FLEX: Fault Localization and Explanation Using Open-Source Large Language Models in Powertrain Systems

Herbert Muehlburger
Institute of Software Technology, Graz University of Technology, Austria

Franz Wotawa
Institute of Software Technology, Graz University of Technology, Austria

Abstract

Cyber-physical systems (CPS) are critical to modern infrastructure, but are vulnerable to faults and anomalies that threaten their operational safety. In this work, we evaluate the use of open-source Large Language Models (LLMs), such as Mistral 7B, Llama3.1:8b-instruct-fp16, and others to detect anomalies in two distinct datasets: battery management and powertrain systems. Our methodology utilises retrieval-augmented generation (RAG) techniques, incorporating a novel two-step process where LLMs first infer operational rules from normal behavior before applying these rules for fault detection. During the experiments, we found that the original prompt design yielded strong results for the battery dataset but required modification for the powertrain dataset to improve performance. The adjusted prompt, which emphasises rule inference, significantly improved anomaly detection for the powertrain dataset. Experimental results show that models like Mistral 7B achieved F1-scores up to 0.99, while Llama3.1:8b-instruct-fp16 and Gemma 2 reached perfect F1-scores of 1.0 in complex scenarios. These findings demonstrate the impact of effective prompt design and rule inference in improving LLM-based fault detection for CPS, contributing to increased operational resilience.

2012 ACM Subject Classification Software and its engineering → Software testing and debugging; Software and its engineering → Software fault tolerance; Computing methodologies → Machine learning; Software and its engineering; Computing methodologies → Natural language processing

Keywords and phrases Fault detection, anomaly detection, powertrain systems, large language models, open-source LLMs

Digital Object Identifier 10.4230/OASIcs.DX.2024.25

Category Short Paper

Supplementary Material *Software (Source Code and Data)*: `https://github.com/muehlburger/dx-2024-flex` [16], archived at `swh:1:dir:e12a10f926844b17fac4162c73f2aa3a3d56b605`

Funding This study was conducted within the framework of the project ARCHIMEDES, receiving funding from the Key Digital Technologies Joint Undertaking (KDT JU) - the Public-Private Partnership for research, development, and innovation under Horizon Europe – and National Authorities under Grant Agreement No 101112295 funded by the European Union and the FFG under Grant Agreement No FO999899377.

1 Introduction

Cyber-physical systems (CPS) integrate hardware and software with their physical environment, forming the backbone of critical infrastructures such as transportation and energy. Faults in CPS can lead to significant operational and safety risks, making robust fault detection and diagnosis systems essential. Advances in deep learning techniques, such as transformers [21] applied to time-series data [22], have significantly improved the ability to monitor these systems. Fault detection can be applied during system development [13] or real-time operation, where diagnosing faults promptly is crucial for safety and reliability [15]. In our previous work [14], we evaluated the effectiveness of open-source large language models (LLMs) for fault detection in cyber-physical systems, with a focus on battery management systems. The current work builds on these findings by extending the evaluation to additional system components, such as powertrain systems, and introducing novel prompt designs tailored to different types of anomalies. This allows for more comprehensive fault detection and localization across a wider range of systems.

We explore whether LLMs can be effectively applied for fault detection and diagnosis in CPS. Traditional fault detection approaches have primarily relied on machine learning models and deep learning techniques such as autoencoders [2]. While LLMs like BERT [8] and GPT-3 [5] have shown success in tasks such as rule learning [24] and anomaly detection [17], their use in real-time CPS fault diagnosis remains underexplored.

A key consideration in this study is the exclusive use of open-source LLMs. In contrast to commercial models, open-source LLMs ensure transparency, accessibility, and reproducibility – fundamental requirements in scientific research. Open-source models like Mistral 7B, Llama3.1:8b-instruct-fp16, and Gemma 2 offer the advantage of being freely available and adaptable for diverse applications, making them ideal for research purposes. This choice allows researchers to replicate, validate, and extend the findings, which is crucial for building reliable fault detection systems in CPS.

We focus on using retrieval-augmented generation (RAG) [12] to enhance fault detection by integrating external knowledge through search strategies such as maximal marginal relevance (MMR) and cosine similarity. A central component of our approach is the inference of operational rules: LLMs first infer rules from normal CPS behavior before applying them for fault detection. This two-step approach ensures that the models can identify deviations from normal operations and accurately diagnose faults.

Our main contributions are summarised as follows:

- We assess the performance of six open-source models, including Mistral 7B, Gemma 2, Llama3.1:8b-instruct-fp16, and others for fault detection in CPS datasets.
- We propose a two-step process in which LLMs first infer operational rules from normal CPS behavior and then apply these rules to diagnose faults, highlighting the critical role of rule inference in improving detection accuracy.
- We evaluate different search strategies, including maximal marginal relevance (MMR) and cosine similarity, to retrieve relevant information during the fault detection process in the RAG framework.
- We provide a detailed analysis of the models' ability to generalise from inferred rules and accurately diagnose faults in CPS, offering insights into their performance across different datasets.

