2024 IEEE/ACM 2nd International Workshop on Interpretability, Robustness, and Benchmarking in Neural Software Engineering (InteNSE)

An Exploratory Study on How Non-Determinism in Large Language Models Affects Log Parsing

Merve Astekin
merve@simula.no
Simula Research Laboratory
Oslo, Norway

Max Hort
maxh@simula.no
Simula Research Laboratory
Oslo, Norway

Leon Moonen
leon.moonen@computer.org
Simula Research Laboratory &
BI Norwegian Business School
Oslo, Norway

ABSTRACT

Most software systems used in production generate system logs that provide a rich source of information about the status and execution behavior of the system. These logs are commonly used to ensure the reliability and maintainability of software systems. The first step toward automated log analysis is generally *log parsing*, which aims to transform unstructured log messages into structured log templates and extract the corresponding parameters.

Recently, Large Language Models (LLMs) such as ChatGPT have shown promising results on a wide range of software engineering tasks, including log parsing. However, the extent to which non-determinism influences log parsing using LLMs remains unclear. In particular, it is important to investigate whether LLMs behave consistently when faced with the same log message multiple times.

In this study, we investigate the impact of non-determinism in state-of-the-art LLMs while performing log parsing. Specifically, we select six LLMs, including both paid proprietary and free-to-use models, and evaluate their non-determinism on 16 system logs obtained from a selection of mature open-source projects. The results of our study reveal varying degrees of non-determinism among models. Moreover, they show that there is no guarantee for deterministic results even with a temperature of zero.

CCS CONCEPTS

• **Software and its engineering → Consistency**.

KEYWORDS

log parsing, large language model, robustness, non-determinism, consistency

ACM Reference Format:
Merve Astekin, Max Hort, and Leon Moonen. 2024. An Exploratory Study on How Non-Determinism in Large Language Models Affects Log Parsing. In *2024 ACM/IEEE 2nd International Workshop on Interpretability, Robustness, and Benchmarking in Neural Software Engineering (InteNSE '24), April 15, 2024, Lisbon, Portugal.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3643661.3643952

InteNSE '24, April 15, 2024, Lisbon, Portugal

ACM ISBN 979-8-4007-0564-9/24/04.
https://doi.org/10.1145/3643661.3643952

1 INTRODUCTION

The increased complexity of today's software-intensive systems makes it impossible to anticipate all possible issues using traditional Verification&Validation (V&V) techniques. Unanticipated faults that can remain after conventional V&V can be detected by automated analysis of log data generated by software systems during execution. Log data contain textual and numeric information regarding the status of system components and various events that occur at runtime. This information can help developers and operators understand system execution and perform fault detection, prediction, and diagnosis activities.

Log parsing is the first step of log analysis to gain more insight into the run-time behavior of software systems before going into further analysis. Through this process, unstructured raw log messages are transformed into structured data by extracting corresponding log templates and parameters. An example log parsing workflow is depicted in Figure 1. As the scale increases, software systems can generate huge volumes of log messages from several software components. Thus, it becomes infeasible to parse log data efficiently and effectively by traditional parsing approaches such as manual rule-based parsing.

Over the years, many methods and techniques have been proposed for automatic log parsing [1, 6, 27, 29]. However, heterogeneous and unstructured log data generated by multiple distributed systems cause a generalization issue. It is known that most existing log parsers do not perform well on different datasets [29].

The similarities between software system logs and natural language have led researchers to explore the use of language models for log analysis. Several studies have adopted pre-trained language models (PTMs) for different phases of log analysis, including log parsing [12, 16], and anomaly/fault detection [2, 4]. PTMs require training from scratch or fine-tuning for downstream tasks, which may not be feasible given the limited availability of labeled data and computing resources.

Recent studies exploit Large-Language Models (LLMs) for several software engineering tasks [15], including log parsing [8, 11, 26], and log anomaly detection [13]. While LLMs, such as ChatGPT, have been applied for the task of log parsing and achieved promising results, it is unclear how deterministic these results are (i.e., do we get different results when asking LLMs to extract templates for the same log message repeatedly?). In particular, a recent study by Ouyang et al. [17] showed that ChatGPT produced unstable results for code generation. This threatens the reliability but also validity of results produced by research on log parsing with LLMs. Therefore,

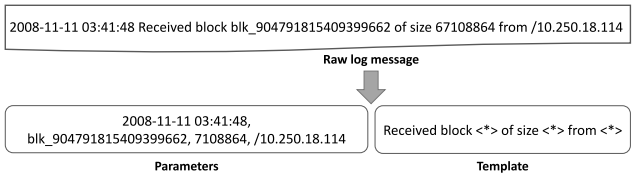

Figure 1: An example workflow of log parsing.

we investigate the following question: **How deterministic are LLMs for log parsing?**

For this purpose, we conduct an empirical study on six LLMs, two prompt types, and two temperatures. Our results show that the determinism of LLMs for log parsing varies based on the LLM type but also on the respective log message for extracting. Reducing temperature increases determinism, but even a temperature of 0 does not guarantee deterministic results.

