The Effectiveness of Compact Fine-Tuned LLMs in Log Parsing

Maryam Mehrabi
ECE, Concordia University
Montreal, QC, Canada
me_marya@live.concordia.ca

Abdelwahab Hamou-Lhadj
ECE, Concordia University
Montreal, QC, Canada
wahab.hamou-lhadj@concordia.ca

Hossein Moosavi
Cisco Systems
Ottawa, ON, Canada
smoosavi@cisco.com

***Abstract*—Log parsing is defined as the process of extracting structured information from unstructured log data. It is an important step prior to many log analytics tasks. The emergence of Large Language Models (LLMs), like Generative Pre-trained Transformers (GPTs), has driven the development of novel log parsing methods. Existing studies have examined the effectiveness of large-scale general-purpose LLMs in log parsing. In this paper, we argue that the long-term adoption of such LLMs pose challenges of data privacy, cost, and tool integration. To address these challenges, we explore the viability of supervised fine-tuning of an open-source compact LLM for log parsing as a prospective alternative. To this end, we fine-tune the Mistral-7B-Instruct LLM on a diverse set of log files and evaluate its performance, in terms of both accuracy and robustness, against OpenAI's GPT-4-Turbo using different configuration settings. We apply two evaluation approaches, namely metric-based and LLM-based. Our overall findings show that fine-tuning a compact LLM such as Mistral-7B provides similar and sometimes better results than using a large-scale LLM, in our case GPT-4-Turbo. These findings are important because they enable companies to use a smaller LLM that they can readily adapt to parsing their log data, and integrate into their log analytics tools, without the need to rely on third-party LLM providers.**

***Index Terms*—Log Parsing, Large Language Models, Machine Learning, Software Maintenance and Evolution**

I. Introduction

Logs are used by software development and operations teams to understand and analyze the behaviour of software systems [1]. Log analysis is instrumental in facilitating a variety of tasks including the detection of system anomalies [2], [3], root cause analysis of system faults [4], [5], and performance optimization [6].

Logs, however, are largely unstructured [7] [8], [9], making it challenging to analyze their content [10]. To address this challenge, several log parsing techniques have been proposed (e.g., [11] [12]) with the main objective to automatically extract log event structures (also called event templates) from unstructured log files. Log parsing is a prerequisite to many log analytics tasks as discussed in [9], [13].

As an example, consider the log event in Figure 1, which was generated from the Hadoop Distributed File System (HDFS). This log event contains a mix of static tokens (`PacketResponder, for, terminating`) and dynamic tokens, which correspond to the values of the logged variables (`1, blk_5017373558217225674`). The challenge of log parsing is to automatically distinguish between the static tokens and dynamic tokens. One possible solution would be to use regular expressions. The problem is that typical log files may contain thousands of event templates [8]. In addition, any modifications to the system would require constant changes to the regular expressions. To make things worse, it is common for companies to have several types of log files, further complicating the development and maintenance of regular expressions to parse log data [14] [10].

Log Event: `PacketResponder 1 for block blk_5017373558217225674 terminating`

Event Template: `PacketResponder <*> for block <*> terminating`

Fig. 1: An example of a log event and its corresponding event template

Recently, Large Language Models (LLMs) have been used to support a variety of software engineering tasks. In the context of log analytics, LLMs have been used for automatic generation of logging statements [15], log parsing [16]–[19], and root cause analysis [20]–[22]. Existing studies have reported satisfactory results, showcasing the analytical capabilities of these models for log analytics.

Recent LLM-based log parsing studies have mainly leveraged large-scale general-purpose LLMs such as the GPT family of languages [16]–[19], [23]. The use of such large-scale LLMs for log analytics pose three challenges related to privacy, tool integration, and cost:

1) **Privacy:** Logs often contain private information such as user names, IP addresses, MAC addresses, etc. Using a proprietary LLM (e.g., GPT-4) would require from companies to send their logs to be processed on third-party LLM servers, increasing the risk of violating privacy regulations that govern the use of personal data.
2) **Tool Integration:** Log parsing is only one step in the entire log analytics pipeline. From the development perspective, integrating a third-party LLM with the existing log analytics tooling can be challenging. Companies

should be able to download the LLM, fine-tune it to their specific needs, and integrate it with their tool suite.

3) **Cost:** The high-performing LLMs tend to be expensive when used with large data [24] [25]. Log files are notoriously known to be considerably large [26], potentially resulting in high processing costs.

To address these challenges, in this paper, we investigate the use of a compact open-source LLM as an alternative solution. More particularly, we show how fine-tuning Mistral-7B-Instruct[1], a 7 billion parameter model, can perform log parsing tasks as well as OpenAI's GPT-4-Turbo, a large-scale state-of-the-art LLM speculated to have more than 1 trillion parameters [27]. We choose Mistral-7B[2] as one of the top performer LLMs for its size on the Huggingface Open LLM Leaderboard[3]. To support our findings, we experimented with several log files. Our evaluation includes an examination of both the accuracy and robustness of the results using two complementary approaches, namely metric-based and LLM-based. For the metric-based method, we quantitatively measure accuracy and robustness. However, because language models (particularly the smaller ones) have the potential to produce extraneous tokens, metric-based evaluation methods tend to be sensitive to such noise. To address this, we use an LLM-based evaluation method where we ask GPT-4 to act as an expert and evaluate the accuracy and robustness of the models.

The contributions of this paper are as follows:

- To our knowledge, this is the first study that explores the applicability of a compact fine-tuned LLM (in our case Mistral-7B) for log parsing, as an alternative to a general-purpose proprietary LLM such as GPT-4.
- We use a comprehensive evaluation framework comprising two complementary methods that are designed to ascertain whether a smaller, fine-tuned model can achieve the performance of a large-scale model in log parsing tasks in terms of accuracy and robustness.

II. Background on Large Language Models

LLMs are large-scale language models that are pre-trained on an extensive corpus of language data [28]. These models are fundamentally categorized into two types based on their architectural approach: autoregressive models and masked token models. Autoregressive models, exemplified by the GPT series, employ a decoder-only architecture. In this setup, the prediction of the next token regarding preceding tokens. Masked token models, on the other hand, are designed to predict the value of masked tokens within a given input sequence by understanding the context provided by the unmasked tokens [29], [30]. In this paper, we focus on two tiers of autoregressive language models in terms of number of model parameters and specialization.

[1]https://huggingface.co/NousResearch/Yarn-Mistral-7b-128k

[2]From this point forward, any reference to *GPT-4* and *Mistral-7B* will mean *GPT-4-Turbo* and *Mistral-7B-Instruct*, respectively.

[3]https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard

A. Large-scale general-purpose LLMs

These models are noteworthy for their scale in terms of the number of training parameters and the sources of data used for their training. As an example GPT-4 speculated to have more than one trillion parameters and is trained on a diverse array of data sources, from Wikipedia texts to programming projects on GitHub. As a result, they have achieved state-of-the-art performance across a multitude of natural language processing (NLP) tasks, including translation, summarization, and text generation, showcasing their capacity to handle complex language-related challenges efficiently [31].