The remainder of this paper is structured as follows: Section 2 reviews related work. Section 3 outlines the methodology and datasets used. Section 4 presents and discusses the experimental results. Finally, Section 5 concludes the paper and suggests directions for future research.

2 Related Work

Our work is at the intersection of research on a) security in cyber-physical systems and b) machine learning in fault detection, specifically the application of large language models in anomaly detection.

CPS play a critical role in various essential infrastructure sectors, such as manufacturing and smart grids [7]. Ensuring the security of CPS involves fault detection mechanisms to maintain operational integrity. Previous studies have investigated fault detection in CPS using traditional machine learning techniques, highlighting both achievements and constraints [18]. However, there is a research gap on the use of LLMs for anomaly detection in CPS, particularly in fault detection scenarios [11].

Recent discussions have surfaced about the use of open-source LLMs in cybersecurity, offering theoretical insights and initial findings on their potential applications [11]. Furthermore, research has examined textual encoding and its impact on the performance of machine learning models, especially in contexts relevant to CPS security [11]. Recognising this research gap, this work aims to explore the application of open-source LLMs to fault detection in CPS, presenting avenues for future investigations [11].

In the domain of fault detection and isolation, researchers have proposed robust methods for fault diagnosis in CPS, focusing on fault detection, isolation, identification, and risk assessment processes [4]. Furthermore, studies have explored anomaly detection in CPS using Bayesian networks to establish cause-and-effect relationships between cyber components and physical systems [11]. These integrated approaches are crucial to effectively detect anomalies and distinguish between cyber attacks and physical faults in CPS [11].

Previous studies, including our own work [14], have explored the application of LLMs to fault detection in CPS, with a specific focus on battery management systems. In that work, we demonstrated the efficacy of open-source LLMs such as Mistral for fault detection using retrieval-augmented generation (RAG) techniques. However, that study primarily dealt with fault detection in isolated systems (i.e., the battery management system), whereas this work expands the focus to more complex, multi-component systems like powertrains. Additionally, this paper introduces new prompt designs optimized for fault localization in systems with interdependent components.

The research discussed in this section highlights the importance of fault detection to ensure the security and reliability of CPS. The application of machine learning, specifically LLMs, provides opportunities to further enhance the field of CPS security and fault detection.

3 Methodology

Building on our previous evaluation of LLMs for fault detection in battery management systems [14], we adapted the methodology to handle the more complex fault localization requirements of powertrain systems. While the initial study focused on detecting anomalies in a single system component, this work extends the approach by designing new prompts that capture interactions between multiple components (battery, H-Bridge, and motor) to improve the accuracy and explanatory power of fault detection. In this study, we implement a fault detection system for powertrain components using a combination of LLMs, embeddings, and vector search. Figure 1 visualizes FLEX, our proposed approach to using LLMs for fault detection. In this section, we further describe the dataset used in our experiments. We go into more detail about the data collection procedure, the data preprocessing steps, the model selection, and the experimental setup. Finally, we highlight some limitations of our work.

3.1 Datasets

We use two datasets that represent normal and faulty behavior of two different CPS. The first CPS is a battery management system from an electrified vehicle, which we refer to as the “battery” dataset. The second dataset, called “powertrain,” contains simulated time-series data of various powertrain components, such as battery current, voltage, motor speed, and H-Bridge mode.

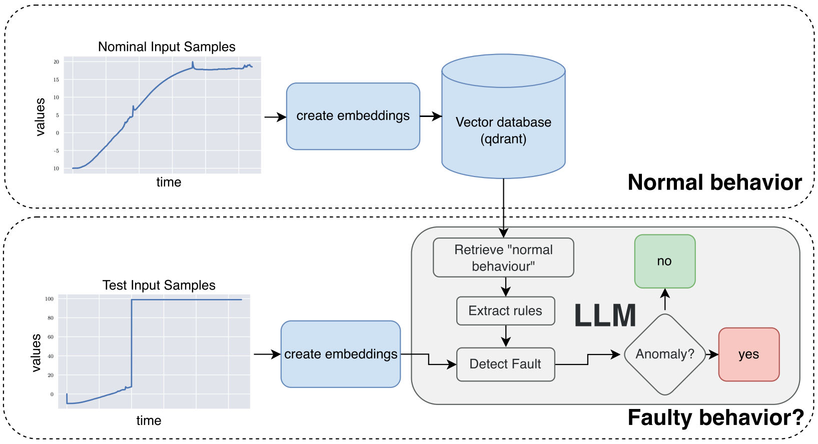

Figure 1 The FLEX approach for fault detection. Nominal (normal) input samples are text encoded and embedded into a vector database. Test samples undergo the same encoding and embedding process. A Retrieval-Augmented Generation (RAG) mechanism is used to query the vector database for similar normal behavior samples. The LLM extracts rules from these normal samples and applies them to detect anomalies in the test samples.

