and semantically meaningful variable names.

The process of our method is illustrated in Fig. 1. In the initial step of our method, we apply a Log Abstraction Technique (such as Drain[5], LogMine[6], Logram[1], etc.) to an example log sample from the Apache system. For instance, the log sample `jk2_init() Found child 6733 in scoreboard slot 7` is abstracted into a log template `jk2_init() Found child <*> in scoreboard slot <*>`. Following this, our prompting technique is used to generate a prompt. In the prompting stage, our framework converts the derived log template into a python f-string formatted logging statement, leveraging high capability of LLM about understanding and generation of code. The well-structed prompt is then fed into the LLM, which generates a logging code snippet that declares the variables with semantic names. In the final step, the framework executes the generated code and checks if the output log message is identical to the input log message. Once the output is validated, the code is parsed to retrieve final semantic log template `jk2_init() Found child {child} in scoreboard slot {slot}`.

In the experiments of this study, we implemented one-shot semantic log abstraction, utilizing a single log message and a log template derived from log abstraction tools as inputs for the prompt. However, this prompting strategy can be adjusted according to the needs of practitioners, leading to more efficient or effective outcomes. For instance, if a practitioner wishes to perform semantic log abstraction solely with log messages, bypassing the use of existing log abstraction tools, the prompt could be tailored to ask the LLM to generate Python code in f-string format that can print the specific log.

In cases where a more sophisticated variable naming is required for a log template, several corresponding log messages (e.g., three) could be sampled and added to the prompt, or domain knowledge of the system could be incorporated. Essentially, our methodology offers a high level of flexibility to meet the diverse needs of practitioners, thanks to its adjustable prompting strategy.

III. Related Work

There is a considerable amount of previous research focused on log abstraction. Techniques based on frequent pattern mining, such as Logram[1], leverage the observation that static words occur more frequently than dynamic parameters. A study conducted by Yu et al [7] harnessed the self-attention scores of BERT for each token, taking advantage of the high frequency of static words. Several studies applied clustering algorithms with the principle that similar logs are likely to correspond to the same template [6]. Some other research employed heuristic-based parsing rules in conjunction with techniques like fixed depth parse trees, as demonstrated in Drain [5]. However, these previous studies are primarily centered around distinguishing between static and dynamic elements, with little to no attention given to the semantics of the variables.

More recently, research efforts have started acknowledging the importance of the semantics of dynamic variables in log abstraction. A study, VALB [2], manually analyzed the log dataset to understand the characteristics of dynamic variables and delineated ten different categories. They introduced an LSTM-based framework that utilizes a Named Entity Recognition scheme to categorize each word in the log. The study emphasized that the dynamic values embedded in the log offer valuable insights for interpreting and analyzing the log data.

SemParser [8], on the other hand, manually labeled the concepts of each dynamic value or instance in the log, and proposed a Semantic Miner, an LSTM-based tool, to predict the concepts at the word level. The authors also introduced a Joint Parser that uses a domain knowledge repository to identify instances not only within individual logs but also across multiple logs, based on the pairs of (concept, instance).

Our method distinguishes itself by leveraging Large Language Models (LLMs) for semantic log abstraction, reducing the manual effort required. Unlike previous studies focusing on static and dynamic elements separation or manual categorization, our method leverages the LLM's understanding and generation capacity for both natural language and code. We transform log abstraction into a code generation task, exploiting developers' habit of assigning meaningful names to variables. The LLM generates semantically meaningful variable names from log templates as Python logging statements, thereby streamlining the semantic log abstraction process. This innovative approach sets our research apart in the realm of automated log analysis.

IV. Conclusion and Future Work

Our research introduced a unique approach for semantic log abstraction using Large Language Models (LLMs). By capitalizing on developers' tendency to assign meaningful names to variables during coding, we significantly reduce manual effort in semantic log analysis. This paper presented our method with examples and scenarios illustrating its flexibility and capacity to generate meaningful variable names.

Future work will focus on refining our model's ability to interpret context and exploring ways to incorporate diverse types of knowledge into the prompting process. We also aim to test our method across different logs and domains to examine its generalizability.

In conclusion, our research offers a promising pathway to automated semantic log abstraction. As we continue to develop this work, we expect our approach to significantly advance automated log analysis and related applications.

Acknowledgment

This work was supported by the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT) (NRF-2023R1A2C1006390).

References

[1] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, "Logram: Efficient Log Parsing Using nn-Gram Dictionaries," *IEEE Trans. Softw. Eng.*, vol. 48, no. 3, pp. 879–892, Mar. 2022, doi: 10.1109/TSE.2020.3007554.

[2] Z. Li *et al.*, "Did We Miss Something Important? Studying and Exploring Variable-Aware Log Abstraction." in ICSE '23. arXiv, Apr. 22, 2023. Accessed: May 01, 2023. [Online]. Available: http://arxiv.org/abs/2304.11391

[3] L. Ouyang *et al.*, "Training language models to follow instructions with human feedback." arXiv, Mar. 04, 2022. Accessed: Jul. 01, 2023. [Online]. Available: http://arxiv.org/abs/2203.02155

[4] T. B. Brown *et al.*, "Language Models are Few-Shot Learners," *NeurIPS*, Jul. 2020, Accessed: Mar. 07, 2022. [Online]. Available: http://arxiv.org/abs/2005.14165

[5] 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. doi: 10.1109/ICWS.2017.13.

[6] H. Hamooni, B. Debnath, J. Xu, H. Zhang, G. Jiang, and A. Mueen, "LogMine: Fast Pattern Recognition for Log Analytics," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, Indianapolis Indiana USA: ACM, Oct. 2016, pp. 1573–1582. doi: 10.1145/2983323.2983358.

[7] S. Yu, N. Chen, Y. Wu, and W. Dou, "Self-supervised log parsing using semantic contribution difference," *J. Syst. Softw.*, vol. 200, p. 111646, Jun. 2023, doi: 10.1016/j.jss.2023.111646.

[8] Y. Huo, Y. Su, C. Lee, and M. R. Lyu, "SemParser: A Semantic Parser for Log Analysis." in ICSE '23. arXiv, Feb. 05, 2023. Accessed: Apr. 21, 2023. [Online]. Available: http://arxiv.org/abs/2112.12636