B. Compact fine-tuned LLMs

Compact models are designed to provide substantial language understanding and generation capabilities while having lower compute requirements for training and inference compared to the larger-scale models. An illustrative example within this category is the Llama2 collection, which offers models with parameter counts ranging from 7 billion to 70 billion [32]. Despite their relatively lower parameter count and training on a more concise corpus of data, these models consistently produce high-quality inferences. Their open-source nature and their resource-efficient design, renders them particularly advantageous for fine-tuning and the development of specialized, task-specific models.

In-context learning and fine-tuning are two distinct methodologies that facilitate the adaptation of LLMs to specific tasks.

1) In-context Learning

This approach enables a model to learn specific tasks utilizing a set of examples, instead of explicit retraining or fine-tuning on task-specific datasets. The core mechanism of in-context learning involves the following steps: 1) providing a prompt containing several examples illustrate the task and guiding the model regarding the expected outcomes. 2) Utilizing these examples, the model deduces the task's requirements and crafts a response that aligns with the task [33].

2) Fine-tuning

Fine-tuning is when a pre-trained model undergoes additional training on a smaller, specific dataset relevant to a particular task or domain. Fine-tuning is generally more effective than in-context learning in terms of performance and is more resource-efficient compared to training a model from scratch [34].

III. Experiment Setup

A. Study Objective

The objective of this study is to investigate the performance of Mistral-7B, a compact LLM, for parsing log events as compared to GPT-4, a large-scale LLM. The study addresses the following research questions.

- **RQ1:** What is the accuracy of Mistral-7B compared to GPT-4 across various configuration settings using a metric-based evaluation method?
- **RQ2:** What is the robustness of Mistral-7B compared to GPT-4 across various configuration settings using a metric-based evaluation method?

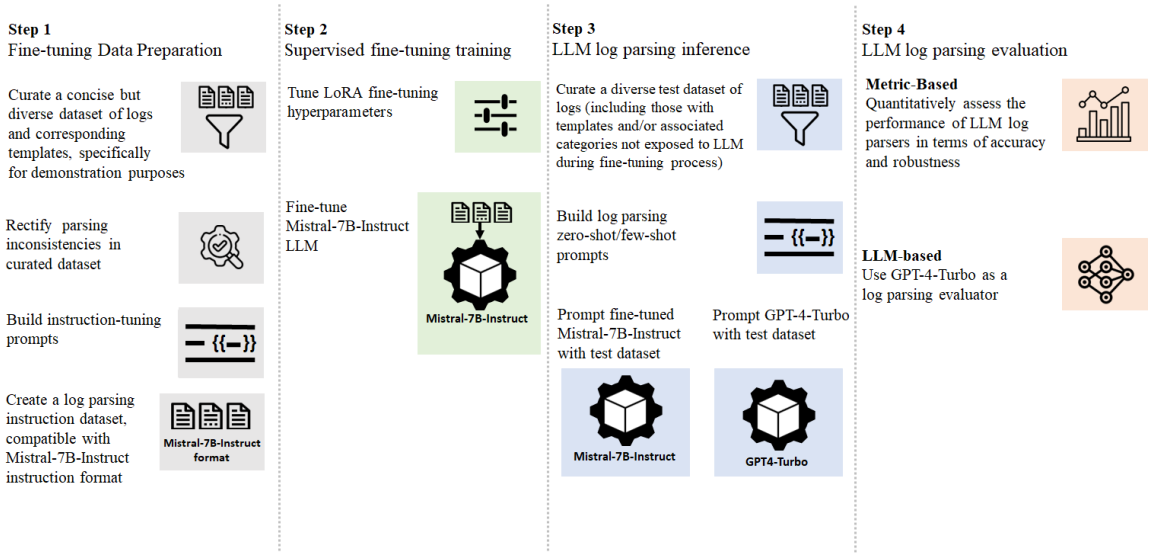

Fig. 2: Overview of our approach

- **RQ3:** What is the accuracy and robustness of Mistral-7B compared to GPT-4 using an LLM-based evaluation method?

For RQ1 and RQ2, we examine the relative accuracy and robustness of a fine-tuned compact LLM, Mistral-7B, compared to GPT-4, a large-scale general-purpose LLM, when applied to the task of log parsing in various configurations. Additionally, in RQ3, we explore the need for LLM-based evaluation as a complementary approach to quantitative metric-based methods for assessing the performance of a log parser.

B. Approach

Figure 2 shows our approach to answer the RQs, which consists of four steps. In Step 1, we curate a diverse subset of logs with their corresponding templates from the LogPai log file repository [10] to create the fine-tuning dataset. We made some corrections to the ground truth log templates to ensure the fine-tuning dataset is accurate and consistent. Then, we build the instruction-tuning prompts and create a fine-tuning dataset for the Mistral-7B model. In Step 2, we first tune the LLM fine-tuning hyperparameters using Low-Rank Adaptation (LoRA), which is a parameter-efficient method to update a smaller subset of the model's parameteres [35]. Then we fine-tune the Mistral-7B model with the log parsing instruction dataset prepared in Step 1. During Step 3, we select a diverse dataset from the LogPai repository for testing. The test data includes the templates seen in the fine-tuning dataset as well as unseen templates. Then, we prepare log parsing prompts for different configuration settings, and use the Mistral-7B and GPT-4 to infer the corresponding templates for each log event. In Step 4, we evaluate the effectiveness of the selected models using metric-based and LLM-based evaluation methods. The metric-based method focuses on quantifiable aspects, whereas the LLM-based method examines the model performance by using GPT-4 as an evaluator.

C. Data Preparation

Fine-tuning Dataset: The log files used in this study are from the LogPai benchmark [10]. The benchmark contains 16 log files that are generated from different types of systems, namely distributed systems, supercomputers, operating systems, mobile applications and server applications. These datasets are used extensively in log parsing studies.

To construct the fine-tuning dataset, we select randomly five log files, the ones generated from Apache, BGL, HPC, Proxifier, and Zookeeper. Each log file comes with a subset of 2,000 log events that have been parsed manually. The event templates were identified, and each log event out of the 2,000 events was associated with a specific event template. These labelled log files are used as ground truth against which we can check the accuracy and robustness of our models.

In addition, to construct a well-defined fine-tuning dataset, we randomly choose 80% of the event templates from each of the five log files (see Table I). Subsequently, for each event template, we select randomly 15 log events that we will use to fine-tune the LLM models. A different number of events could have been chosen, but we do not believe that this would have much effect on the results. What is important is to have the same distribution for each event template to ensure that the LLM learns the patterns in the data across multiple log files. In cases where the number of log events for a certain template was fewer than 15, we used oversampling to ensure an equal number of log events across each event template. The detailed composition of the fine-tuning dataset is shown in Table I.