Battery Dataset. The battery dataset is a multivariate time-series obtained from a simulated industrial battery management system. It contains 90 minutes (5,400.5 seconds) of continuous readings across 13 different signals, including temperature sensors, power usage, velocity, and airflow from cooling systems, amounting to 10,801 samples.
We use the training dataset, which contains no anomalies, to represent the normal behavior of the system in a semi-supervised setting. This training data is stored in a vector database for Retrieval-Augmented Generation (RAG) purposes, allowing context retrieval during LLM prompting. A separate test dataset containing anomalies, also with 10,801 samples, is used for model evaluation. The experimental setup is further detailed in Section 3.4.

Powertrain Dataset. The powertrain dataset is derived from a simplified simulation model of an electrified powertrain system. [23] This system consists of a battery, an H-bridge, an electric motor, and a load. The powertrain model is used to generate both correct and faulty behavior data for different loads and faults in the system.
The training dataset includes normal behavior under various load conditions, while the test dataset introduces specific faults in the battery, H-bridge, and motor components. Faults are introduced at various time points in the simulation. For example, faults such as an empty battery or malfunctioning H-bridge switches are simulated. The dataset includes battery voltage, current, motor rotational speed, and H-bridge mode over time for both correct and faulty behaviors, allowing for precise fault diagnosis.

3.2 Data Pre-processing

The pre-processing of our dataset is a crucial step toward leveraging the capabilities of pre-trained LLMs for anomaly detection within CPS. This process involves transforming raw data into a format that is more suitable for analysis by these models. Our pre-processing pipeline consists of several stages outlined below and in Figure 2:

1. **Rounding:** Each numerical value in the dataset is rounded to two decimal places. This simplification helps in standardising the data format and reduces noise in the data that is insignificant to the model's performance.

2. **Text Encoding:** We convert the dataset from numerical to textual representation to align with the input format expected by LLMs. This encoding process involves:
   a. Prefixing each value with its corresponding column name, for example, converting a temperature reading column to "temperature_5_deg".
   b. Appending the text " is " followed by the rounded value of the reading, resulting in entries such as "temperature_5_deg is -9.99".
   c. Convert every number into a word representation resulting in entries such as "temperature_five_deg is minus nine point nine nine".

This textual transformation helps in aligning the dataset with the linguistic input format preferred by LLMs, thereby facilitating more effective anomaly detection by enabling the models to process and analyze the data within a contextually richer format.

3.3 Model Selection and Implementation Strategy

In this study, we deliberately opted for open-source LLMs instead of proprietary options like ChatGPT. Our goal was to ensure reproducibility and openness, making the models we used accessible and easy to replicate for future research. The specific LLMs selected for this study include *Llama3* [9], *Llama3.1:8b-instruct-fp16* [9], *Gemma 2* [20], *Phi3:14b* [1], *Mistral* [10], and *Mistral NeMo* [3]. These models were specifically chosen for their capabilities in anomaly detection within CPS. By selecting a diverse range of open-source models, we were able to explore the strengths and limitations of each in the context of fault detection in CPS. This approach allowed us to evaluate how well these models can handle the complexities of CPS data and their effectiveness in identifying potential faults.

To enhance model performance, we integrated these LLMs into a Retrieval Augmented Generation (RAG) framework. This framework enabled the models to access relevant, anomaly-free operational data from a vector store, improving their ability to detect deviations from normal behavior. The underlying hypothesis was that by providing LLMs with precise representations of normal system operations, they would be better equipped to identify faults.

3.4 Experimental Setup

Our experimental setup[1], as illustrated in Figure 1, uses data represented as multivariate time series in CSV files. Each row contains multiple sensor readings and represents one training sample. We implemented a Retrieval Augmented Generation (RAG) [12] application that uses training samples to provide a relevant context for LLMs. The training data are transformed into an embedding and stored in a vector database. Before prompting the LLM, we query the vector database to find values similar to the test data. Retrieval-augmented generation (RAG) models enhance the capabilities of generative models by incorporating relevant information extracted from an external knowledge base during the generation process. A critical step in this approach is the selection of relevant documents or text snippets from the knowledge base, which is achieved using similarity measures. We used a retriever to fetch relevant documents from the vector store using similarity measures, such as cosine similarity and Maximal Marginal Relevance (MMR) [6]. The retrieved documents were used as context for the LLM, which then processed each measurement from the test dataset, applying the previously inferred rules. The LLM was executed in a pipeline, taking the test data and context as input and producing a fault diagnosis for each time series point.

[1] Our code is available open-source on Zenodo: `https://doi.org/10.5281/zenodo.13879936`

