XRAGLog: A Resource-Efficient and Context-Aware Log-Based Anomaly Detection Method Using Retrieval-Augmented Generation

Lingzhe Zhang[1], Tong Jia[1*], Mengxi Jia[2], Yifan Wu[1], Hongyi Liu[1], Ying Li[1*]

[1]Peking University, Beijing, China
[2]Institute of Artificial Intelligence, Beijing, China
zhang.lingzhe@stu.pku.edu.cn, jia.tong@pku.edu.cn, jiamx1@chinatelecom.cn, yifanwu@pku.edu.cn, hongyiliu@pku.edu.cn, li.ying@pku.edu.cn

Abstract

As large language models (LLMs) become more sophisticated and pervasive, ensuring their operational stability has become increasingly critical. Consequently, the need for accurate and reliable anomaly detection has grown significantly. Leveraging the rich semantic information within log data, LLMs have proven to be powerful tools for anomaly detection. However, existing log-based anomaly detection methods that utilize LLMs are resource-intensive and often overlook the interrelationships between log entries. To address these limitations, we propose a resource-efficient and context-aware log-based anomaly detection approach. This method combines hierarchical log compression with context-aware retrieval-augmented generation to enhance efficiency and accuracy. Experiments on various public and real-world datasets demonstrate that our approach significantly improves anomaly detection accuracy while dramatically reducing token consumption.

Introduction

With the rapid proliferation of large language models (LLMs), the complexity of their underlying software clusters has increased significantly, with growing interdependencies between components across clusters. A malfunction in one server can potentially trigger a cascade of failures throughout multiple servers, leading to instability in the entire LLM system (Diaz-De-Arcaya et al. 2024; Tantithamthavorn et al. 2025). Ensuring the stability and security of these systems requires timely fault detection and mitigation to prevent cascading failures. However, the sheer scale and complexity of modern software environments make manual fault diagnosis impractical and prone to errors. As a result, AI-based automated anomaly detection algorithms have become essential in addressing these challenges (Lin et al. 2020; Kang et al. 2022).

System logs meticulously track the states and significant events of actively running processes, making them a valuable resource for anomaly detection. However, log data is often vast and rich in semantic information, making it resource-intensive and time-consuming to train and predict using such data. Additionally, different systems typically have unique log characteristics, which means that operations personnel must train distinct log-based anomaly detection models for each system, significantly limiting the practical implementation of these models in industrial settings (Zhang et al. 2024a,b,c).

Fortunately, large language models (LLMs), pre-trained on natural language understanding tasks, offer a promising solution to these challenges. Due to their inherent strong semantic analysis and logical reasoning capabilities, many recent approaches have leveraged LLMs for log-based anomaly detection, thereby avoiding the resource-intensive training processes typically required (Qi et al. 2023; Pan, Wong, and Yuan 2023; Zhang et al. 2023; Egersdoerfer, Zhang, and Dai 2023; Chen et al. 2024; Zhang et al. 2024d). These works utilize LLMs in various ways for log-based anomaly detection. Chris et al. (Egersdoerfer, Zhang, and Dai 2023) conducted a study on zero-shot log-based anomaly detection using ChatGPT. LogPrompt (Zhang et al. 2023) employs self-explanation prompts to guide the pre-trained language model in better understanding the semantic and sequential information in logs. RAGLog (Pan, Wong, and Yuan 2023) enhances LLM-based anomaly detection by integrating retrieval-augmented generation techniques with vector databases, thereby improving detection accuracy. LogGPT (Qi et al. 2023) utilizes chain-of-thought (CoT) prompting to boost the LLM's performance in anomaly detection, enabling more nuanced and effective analysis.

