Leveraging Few-Shot In-Context Learning for Scaling Railway Log Anomaly Detection

Quentin Possamaï[1], Rajesh Bonangi[2], Alexandre Trilla[3], Ossee Josepha Charlesia Yiboe[4], Kenza Saiah[4], and Nenad Mijatovic[4]

[1] *Alstom, Villeurbanne, 69100, France*
quentin.possamai@alstomgroup.com

[2] *Alstom, Bengaluru, 560005, India*
rajesh.bonangi@alstomgroup.com

[3] *Alstom, Santa Perpètua de Mogoda, 08130, Spain*
alexandre.trilla@alstomgroup.com

[4] *Alstom, Saint-Ouen, 93070, France*
ossee-josepha-charlesia.yiboe@alstomgroup.com
kenza.saiah@alstomgroup.com
nenad.mijatovic@alstomgroup.com

ABSTRACT

This paper presents a scalable, data-driven approach for anomaly detection in railway signaling logs using Large Language Models (LLMs) and in-context learning. By classifying log keys — the structural templates of log messages — instead of individual messages, the method dramatically reduces the number of required model calls, thereby lowering computational costs (in terms of energy or monetary resources). Expert-labeled log keys are incorporated into LLM prompts to help the models differentiate between normal and abnormal log messages. Multiple state-of-the-art LLMs are evaluated on this task, revealing that performance increases as more labeled examples are added to the prompt, although the improvement gain diminishes with each additional label. Further analysis indicates that GPT-4.1 offers the best balance of monetary cost, response time, and F_1 score for this application. The study highlights both the advantages and limitations of in-context learning for railway log anomaly detection, notably its ability to leverage expert-labeled examples without additional model training, but also its sensitivity to data imbalance and exclusion of parameter values. It further discusses avenues for future improvement, such as model fine-tuning, prompt enrichment with additional contextual information, and the potential use of Retrieval-Augmented Generation (RAG) or self-feedback strategies to enhance classification performance.

1. INTRODUCTION

Railway signaling is a critical component of systems designed to manage railway traffic and ensure the safety and efficiency of transportation services. One of the main implementations of a signaling solution is the train control system, which supervises and manages trains by collecting data from the tracks and transmitting it to wayside processing units for real-time decision-making as presented in (*European Rail Traffic Management System*, 2018).

In the event of a system failure, time is of the essence, as service is disrupted and penalties may be incurred. A dedicated team of expert troubleshooters is then deployed to debug the railway computer. Moreover, as the system grows in functionality and complexity, the diagnosis team often faces a wide range of potential root causes interspersed with millions of textual messages produced sequentially by the Train Control system also referred to as *logs* (Ding, Yang, Hu, & Liu, 2017; Klumpenhouwer & Shalaby, 2022; Mohamad, Hashim, Abdul Hamid, & Ismail, 2021; Mannhardt & Landmark, 2019). In those records, the most recent events within a defined historical window are archived and sent for troubleshooting. As a consequence, the distribution of the studied time series of logs is consistently influenced by abnormal system behavior. Moreover, their sheer volume challenges the time and effort required to identify the root cause of a failure.

Our paper introduces a data-driven approach to efficiently identify the most abnormal events from real-world production logs, thereby narrowing the search space and reducing investigation time. By harnessing the embedded knowledge of pre-trained machine learning models and their in-context learning abilities, we assess the effectiveness and cost-efficiency of large language models for scalable anomaly detection. Our method minimizes the number of required queries by grouping structurally similar events and leveraging feedback from previously human-labeled examples, as illustrated in figure 1. Initially, log groups are identified through log parsing. These groups — referred to as log keys — are first partially labeled by expert troubleshooters, with the classification process then extended using an LLM via in-context learning. Our empirical results show that including more labeled log keys in the prompt enhances performance, though the improvement in F_1 score diminishes as more labels are added.