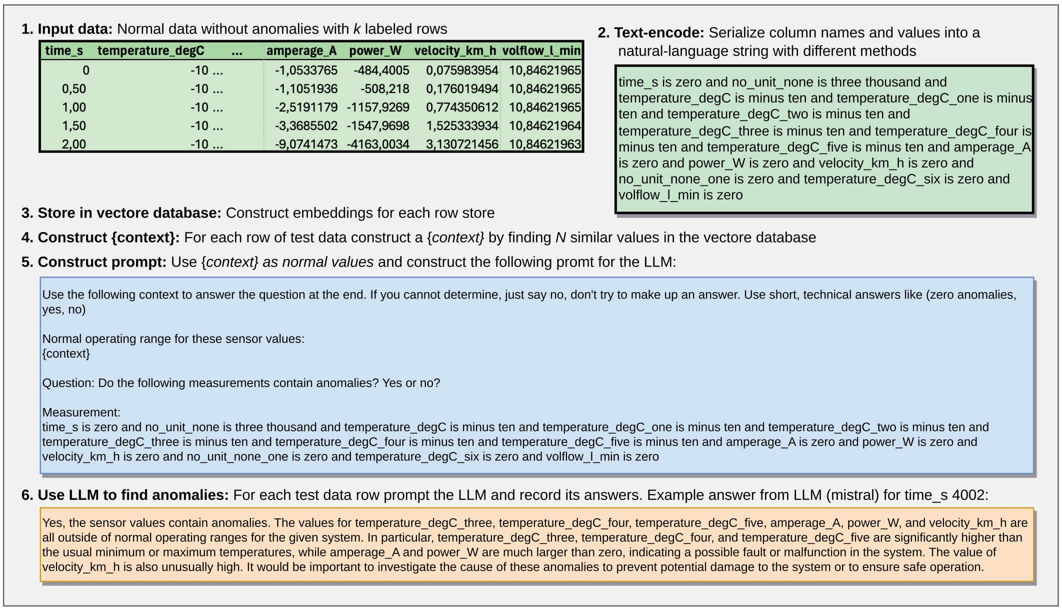

Figure 2 Schematic representation of the anomaly detection workflow using the LLM Mistral for the "battery" dataset. The process begins by ingesting time series data (1), which is transformed into descriptive natural language through text-encoding feature names and their corresponding values (2). These textual representations are embedded using the *SentenceTransformer* library and stored in a vector database (3). The LLM retrieves contextually relevant data by querying this database with test data (4). The retrieved context is then used to prompt the LLM for anomaly detection (5), where the LLM assesses each row by comparing sensor readings with normal operational ranges. The model identifies anomalies, assigns an anomaly rating, and provides explanations for any deviations (6). The example shows a detailed response from Mistral 7B analysing time step 4002, highlighting the model's ability to detect and explain faults in the "battery" dataset.

3.4.1 Application in Retrieval-Augmented Generation

In the context of RAG, similarity measures, such as cosine similarity and MMR, are used to identify documents or text snippets within the knowledge base that are most relevant to the given query. Specifically, the query and each candidate document are represented as vectors in a high-dimensional space, where each dimension corresponds to a feature represented as an embedding. The similarity between the query vector and each document vector is calculated, and the documents are ranked based on their similarity scores. The highest-scoring documents are considered the most relevant and are subsequently used to augment the generative process, generating more informed and contextually relevant text. Cosine similarity and MMR represent different similarity measures, leading to different rankings.

3.4.2 Fault Detection Prompt Design

Initially, we designed a simple prompt for the battery system, focusing on voltage stability as the main indicator of faults. However, when we applied this prompt to the powertrain system, it did not produce satisfactory results. The powertrain system is more complex, involving multiple interdependent components like the battery, H-Bridge, and motor, each with distinct fault patterns. Using the same battery prompt for powertrain would not enable the LLM to effectively capture and differentiate these complex fault scenarios.

Listing 1 Prompt Template for “battery” dataset. The prompt contains a {context} with samples that do not contain anomalies retrieved from the vector database. We search for the ten most similar entries in the vector database compared to the test data {measurement}.

```
Use the following context to answer the question at the end. If you cannot determine, just say no, don't try to make up an answer. Use short, technical answers like (zero anomalies, yes, no). Write a response that appropriately completes the request.

Context: Normal operating range:
{context}

Question: Do the following measurements contain anomalies? Yes or no?

Test Measurement:
{measurement}

Provide an anomaly rating between 0-100 based on the following criteria:

1) Values are in a reasonable range of the provided context.
2) Values do not contain anomalies.
3) Values are consistent with the given context.

Check *very carefully* with the given ranges before giving your rating.

Answer:
- Anomaly: e.g. yes, no
- Anomaly-Rating:
- Explanation:
```

We also chose not to use the more complex powertrain prompt for the battery system, as it emphasises interactions between components, which are not necessary for detecting battery faults. The battery system relies primarily on voltage and current for fault detection, making a simpler approach more efficient. Applying the more detailed powertrain prompt to the battery system would add unnecessary complexity and reduce efficiency, but could be explored in future work.

Given these differences, we constructed a two-part prompt specifically for the powertrain system. In the first part, the LLM infers operational rules from normal behavior, identifying expected ranges for key variables like battery current, voltage, motor speed, and H-Bridge mode. These rules are strictly based on time-series data and do not assume any values outside the provided context. In the second part, the LLM uses the inferred rules to detect faults by comparing test data against the established normal behavior. Faults are only reported when there is a clear deviation from normal operations, ensuring a high level of precision in the fault detection process. This strict two-step approach helped improve fault detection accuracy in the powertrain system. We used the prompt template from Listing 1 for the “battery” dataset and the prompt templates from Listings 2 and 3 for the “powertrain” dataset. These prompts were implemented using the Langchain[2] Python framework and LLMs were accessed locally through Ollama[3].