Although these LLM-based anomaly detection methods have shown promising results, they still face the following practical challenges when applied to industrial scenarios:

- **Resource-Intensive:** Current methods typically input the logs requiring detection directly into the LLM along with the prompt. However, given the massive volume of log data, this approach consumes a large number of tokens, leading to significant resource waste.
- **Lack of Consideration for Log Interrelationships:** Some approaches focus on determining whether a single log entry is anomalous without considering the interrelationships between different logs. In real industrial settings, detecting anomalies often requires analyzing the sequence of logs over a period of time to identify irregularities.

To fill this significant gap, we develop a resource-efficient

*Corresponding author

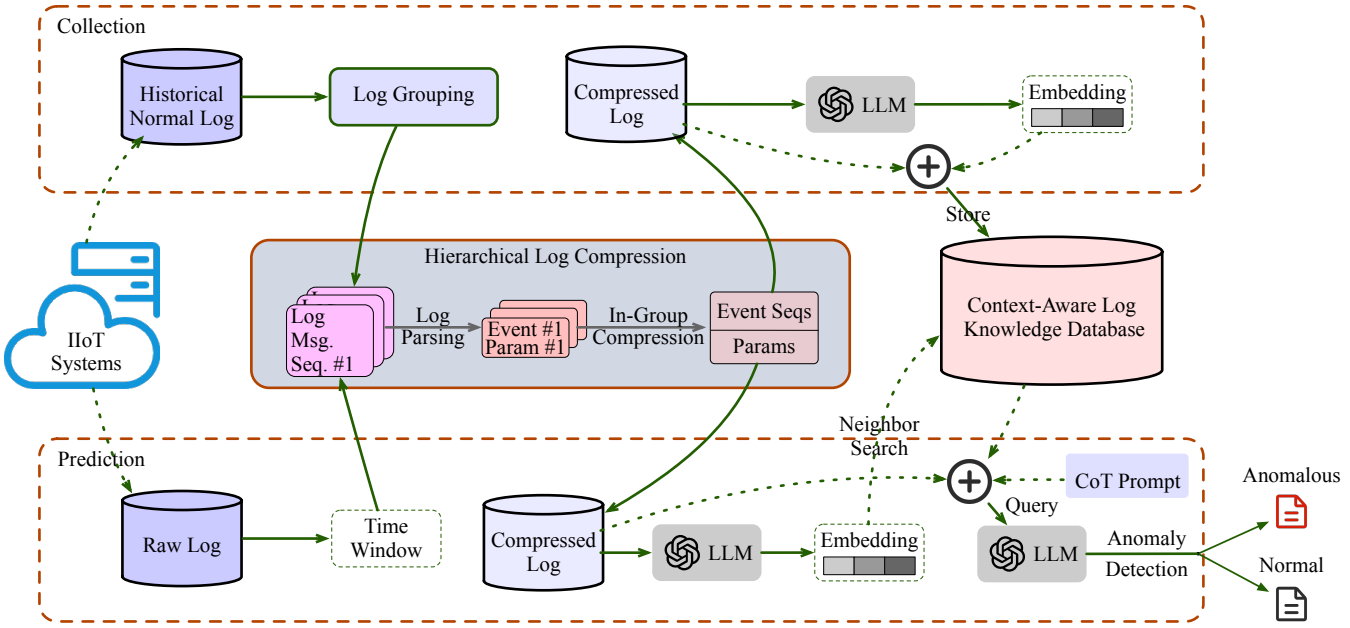

Figure 1: XRAGLog Architecture

and context-aware log-based anomaly detection method using retrieval-augmented generation, named XRAGLog. XRAGLog first employs a Hierarchical Log Compression method to perform compression of log data. The compression process begins with log parsing, extracting log templates and parameters, and then groups the logs together for in-group merging and compression. Next, XRAGLog creates a Context-Aware Log Knowledge Database using anomaly-free log groups, storing essential contextual information. Finally, XRAGLog compresses each input log group and queries the Context-Aware Log Knowledge Database for similar log contexts, merging them into a chain-of-thought prompt for querying the large language model.

