System Logs Anomaly Detection using Fine-Tuned LLMs

Yue Fang (yfang53@asu.edu)
Afshiya Roohi Shaik (ashaik89@asu.edu)
Sivasanker Nandakumar Padmapriya (snanda14@asu.edu)

1. Abstract

As software systems continue to grow, the volume of logs generated has also increased exponentially, making real time analysis of logs a crucial aspect in maintaining a stable and reliable system. Existing systems rely on predefined patterns which are explicitly programmed or trained to identify a certain kind of anomaly pattern, making it unreliable against unseen issues and needs manual intervention. This project aims to develop fine-tuned Large Language Models (LLMs), where we use its natural language understanding capabilities to classify logs as 'normal' or 'anomalous' without relying on predefined rules or patterns. We will also explore the computational cost, tradeoffs and practicality of LLM based anomaly detection in real time log analysis, by comparing it to widely adopted existing techniques. We propose a system that scales along with the evolving system with improved accuracy and minimal human interference.

2. Project Steps

2.1 Steps To Follow

a. **Data Collection & Preprocessing**
We will be using two labeled datasets Hadoop Distributed File System Logs (HDFS) and BlueGene/L Supercomputer Logs (BGL) to fine tune our LLMs. They are huge datasets and as part of training dynamic log fields are replaced and downsampling is done to reduce complexity and avoid model bias.

b. **Model Selection & Fine-Tuning LLMs**
We will be using two pretrained LLMs for our task: BERT (encoder) & Llama 2 3B (decoder). These models are trained on general text, so we have to fine tune them against logs in order to avoid model generalisation.

c. **Model Evaluation & Comparison**
We will be accessing the model performance by calculating Precision, Recall and F1 score. Also, we will compare the model against different transformer models like GPTs and the results are recorded.

d. **User Interface**
We will be developing an application using Tkinter which allows the user to upload a log file as input and the model detects anomalies.

e. **Report the findings**
Summarize insights, discuss computational requirements, cost, trade offs, and practical applications.

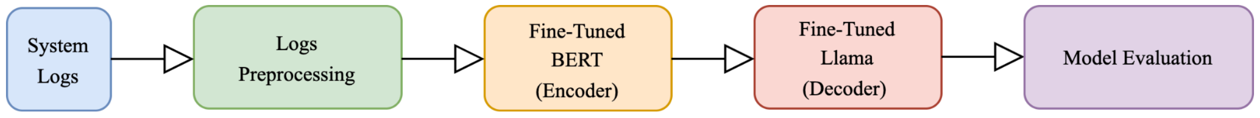

Figure 1: Log Anomaly Detection Pipeline.

2.2 Work Load Distribution

Table 1: Work load distribution.

Task	**Team Members**
Data collection, Data Preprocessing & UI development	Yue Fang
Fine-tune ‘BERT’ for feature extraction, model testing & evaluation	Afshiya Roohi Shaik
Fine-tune ‘Llama 2 3B’ for sequence analysis, model testing & evaluation	Sivasanker Nandakumar Padmapriya
Performance analysis and final report drafting	- Yue Fang - Afshiya Roohi Shaik - Sivasanker Nandakumar Padmapriya

2.3 Time Table

Table 2: Time table.

Task	**Weeks**
Finalize the project scope, roles & outline the requirements	Week 1
Collect and preprocess datasets	Weeks 2-3
Fine-tune LLM models	Weeks 4-6
Run, analyze result & conduct internal peer reviews	Week 7
Develop UI and integrate model	Week 8
Continuous report writing & document results	Week 9
Final report review & project submission	Week 10

2.4 Expected challenges and how to handle them

- **Computational bottlenecks**: There are a million log entries and fine-tuning LLMs like ‘Llama 2 3B’ can be computationally demanding.
  Mitigation: We will be applying downsampling and only a subset of the actual dataset is chosen. This makes model training more feasible.

- **Model Bias**: When training on imbalanced datasets, the model can become biased towards the majority class which can distort the model prediction.
  Mitigation: We will be adopting appropriate sampling methods to maintain a balanced ratio between ‘normal’ and ‘anomalous’ log.

3. Evaluation plan

3.1 Evaluate the outcome

To evaluate the model created, we need to make sure the model is not overfitting or under fitting. We can compare training metrics and validation metrics to find if the model is a perfect choice. Then a test dataset that is split from the original dataset will be used to test the model. After testing it, we will build a confusion matrix to know True positive, True negative, False positive, and False negative. Then we will calculate each metric, and some metrics will be picked and analyzed. The first one we choose is precision, which measures the proportion of predicted anomalies that are actual anomalies. For this model, the ideal value of precision will be greater than 0.9. The second is recall, which measures the proportion of actual anomalies correctly identified by the model. For this model, the ideal value of recall will be greater than 0.95. The third one is F1-score, which is the harmonic mean of Precision and Recall, providing a balanced performance measure. For this model, the ideal value of precision will be greater than 0.85. After analyzing these metrics, a ROC curve will also be created to help the team evaluate if the model is good enough. The area under the curve (AUC), should be higher than 0.5, and for this model, the ideal ROC AUC will be 0.8-0.9.

3.2 Analyze the results yielded by the method(s), the expected contributions or takeaways

To analyze the result yielded by our methods,

a. **Compare with existing models**: We will be comparing our model against pretrained transformer models like GPT, RoBERTa and verify whether they are outperformed. We will compare the metrics we get from each model, such as their precision, recall and F1-score to see which model is better. Based on these results, we will analyze the current models and make changes to our model to improve it.

b. **Error pattern**: We will be analyzing the error patterns to identify how often false positives and false negatives are found and the types of data that led to misclassification. After we find the reason, we will make improvements to our preprocessing techniques in order to decrease the number of false positives and false negatives.

c. **Real world practicality**: We will assess and document the real world feasibility of deploying our model and the computational requirement to analyze logs in real time. We will use the newest real logs to test the model to see if the model can work well to analyze these logs. If possible, we will test our model in a real situation, not just use existing logs to test it. These testing will help us to make sure the model can work in the real world. We can make improvements to our models based on these tests.

4. References

[1] BGL Dataset: LogHub. (n.d.). logpai/LogHub at master   logpai/loghub. GitHub.
https://github.com/logpai/loghub/blob/master/BGL

[2] HDFS Dataset: LogHub. (n.d.). logpai/LogHub at master · logpai/loghub. GitHub.
https://github.com/logpai/loghub/tree/master/HDFS

[3] Minghua He; Tong Jia; Chiming Duan; Huaqian Cai; Ying Li; Gang Huang. "LLMeLog: An approach for anomaly detection based on LLM-enriched log events." 2024 IEEE International Symposium on Software Reliability Engineering (ISSRE) IEEE, 2024.

[4] Wei Guan, Jian Cao, Shiyou Qian, Jianqi Gao, Chun Ouyang. "LogLLM: Log-based anomaly detection using large language models" arXiv, 2024.