[2] H. Chase. “Langchain.” python.langchain.com. `https://python.langchain.com/docs/get_started/introduction` (accessed Sep. 05, 2024).

[3] Ollama. “Ollama v0.3.9.” `https://ollama.com` (accessed Sep. 05, 2024).

Listing 2 System Prompt template for “powertrain” dataset. The system prompt provides strict instructions for rule inference and fault detection, to prevent hallucination of faults.

```
Your goal is to evaluate the overall status of the battery based on the provided data. The battery status should only be ‘‘OK’’ or ‘‘Empty.’’ Do not mention individual sensor values such as voltage or current in the final report. The output should be in a concise, structured format and only focused on the battery as a whole component.

### Battery Fault Models:
- OK: The battery is functioning as expected.
- Empty: The battery provides no voltage (Battery Voltage = 0).

### For inferred rules:
- You must infer rules based on the provided time series data.
- Rules should represent the overall battery behavior as either ‘‘OK’’ or ‘‘Empty.’’

Example format for inferred rules:
- Battery: OK (the battery voltage remains stable)

### For fault detection:
- You MUST NOT infer faults that are not represented by the data. Report faults only if the data clearly shows that the battery provides no voltage (Battery Empty).

Output format must strictly be:
component,fault,reason
battery,OK/Empty,Explanation
```

Listing 3 User Prompt template for “powertrain” dataset. The user prompt guides the process of rule inference and fault detection based on time series data.

```
# Part 1: Infer Rules from Normal Battery Behavior

Use the following time series data to infer the overall behavior of the battery as either ‘‘OK’’ or ‘‘Empty.’’

# Input Data:
{context}

---

# Part 2: Battery Fault Detection Based on Inferred Rules

Now, apply the inferred rules strictly to the following test data to detect if the battery is ‘‘OK’’ or ‘‘Empty.’’

# Battery Fault Models:
- OK: The battery is functioning as expected.
- Empty: The battery provides no voltage (Battery Voltage = 0).

# Input Data:
{measurement}

# Task:
1. Based ONLY on the inferred rules, determine whether the battery is ‘‘OK’’ or ‘‘Empty.’’
2. Report the results in the following format:
component,fault,reason
battery,OK/Empty,Explanation
```

3.5 Evaluation Metrics

In this work, we assess the performance of open-source LLMs in diagnosing faults in CPS using two datasets: "battery" and "powertrain." Our evaluation is based on two key experiments: the first evaluates models Mistral and Llama3 on 100 samples from the "battery" dataset, while the second evaluates the models Mistral, Mistral NeMo, Llama3.1:8b-instruct-fp16, Gemma 2, and Phi3:14b on 10 samples from the "powertrain" dataset.

Seliya et al. [19] proposed several metrics for classifier evaluation. For our study, we selected precision, recall, and F1-score to compare the performance of the models in fault detection tasks. These metrics are defined as follows:

Precision measures the proportion of correctly identified faults (true positives) out of all instances flagged as faulty, i.e.: Precision $= \frac{TP}{TP+FP}$. High precision is crucial in fault detection, as it minimises false positives, preventing unnecessary maintenance or downtime.

Recall, or sensitivity, measures the proportion of actual faults that were correctly identified, i.e.: Recall $= \frac{TP}{TP+FN}$. High recall ensures that most faults are detected, even if some false positives are generated.

The **F1-score** is the harmonic mean of precision and recall, balancing the two, i.e.: $F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$. A high F1-score indicates that the model achieves a good balance between precision and recall, which is essential for accurate fault detection, managing both false positives and false negatives effectively.

4 Results

As demonstrated in our previous work [14], open-source LLMs such as Mistral perform exceptionally well in detecting anomalies in battery management systems. However, our results indicate that applying the same prompt structure to powertrain systems does not yield similarly performance. Therefore, this paper introduces a new two-part prompt designed specifically for powertrain systems, leading to improved fault detection performance. This section presents the performance of several LLMs on anomaly detection in CPS, considering the role of prompt design in shaping model performance. The experiments were conducted using two datasets: "battery" and "powertrain," with different prompt structures for each. The models were evaluated based on F1-score, Precision, and Recall, as shown in Tables 1 and 2.

4.1 Effect of Prompts on Model Performance

The prompts used for each dataset had a significant impact on the performance of the models. In the "battery" dataset, the prompt (Listing 1) was straightforward, asking the LLMs to identify anomalies by comparing test measurements with normal system behavior retrieved from the vector database. In contrast, the "powertrain" dataset utilized more complex prompts (Listings 2 and 3) that required the LLMs to first infer rules from normal system behavior before applying them to detect faults. This two-step process introduced additional cognitive complexity, which likely influenced the models' performance differently across the datasets.