Test Dataset: We divide the test dataset into three distinct categories to ensure a comprehensive evaluation of the performance of the models across a variety of scenarios with different levels of familiarity with the data. Specifically, one third of the event templates are extracted directly from the training dataset (i.e., 80% of the event templates). Another

TABLE I: Summary of the fine-tuning dataset

Log File	#Total Templates	#Selected Templates	#Selected Log Events
Apache	6	4	60
BGL	120	96	1440
HPC	46	36	540
Proxifier	8	6	90
Zookeeper	50	40	600
Total number of train log events			2730
#Selected Templates = 0.8 * #Total Templates #Selected Log Events = 15 * #Selected Templates			

third is meticulously selected from the remaining 20% of the training dataset. The final third is comprised of event templates drawn from three new log files from the LogPai benchmark that were not part of the training process. For this last third, we select randomly three log files, namely the Hadoop, OpenStack, and Spark log files; each contributing 16 event templates to the test dataset. The motivation behind using unseen log files in this study is to investigate the ability of LLMs to recognize the structure of log files beyond those used during training. For each event template within these categories, three log events are randomly selected to represent a wide range of potential scenarios. In situations where the available log events for a given template were fewer than three, oversampling is employed and thus standardizes the dataset, ensuring a uniform number of log events for each event template. The selection and organization of these event templates and log events are detailed in Table II.

TABLE II: Summary of the test dataset

Category	#Selected Templates	#Selected Log Events
From the training dataset and included in training	48	144
From the training dataset but not included in training	48	144
From new log files (Hadoop, Spark, OpenStack)	48	144
Total Number of test log events		432
#Log events = #Templates * 3		

Dataset Correction: Our initial examination of the log files used for training and testing revealed several inconsistencies in the labelled ground truth. This was also noticed in other studies [36]. To address these issues, we reviewed the ground truth and rectified small ground truth inconsistencies to ensure that event templates corresponding to similar log events adhere to uniform extraction patterns. For example, the ground template `Found block rdd_<*> locally` should be `Found block <*> locally`. The token `rdd_` is part of the dynamic token that was mistakenly classified as a static token. Examples of changes to the ground truth are included on the GitHub repository[4].

Building the LLM System Prompts: The preparation of the training data for fine-tuning Mistral-7B , testing purposes, and LLM-based evaluation required the development of standardized prompt templates. These templates are subjected to iterative optimization processes, with modifications made to various details to identify configurations that yielded the most favorable results during model inference.

To cover all targeted configurations, namely zero-shot, few-shot, and fine-tuned settings, we prepare distinct prompts tailored to each scenario. The prompt used for Mistral-7B zero-shot configuration is identical to that used in the fine-tuning process. It is designed to assess the model's innate capabilities without prior exposure to specific examples. For the few-shot setting in both Mistral-7B and GT-4 models, we use identical prompts that are augmented with two illustrative examples. It is intended to facilitate the models' learning from these minimal yet targeted inputs. The prompt prepared for the LLM-based evaluation also include the definition of the evaluation metrics and two examples. In each example, we teach the model the given log event, inferred event template, and ground truth template and the appropriate score for the targeted evaluation metric.

D. Fine-tuning Configuration

We fine-tune Mistral-7B with HuggingFace's Parameter-Efficient Fine-Tuning (PEFT)[5] and Low-Rank Adaptation (LoRA). LoRA helps in reducing the computational and memory overhead associated with traditional fine-tuning methods and has been shown to perform on par or even better than full fine-tuning in some cases, despite having fewer trainable parameters and higher training throughput [35]. Our code, prompts, and experimental results are available on GitHub[6].

E. Inference Configuration

Model Selection The selection of the LLMs for our study is based on an evaluation of its capabilities and performance benchmarks, particularly emphasizing LLMs with the minimum viable number of parameters. As we discussed earlier, for the compact LLM, we select the Mistral model with 7 billion parameters. This model was distinguished by its exceptional performance among the models with 7 billion parameters on the Huggingface Leaderboard at the time our experiments were conducted. For the large-scale LLM, we select the GPT-4 model.

F. Evaluation

We employ two synergistic methodologies for the evaluation of our LLM-based log parsers: a metric-based approach and an LLM-based approach.

1) Metric-based Approach

We use accuracy and robustness as the primary metrics to gauge the performance of a log parser[7].

[4]https://github.com/maryam-mrb/logparsing-compactLLMs/tree/main/RectifiedTemplates

[5]https://huggingface.co/docs/peft/en/index

[6]https://github.com/maryam-mrb/logparsing-compactLLMs

[7]A third dimension for measurement is computational efficiency, that is, how fast the extracted templates are generated and how much computational resources are consumed. Computational efficiency can be evaluated by metrics such as run time, memory usage, and scalability, which is beyond the scope of this work.

We quantitatively assess the accuracy of each LLM-based log parser using the following metrics.

- **Message Level Accuracy (MLA)** is defined as the ratio of the number of log events for which event templates that precisely match the ground truth data were successfully inferred to the total number of log events analyzed [37].
- **Levenshtein Edit Distance (ED)** measures the similarity between the inferred event template and ground truth by quantifying the minimum number of single-character edits (*i.e.*, insertions, deletions, or substitutions) required to transform the inferred event template into ground truth [37]. A lower ED indicates a more accurate log parser.
- **F1 Score** is the harmonic mean of precision and recall. For each template, precision measures the percentage of log messages correctly identified as belonging to that certain template among all the messages identified as belonging to that template, while recall measures the percentage of messages correctly identified as belonging to the template among all the messages actually belonging to that template in the ground truth. The F1 Score ranges from 0 to 1, where 0 means no similarity and 1 means perfect match. A higher F1 Score indicates a more accurate log parser.

In evaluating robustness, we measure the efficacy with which a log parser manages diverse and complex log data. This includes logs characterized by undefined or poorly-defined templates, multiple parameters, or noisy messages. We assess the robustness of each LLM-based log parser in relation to the ground truth template size (quantified by the number of characters), the log dataset to which the template belongs, and whether the template and its associated log dataset were exposed to the log parser during the fine-tuning process.

2) LLM-based Approach

The LLM-based evaluation involves using GPT-4 as a log parsing evaluator, to rate the inferred event templates given the ground truth. More specifically, we prompt GPT-4 to subjectively evaluate each individual extracted log template in terms of accuracy and robustness and give a score between 0 and 5 for each evaluation criterion. The complete system prompts are provided as part of the reproduction package.

IV. Evaluation Results

In this section, we present the findings of our experimental analysis, aimed at addressing the research questions.

A. RQ1: What is the accuracy of Mistral-7B compared to GPT-4 across various configuration settings using a metric-based evaluation method?

Table III details the accuracy assessment for the Mistral-7B and GPT-4 models across various configurations, using MLA, ED, and F1 score as metrics. For the Mistral-7B model, there is a noticeable drop in MLA from one-shot (22.9%) to two-shot (14.1%); however, fine-tuning the model boosts the MLA to 74.8%. The fine-tuned version shows a mean ED of 7.2 and a median of 0.0, and an F1 score with a mean of 0.74 and a perfect median of 1.0, indicating highly accurate performance on most inputs after fine-tuning. GPT-4, in its zero-shot configuration, achieves a MLA of 47.2%, improving to 72.2% in a two-shot scenario. The model's ED scores are low, with means of 6.4 in the zero-shot and 9.2 in the two-shot, both with a median of 0.0, suggesting precision in edits. For the F1 score, GPT-4 scores a mean of 0.46 in zero-shot and improves to 0.71 in two-shot, demonstrating less but competitive score compared to Mistral-7B's.

Finding RQ1: The results show that fine-tuned Mistral-7B achieves better accuracy compared to GPT-4 using all three metric-based assessment, MLA, ED, and F1 Score.