We evaluate XRAGLog's effectiveness on various public datasets and real-world measured datasets. The evaluation results indicate that, compared to state-of-the-art models, XRAGLog improves the F1-score in anomaly detection by approximately 40%, while reducing token consumption by 90%. To summarize, our key contributions are as follows:

- We employ a Hierarchical Log Compression method to reduce prompt consumption during the llm-based anomaly detection process.
- We introduce a Context-Aware Log Knowledge Database to assist in retrieval-augmented generation, effectively taking log interrelationships into account.
- The effectiveness of XRAGLog is validated based on various public datasets and real-world measured datasets.

Methodology

The architecture of XRAGLog, as depicted in Figure 1, is divided into two main components: the collection part and the prediction part. The collection component is responsible for log grouping of historical normal logs, compressing each log group using Hierarchical Log Compression. The compressed logs are then processed using an LLM to compute embeddings, which are stored in the Context-Aware Log Knowledge Database, a vector database. The prediction component operates in real-time within the LLM system. After each fixed time window, the logs within that window are compressed using Hierarchical Log Compression, and embeddings are computed using the LLM. These embeddings are used to perform a neighbor search in the Context-Aware Log Knowledge Database to find similar compressed normal logs. Finally, these logs, along with the current time window's logs, are used to construct a CoT prompt, which is input into the LLM for anomaly detection.

Hierarchical Log Compression

The Hierarchical Log Compression consists of two steps: first, log parsing is performed on the raw logs, followed by in-group compression of the parsed results.

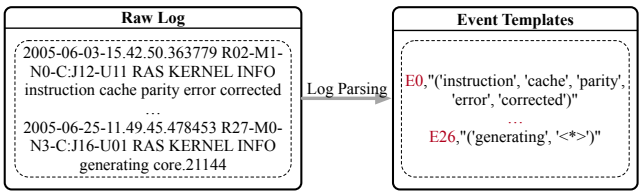

Figure 2: Log Parsing Example

Log Parsing Raw logs consist of semi-structured text encompassing various fields like timestamps and severity levels. For the benefit of downstream tasks, log parsing is employed to transform each log message into a distinct event template, which includes a constant part paired

with variable parameters. For example, the log template "E0,('instruction', 'cache', 'parity', 'error', 'corrected')" can be extracted from the log message "2005-06-03-15.42.50.363779 R02-M1-N0-C:J12-U11 RAS KERNEL INFO instruction cache parity error corrected" in figure 2. Formally, for a given log sequence $S = (s_1, s_2, ..., s_N)$, where s_n represents an individual log entry. After log parsing, each log s_n can be represented as an event e_i with corresponding parameters p_n. The entire collection of unique events is denoted as $\omega = \{e_1, e_2, ..., e_n\}$.

In-Group Compression Building on log parsing, we further compress the logs by merging consecutive events within a log group and then performing a secondary compression on the corresponding parameters of these events. Formally, given a log group $G = (e_1, e_2, ..., e_M)$, where e_i represents individual events, we first identify consecutive occurrences of the same event type and combine them into a single event with the number of occurrences denoted as $e'_i = e_i \times c$, the log group is then converted into $G' = (e'_1, e'_2, ..., e_C)$, where $C << M$. The parameters associated with these merged events are subsequently aggregated using a state-of-art NLP compression model for LLMs (Jiang et al. 2023b; Li et al. 2023; Laban et al. 2021; Pan et al. 2024; Jiang et al. 2023a), resulting in a more compact representation.

Knowledge Database Construction

The primary task of the collection component is to construct a Context-Aware Log Knowledge Database using Hierarchical Log Compression. This process begins with historical normal logs, which are first grouped based on a fixed time window. Each log group is then compressed using Hierarchical Log Compression. The resulting compressed logs are processed by an LLM to compute embeddings, which are stored in a vector database, referred to as the Context-Aware Log Knowledge Database.