4.2 Battery Dataset Results

The "battery" dataset prompt provided a clear context of normal behavior and asked the LLMs to determine whether test measurements contained anomalies based on this context. As shown in Table 1, *Mistral* consistently outperformed *Llama3* across different configurations

Table 1 Performance of Large Language Models (LLMs) on anomaly detection tested using 100 samples from the "battery" dataset. The table evaluates two models, Mistral and Llama3, across two retrieval strategies: Maximal Marginal Relevance (MMR) and cosine similarity. The models were assessed with varying numbers of retrieved results (ranging from 1 to 1000), with performance measured using F1-score, precision, and recall. Notably, Mistral consistently achieved near-perfect scores (F1=0.99, Precision=1.00, Recall=0.98) across multiple configurations, while Llama3 showed lower precision when fewer results were retrieved, despite maintaining perfect recall.

Model	Strategy	# Results	F1	Precision	Recall
llama3	mmr	1	0.68	0.52	1.00
llama3	similarity	1	0.68	0.52	1.00
mistral	similarity	1000	0.81	0.69	1.00
mistral	similarity	500	0.86	0.77	0.98
mistral	similarity	10	0.96	0.94	0.98
llama3	mmr	5	0.98	1.00	0.96
mistral	mmr	1000	0.98	0.98	0.98
mistral	mmr	500	0.98	0.98	0.98
llama3	mmr	10	**0.99**	1.00	0.98
mistral	mmr	1	**0.99**	1.00	0.98
mistral	similarity	1	**0.99**	1.00	0.98
mistral	mmr	10	**0.99**	1.00	0.98
mistral	mmr	5	**0.99**	1.00	0.98
mistral	similarity	5	**0.99**	1.00	0.98

and retrieval strategies (MMR and similarity). *Mistral* achieved an F1-score of 0.99 when retrieving 1, 5, or 10 documents, with near-perfect precision (1.00) and recall (0.98). This high performance can be attributed to the direct nature of the prompt, which simply asked the models to compare test data with retrieved normal data without needing to infer additional rules.

In contrast, *Llama3* showed more variability in performance, especially when fewer documents were retrieved. The best F1-score (0.99) for *Llama3* was observed when 10 documents were retrieved with the MMR strategy, yielding a precision of 1.00 and recall of 0.98. However, when only 1 document was retrieved, the F1-score dropped to 0.68, with a precision of 0.52. This variability suggests that *Llama3* relies more heavily on sufficient contextual information to perform optimally. The straightforward nature of the battery prompt worked in favor of *Mistral*, while *Llama3* struggled with limited context, leading to more false positives.

4.3 Powertrain Dataset Results

The "powertrain" dataset employed a more sophisticated two-step prompt, where the LLMs first had to infer rules from the context of normal behavior and then apply these rules to detect faults in the test data. This rule inference process significantly impacted the models' performance, as it required a deeper understanding of normal behavior and strict adherence to the inferred rules before making fault predictions.

As detailed in Table 2, *Mistral-NeMo* struggled with this complexity, achieving an F1-score of only 0.33, with perfect precision (1.00) but very low recall (0.20). The need to infer rules and then apply them to the test data may have caused *Mistral-NeMo* to miss many true anomalies, leading to poor recall. This highlights the model's difficulty in managing tasks that require both rule inference and anomaly detection.

Table 2 Comparative performance of various Large Language Models (LLMs) on anomaly detection, tested with 10 samples from the "powertrain" dataset. The table evaluates models such as Mistral-Nemo, Phi3:14b, Gemma2, Llama3.1:8b-instruct, and Mistral across two retrieval strategies: Maximal Marginal Relevance (MMR) and cosine similarity. The performance is measured using F1-score, precision, and recall. Notably, models like Gemma2, Llama3.1:8b-instruct, and Mistral achieved perfect performance (F1=1.00, Precision=1.00, Recall=1.00), while Mistral-Nemo struggled with lower recall (0.20) despite perfect precision, indicating difficulties in detecting all anomalies in the test samples.

Model	Strategy	# Results	F1	Precision	Recall
mistral-nemo	mmr	10	0.33	1.00	0.20
mistral-nemo	similarity	10	0.33	1.00	0.20
phi3:14b	mmr	10	0.89	1.00	0.80
phi3:14b	similarity	10	0.89	1.00	0.80
gemma2	mmr	10	**1.00**	1.00	1.00
gemma2	similarity	10	**1.00**	1.00	1.00
llama3.1:8b-instruct-fp16	mmr	10	**1.00**	1.00	1.00
llama3.1:8b-instruct-fp16	similarity	10	**1.00**	1.00	1.00
mistral	mmr	10	**1.00**	1.00	1.00
mistral	similarity	10	**1.00**	1.00	1.00

In contrast, *Phi3:14b* performed much better, achieving an F1-score of 0.89 with both MMR and similarity strategies, balancing perfect precision (1.00) with a strong recall of 0.80. This model was more adept at handling the rule inference step, allowing it to maintain a good balance between detecting true anomalies and avoiding false positives.

Both *Gemma 2* and *Llama3.1:8b-instruct-fp16* performed flawlessly in the "powertrain" dataset, achieving perfect F1-scores (1.00) across all retrieval strategies. These models successfully inferred rules and applied them to detect anomalies with no errors, suggesting that they are particularly well-suited for tasks that involve complex, multi-step prompts like the one used in the "powertrain" dataset.