Our paper is organized as follows: Section 2 reviews related work in log parsing and various log analysis methods used to detect root causes or anomalies. Section 3 describes the proposed methodology: parsing the logs, constructing a human-labeled dataset, and conducting experiments to classify log keys. Section 4 presents classification performance as a function of labeled input, and compares model costs. Finally, Section 5 discusses the implications of the findings, our method limitations, and suggests directions for improvement.

2. RELATED WORK

This section reviews the main approaches to automatic log analysis.

Log Parsing — System logs are structured with a fixed pattern and variable parameters (e.g. IP addresses, memory pointers...). Several studies (Yu, He, Chen, & Wu, 2023; Du & Li, 2016) aim to separate them for following analysis such as grouping logs by their common pattern and analyzing peculiar parameters. A parser benchmark in (He, Zhu, He, Li, & Lyu, 2016; Zhu et al., 2019) tests them on various log datasets coming from operating systems or popular applications.

Normal Log Sequence Modeling — A popular log anomaly detection method models the evolution of log keys and parameters over time while the system behaves normally, leveraging prediction error when an abnormal event occurs (Du, Li, Zheng, & Srikumar, 2017). However, according to our domain knowledge, the nature of the studied logs differs as anomalies are not sequence-based but instead depend solely on the semantics of the message.

Causal Graphs — Another area of research aims to automatically discover causal graphs from event sequences to determine potential root causes as presented in (Kobayashi,

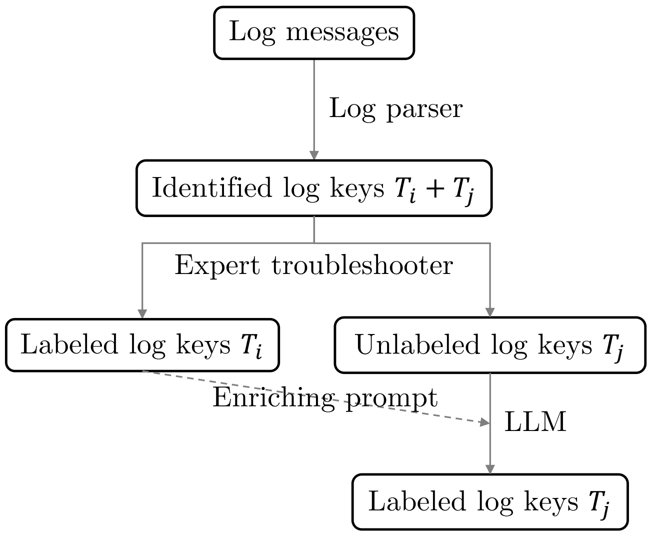

Figure 1. Log keys labeling workflow merging human and large language model classification

Otomo, Fukuda, & Esaki, 2018). It first uses log parsers to extract log keys and uses the Peter-Clark (PC) algorithm on a sequence of templates to identify a causal graph described in (Spirtes, Glymour, & Scheines, 2001). However, the number of log keys exceeds the number of failure occurrences by an order of magnitude which decreases the identifiability of the graph. Furthermore, this method is prone to false positives and does not leverage the semantics of the message.

Semantic-based anomaly detection — The combination of the log message semantic with modeling the normal system behavior has been studied in (Meng et al., 2019) and (Huang et al., 2020). The latter uses a pre-trained Bidirectional Encoder Representations from Transformers (BERT) model, as in (Devlin, Chang, Lee, & Toutanova, 2019), to extract the semantic meaning of the log messages. It then chooses a transformer model that takes sequences of embedded templates and parameters for anomaly classification.

Large scale multimodal models — While a lot of benchmarks evaluate LLMs on reasoning, coding, and natural language understanding tasks as in (Hendrycks et al., 2020), several works evaluate LLMs on parsing, mining, anomaly detection, event prediction, and summarization on system logs as in (Y. Liu et al., 2024) and in (Mudgal & Wouhaybi, 2024). The latter empirically shows strong performances on log parsing but is limited to other tasks. Both use GPT 3.5 Turbo but do not consider the costs of calling a large language model for each (or a batch of) log messages.