B. RQ2: What is the robustness of Mistral-7B compared to GPT-4 across various configuration settings using a metric-based evaluation method?

To address this question, we evaluate robustness of each LLM-based log parser in relation to the the ground truth template size, the log dataset to which the template belongs, and whether the template and its associated log dataset were seen by the LLM during the fine-tuning process. We use different parsing accuracy metrics, including MLA, ED, and F1 Score, to assess robustness of each LLM-based parser under various configurations.

Template Size: Figures 3, 4, and 5 demonstrate the MLA, ED, and F1 Score variations across log template sizes, respectively.

For MLA (Figure 3), both models exhibit a general trend of decreasing accuracy with an increase in template size, although the fine-tuned Mistral-7B maintains relatively higher accuracy across the board, suggesting robustness in handling complex parsing tasks.

In terms of ED (Figure 4), the fine-tuned Mistral-7B presents lower edit distances, implying that its outputs are closer to the desired results with fewer modifications needed, a clear indicator of its robust nature post fine-tuning. The GPT-4 model, particularly in the two-shot configuration, shows a smaller increase in ED with the rise in template size, which suggests a certain degree of robustness in adapting to the increased complexity.

The F1 Score (Figure 5), is notably high for the fine-tuned Mistral-7B across all template sizes, underscoring its robust performance. GPT-4 demonstrates competitive F1 Scores, especially in the two-shot configuration, indicating a robust design that efficiently leverages additional context for performance enhancement.

Diversity of Datasets: Figures 6, 7, and 8 compare the robustness of the Mistral-7B and GPT-4 models across various datasets using MLA, ED, and F1 Score as evaluation metrics. It is worth to mention that Hadoop, Spark, and Open Stack datasets did not used as training datasets during fine-tuning the Mistral-7B model.

When observing MLA (Figure 6), the fine-tuned Mistral-7B demonstrates a higher accuracy, indicating effective learning

TABLE III: Metric-based evaluation results for assessing the accuracy

Model	MLA	ED		F1 Score	
		Mean	Median	Mean	Median
Mistral-7B (0-shot)	22.9%	21.9	12.0	0.23	0.0
Mistral-7B (2-shot)	14.1%	54.2	55.5	0.13	0.0
Mistral-7B (Fine-tuned)	74.8%	7.2	0.0	0.74	1.0
GPT-4 (0-shot)	47.2%	6.4	2.0	0.46	0.0
GPT-4 (2-shot)	72.2%	9.2	0.0	0.71	1.0

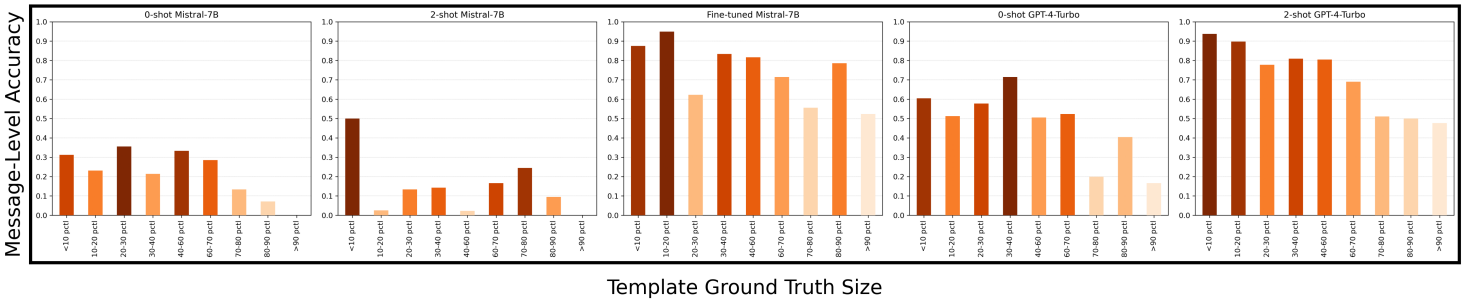

Fig. 3: Message Level Accuracy across template ground truth sizes

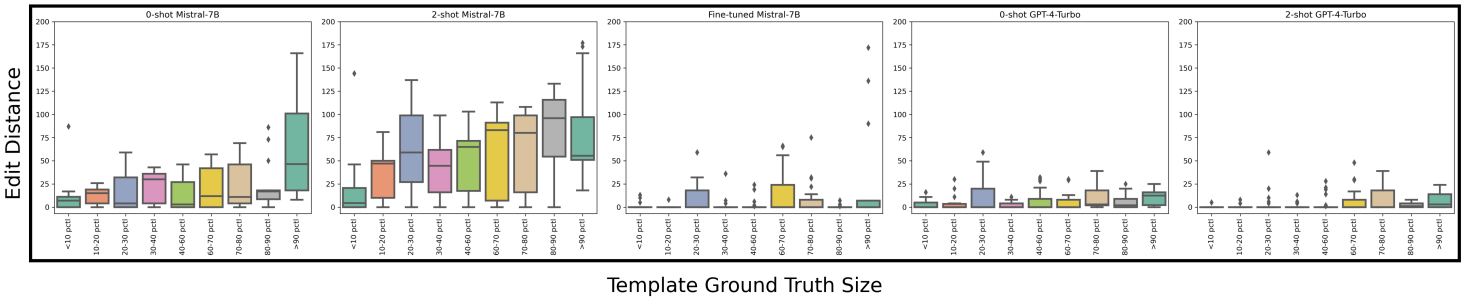

Fig. 4: Edit Distance across template ground truth sizes

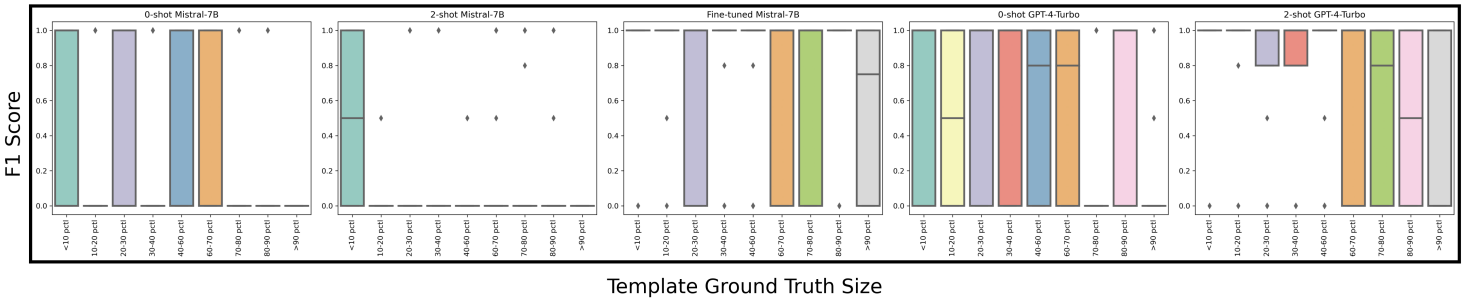

Fig. 5: F1 Score across template ground truth sizes

and adaptation to various datasets. This LLM performs even better than GPT-4 in zero-shot and two-shot settings facing unseen data from Spark dataset. It showcases a better performance than GPT-4 in zero-shot setting for OpenStack and Hadoop dataset that highlights the robustness of fine-tuned LLM versus a large-scale LLM.