4.4 Implications of Rule Inference on Model Behavior

The rule inference step in the "powertrain" dataset introduced an additional layer of complexity that had clear implications for model performance. Models that were able to handle this step effectively, such as *Gemma 2*, *Llama3.1:8b-instruct-fp16*, and *Phi3:14b*, performed well in both precision and recall, demonstrating their ability to generalize from inferred rules and apply them accurately to detect faults. This process required models to not only understand normal behavior but also apply that understanding in a precise manner to detect deviations.

On the other hand, models like *Mistral-NeMo* struggled with this process, leading to a significant drop in recall. This suggests that models less capable of managing multi-step tasks or those that struggle with rule inference may fail to detect all anomalies, particularly in complex scenarios. This limitation highlights the importance of choosing models that can effectively handle not only anomaly detection but also rule-based reasoning tasks in more sophisticated use cases.

4.5 Cross-Dataset Comparison

Across both datasets, *Mistral* proved to be the most consistent performer, achieving strong results in the simpler "battery" prompt and perfect results in the more complex "powertrain" prompt. *Llama3* displayed high sensitivity to the number of retrieved documents in the "battery" dataset, performing well when given sufficient context but struggling when that context was limited. However, the *Llama3.1:8b-instruct-fp16* variant performed perfectly in the "powertrain" dataset, suggesting that specific configurations of *Llama3* are better suited to complex, multi-step tasks.

The performance of *Mistral-NeMo* in the "powertrain" dataset illustrates how rule inference can overwhelm models that are not well-equipped for this type of reasoning. In contrast, models like *Gemma 2* and *Phi3:14b* thrived in this context, handling both the rule inference and fault detection steps effectively.

4.6 Key Insights

- **Prompt Complexity Influences Results**: The simpler prompt in the "battery" dataset led to more consistent performance, while the complex rule inference prompt in the "powertrain" dataset revealed varying levels of model proficiency. Models that managed rule inference well performed better in the "powertrain" dataset.
- **Mistral's Robustness**: *Mistral* maintained high performance across both datasets, demonstrating its ability to manage both straightforward and more complex tasks.
- **Rule Inference Sensitivity in Llama3**: While *Llama3* performed well with sufficient context in the "battery" dataset, its *Llama3.1:8b-instruct-fp16* variant excelled in the more complex "powertrain" task, suggesting its suitability for tasks that require rule-based reasoning.
- **Mistral-NeMo's Struggle with Complexity**: *Mistral-NeMo* struggled in the "powertrain" dataset due to the complexity of rule inference, indicating the need for more tuning or alternative approaches when handling multi-step tasks.

4.7 Insights from Model Responses

Analysis of the models' generated responses offers insights into their decision-making processes. For example, Mistral's responses provide a structured output comprising a Boolean flag for anomaly detection, an anomaly rating quantifying detection certainty, and a textual explanation justifying the decision. Interestingly, an examination of Mistral's reasoning reveals occasional inaccuracies in its assessments. The LLM detected an anomaly and provided an explanation arguing that a value of -1.42 is not part of the range of -10 to 0.8. This argument is clearly wrong. Interestingly, if we provide the same answer to the same LLM in a second round, it notices that the answer is not correct and the value is within the given range. This discrepancy underscores the importance of iterative validation and refinement in improving the interpretability and reliability of the model in anomaly detection tasks.

The decision-making process in battery fault detection also showcases the nuances in model behavior. Let us illustrate how Llama3.1:8b-instruct-fp16 generates detailed, multi-step diagnostics based on time series and test data, inferring the state of the battery as either "OK" or "Empty" considering two queries. In the first query, the model analyzes a scenario where the battery's voltage remains stable at 14, with no signs of discharge or abnormal behavior. By applying these inferred rules to the test data, Llama3.1:8b-instruct-fp16 concludes that the battery is functioning normally, providing stable voltage, and classifies it as "OK." This demonstrates the model's ability to detect stable battery behavior based on consistent voltage readings and an absence of current flow. In another query, Llama3.1:8b-instruct-fp16 identifies a battery fault. The model, after inferring the same rules from normal battery behavior, detects that the voltage has dropped to zero in the test data. Based on this observation, the model correctly concludes that the battery is "Empty," citing the zero voltage as the primary reason for the fault. These example illustrates the model's application of learned rules to identify faults when deviations from normal behavior occur.

5 Conclusion & Future Work

This paper explored the use of open-source LLMs for fault detection in CPS, focusing on battery management and powertrain datasets. Models like *Mistral 7B* and *Llama3.1:8b-instruct-fp16* performed well, with *Gemma 2* and *Phi3:14b* achieving perfect F1-scores in complex scenarios. The results highlight the importance of prompt design and rule inference for accurate fault detection. Challenges remain, as some models, like *Mistral-NeMo*, struggled with more complex tasks, underscoring the need for model selection tailored to task complexity. Future work should explore more sophisticated fault diagnosis tasks and adaptive retrieval methods that further enhance LLM performance in dynamic CPS environments.

Ethical Statement

This research does not involve any ethical concerns or conflicts of interest.