In-context learning — Other work demonstrated that LLMs can learn to perform and improve on tasks by adding a few examples of the task and its solution in the prompt (Brown et al., 2020). The model can get increased performances for the task without any additional weight updates. This is particularly useful for tasks where labeled data is scarce, expensive to obtain, or with few computational resources compared with fine-tuning or Low-Rank Adaptation (LoRA) as in (Hu et al., 2021).

Moreover, (Lewis et al., 2020) introduces the RAG framework, which combines a retriever and a generator to improve the performance of LLMs on knowledge-intensive tasks by enriching the prompt for text generation with relevant information from a large corpus of documents. Leveraging RAG, some works propose a log anomaly detection method that combines a log parser with a context-aware RAG knowledge database to enrich the prompt of a text generation model (Zhang et al., 2025) and (Pan, Liang, & Yidi, 2024). RAG is especially useful when the prompt reaches the maximum context size by retrieving the most pertinent knowledge.

3. METHOD

As presented in figure 1, due to the semi-structured nature of the logs, the method presented in this document proposes to build the anomaly detection system by first applying a log parser to group messages by their longest common structure, known as the log key, while ignoring associated parameters. Expert troubleshooters then classify each log key as either normal or abnormal, thereby constructing a labeled dataset. Finally, a pre-trained LLM is deployed to discriminate anomalous log key patterns. To this end, several LLMs are benchmarked on the anomaly classification task using a subset of labeled log keys to enrich the prompt and another subset as ground truth for a test set in a cross-validation configuration. While smaller LLMs have been tested, only Generative Pre-trained Transformer (GPT) neural networks — GPT-4.1, GPT-4o, GPT-o1, and GPT-o1-mini — are included in the study due to having enough output structure stability. At the time of the experiments, other competitive models such as Gemini or Claude were not available in the company's environment.

3.1. Log Parsing

Since the method classifies templates rather than unique messages, it reduces the number of expensive calls to the model by several orders of magnitude.

Each event includes a timestamp and a textual message. The events follow the RFC 5424 Syslog protocol, similar to Linux system logs (Gerhards, 2009). Logs are generated using a hard-coded sequence of words with varying parameters. For example, `Start education on UDP {IP}:{port} {IP_mask1}<=>{IP_mask2}` where the IP, port, and IP masks are parameters and the rest constitute the true log key.

To identify log keys, the literature proposes several unsupervised algorithms for automatically extracting the fixed structure. The following work (Yu et al., 2023) introduces a new log parser called Brain, which achieves strong performance on the LogPAI log parser benchmark (He et al., 2016; Zhu et al., 2019).

In the Brain algorithm, a *word* is defined as a sequence of characters separated by a whitespace, and the *position* of a word is its index in the log message. A *pattern* is a sequence of word-position pairs shared across a group of logs. The algorithm groups logs based on the longest common pattern and then refines it using constant words. Once finished we found the patterns of the identified log keys. Each log key is converted to a regular expression to match new logs. Grouping is performed on a training set and evaluated on unseen data to count unmatched logs. This method tends to cluster logs that share common (word, position) pairs. To improve log key quality and interpretability, frequent irrelevant words — such as isolated pipes `|`, `info`, `warning` ... — are removed for grouping.

3.2. Log Key Human Labeling

One advantage of using log keys for classification is their interpretability by domain experts. The removed parameters are replaced by a wildcard, `<*>`, at their respective positions; for example, `Start education on UDP <*>:<*> <*><=><*>`.

Human feedback was collected via a custom web application, allowing expert troubleshooters to label a subset of the log keys as either normal or abnormal, along with textual justifications. These labeled log keys are subsequently used for enriching the prompt and evaluating the classifier.

3.3. Log Key Classification

Log key classification is performed by prompting the LLM and parsing its output to obtain the predicted class. The prompt is composed of two sequential parts. The first is a labeled set in which each entry consists of the index, the log key, the expert explanation, and the ground truth label (abnormal or not). The second is a test set where each entry includes only the index and the log key. The prompt concludes with an instruction asking the LLM to generate a list of predictions, each containing the index and a boolean indicating whether the corresponding test log key is abnormal.