Regarding ED (Figure 7), the fine-tuned Mistral-7B model demonstrates consistently lower distances across the majority

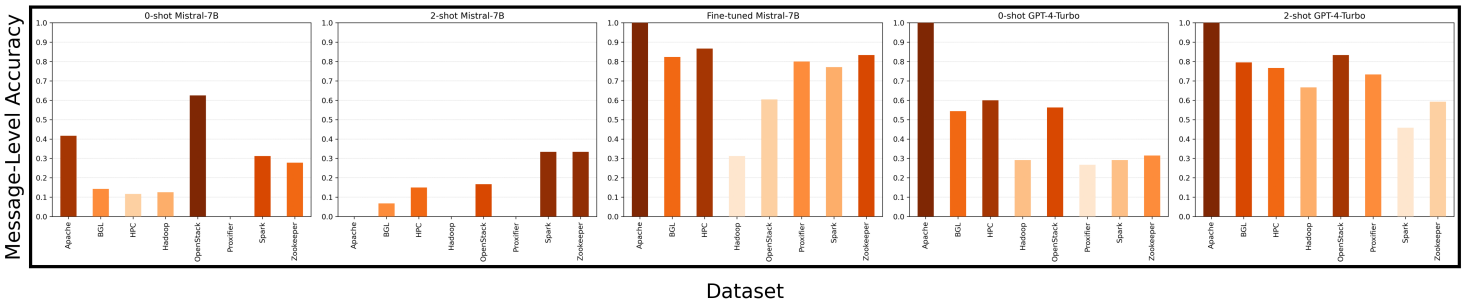

Fig. 6: Message Level Accuracy across different log files

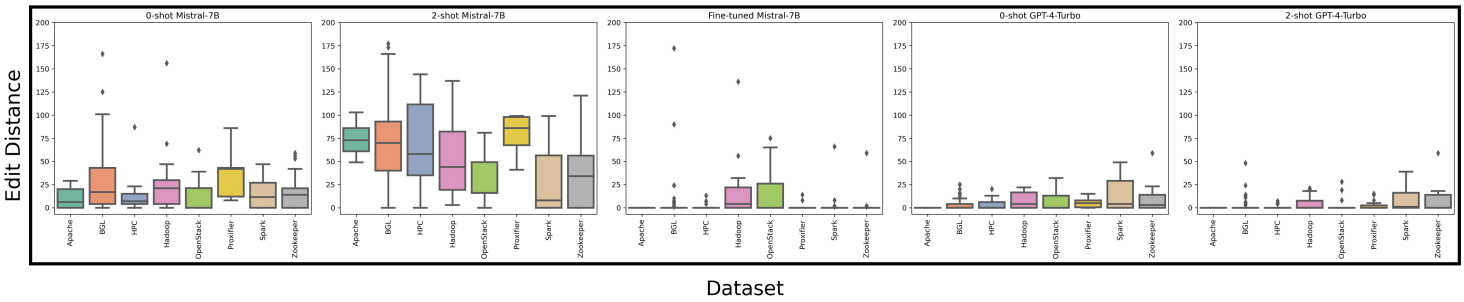

Fig. 7: Edit Distance across different log files

of datasets. As anticipated, it exhibits higher edit distances for the Hadoop and OpenStack datasets, which were not included in its fine-tuning. However, the templates inferred for the unseen log events from Spark aligned perfectly with the ground truth. In contrast, GPT-4, in both zero-shot and two-shot configurations, recorded higher edit distances for the Spark dataset. Nonetheless, GPT-4 generally exhibits greater robustness in the two-shot setting as compared to zero-shot.

Regarding the F1 Score (Figure 8), the fine-tuned Mistral-7B achieves higher scores across the majority of datasets. As anticipated, its performance is lower on new datasets; however, overall, it has the best scores compared to GPT-4 in both zero-shot and two-shot settings. GPT-4 maintains competitive F1 Scores, especially in the two-shot setting, which highlights its inherent ability to comprehend and adapt to various data structures and complexities.

Exposure during fine-tuning: To investigate the extent of the fine-tuned Mistral-7B's reliance on memorization, we divided the test data into different levels of familiarity: 1) log events with event templates that are present in the training data; 2) log events originating from datasets included in the training phase, yet featuring event templates that were not part of the training set; and 3) completely unseen event logs derived from datasets that were excluded from the training process.

In terms of MLA (Figure 9), log events with templates that were seen during training has the highest score, suggesting an excellent retention of the learned patterns. For log events derived from datasets that were included in the training phase but whose specific templates were not part of the training set, the fine-tuned LLM exhibits satisfactory accuracy. However, there is a noticeable decline in MLA for log events from unseen datasets with new templates.

For ED (Figure 10), the fine-tuned Mistral-7B shows minimal need for edits when dealing with familiar both templates and novel templates from datasets that were part of the training. Conversely, the model incurs a higher ED when confronted with entirely unseen data, suggesting an area for potential enhancement in model generalization capabilities beyond the training domain.

The F1 Score (11) follows a similar pattern, the highest score relates to familiar data and there is a gradual decline as the data becomes less familiar. The sustained performance on unseen templates from known datasets suggests the LLM's ability to apply learned knowledge to novel scenarios within familiar contexts. However, the lower F1 Score for entirely new datasets reflects the model's challenge in extrapolating beyond its training scope.

C. *RQ3: What is the accuracy and robustness of Mistral-7B compared to GPT-4 using an LLM-based evaluation method?*

In the application of language models for extracting event templates within log parsing tasks, it is observed that particularly the smaller models may append or prepend extraneous

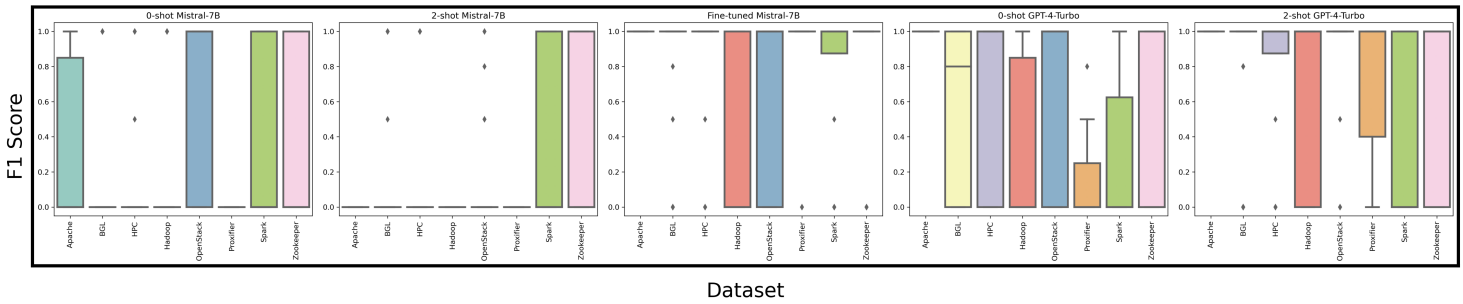

Fig. 8: F1 score across different log files

Finding RQ2: The results show that fine-tuned Mistral-7B achieves the best robustness in metric-based assessment with different template sizes and different datasets. It also has a satisfactory robustness when used with familiar datasets. However, it requires enhancement in order to be more robust when used with new and unseen log files.

tokens relative to the target event templates. This can lead to instances where the models struggle to identify the correct sequence in their template generation. In the metric-based evaluation, the accuracy and edit distance of inferred event templates are determined by syntactic alignment with ground truth templates. This necessitates a thorough post-processing procedure to extract the correct part of the generated text to achieve the template. The challenge intensifies due to the absence of a standardized pattern in the generation of extraneous tokens by these models, rendering the post-processing not only challenging but also time-intensive.