References

1. Marah Abdin, Jyoti Aneja, and Hany Awadalla. Phi-3 Technical Report: A Highly Capable Language Model Locally on Your Phone, August 2024. `doi:10.48550/arXiv.2404.14219`.
2. Abhaya Abhaya and Bidyut Kr. Patra. An efficient method for autoencoder based outlier detection. *Expert Systems with Applications*, page 118904, September 2022. `doi:10.1016/j.eswa.2022.118904`.
3. Mistral AI-Team. Mistral NeMo. https://mistral.ai/news/mistral-nemo/, July 2024.
4. Joseph Osedome Boi-Ukeme. A Robust Discrete Event Method for the Design of Cyber-Physical Systems, January 2023.
5. Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners, July 2020. `arXiv:2005.14165`, `doi:10.48550/arXiv.2005.14165`.
6. Jaime Carbonell and Jade Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 335–336, Melbourne Australia, August 1998. ACM. `doi:10.1145/290941.291025`.
7. Marco Cook, Cory Paterson, Angelos K. Marnerides, and Dimitrios Pezaros. Anomaly Diagnosis in Cyber-Physical Systems. In *ICC 2022 - IEEE International Conference on Communications*, pages 5445–5450, May 2022. `doi:10.1109/ICC45855.2022.9838968`.
8. Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. `arXiv:1810.04805`, `doi:10.48550/arXiv.1810.04805`.
9. Abhimanyu Dubey, Abhinav Jauhri, and Pandey. Llama 3 Herd of Models, August 2024. `arXiv:2407.21783`.

10 Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, October 2023. doi:10.48550/arXiv.2310.06825.

11 Sudha Krishnamurthy, Soumik Sarkar, and Ashutosh Tewari. Scalable Anomaly Detection and Isolation in Cyber-Physical Systems Using Bayesian Networks. In *ASME 2014 Dynamic Systems and Control Conference*. American Society of Mechanical Engineers Digital Collection, December 2014. doi:10.1115/DSCC2014-6365.

12 Patrick Lewis, Ethan Perez, Aleksandara Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Kuttler, M. Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *ArXiv*, May 2020.

13 Herbert Muehlburger and Franz Wotawa. A Passive Testing Approach using a Semi-Supervised Intrusion Detection Model for SCADA Network Traffic. In *2022 IEEE International Conference On Artificial Intelligence Testing (AITest)*, pages 42–47, August 2022. doi:10.1109/AITest55621.2022.00015.

14 Herbert Muehlburger and Franz Wotawa. FaultLines: Evaluating the efficacy of open-source large language models for fault detection in cyber-physical systems. In *2024 IEEE International Conference on Artificial Intelligence Testing (AITest)*, Shanghai, China, July 2024. IEEE Computer Society.

15 Herbert Muehlburger and Franz Wotawa. Making Systems Fail-Aware: A Semi-Supervised Machine Learning Approach for Identifying Failures by Learning the Correct Behavior of a System. *IFAC-PapersOnLine*, 58(4):7–12, January 2024. doi:10.1016/j.ifacol.2024.07.185.

16 Herbert Mühlburger and Franz Wotawa. muehlburger/dx-2024-flex. Software, swhId: swh:1:dir:e12a10f926844b17fac4162c73f2aa3a3d56b605 (visited on 2024-11-14). URL: https://github.com/muehlburger/dx-2024-flex.

17 Jonathan Pan, Swee Liang Wong, and Yidi Yuan. RAGLog: Log Anomaly Detection using Retrieval Augmented Generation, November 2023. doi:10.48550/arXiv.2311.05261.

18 Vasso Reppa, Marios M. Polycarpou, and Christos G. Panayiotou. Distributed Sensor Fault Diagnosis for a Network of Interconnected Cyberphysical Systems. *IEEE Transactions on Control of Network Systems*, 2(1):11–23, March 2015. doi:10.1109/TCNS.2014.2367362.

19 Naeem Seliya, Taghi M. Khoshgoftaar, and Jason Van Hulse. A Study on the Relationships of Classifier Performance Metrics. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 59–66, November 2009. doi:10.1109/ICTAI.2009.25.

20 Gemma Team, Morgane Riviere, and Shreya Pathak. Gemma 2: Improving Open Language Models at a Practical Size, August 2024. doi:10.48550/arXiv.2408.00118.

21 Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. arXiv:1706.03762.

22 Qingsong Wen, Tian Zhou, Chaoli Zhang, Weiqi Chen, Ziqing Ma, Junchi Yan, and Liang Sun. Transformers in Time Series: A Survey, May 2023. arXiv:2202.07125.

23 Franz Wotawa, David Kaufmann, Adil Amukhtar, Iulia Nica, Florian Klück, Hermann Felbinger, Petr Blaha, Matus Kozovsky, Zdenek Havranek, and Martin Dosedel. *Real-Time Predictive Maintenance – Model-Based, Simulation-Based and Machine Learning Based Diagnosis*, pages 63–81. River Publishers, New York, 1 edition, September 2022. doi:10.1201/9781003337232-7.

24 Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, and Hanjun Dai. Large Language Models can Learn Rules, October 2023. doi:10.48550/arXiv.2310.07064.