2 RELATED WORK

Robustness and Non-Determinism: Robustness addresses the ability of machine learning models, in this context LLMs, to behave consistently when dealing with changing situations (e.g., do not change the output if changes to the input are small and do not require modifications) [14]. Hereby, language models are tested with slightly perturbed inputs, which can also be described as adversarial attacks [25]. A robust language model is expected to return similar or identical responses for slightly altered inputs.

To compare the robustness of machine learning models, Rauber et al. [18] proposed Foolbox, a Python package aimed at finding the smallest input perturbation which leads to adversarial samples. Going a step beyond the smallest perturbations, Ouyang et al. [17] investigated the behavior of ChatGPT when faced with an identical prompt repeatedly. This consideration of robustness with regard to identical inputs is also called determinism or non-determinism (e.g., an LLM which returns different responses to identical prompts is non-deterministic). In particular, Ouyang et al. [17] assessed the non-determinism of ChatGPT for generating source code and found variety in the responses, even at a temperature of 0, which is a threat to the validity of results.

In this study, we carry out experiments similar to those of Ouyang et al. [17]. We repeatedly query LLMs with the same prompt to measure response variability under different temperature settings. In contrast to their work, we investigate the non-determinism of language models for the task of log parsing. Furthermore, we consider six LLMs, including two LLMs available through paid APIs and four openly accessible LLMs (see Section 4.1 for more details).

Log Parsing: System logs are unstructured or semi-structured data, consisting of natural language text written by developers and dynamic variables automatically generated during system execution. Since automated log analysis techniques generally require structured data, they need to be transformed into a structured form before being used for further analysis. Although there exist a multitude of approaches for automatic log parsing (e.g., clustering [3, 5, 20], pattern mining [23, 24], heuristic [6, 9]), this section outlines LLM-based methods as these are concerned with the techniques we explore.

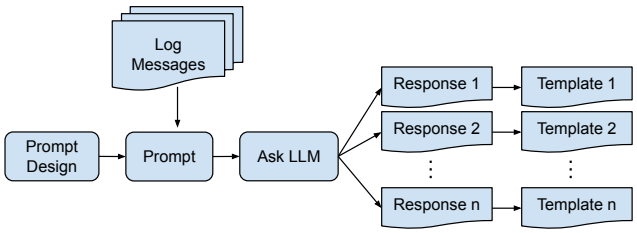

Figure 2: Overview of the experimental method.

Jiang et al. [8] proposed LLMParser, a framework for log parsing. LLMparser queries ChatGPT (gpt-3.5-turbo-0613) with few-shot learning. The respective example logs used for few-shot prompts are selected based on their relevance and similarity to the queried log. Mugdal and Wouhaybi [15] considered seven prompts to evaluate ChatGPT's ability on the tasks of log parsing, log analytics, and log summarization. Other works used prompt tuning to retrain a RoBERTa model for log parsing [12], or studied ChatGPT for log parsing under consideration of different prompting strategies [11].

Unlike existing works, we take non-determinism into account and consider a wider range of LLMs, with two paid models and four free-to-use ones. The majority of log parsing approaches with LLMs were focused on GPT [8, 11, 15]. Claude, CodeLlama, CodeUp, Llama2, and Zephyr (Section 4.1) have not been investigated.

3 METHODOLOGY

Figure 2 shows an overview of the methodology used in this study. We first design a prompt prefix with log parsing instructions in a zero-shot or few-shot setting (Figure 3). Then, we insert a log message into the designed prompt prefix and send this prompt to an LLM to extract the template part of the log message. For a single prompt, we repeat this procedure 50 times such that an LLM generates 50 responses for the same prompt. Subsequently, we extract and analyze the templates from the 50 responses.