4. RESULTS

For log parsing, 468 log keys were discovered among approximately five million logs. The parser was then tested on around two million unseen logs to estimate the number of ungroupable messages. With only 0.41% of messages unable to be matched due to a new log structure appearing, we assume parsing to be acceptable. Then, out of 468 log keys, 212 were labeled by expert troubleshooters.

Regarding anomaly classification, GPT-4.1 (2025-04-14), GPT-4o (2024-05-13), GPT-o1 (2024-12-17), and GPT-o1-mini (2024-09-12) are benchmarked to classify log keys as normal or abnormal. The dates in parentheses indicate the models' release. GPTs are multimodal text generation models achieving strong performances on massive multitask language understanding (MMLU), biology, physics, chemistry (GPQA), maths (MATH and MGSM), code generation (HumanEVAL), and complex question reasoning (DROP) benchmarks as presented in (OpenAI et al., 2024; OpenAI, 2025, 2024b, 2024c, 2024a). GPT-4.1 is mainly significantly better on coding tasks than GPT-4o. GPT-o1 outputs reasoning tokens helping itself to generate the answer leading to improved results on MMLU. Finally, GPT-o1-mini is smaller than o1 and is designed for cost-efficiency.

4.1. Experiments

Due to the stochastic nature of LLMs, the binary classification experiment is repeated using four random seeds for splitting the data. For each seed, a 5-fold cross-validation (80/20%) is applied to a total of 212 labeled log keys, which are split into prompt and test sets. Prompt set sizes are linearly varied from 0 to 170 in 21 steps. For each model, 420 generations are performed (21 prompts $\times$ 5 folds $\times$ 4 seeds). Each model receives the same prompt and test set for a given seed, fold, and prompt size.

The model predicts a log key either being abnormal (positive) or normal (negative). The true positives (TP) are log keys correctly classified as abnormal, while false positives (FP) are normal log keys incorrectly classified as abnormal. Conversely, true negatives (TN) are normal log keys correctly classified as normal, and false negatives (FN) are abnormal log keys incorrectly classified as normal. The precision shows the portion of the correct positive predictions, while the recall shows the portion of the truly positive cases that are correctly detected. They are defined as

$$\text{precision} = \frac{TP}{TP+FP}, \quad \text{recall} = \frac{TP}{TP+FN} \tag{1}$$

Classification performance is evaluated using the F_1 score, a suitable metric for imbalanced datasets, as only 8% of labeled log keys are abnormal. The F_1 score is the harmonic mean of the precision and recall and is calculated as

$$F_1 = \frac{2}{1/\text{precision} + 1/\text{recall}} \tag{2}$$

With a value between 0 and 1, the higher the score the better. The baseline is set at 0.14 and corresponds to a model that always predicts "abnormal" (positive class) which is the model that maximizes the F_1 score with a random decision. A model always predicting "normal" (negative class) does not yield a valid F_1 score, as it would result in a division by zero for the precision.

The results are shown in figure 2. While first increasing as more labels are added, starting from 125 labeled log keys in the prompt, GPT-4.1, GPT-4o, and GPT-o1 appear to plateau at an F_1 score of approximately 0.6. The GPT-o1-mini model does not benefit from in-context learning for this task and shows no significant improvement as more labels are added to the prompt.

The F_1 score distributions of all models have a high variance. One model can perform well on one fold and poorly on another, even more so with an imbalanced dataset. Consequently, the averaged F_1 score shows only a trend and not a strong correlation with the number of labels in the prompt.

Table 1 shows the evolution of recall and precision across models when given the maximum number of labels in the prompt. While GPT-4.1 and GPT-o1 balance the two, GPT-4o and GPT-o1-mini favor recall over precision, leading to many false positives but fewer missed anomalies.

4.2. Costs

With standard pricing, the cost of using language models is linear with the number of tokens in the prompt, output, and cache. Cached tokens are a subset of current prompt tokens that were previously used and can be reused to reduce computation and cost. The public model cost is defined as

$$\text{cost} = a_1 \times \text{prompt} + a_2 \times \text{output} - a_3 \times \text{cached}, \tag{3}$$

where the cost per token coefficients a_i are strictly positive real numbers.

