Early Exploration of Using ChatGPT for Log-based Anomaly Detection on Parallel File Systems Logs

Chris Egersdoerfer
cegersdo@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

Di Zhang
dzhang16@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

Dong Dai
ddai@uncc.edu
University of North Carolina at Charlotte
Charlotte, NC, USA

ABSTRACT

Log-based anomaly detection has been extensively studied to help detect complex runtime anomalies in production systems. However, existing techniques exhibit several common issues. First, they rely heavily on expert-labeled logs to discern anomalous behavior patterns. But labelling enough log data manually to effectively train deep neural networks may take too long. Second, they rely on numeric model prediction based on numeric vector input which causes model decisions to be largely non-interpretable by humans which further rules out targeted error correction.

In recent years, we have witnessed groundbreaking advancements in large language models (LLMs) such as ChatGPT. These models have proven their ability to retain context and formulate insightful responses over entire conversations. They also present the ability to conduct few-shot and in-context learning with reasoning ability. In light of these abilities, it is only natural to explore their applicability in understanding log content and conducting anomaly classification among parallel file system logs.

ACM Reference Format:
Chris Egersdoerfer, Di Zhang, and Dong Dai. 2023. Early Exploration of Using ChatGPT for Log-based Anomaly Detection on Parallel File Systems Logs. In *Proceedings of the 32nd International Symposium on High-Performance Parallel and Distributed Computing (HPDC '23), June 16–23, 2023, Orlando, FL, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3588195.3595943

1 DESIGN AND PRELIMINARY RESULTS

1.1 Key Design and Implementation.

To detect the anomalies, we used a fixed size window to scan the incoming logs. For each log window, there are two primary parts: *Context Creation* and *Active Analysis*. Both of these are conducted by directly querying the LLM (i.e., ChatGPT) [3]. During *Context Creation*, we seek to maintain a useful memory of past logs (long-term context) to be used during the analysis of the current window of logs. This is done by requiring the LLM to constantly re-summarize the important system events described in historical logs. During *Active Analysis*, we use the previously described summary, as well as the immediate to-be-analyzed logs, as input to prompt the model to create an output which includes a prediction of current system status and a detailed description of reasoning behind the prediction. Additionally, we ask the model to produce a list of log indices in the current window which are anomalous in the same output.

The key benefits to our solution are twofold: 1) avoiding the need for expert labelled training data entirely; 2) creating highly interpretable and useful output which can be easily understood by system administrators.

"""
Progressively summarize the lines of conversation provided, adding onto the previous summary returning a new summary. If the summary is empty just disregard it because it does not yet exist.

DO NOT INCLUDE LOG INDEX NUMBERS IN THE SUMMARY. Only include general information about processes and events that are occuring in the logs.

Current summary:
{*summary*}
New lines of conversation:
{*new_lines*}
New summary:
"""

Figure 1: Context Creation summary prompt template

1.2 Context Creation

One of the primary limitations of today's LLMs is their re-initialization at each query, limiting their immediate context to the static token input length they were designed with. For instance, ChatGPT maximally contains 4096 tokens. To circumvent this issue, we maintain a summary-based memory which the LLM updates with each sequential window and carries to the next round of prompting. More specifically, when the first window of logs arrives, the LLM is prompted to summarize the content of the logs. When the next window of logs arrives, the LLM is prompted to create a combined summary of the summary from the previous window and the new log messages which have arrived. This process is repeated for each window of logs and provides useful information to the analysis of the current window regarding what has happened in previous windows. In practice, the LLM often uses this information to recognize repetitive system behavior and even to classify anomalies as part of systemic issues as opposed to local errors. The specific prompt given to the LLM to generate and maintain this summary is shown in Figure 1.

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
HPDC '23, June 16–23, 2023, Orlando, FL, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0155-9/23/06.
https://doi.org/10.1145/3588195.3595943

1.3 Active Analysis

After creating the summary-based context, we include it as part of the active analysis prompt used for anomaly detection. Figure 2 shows the complete prompt template. Within this template, {*historic_summary*} represents the aforementioned summary-based context, {*window_size*} represents the number of logs in each log window, {*file_system*} represents the name of the file system from which the logs originate, and {*log_window*} represents the actual logs in the current window of logs. This prompt consistently returns highly accurate and detailed responses without log preprocessing or model training.

"""
You will be given a window of {*window_size*} {*file_system*} logs separated by newlines in addition to a summary of all previously seen windows of logs.