Prompt Design: We design the prompts based on two different approaches that have been applied in related works: zero-shot [11, 15], and few-shot [8, 11, 12, 26]. We first construct a prompt prefix, which is a fixed string that contains general information and instructions about the log parsing task. In the zero-shot setting, we use the prompt prefix as is, while in the few-shot setting, we add four examples to the prompt prefix. Both prompt prefixes can be seen in Figure 3. In the figure, the area shaded in green is only included in the few-shot scenarios, while the remaining area is used for both the zero- and few-shot scenarios. The four examples in the area shaded in green are different from the actual log messages that are used as inputs in the experiment. Two of the examples are taken from the experimental dataset [28] (Section 4.2) while another two are from unrelated datasets. The examples are designed as Q&A pairs where each consists of two parts: the log message in the question and the corresponding template as an answer. Furthermore, we add instructions to the prompt to generate customized responses with specific tags to ensure that the LLMs do not produce explanations in their responses about the construction process or the parameters of the template, and to distinguish the extracted template in the responses from the plain text, if present.

Table 1: Overview of selected large language models.

Model	Version/ Release time	Base model	Availability
GPT-3.5	gpt-3.5-turbo-0613/ June, 2023	-	Closed-source
Claude 2	July 2023	-	Closed-source
Llama 2	llama2:7b/ July, 2023	-	Public
CodeLlama	codellama:7b-instruct/ August, 2023	Llama 2	Public
Zephyr Beta	zephyr:7b/ October, 2023	Mistral-7B	Public
CodeUp	codeup:13b/ July, 2023	Llama 2	Public

We instruct the LLMs to print the log template corresponding to the input surrounded by a <TPL> and </TPL> pair. To indicate the log message to be parsed, we use a similar tagging approach in the form of a <MSG> and </MSG> pair.

Prompt: We combine the designed prompt prefix with the log message and construct the final prompt. We insert only one log message to be parsed per prompt.

Ask LLM: We send requests for each prompt and ask LLMs to extract the log template in the defined format. We repeat this step 50 times to obtain 50 predictions from the LLMs with the same prompt. When we send requests to the LLMs, we specify different values of the temperature parameter of the LLMs to control the randomness of the predictions.

Template Extraction: After receiving the responses from the LLMs, we apply template extraction to retrieve the templates from the generated text. We extract the text part surrounded by <TPL> and </TPL> pairs in the response text, and directly use them for the evaluation without making any modifications.

Evaluation: We measure the quality of the extracted templates in two aspects: *correctness* and *determinism*. The *correctness* is evaluated with the number of correctly parsed log messages. A log message is considered correctly parsed only when each token in the log message is correctly recognized as a template or variable based on the ground truth. The *non-determinism* is determined by the number of unique templates extracted by LLMs for an identical log message for the same setting (i.e., temperature, prompt type).

You will be provided with a log message delimited by **<MSG>** and **</MSG>**.
The log texts describe various system events in a software system. A log message usually contains a header that is automatically produced by the logging framework, including information such as timestamp, class, and logging level (INFO, DEBUG, WARN etc.).

The log message typically consists of two parts:
1. Template - message body, that contains constant strings (or keywords) describing the system events;
2. Parameters/Variables - dynamic variables, which reflect specific runtime status.
You must identify and abstract all the dynamic variables in the log message with suitable placeholders inside angle brackets to extract the corresponding template.

You must output the template corresponding to the log message. Print only the input log's template surrounded by **<TPL>** and **</TPL>**. Never print an explanation of how the template is constructed.

Here are a few examples of log messages (labeled with Q:) and corresponding templates (labeled with A:):

Q: <MSG>[*081109 204453 34 INFO dfs.FSNamesystem: BLOCK* NameSystem.addStoredBlock: blockMap updated: 10.250.11.85:50010 is added to blk_2377150260128098806 size 67108864*]**</MSG>**
A: <TPL>[*BLOCK* NameSystem.addStoredBlock: blockMap updated: <*>:<*> is added to <*> size <*>*]**</TPL>**

Q: <MSG>*- 1129734520 2005.10.19 R17-M0-N0-I:J18-U01 2005-10-19-08.08.40.058960 R17-M0-N0-I:J18-U01 RAS KERNEL INFO shutdown complete***</MSG>**
A: <TPL>*shutdown complete***</TPL>**

Q: <MSG>*20231114T101914E ERROR 14 while processing line 123: cannot find input '42'***</MSG>**
A: <TPL>*ERROR <*> while processing line <*>: cannot find input <*>***</TPL>**

Q: <MSG>*2023-01-14 23:05:14 INFO: Reading data from /user/input/file.txt***</MSG>**
A: <TPL>*Reading data from <*>* **</TPL>**

Here is the input log message:**<MSG>...</MSG>**
Please print the corresponding template.