When leveraging GPT-4 to evaluate inferred templates, its extensive pre-training on code and logs provides a significant advantage. This large-scale LLM brings a depth of understanding to the domain, allowing to evaluate beyond mere syntactic correctness and incorporate semantic interpretation. Through in-context learning, this LLM is equipped to perform assessments with a level of expertise, recognizing the specific, relevant components within the templates. This proficiency enables a more nuanced and accurate analysis of the inferred log data, like the evaluation expected from a human expert. Consequently, the integration of LLM-based evaluation alongside metric-based method as a complementary strategy enhances the precision of the assessment.

Table IV presents LLM-based evaluation results for accuracy and robustness, providing scores on a scale from 0 to 5 for the Mistral-7B and GPT-4 LLMs under various configurations. The Mistral-7B, in its zero-shot form, starts with lower mean scores for accuracy and robustness, but these scores improve in the two-shot setting. The most significant enhancement is observed when Mistral-7B is fine-tuned, reflected by near-perfect mean scores and a consistent median of 5.0, indicating highly reliable performance. On the other hand, GPT-4 demonstrates strong capabilities from the outset in the zero-shot setting, with both accuracy and robustness mean scores exceeding 4.2 and a median of 5.0, suggesting a considerable degree of inherent robustness. When given additional context in the two-shot configuration, GPT-4's mean scores rise slightly, suggesting marginal gains with more information. This LLM-based evaluation underscores the significant impact of fine-tuning on the Mistral-7B and the robust initial performance of the GPT-4, which is further enhanced with two-shot learning.

A comparison between the results of metric-based evaluation (Figures 14, 15, and 16) and LLM-based evaluation (Figures 12 and 13) shows why we need the LLM-based approach as a complementary approach. Both approaches confirm that Mistral-7B in fine-tuned setting and GPT-4 in two-shot setting are too close and have the highest accuracy and robustness. However, in metric-based, fine-tuned Mistral-7B shows a better performance and in the LLM-based approach GPT-4 is slightly better. Also, Mistral-7B exhibits superior accuracy and robustness when operating in a zero-shot setting compared to two-shot setting. Conversely, in LLM-based assessment the results are different. Here, the two-shot setting demonstrates enhanced accuracy and robustness. The reason is that GPT-4 has the ability to analyze both syntax and semantics. This capability ensures that the LLM's judgement remains robust against the absence of an intense post processing.

Finding RQ3: The LLM-based results confirm that fine-tuned Mistral-7B achieves a similar result than GPT-4 both in terms of accuracy and robustness. This result demonstrates that a fine-tuned compact LLM such as Mistral-7B can be effectively used as an alternative solution to GPT-4, a large scale LLM.

V. Discussion

Efficiency vs. Effectiveness: In this study, we carried out a comprehensive evaluation to compare the effectiveness in terms of accuracy and robustness of a fine-tuned compact LLM with a large-scale LLM. However, it is worth noting that one of the critical metrics for comparing log parsers is their efficiency

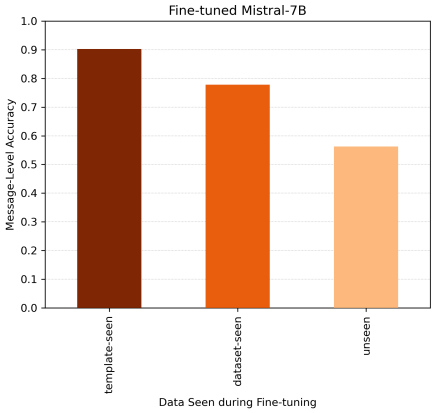

Fig. 9: Message level accuracy of fine-tuned Mistral-7B across different familiarity levels of the test data

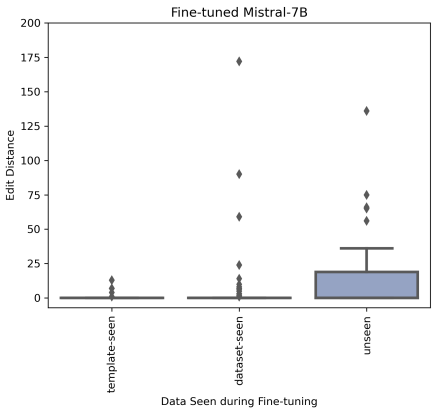

Fig. 10: Edit distance of fine-tuned Mistral-7B model across different familiarity levels of the test data

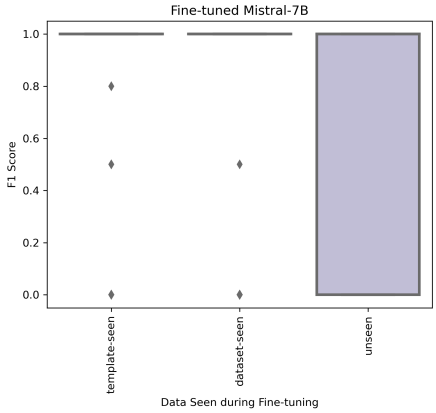

Fig. 11: F1-score of fine-tuned Mistral-7B model across different familiarity levels of the test data

TABLE IV: GPT-4 evaluated accuracy and robustness scores

Model	Accuracy		Robustness	
	Mean	Median	Mean	Median
Mistral-7B (0-shot)	2.23	2.0	2.50	2.0
Mistral-7B (2-shot)	3.34	4.0	3.25	4.0
Mistral-7B (Fine-tuned)	4.45	5.0	4.44	5.0
GPT-4 (0-shot)	4.21	5.0	4.40	5.0
GPT-4 (2-shot)	4.61	5.0	4.63	5.0

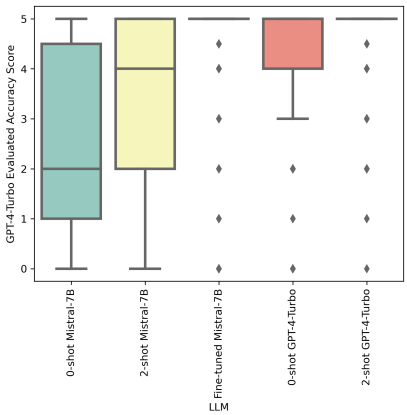

Fig. 12: Accuracy Score assessment by GPT-4 as evaluator

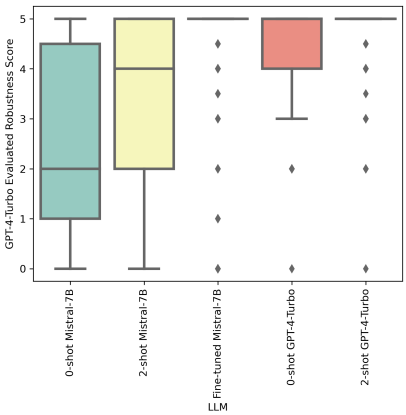

Fig. 13: Robustness Score assessment by GPT-4 as evaluator

(specifically in terms of running time). Since OpenAI employs advanced hardware for its operation, whereas our inference is conducted on standard platforms, a direct comparison of the efficiency between these two LLMs is not fair.