The GPT-o1, GPT-4o, and GPT-4.1 models use the same tokenizer resulting in similar token counts across prompts and outputs. In contrast, GPT-o1-mini tokenizes text differently, producing approximately 2% more tokens on average among all experiments. However, the a_i coefficients differ significantly across models (see table 2).

While GPT-4.1 and GPT-4o produce similar output lengths, GPT-o1 and GPT-o1-mini generate verbose responses with additional reasoning. On average, GPT-o1-mini produces 15.8 times more output tokens, and GPT-o1 produces 22.8 times more than the others. Ignoring cost savings from cached tokens (which depend on previous queries), the cost

Model	Recall %	Precision %	Cost per TP €
GPT-4.1	65	65	0.013
GPT-4o	79	57	0.020
GPT-o1	61	62	0.333
GPT-o1-mini	83	31	0.017

Table 1. The recall, precision, and cost per true positive for each model when given the maximum number of labels in the prompt.

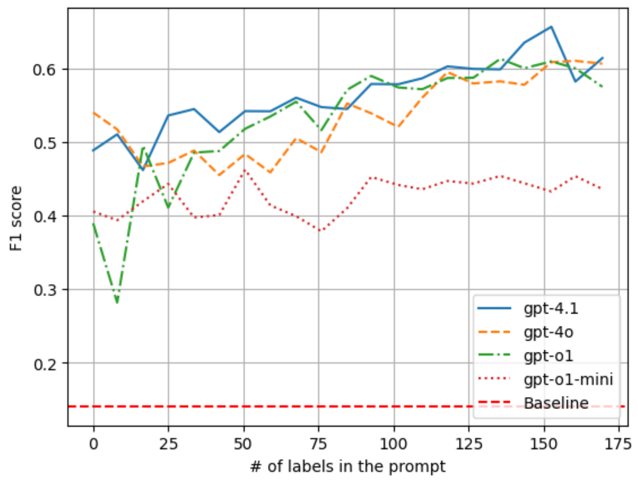

Figure 2. Average F_1 score as a function of the number of labels in the prompt for the task of classifying log keys as abnormal like an expert troubleshooter. The standard deviation of each signal is around 0.15.

Table 2. Cost per token coefficients for the different models (in 10^{-7}€/token) as of April 2025.

Model	a_1 (Prompt)	a_2 (Output)	a_3 (Cached)
GPT-4.1	20	80	15
GPT-4o	25	100	12.5
GPT-o1	150	600	75
GPT-o1-mini	11	44	6.05

of 420 queries is estimated at 12.34€ for GPT-4.1, 15.37€ for GPT-4o, 238.14€ for GPT-o1, and 14.17€ for GPT-o1-mini. Moreover, GPT-o1 queries can take several minutes to complete, whereas the others typically respond within a few seconds. Therefore, GPT-4.1 appears to be the most suitable model for this task, offering the best trade-off between performance, cost, and response time. However, for applications where recall is prioritized over precision, GPT-o1-mini may be a more cost-effective choice despite its lower F_1 score.

In table 1, for each model, the cost per true positive (TP) is calculated by averaging over all experiments the ratio of the total cost of queries to the number of true abnormal template detected. Also note that the cost for classifying template is fixed and does not depend on the number of logs processed, as long as they are parsed into the same number of log keys.

5. DISCUSSION

The approach is highly energy efficient: initial log parsing relies on a trained set of regular expressions, which operate solely on CPUs. Subsequently, resource-intensive neural networks running on GPUs are applied to groups of logs rather than individual messages, further reducing computational demands. In-context learning delivers substantial performance improvements while requiring only a small number of labeled log keys. This is especially advantageous for railway logs, where expert labeling demands deep domain knowledge and can be both time-consuming and costly. The method also enables quick adaptation to new log keys without the need for extensive retraining or fine-tuning of the model.