Formally, given a set of historical normal logs $L = \{l_1, l_2, ..., L_N\}$, these logs are grouped into subsets G_i based on a fixed time window T, such that $G_i = \{l_j | t_j \in T_i\}$, where t_j is the timestamp of log l_j. Each group G_i is then compressed using Hierarchical Log Compression to obtain G'_i. The compressed logs G'_i are processed by an LLM to generate embeddings E_i, which are stored in the Context-Aware Log Knowledge Database $D = \{E_1, E_2, ..., E_M\}$.

Real-Time Anomaly Detection

XRAGLog is ultimately deployed online for real-time anomaly detection using log data. At the end of each fixed time window T, the log data $L_T = \{l_1, l_2, ..., l_N\}$ is compressed using Hierarchical Log Compression. The compressed logs L'_T are then processed by an LLM to compute embeddings E_T. A k-nearest neighbors search is conducted using E_T within the Context-Aware Log Knowledge Database D to identify the most similar embeddings $\{E_{T1}, E_{T2}, ..., E_{Tk}\}$ as well as corresponding compressed normal log $\{G'_{T1}, G'_{T2}, ..., G'_{Tk}\}$, which are then used for anomaly detection in the current time window T.

Finally, we concatenate the compressed log intended for anomaly detection with the similar normal compressed logs retrieved from the database into a CoT prompt for querying. The complete prompt structure is illustrated in Figure 3. The LLM is queried using a function call, and to enhance the interpretability of the generated results, we also ask the LLM to provide coresponding thought process when determining whether an anomaly is present.

Context:
Your task is to determine if a given set of log messages contains an anomaly or not (Sorted by timestamp).
We will provide you with the following logs and ask you to determine if they contain any anomalies.
Additionally, we will provide the most semantically similar normal log for each log entry. Be careful to consider the contextual information of the log sequence.
Use the following format:
Logs: Given a set of log messages here. (a python list of log template id with parameter, each template id will be given corresponding template content at the end)
Similar Logs: Given a set of log messages here. (a python list of log template id with parameter, each template id will be given corresponding template content at the end)
Answer: yes or no (Output your **thought process**)
Input:
Logs: [Log Template Sequence] [Parameters] [Template ID: Content]
Similar Logs: {[Log Template Sequence] [Parameters] [Template ID: Content]} $\times k$

Figure 3: The prompt for anomaly detection

Experiments

Experimental Setup

We evaluate XRAGLog on 3 datasets collected by loghub(He et al. 2023): HDFS, BGL and Thunderbird. We utilize GPT-3.5 as our LLM. We compare XRAGLog with LogGPT (Qi et al. 2023), LogPrompt (Zhang et al. 2023), and RAGLog (Pan, Wong, and Yuan 2023). To demonstrate the advantages of our log compression algorithm, we also evaluate the performance of these approaches when the prompts are compressed using selective context (SC) (Li et al. 2023), llmlingua-2 (Pan et al. 2024), and KIS (Laban et al. 2021) algorithms before anomaly detection.

Experiments Results and Analysis

Model Effectiveness We first conduct a comprehensive evaluation of the model's overall performance. As shown in Figure 1, XRAGLog consistently outperforms other methods, achieving higher F1-scores across all three datasets, even when compared to methods combined with any NLP-based compression techniques. However, for the HDFS dataset, XRAGLog does not achieve the highest precision and recall. In this dataset, the next best performer is the combination of RAGLog and llmlingua-2, with an F1-score only 1.37% lower than that of XRAGLog. Notably, while LogGPT achieves the highest precision, it has significantly lower