Generalizability of the findings: We experimented with Mistral-7B and GPT-4. We also worked with the log files of the LogPai benchmark, which covers a range of software systems. We need to experiment with more LLMs and log files to claim generalizability of the results. This is the main threat to validity of our this study.

Hyperparameter Tuning: While the fine-tuned Mistral-7B with our carefully curated log parsing instruction dataset demonstrates the promise of fine-tuning smaller LLMs for log parsing applications, it encourages us to further optimize the fine-tuning process in order to explore the boundaries of a compact LLM adapting to log parsing tasks in various scenarios. In particular, rank of decomposition, learning rate, and scaling factor are among the most important LoRA hyperparameters to be optimized. This represents an avenue for future work.

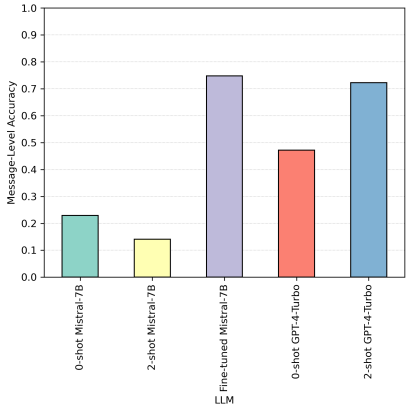

Fig. 14: MLA for Mistral-7B and GPT-4 in different settings

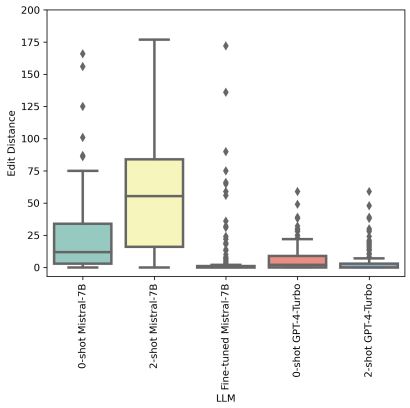

Fig. 15: Edit Distance for Mistral-7B and GPT-4 in different settings

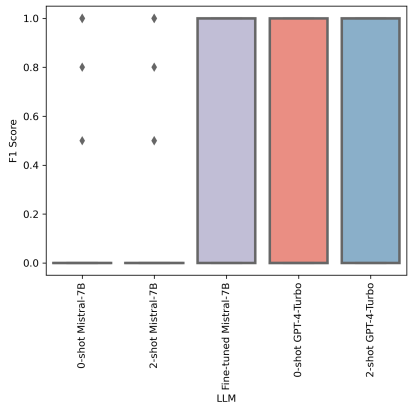

Fig. 16: F1 Score for Mistral-7B and GPT-4 in different settings

Training dataset: The training dataset influences the performance of the fine-tuned LLM. Given the presence of several inconsistencies within the ground truth log files used in this paper, the preparation of a consistent labeled dataset for fine-tuning and pre-training on log data would be beneficial.

VI. Related Work

There are several log parsing tools that use an algorithmic approach [9]. In this section, we only discuss LLM-based log parsing approaches.

Le and Zhang [16] conducted a study into the use of ChatGPT for log parsing. The authors experimented with different configurations, including zero-shot and multi-shot learning, as well as varying the prompts, to assess their impact on the LLM's performance. Xu et al. [18] presented LogDiv, a GPT-3 framework that is adapted through in-context learning to log parsing. The authors implemented a method for the selection of examples for in-context learning, as opposed to manual selection. For few-shot prompting, a pool of distinct samples was created. By receiving a log message as input, the framework automatically selects pertinent examples based on a similarity score, facilitating the LLM's parsing capability.

Liu et al. [17] introduced LogPrompt, a log parsing approach designed for zero-shot scenarios, with the aim of enhancing robustness when encountering unseen data. The authors' method relied on ChatGPT, demonstrating its adaptability. The authors proposed three distinct prompt strategies to optimize performance, namely, the self-prompt strategy, the Chain-of-thought, and the In-context prompt strategy. Jiang et al. [19] introduced LILAC, a framework to address the challenges related to the generation of unstable output templates and the high computational resources required by large-scale general-purpose LLMs for log parsing. LILAC combines in-context learning with an adaptive cache mechanism. A candidate sampling algorithm is used to select examples during in-context learning. Guo et al. [38] addressed the issues of high inference times and the generation of unstable results associated with LLM-based log parsers by proposing Lemur, a LLM-based framework for log parsing. To mitigate these issues, they integrated an entropy sampling approach for clustering log messages and identifying variable components within logs for template generation. Furthermore, they used a chain-of-thought method to merge these templates effectively. Le and Zhang [23] introduced LogPPT, a log parser leveraging prompt-based few-shot learning. LogPPT incorporates an Adaptive Random Sampling algorithm designed to select a subset of log data and their ground truth to serve as training samples. The authors used the fact that most keywords within log statements are valid, dictionary-lookup words, and thus more readily predictable by a language model. This approach contrasts with the challenge of predicting parameters, which are inherently dynamic and less predictable. Consequently, the task of log parsing is redefined as a label token prediction problem, and a pre-trained language model, specifically RoBERTa, is used to predict a designated label token at the parameter positions within log messages.

VII. Conclusion

We examined the use of a fine-tuned compact LLM for log parsing, as a viable alternative to large-scale general-purpose LLMs. To this end, we fine-tuned Mistral-7B, a compact LLM, with various log parsing datasets. Then, we evaluated its accuracy and robustness against GPT-4, a large-scale general-purpose LLM. Our evaluation framework uses two methods: a metric-based method and an LLM-based approach. The findings show that the performance of the fine-tuned compact LLM, Mistral-7B is on par with that of its larger general-purpose counterpart, GPT-4, supporting the fact that a smaller fine-tuned LLM can be used for log parsing as an alternative to a large-scale GPT-4 model.

VIII. Acknowledgment

This work was supported in part by funding from the Innovation for Defence Excellence and Security (IDEaS) program from the Canadian Department of National Defence (DND) and the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] B. Chen and Z. M. Jiang, "A survey of software log instrumentation," *ACM Computing Surveys (CSUR)*, vol. 54, no. 4, pp. 1–34, 2021.

[2] L. Yang, J. Chen, Z. Wang, W. Wang, J. Jiang, X. Dong, and W. Zhang, "Semi-supervised log-based anomaly detection via probabilistic label estimation," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 2021, pp. 1448–1460.

[3] X. Li, P. Chen, L. Jing, Z. He, and G. Yu, "Swisslog: Robust and unified deep learning based log anomaly detection for diverse faults," in *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2020, pp. 92–103.

[4] F. Lin, K. Muzumdar, N. P. Laptev, M.-V. Curelea, S. Lee, and S. Sankar, "Fast dimensional analysis for root cause investigation in a large-scale service environment," *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, vol. 4, no. 2, pp. 1–23, 2020.