Figure 3: The prompt prefix. Few-shot examples are highlighted in green. The log messages are inserted in <MSG>...</MSG> at the bottom of the prompt.

4 EMPIRICAL STUDY

4.1 Large Language Models

We consider six LLMs for our empirical study. Table 1 lists each LLM and provides information on the version, availability, and base model, in case it is built by fine-tuning another LLM. Among the six models, two can be accessed through API calls and require payment (GPT3.5[1] and Claude 2[2]). The remaining four LLMs (Llama2 [21], Code Llama [19], Zephyr [22], CodeUp [7]) are publicly accessible and can be downloaded from Huggingface.[3] For our implementation, we accessed the free-to-use models via Ollama[4] and used APIs for the paid proprietary models. [5]

4.2 Dataset

We conduct experiments on the log messages selected from LogPai's LogHub collection [28]. This collection provides 16 real-world

[1] https://platform.openai.com/docs/models/gpt-3-5
[2] https://www.anthropic.com/index/claude-2
[3] https://huggingface.co/
[4] https://ollama.ai/
[5] The costs for running the GPT-3.5 experiments were approx. USD 1.7, and for the Claude-2 experiments approx. USD 7).

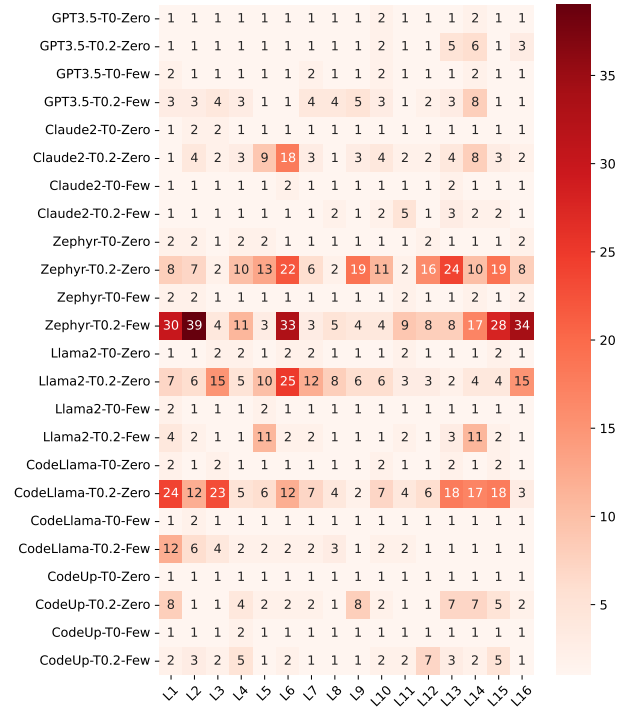

Figure 4: Number of unique templates extracted out of 50 repetitions (*determinism*).

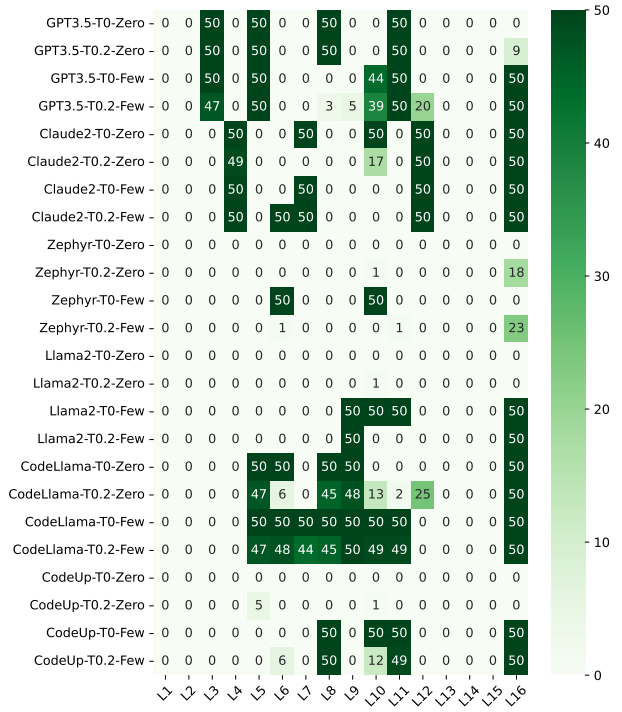

Figure 5: Number of correct templates extracted out of 50 repetitions (*correctness*).

log datasets from various systems, including distributed systems, supercomputers, operating systems, mobile systems, server applications, and standalone software. The experiments are carried out on a total of 16 log messages, where we selected one message from each of the 16 LogHub projects at random (Table 2). Following existing studies [11, 12, 26], we use the templates provided by Khan et al. [10] to determine the correctness of the extracted log templates.