Table 1: Overall Evaluation Results

Method	HDFS			BGL			Thunderbird		
	P	***R***	***F1***	***P***	***R***	***F1***	***P***	***R***	***F1***
LogGPT	0.9487	0.3162	0.4744	0.4359	0.6296	0.5152	0.0513	0.1053	0.0689
LogGPT+SC	**0.9744**	0.3363	0.5000	0.8462	0.3084	0.4521	0.5385	0.3962	0.4565
LogGPT+llmlingua-2	0.9231	0.3103	0.4645	0.2564	0.1587	0.1961	0.3077	0.2308	0.2637
LogGPT+KIS	0.6667	0.2500	0.3636	0.3333	0.1646	0.2203	0.0513	0.2222	0.0833
LogPrompt	0.7692	0.3750	0.5042	0.8462	0.6111	0.7097	0.3846	0.2632	0.3125
LogPrompt+SC	0.6667	0.3824	0.4859	0.5385	0.3750	0.4421	0.3333	0.3023	0.3171
LogPrompt+llmlingua-2	0.6410	0.3571	0.4587	0.7949	0.3647	0.5000	0.4359	0.3400	0.3820
LogPrompt+KIS	0.7692	0.3529	0.4839	0.5128	0.2857	0.3669	0.5897	0.3594	0.4466
RAGLog	0.7436	0.7632	0.7532	0.6410	0.9259	0.7576	0.1538	0.1364	0.1446
RAGLog+SC	0.8205	0.8421	0.8312	0.7179	0.3111	0.4341	0.3077	0.2000	0.2424
RAGLog+llmlingua-2	0.7179	**0.9656**	0.8235	0.6154	0.2791	0.3840	0.4359	0.6800	0.5313
RAGLog+KIS	0.5897	0.3382	0.4299	0.1795	0.3333	0.2333	0.1282	0.4167	0.1961
XRAGLog(*Ours*)	0.9231	0.7659	**0.8372**	**0.8718**	**1.0000**	**0.9315**	**0.9487**	**0.9487**	**0.9487**

recall, indicating a high rate of missed detections.

Furthermore, while not surpassing XRAGLog, the utilization of NLP-based compression methods generally improves anomaly detection performance in most cases. For instance, applying RAGLog with SC or llmlingua-2 on the HDFS dataset led to an approximately 7% improvement in effectiveness compared to using RAGLog alone. However, the effectiveness of these NLP-based compression methods varies across different datasets. While SC and llmlingua-2 enhance model performance on the HDFS and Thunderbird datasets, they both result in a decrease in F1-score on the BGL dataset.

Table 2: Reduced Token Ratio

Method	HDFS	BGL	Thunderbird
SC	1.5101	1.5598	1.4438
llmlingua-2	2.5183	2.3205	2.4612
KIS	1.2972	1.2286	1.2564
XRAGLog(*Ours*)	**3.0860**	**19.3658**	**3.7962**

Token Consumption In addition to improving model performance, XRAGLog significantly reduces resource consumption. We use the average number of tokens required to complete each anomaly detection as the metric for measurement. As shown in Figure 2, XRAGLog dramatically lowers token consumption compared to other methods, outperforming the second-best method, llmlingua-2, by 22.64%, 734.55%, and 54.24% across the three datasets. Notably, the improvement on the BGL dataset is particularly pronounced, as it frequently contains consecutive identical events that can be easily merged.

Conclusions

In this work, to address the issues of resource intensity and lack of contextual log analysis in current log-based anomaly detection methods using LLMs, we propose a resource-efficient and context-aware log-based anomaly detection approach. This method leverages a unique hierarchical log compression and context-aware retrieval-augmented generation. Experiments on various public datasets demonstrate that our method significantly enhances anomaly detection accuracy while drastically reducing token consumption.

Acknowledgement

We sincerely appreciate the invaluable support provided by the Huawei team in the course of this work. This research was supported by the PKU-Huawei Cooperation Research Project.

References

Chen, Y.; Xie, H.; Ma, M.; Kang, Y.; Gao, X.; Shi, L.; Cao, Y.; Gao, X.; Fan, H.; Wen, M.; et al. 2024. Automatic root cause analysis via large language models for cloud incidents. In *Proceedings of the Nineteenth European Conference on Computer Systems*, 674–688.