Based on the current window of logs and the provided summaries, you must predict whether the system is functioning normally or if it is in a critical state, along with a written description of your reasoning.

Additionally, if you find any lines of logs which seem to indicate errors or anomalies in the current window of logs, report the indices of these lines in a python style list and provide a very brief description of what the problems indicate. The format of the current window of logs will be "Index: Log Message"

The following format must be followed:
Window: <current window of logs>
Historical Window Summary: <summary of the current window of logs>
System State: <normal or critical>
Reasoning: <english language description of your reasoning for the prediction>
Anomalies: <list of log indices in the current window of logs which indicate errors or are anomalous; if all lines are anomalous, include all indices in the list; if there are no such lines, just write an empty list like this: []>
Anomaly Descriptions: <english language description of what the error or anomaly indicates; if there are no anomalies write 'N/A'>

Here is the first example:

Window: {*log_window*}
Historical Window Summary: {*historic_summary*}
"""

Figure 2: Active Analysis prompt template

Preliminary Results. Our Preliminary evaluation is set up similarly to common evaluation processes of previous SOTA anomaly detection frameworks. Specifically, we feed a sliding window of 10 logs at a time to our system and parse its four outputs. We use OpenAI's gpt-3.5-turbo model as our LLM and conduct evaluation on a dataset originating from the Lustre file system which has been used in previous work [4]. The results are compared to those of NeuralLog [2], DeepLog [1], and SentiLog. Note that, all of these approaches require varying amounts of pre-processing and model training, both of which are not required in our solution. The results presented in Fig 3 show that our system achieves the best performance among all three approaches. Additionally, our system provides a human interpretable explanation of each error it predicts as well as a written summary of processes recorded in the logs both of which have not been a feature of any previous or related approaches. These allow for far greater overall interpretability of the system and its faults as it is running. A sample of the historic summary as well as all four LLM prompt outputs are represented in Fig 4.

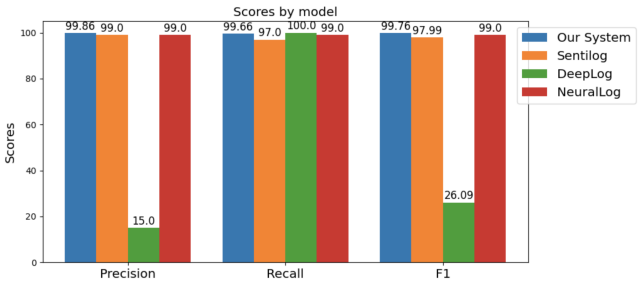
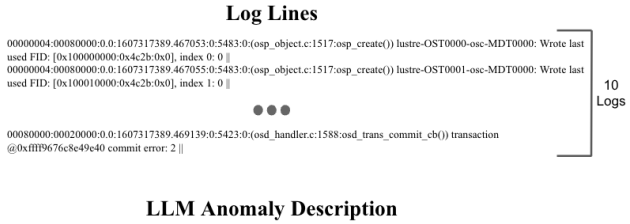


Figure 3: Comparison of performance with other approaches



LLM Anomaly Description

Historic Summary: Continuing from the previous logs, Lustre-OST0000, Lustre-OST0001, and Lustre-OST0002 continue to write their last used FIDs to MDT0000 with incremented indexes. Additionally, multiple transactions with commit errors are recorded in the logs. No other significant events are noted.
system health prediction: Critical
description of system health: The multiple transactions with commit errors indicate a problem with the Lustre file system's ability to commit changes. This could lead to data loss or corruption if not addressed immediately.
Anomalies: [0 0 0 1 1 1 1 1 1 1]
Description: The commit errors in the transactions indicate that Lustre is having issues writing changes to disk. This could cause data inconsistencies or loss and is a critical issue that needs to be addressed.

Figure 4: An output example.

Acknowledgment. We thank the anonymous reviewers for their insightful feedback. This work was supported in part by NSF under grants CNS-2008265 and CCF-1908843. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the view of NSF.

REFERENCES

[1] Min Du, Feifei Li, Guineng Zheng, and Vivek Srikumar. 2017. Deeplog: Anomaly detection and diagnosis from system logs through deep learning. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS'17)*.
[2] Van-Hoang Le and Hongyu Zhang. 2021. Log-based anomaly detection without log parsing. In *2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE'21)*.
[3] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.
[4] Di Zhang, Dong Dai, Runzhou Han, and Mai Zheng. 2021. SentiLog: Anomaly detecting on parallel file systems via log-based sentiment analysis. In *Proceedings of the 13th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage'21)*.