4.3 Results and Findings

We apply six LLMs to evaluate their *non-determinism* and *correctness*, over 50 repeated template extractions for 16 log messages. For each of the six LLMs, we conduct experiments with two prompt types (zero-shot, few-shot) and two temperature values (0, 0.2). Thus, there are four types of experiments for each model. We name the experiments as follows: [model_name]-[temperature]-[prompt_type] (e.g., Zephyr-T0-Zero). The results are summarized in Figure 4 (determinism) and Figure 5 (correctness).

Impact of temperature on determinism: With both types of prompts, each model at temperature 0 produced fewer unique templates than at temperature 0.2. However, this does not always mean that the models at temperature 0 produce only a single type of template for the same log message. We observe that only a single case (CodeUp with zero-shot prompting) provides a single template while the average number of unique templates generated is always more than one in all other cases. Ouyang *et al.* have also a similar finding that setting the temperature to 0 does not guarantee determinism for the code generation task [17]. One reason for this could be non-deterministic floating point calculations of GPUs.[6] Nevertheless, the number of unique templates extracted varies between one and two for all models at temperature 0 while it increases up to 39 for a model (Zephyr) at temperature 0.2.

Impact of prompt on determinism: We observe that the experiments with zero-shot prompting generate a greater number of unique templates than the experiments with few-shot prompting, in 7 out of 12 cases. The results vary based on models, hence it is not the case that the experiments with zero-shot prompting have a higher number of unique templates for all models. For instance, GPT-3.5 experiments with few-shot prompting have a higher number of unique templates than zero-shot prompts, while Llama2 and CodeLlama experiments with zero-shot prompting have a higher number of unique templates. For CodeUp and Zephyr models, results differ based on temperature.

Impact of LLM on determinism: Zephyr has the highest number of unique templates on 13 out of 16 log messages. Moreover, it has the highest number of unique templates on average for all four types of experiments (i.e., prompt-type and temperature). Overall prompts for each model, GPT-3.5 has the least number of unique templates generated for all the log messages. In all the zero-shot experiments, GPT-3.5 has the least number of unique templates at temperature 0.2, while CodeUp has the least number of unique templates at temperature 0. In all the few-shot experiments, Claude-2 has the least number of unique templates at temperature 0.2, while both CodeUp and CodeLlama have the least number of unique templates

[6] https://community.openai.com/t/a-question-on-determinism/8185

Table 2: Overview of the log messages used in the empirical study.