Diaz-De-Arcaya, J.; López-De-Armentia, J.; Miñón, R.; Ojanguren, I. L.; and Torre-Bastida, A. I. 2024. Large Language Model Operations (LLMOps): Definition, Challenges, and Lifecycle Management. In *2024 9th International Conference on Smart and Sustainable Technologies (SpliTech)*, 1–4. IEEE.

Egersdoerfer, C.; Zhang, D.; and Dai, D. 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*, 315–316.

He, S.; Zhu, J.; He, P.; and Lyu, M. R. 2023. Loghub: A large collection of system log datasets towards automated log analytics.

Jiang, H.; Wu, Q.; Lin, C.-Y.; Yang, Y.; and Qiu, L. 2023a.

Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*.

Jiang, H.; Wu, Q.; Luo, X.; Li, D.; Lin, C.-Y.; Yang, Y.; and Qiu, L. 2023b. Longllmlingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *arXiv preprint arXiv:2310.06839*.

Kang, Y.; Huang, X.; Song, S.; Zhang, L.; Qiao, J.; Wang, C.; Wang, J.; and Feinauer, J. 2022. Separation or not: On handing out-of-order time-series data in leveled lsm-tree. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, 3340–3352. IEEE.

Laban, P.; Schnabel, T.; Bennett, P. N.; and Hearst, M. A. 2021. Keep It Simple: Unsupervised Simplification of Multi-Paragraph Text. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*, volume 1.

Li, Y.; Dong, B.; Lin, C.; and Guerin, F. 2023. Compressing Context to Enhance Inference Efficiency of Large Language Models. arXiv:2310.06201.

Lin, S.; Clark, R.; Birke, R.; Schönborn, S.; Trigoni, N.; and Roberts, S. 2020. Anomaly detection for time series using vae-lstm hybrid model. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 4322–4326. Ieee.

Pan, J.; Wong, S. L.; and Yuan, Y. 2023. RAGLog: Log Anomaly Detection using Retrieval Augmented Generation. *arXiv preprint arXiv:2311.05261*.

Pan, Z.; Wu, Q.; Jiang, H.; Xia, M.; Luo, X.; Zhang, J.; Lin, Q.; Rühle, V.; Yang, Y.; Lin, C.-Y.; et al. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *arXiv preprint arXiv:2403.12968*.

Qi, J.; Huang, S.; Luan, Z.; Yang, S.; Fung, C.; Yang, H.; Qian, D.; Shang, J.; Xiao, Z.; and Wu, Z. 2023. Loggpt: Exploring chatgpt for log-based anomaly detection. In *2023 IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application (HPCC/DSS/SmartCity/DependSys)*, 273–280. IEEE.

Tantithamthavorn, C. K.; Palomba, F.; Khomh, F.; and Chua, J. J. 2025. MLOps, LLMOps, FMOps, and Beyond. *IEEE Software*, 42(01): 26–32.

Zhang, L.; Jia, T.; Jia, M.; Li, Y.; Yang, Y.; and Wu, Z. 2024a. Multivariate Log-based Anomaly Detection for Distributed Database. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 4256–4267.

Zhang, L.; Jia, T.; Jia, M.; Yang, Y.; Wu, Z.; and Li, Y. 2024b. A Survey of AIOps for Failure Management in the Era of Large Language Models. *arXiv preprint arXiv:2406.11213*.

Zhang, L.; Jia, T.; Wang, K.; Jia, M.; Yang, Y.; and Li, Y. 2024c. Reducing Events to Augment Log-based Anomaly Detection Models: An Empirical Study. In *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 538–548.

Zhang, L.-Z.; Huang, X.-D.; Wang, Y.-K.; Qiao, J.-L.; Song, S.-X.; and Wang, J.-M. 2024d. Time-tired compaction: An elastic compaction scheme for LSM-tree based time-series database. *Advanced Engineering Informatics*, 59: 102224.

Zhang, T.; Huang, X.; Zhao, W.; Bian, S.; and Du, P. 2023. LogPrompt: A Log-based Anomaly Detection Framework Using Prompts. In *2023 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.