[5] L. Wang, N. Zhao, J. Chen, P. Li, W. Zhang, and K. Sui, "Root-cause metric location for microservice systems via log anomaly detection," in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020, pp. 142–150.

[6] R. Ding, H. Zhou, J.-G. Lou, H. Zhang, Q. Lin, Q. Fu, D. Zhang, and T. Xie, "Log2: A {Cost-Aware} logging mechanism for performance diagnosis," in *2015 USENIX Annual Technical Conference (USENIX ATC 15)*, 2015, pp. 139–150.

[7] Q. Cheng, D. Sahoo, A. Saha, W. Yang, C. Liu, G. Woo, M. Singh, S. Saverese, and S. C. Hoi, "AI for IT Operations (AIOps) on cloud platforms: Reviews, opportunities and challenges," *arXiv preprint arXiv:2304.04661*, 2023.

[8] T. Zhang, H. Qiu, G. Castellano, M. Rifai, C. S. Chen, and F. Pianese, "System log parsing: A survey," *IEEE Transactions on Knowledge and Data Engineering*, 2023.

[9] D. El-Masri, F. Petrillo, Y.-G. Guéhéneuc, A. Hamou-Lhadj, and A. Bouziane, "A systematic literature review on automated log abstraction techniques," *Information and Software Technology*, vol. 122, p. 106276, 2020.

[10] 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)*. IEEE, 2019, pp. 121–130.

[11] 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)*. IEEE, 2017, pp. 33–40.

[12] H. Dai, H. Li, C.-S. Chen, W. Shang, and T.-H. Chen, "Logram: Efficient log parsing using n n-gram dictionaries," *IEEE Transactions on Software Engineering*, vol. 48, no. 3, pp. 879–892, 2020.

[13] S. He, P. He, Z. Chen, T. Yang, Y. Su, and M. R. Lyu, "A survey on automated log analysis for reliability engineering," *ACM computing surveys (CSUR)*, vol. 54, no. 6, pp. 1–37, 2021.

[14] K. Panchal, F. Ait-Mahammed, A. Hamou-Lhadj, Z. Zhu, S. Memon, A. Isac, and P. Krishnamoorthy, "A study on the use of runtime files in handling crash reports in a large telecom company," in *IEEE Future Networks World Forum*. Springer, 2022.

[15] Y. Li, Y. Huo, Z. Jiang, R. Zhong, P. He, Y. Su, and M. R. Lyu, "Exploring the effectiveness of LLMs in automated logging generation: An empirical study," *arXiv preprint arXiv:2307.05950*, 2023.

[16] V. Le and H. Zhang, "Log parsing: How far can ChatGPT go?" in *2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2023, pp. 1699–1704.

[17] Y. Liu, S. Tao, W. Meng, F. Yao, X. Zhao, and H. Yang, "Logprompt: Prompt engineering towards zero-shot and interpretable log analysis," in *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE)*. ACM, 2024.

[18] J. Xu, R. Yang, Y. Huo, C. Zhang, and P. He, "Prompting for automatic log template extraction," *arXiv preprint arXiv:2307.09950*, 2023.

[19] Z. Jiang, J. Liu, Z. Chen, Y. Li, J. Huang, Y. Huo, P. He, J. Gu, and M. R. Lyu, "LILAC: Log parsing using LLMs with adaptive parsing cache," *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, 2024.

[20] T. Ahmed, S. Ghosh, C. Bansal, T. Zimmermann, X. Zhang, and S. Rajmohan, "Recommending root-cause and mitigation steps for cloud incidents using large language models," in *Proceedings of the 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, p. 1737–1749.

[21] Y. Chen, H. Xie, M. Ma, Y. Kang, X. Gao, L. Shi, Y. Cao, X. Gao, H. Fan, M. Wen, J. Zeng, S. Ghosh, X. Zhang, C. Zhang, Q. Lin, S. Rajmohan, D. Zhang, and T. Xu, "Automatic root cause analysis via large language models for cloud incidents," in *Proceedings of the 19th European Conference on Computer Systems (EuroSys)*. ACM, 2024, p. 674–688.

[22] X. Zhang, S. Ghosh, C. Bansal, R. Wang, M. Ma, Y. Kang, and S. Rajmohan, "Automated root causing of cloud incidents using in-context learning with GPT-4," in *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE)*. ACM, 2024, p. 266–277.

[23] V. H. Le and H. Zhang, "Log parsing with prompt-based few-shot learning," in *Proceedings of the 45th International Conference on Software Engineering (ICSE)*. IEEE, 2023, p. 2438–2449.

[24] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized LLMs," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[25] S. Samsi, D. Zhao, J. McDonald, B. Li, A. Michaleas, M. Jones, W. Bergeron, J. Kepner, D. Tiwari, and V. Gadepally, "From words to watts: Benchmarking the energy costs of large language model inference," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2023, pp. 1–9.

[26] A. Miranskyy, A. Hamou-Lhadj, E. Cialini, and A. Larsson, "Operational-log analysis for big data systems: Challenges and solutions," *IEEE Software*, vol. 33, no. 2, pp. 52–59, 2016.

[27] J. A. Baktash and M. Dawodi, "GPT-4: A review on advancements and opportunities in natural language processing," *arXiv preprint arXiv:2305.03195*, 2023.

[28] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang, "Pre-trained models for natural language processing: A survey," *Science China Technological Sciences*, vol. 63, no. 10, pp. 1872–1897, 2020.

[29] Y. Cao, S. Li, Y. Liu, Z. Yan, Y. Dai, P. S. Yu, and L. Sun, "A comprehensive survey of AI-Generated Content (AIGC): A history of generative AI from GAN to ChatGPT," *arXiv preprint arXiv:2303.04226*, 2023.

[30] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[31] Y. Liu, T. Han, S. Ma, J. Zhang, Y. Yang, J. Tian, H. He, A. Li, M. He, Z. Liu, Z. Wu, L. Zhao, D. Zhu, X. Li, N. Qiang, D. Shen, T. Liu, and B. Ge, "Summary of ChatGPT-related research and perspective towards the future of large language models," *Meta-Radiology*, vol. 1, no. 2, 2023.

[32] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[33] Q. Dong, L. Li, D. Dai, C. Zheng, Z. Wu, B. Chang, X. Sun, J. Xu, and Z. Sui, "A survey on in-context learning," *arXiv preprint arXiv:2301.00234*, 2022.

[34] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[36] I. Sedki, A. Hamou-Lhadj, O. Ait-Mohamed, and M. A. Shehab, "An effective approach for parsing large log files," in *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2022, pp. 1–12.

[37] S. Nedelkoski, J. Bogatinovski, A. Acker, J. Cardoso, and O. Kao, "Self-supervised log parsing," in *Machine Learning and Knowledge Discovery in Databases: Applied Data Science Track*. Springer, 2021, pp. 122–138.

[38] H. Guo, W. Zhang, A. Le, J. Yang, J. Liu, Z. Li, T. Zheng, S. Xu, R. Zang, L. Zheng *et al.*, "Lemur: Log parsing with entropy sampling and chain-of-thought merging," *arXiv preprint arXiv:2402.18205*, 2024.