ID	Project	Log message
L1	Android	03-17 16:13:38.811 1702 2395 D WindowManager: printFreezingDisplayLogsopening app wtoken = AppWindowToken9f4ef63 token=Tokena64f992 ActivityRecordde9231d u0 com.tencent.qt.qtl/.activity.info.NewsDetailXmlActivity t761, allDrawn= false, startingDisplayed = false, startingMoved = false, isRelaunching = false
L2	Hadoop	2015-10-18 18:06:26,139 ERROR [eventHandlingThread] org.apache.hadoop.yarn.YarnUncaughtExceptionHandler: Thread Thread[eventHandlingThread,5,main] threw an Exception.
L3	Mac	Jul 1 11:53:49 authorMacBook-Pro networkd[195]: -[NETClientConnection effectiveBundleID] using process name apsd as bundle ID (this is expected for daemons without bundle ID
L4	Windows	2016-09-28 04:30:31, Info CBS SQM: Cleaning up report files older than 10 days.
L5	Apache	[Sun Dec 04 17:43:08 2005] [error] jk2_init() Can't find child 1566 in scoreboard
L6	BGL	- 1130255740 2005.10.25 R11-M1-N2-C:J09-U11 2005-10-25-08.55.40.146416 R11-M1-N2-C:J09-U11 RAS KERNEL INFO critical input interrupt (unit=0x0b bit=0x0b): warning for torus z- wire
L7	HDFS	081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk_38865049064139660 terminating
L8	HealthApp	20171223-22:19:58:415|HiH_HiSyncControl|30002312|checkInsertStatus stepStatSum or calorieStatSum is enough
L9	HPC	289739 Interconnect-0N00 switch_module fan 1121303273 1 Fan speeds (3552 3534 3375 4245 3515 3479)
L10	Linux	Jul 27 14:41:57 combo kernel: CPU 0 irqstacks, hard=02345000 soft=02344000
L11	OpenSSH	Dec 10 11:03:53 LabSZ sshd[25457]: fatal: Write failed: Connection reset by peer [preauth]
L12	OpenStack	nova-compute.log.1.2017-05-16_13:55:31 2017-05-16 00:00:17.754 2931 INFO nova.virt.libvirt.driver [-] [instance: b9000564-fe1a-409b-b8cc-1e88b294cd1d] Instance destroyed successfully.
L13	Proxifier	[10.30 17:02:17] putty.exe - 183.62.156.108:22 open through proxy socks.cse.cuhk.edu.hk:5070 SOCKS5
L14	Spark	17/06/09 20:10:46 INFO rdd.HadoopRDD: Input split: hdfs://10.10.34.11:9000/pjhe/logs/2kSOSP.log:21876+7292
L15	Thunderbird	- 1131567052 2005.11.09 #8# Nov 9 12:10:52 #8#/#8# sshd2[4769]: Now running on #29#'s privileges.
L16	Zookeeper	2015-08-10 18:23:52,646 - INFO [NIOServerCxn.Factory:0.0.0.0/0.0.0.0:2181:Learner@107] - Revalidating client: 0x14f05578bd80018

at temperature 0. At temperature 0, Codeup produces the least variation for the unique templates whereas GPT-3.5 generates the least variation at temperature 0.2.

Determinism and correctness: Increasing the temperature helps to find a correct solution among a variety of predictions. However, it decreases determinism. For example, GPT-3.5 at temperature 0.2 with few-shot prompting correctly identifies three templates more, but this happens only in three or five cases out of the 50 repetitions, for the messages L8 and L9 respectively. Using the increased temperature as an opportunity to find the correct template, requires a selection of the best template when multiple ones are available. On the other hand, the results of the experiments with temperature 0 are consistent since they return the correct template 50 times or 0 times. Here the only exceptional case is for the results of GPT-3.5 experiments with few-shot prompting on the message L11.

At temperature 0, the template of the message L10 is correctly extracted at least 44 times by all the models in either zero-shot or few-shot settings. At temperature 0.2, the template of L10 is correctly identified at least one time and a maximum of 49 times. At the same level of temperature, the extracted template of L16 is correct for at least 9 and at most 50 times. The messages L1, L2, L13, L14, and L15 are not parsed correctly by any model.

We note that syntactically different templates can be semantically identical. This happens when an LLM represents placeholders differently. One example is Zephyr applied to L16 with a temperature of 0.2 and the few-shot prompt. Out of the 34 unique templates, six are correct but written differently: Revalidating client: <*>, Revalidating client: <*clientID*>, Revalidating client: <CLIENT>, Revalidating client: <*clientaddr*>, Revalidating client: <*client_id*>, Revalidating client: <*client*>. We decided to treat these cases as different to highlight the non-determinism of LLMs to generate different responses.

Lastly, we observe that CodeLlama produces the highest number of correctly identified templates for all four settings.

4.4 Threats to Validity

Internal validity refers to threats based on our implementation and analysis of results. To use LLMs, we either accessed them via official APIs (GPT-3.5, Claude 2) or used Ollama, which allowed consistent access the the four open-source LLMs. The template extraction from the generated responses was fully automated for four out of six models. Zephyr and Llama2 required manual extractions for some of the responses, which caused manual influence. To minimize subjective choices, we extracted the first valid template, with the placeholder if possible, when processing messages manually.

The threats to *external* validity depend on the metrics and dataset used for our study. While the correctness of log templates is a popular metric, there are other metrics that could have been considered [10]. We used the LogHub dataset [28] to extract a total of 16 log messages, one for each project. This is a subset of the available 2, 000 log messages per project. An increased number of log messages per project could improve the external validity but was omitted due to the manual extraction effort required.