While the results offer a comparative assessment of various LLMs, this comparison is limited to the specific anomaly classification task and should not be extrapolated to their overall capabilities. A key limitation of the approach is its exclusion of anomalous parameter values within log messages, which may contain critical information. Additionally, the method yields a modest F_1 score and a considerable number of false positives. Notably, GPT-o1-mini achieves the highest recall but generates many false positives, with 40% corresponding to warnings. In this industrial setting, minimizing false negatives is essential—ensuring that no critical bugs are missed. Including some irrelevant information is acceptable if it aids root cause analysis, but overlooking significant events is unacceptable. Given the highly imbalanced dataset, false positives are less problematic here than in fields such as medical diagnostics or fraud detection.

Alternatives — Different models were evaluated to enhance performance or reduce computational cost. CodeGemma (Zhao et al., 2024), optimized for code generation; LLaVA-LLaMA3 (H. Liu, Li, Li, & Lee, 2024), a multimodal model for natural language understanding and generation; as well as GPT-3.5 Turbo 8k, LLaMA 3.1:8B (Grattafiori et al., 2024), Phi 3.5, and Phi 4, which are designed for general-purpose reasoning and language comprehension. The primary advantage of these models lies in their relatively small size, enabling deployment on systems with limited computational resources. However, none demonstrated sufficient output structure stability to be viable for this specific task and were therefore excluded from the study. In this study, only in-context learning was leveraged. Alternatively, the dataset could be used for fine-tuning or for low-rank adaptation (i.e., training newly inserted weights) on a pre-trained LLM. However, the available labeled data is extremely limited with only 212 log keys in the context, and for each, a ”True” or a ”False” converts to approximately 1000 tokens output. Especially facing the scale of the evaluated models which for some exceeds 600 billion parameters.

Improving Performance — Several strategies could be explored to improve performance. First, model performance is largely influenced by its training data, which — in the case of GPT models — is proprietary, though claimed to be large and diverse. A model specifically trained for log and code understanding, while retaining strong reasoning capabilities, may yield better performance.

All enriched prompts remained within the context window of the evaluated models, so retrieval-augmented generation

(RAG) was not required in our experiments. However, providing additional contextual information about the logs in the prompt could still be advantageous. If the context were to exceed the model's capacity, RAG could then be used to selectively retrieve the most relevant portions to improve the classification of log keys.

Another potential improvement is to increase the number of labeled log keys in the prompt without additional human effort, by auto-regressively reusing model-predicted classifications in future prompts (a self-feedback loop). However, given the relatively low F_1 score, such an approach risks compounding errors and may degrade in-context learning performance.

CONCLUSION

We present a method for scaling anomaly detection in railway logs by leveraging LLMs and in-context learning. By classifying log keys instead of individual log messages, we significantly reduce the number of queries to the model, thereby decreasing computational load and costs. Our empirical results demonstrate that LLMs can effectively classify log keys as normal or abnormal, with performance improving as more labeled log keys are included in the prompt. However, the method's performance is limited by the imbalanced dataset and the exclusion of parameters from classification. Future work could explore alternative models, fine-tuning techniques, and additional data sources to enhance performance.

REFERENCES

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., … Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics.*

Ding, X., Yang, X., Hu, H., & Liu, Z. (2017, April). The safety management of urban rail transit based on operation fault log. *Safety Science*.

Du, M., & Li, F. (2016, December). Spell: Streaming Parsing of System Event Logs. In *2016 IEEE 16th International Conference on Data Mining (ICDM).*

Du, M., Li, F., Zheng, G., & Srikumar, V. (2017, October). DeepLog: Anomaly Detection and Diagnosis from System Logs through Deep Learning. In *Proceedings of the 2017 ACM CCS Conference on Computer and Communications Security.* Association for Computing Machinery.

European Rail Traffic Management System. (2018). European Union Agency for Railways.

Gerhards, R. (2009, March). *The Syslog Protocol.* Internet Engineering Task Force.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., … Ma, Z. (2024, November). The Llama 3 Herd of Models. *arXiv*.