5 CONCLUSION

In this work, we investigated the non-determinism of six LLMs for log parsing. Our empirical study on 16 log templates showed that non-determinism depends on the model type, log message, temperature, and prompt. Among the six LLMs, GPT-3.5 is the model with the highest degree of determinism, and CodeUp shows the second highest determinism. However, we found that a temperature of 0 does not guarantee deterministic results, which agrees with the findings by Ouyang et al. [17], who among others suggested to report averaged results to combat non-determinism.

Moreover, we found that a higher temperature decreases determinism and it is less likely for LLMs to consistently find the correct log template (e.g., extracting it correctly for all 50 repetitions). However, an increased temperature could be seen as an opportunity for finding the correct template at least once, as we have observed for GPT-3.5. To benefit from a wider range of log templates, one either requires an oracle to select the correct template or a prioritization approach. This is an interesting avenue to explore in future work. In our future work, we also plan to comprehensively evaluate the determinism of LLMs with an increased number of log messages.

ACKNOWLEDGMENTS

This work is supported by the Research Council of Norway through the cureIT (IKTPLUSS #300461) and the secureIT projects (IKTPLUSS #288787). The empirical evaluation presented in this paper was performed on the Experimental Infrastructure for Exploration of Exascale Computing (eX3), financially supported by the Research Council of Norway under contract #270053.

REFERENCES

[1] Merve Astekin, Harun Zengin, and Hasan Sözer. 2018. DILAF: A framework for distributed analysis of large-scale system logs for anomaly detection. *Software: Practice and Experience* 49 (2018), 153 – 170. https://api.semanticscholar.org/CorpusID:59222641

[2] Song Chen and Hai Liao. 2022. BERT-Log: Anomaly Detection for System Logs Based on Pre-trained Language Model. *Applied Artificial Intelligence* 36, 1 (2022), 2145642. https://doi.org/10.1080/08839514.2022.2145642 arXiv:https://doi.org/10.1080/08839514.2022.2145642

[3] Qiang Fu, Jian-Guang Lou, Yi Wang, and Jiang Li. 2009. Execution Anomaly Detection in Distributed Systems through Unstructured Log Analysis. In *2009 Ninth IEEE International Conference on Data Mining*. 149–158. https://doi.org/10.1109/ICDM.2009.60

[4] Haixuan Guo, Shuhan Yuan, and Xintao Wu. 2021. LogBERT: Log Anomaly Detection via BERT. In *2021 International Joint Conference on Neural Networks (IJCNN)*. 1–8. https://doi.org/10.1109/IJCNN52387.2021.9534113

[5] Hossein Hamooni, Biplob Debnath, Jianwu Xu, Hui Zhang, Guofei Jiang, and Abdullah Mueen. 2016. LogMine: Fast Pattern Recognition for Log Analytics. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. Association for Computing Machinery, New York, NY, USA, 1573–1582. https://doi.org/10.1145/2983323.2983358

[6] Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. 2017. Drain: An Online Log Parsing Approach with Fixed Depth Tree. In *2017 IEEE International Conference on Web Services (ICWS)*. 33–40. https://api.semanticscholar.org/CorpusID:4776668

[7] Juyong Jiang and Sunghun Kim. 2023. CodeUp: A Multilingual Code Generation Llama2 Model with Parameter-Efficient Instruction-Tuning. https://github.com/juyongjiang/CodeUp.

[8] Zhihan Jiang, Jinyang Liu, Zhuangbin Chen, Yichen Li, Junjie Huang, Yintong Huo, Pinjia He, Jiazhen Gu, and Michael R. Lyu. 2023. LLMParser: A LLM-based Log Parsing Framework. https://doi.org/10.48550/arXiv.2310.01796 arXiv:2310.01796 [cs]

[9] Zhen Ming Jiang, Ahmed E. Hassan, Gilbert Hamann, and Parminder Flora. 2008. An Automated Approach for Abstracting Execution Logs to Execution Events. *J. Softw. Maint. Evol.* 20, 4 (jul 2008), 249–267.