He, P., Zhu, J., He, S., Li, J., & Lyu, M. R. (2016, June). An Evaluation Study on Log Parsing and Its Use in Log Mining. In *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN).*

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., & Steinhardt, J. (2020, October). Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations.*

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., … Chen, W. (2021, October). LoRA: Low-Rank Adaptation of Large Language Models. In *International Conference on Learning Representations.*

Huang, S., Liu, Y., Fung, C., He, R., Zhao, Y., Yang, H., & Luan, Z. (2020, December). HitAnomaly: Hierarchical Transformers for Anomaly Detection in System Log. *IEEE Transactions on Network and Service Management*.

Klumpenhouwer, W., & Shalaby, A. (2022, September). Using Delay Logs and Machine Learning to Support Passenger Railway Operations. *Transportation Research Record*.

Kobayashi, S., Otomo, K., Fukuda, K., & Esaki, H. (2018, March). Mining Causality of Network Events in Log Data. *IEEE Transactions on Network and Service Management*.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., … Kiela, D. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems.* Curran Associates, Inc.

Liu, H., Li, C., Li, Y., & Lee, Y. J. (2024, June). Improved Baselines with Visual Instruction Tuning. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

Liu, Y., Tao, S., Meng, W., Yao, F., Zhao, X., & Yang, H. (2024, April). LogPrompt: Prompt Engineering Towards Zero-Shot and Interpretable Log Analysis. In *2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion).*

Mannhardt, F., & Landmark, A. D. (2019, January). Mining railway traffic control logs. *Transportation Research Procedia*.

Meng, W., Liu, Y., Zhu, Y., Zhang, S., Pei, D., Liu, Y., … Zhou, R. (2019). LogAnomaly: Unsupervised Detection of Sequential and Quantitative Anomalies in Unstructured Logs. In *International Joint Conference on Artificial Intelligence.*

Mohamad, N. H., Hashim, H., Abdul Hamid, N. A., & Ismail, M. I. A. (2021, September). Dashboard for analyzing SCADA data log: A case study of urban railway in Malaysia. *International Journal of Advances in Applied Sciences*.

Mudgal, P., & Wouhaybi, R. (2024). An Assessment of ChatGPT on Log Data. In F. Zhao & D. Miao (Eds.), *AI-generated Content*. Singapore: Springer Nature.

OpenAI. (2024a, May). *Hello GPT-4o.*

OpenAI. (2024b, September). *Learning to reason with LLMs.*

OpenAI. (2024c, September). *OpenAI o1-mini.*

OpenAI. (2025, April). *Introducing GPT-4.1 in the API.*

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., … Zoph, B. (2024, March). GPT-4 Technical Report. *arXiv*.

Pan, J., Liang, W. S., & Yidi, Y. (2024, May). RAGLog: Log Anomaly Detection using Retrieval Augmented Generation. In *IEEE World Forum on Public Safety Technology (WFPST).*

Spirtes, P., Glymour, C., & Scheines, R. (2001). *Causation, Prediction, and Search*. The MIT Press.

Yu, S., He, P., Chen, N., & Wu, Y. (2023, September). Brain: Log Parsing With Bidirectional Parallel Tree. *IEEE Transactions on Services Computing*.

Zhang, L., Jia, T., Jia, M., Wu, Y., Liu, H., & Li, Y. (2025, February). XRAGLog: A Resource-Efficient and Context-Aware Log-Based Anomaly Detection Method Using Retrieval-Augmented Generation. In *AAAI 2025 Workshop on Preventing and Detecting LLM Misinformation (PDLM).*

Zhao, H., Hui, J., Howland, J., Nguyen, N., Zuo, S., Hu, A., … Huffman, S. (2024, June). CodeGemma: Open Code Models Based on Gemma. *arXiv*.

Zhu, J., He, S., Liu, J., He, P., Xie, Q., Zheng, Z., & Lyu, M. R. (2019, January). Tools and Benchmarks for Automated Log Parsing. In *International Conference on Software Engineering (ICSE).*