[10] Zanis Ali Khan, Donghwan Shin, Domenico Bianculli, and Lionel Briand. 2022. Guidelines for Assessing the Accuracy of Log Message Template Identification Techniques. In *Proceedings of the 44th International Conference on Software Engineering*. ACM, Pittsburgh Pennsylvania, 1095–1106. https://doi.org/10.1145/3510003.3510101

[11] Van-Hoang Le and Hongyu Zhang. 2023. Log Parsing: How Far Can ChatGPT Go? https://doi.org/10.48550/arXiv.2306.01590 arXiv:2306.01590 [cs]

[12] Van-Hoang Le and Hongyu Zhang. 2023. Log Parsing with Prompt-Based Few-Shot Learning. In *Proceedings of the 45th International Conference on Software Engineering (ICSE '23)*. IEEE Press, Melbourne, Victoria, Australia, 2438–2449. https://doi.org/10.1109/icse48619.2023.00204

[13] Yilun Liu, Shimin Tao, Weibin Meng, Jingyu Wang, Wenbing Ma, Yanqing Zhao, Yuhang Chen, Hao Yang, Yanfei Jiang, and Xun Chen. 2023. LogPrompt: Prompt Engineering Towards Zero-Shot and Interpretable Log Analysis. https://doi.org/10.48550/arXiv.2308.07610 arXiv:2308.07610 [cs]

[14] Milad Moradi and Matthias Samwald. 2021. Evaluating the Robustness of Neural Language Models to Input Perturbations. arXiv:2108.12237 [cs]

[15] Priyanka Mudgal and Rita Wouhaybi. 2023. An Assessment of ChatGPT on Log Data. https://doi.org/10.48550/arXiv.2309.07938 arXiv:2309.07938 [cs]

[16] Sasho Nedelkoski, Jasmin Bogatinovski, Alexander Acker, Jorge Cardoso, and Odej Kao. 2021. Self-supervised Log Parsing. In *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track (Lecture Notes in Computer Science)*, Yuxiao Dong, Dunja Mladenić, and Craig Saunders (Eds.). Springer International Publishing, Cham, 122–138. https://doi.org/10.1007/978-3-030-67667-4_8

[17] Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. 2023. LLM Is Like a Box of Chocolates: The Non-determinism of ChatGPT in Code Generation. https://doi.org/10.48550/arXiv.2308.02828 arXiv:2308.02828 [cs]

[18] Jonas Rauber, Wieland Brendel, and Matthias Bethge. 2018. Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models. arXiv:1707.04131 [cs, stat]

[19] Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2023. Code Llama: Open Foundation Models for Code. arXiv:2308.12950 [cs]

[20] Liang Tang, Tao Li, and Chang-Shing Perng. 2011. LogSig: Generating System Events from Raw Textual Logs. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management* (Glasgow, Scotland, UK) *(CIKM '11)*. Association for Computing Machinery, New York, NY, USA, 785–794. https://doi.org/10.1145/2063576.2063690

[21] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs]

[22] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct Distillation of LM Alignment. arXiv:2310.16944 [cs]

[23] R. Vaarandi. 2003. A data clustering algorithm for mining patterns from event logs. In *Proceedings of the 3rd IEEE Workshop on IP Operations & Management (IPOM 2003) (IEEE Cat. No.03EX764)*. 119–126. https://doi.org/10.1109/IPOM.2003.1251233

[24] Risto Vaarandi and Mauno Pihelgas. 2015. LogCluster - A data clustering and pattern mining algorithm for event logs. In *2015 11th International Conference on Network and Service Management (CNSM)*. 1–7. https://doi.org/10.1109/CNSM.2015.7367331

[25] Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. 2022. Adversarial GLUE: A Multi-Task Benchmark for Robustness Evaluation of Language Models. arXiv:2111.02840 [cs]

[26] Junjielong Xu, Ruichun Yang, Yintong Huo, Chengyu Zhang, and Pinjia He. 2023. Prompting for Automatic Log Template Extraction. https://doi.org/10.48550/arXiv.2307.09950 arXiv:2307.09950 [cs]

[27] Wei Xu, Ling Huang, Armando Fox, David Patterson, and Michael I. Jordan. 2010. Detecting Large-Scale System Problems by Mining Console Logs. In *Proceedings of the 27th International Conference on International Conference on Machine Learning* (Haifa, Israel) *(ICML'10)*. Omnipress, Madison, WI, USA, 37–46.

[28] 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. arXiv:2008.06448 [cs]

[29] Jieming Zhu, Shilin He, Jinyang Liu, Pinjia He, Qi Xie, Zibin Zheng, and Michael R. Lyu. 2019. Tools and Benchmarks for Automated Log Parsing. In *Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice* (Montreal, Quebec, Canada) *(ICSE-SEIP '19)*. IEEE Press, 121–130. https://doi.org/10.1109/ICSE-SEIP.2019.00021