cubic Transportation Systems

AIDA Lab @ IMPERIAL

AI Unlocks Predictive Maintenance in Public Transportation

Independent analysis provided via Imperial Consultants by:
Giorgos Iacovides, Wuyang Zhou, Tony G. Constantinides and Danilo P. Mandic
AIDA Lab, Imperial College London

June 30, 2025*

Contents

1 Executive Summary **2**

2 Introduction **4**

3 Related Work and Preliminaries **4**

4 Problem Definition **5**

5 Dataset and Pre-Processing **6**
5.1 Corpus Overview 6
5.2 Tokenization and Part Class Assignment 6

6 Experimental Design **6**
6.1 Parameter-Efficient Fine-Tuning with Low-Rank Adaptation (LoRA) 7
6.2 Model Training 8

7 Results **9**
7.1 Performance Comparison: Base Model vs. Our Fine-Tuned Model 9
7.2 Robustness to Input Style: Human-Written vs. Machine-Generated Descriptions 9
7.3 Per-Class Top-k Accuracy Analysis 10
7.4 Cumulative Accuracy and Coverage Analysis Across Part Labels 11

8 Conclusion **12**

9 Limitations and Future Works **12**

1 Executive Summary

Fare collection gates accommodate millions of daily passengers, which puts significant operational stress on their mechanisms and payment terminals. For example, the Transportation for London reported 3,588 million completed journeys for the full year 2024/25 across its services [1]. Such heavy usage, compounded by exogenous factors, results in frequent hardware failures that disrupt normal services. Today, incident triage relies on engineers manually reviewing thousands of incident reports and error codes or visiting on-site each month to infer the required replacement part for repairs. This manual process introduces considerable delays, increases operational expenditure, and contributes to system inefficiency and passenger dissatisfaction.

When an on-site issue arises, a "customer problem description", generated by either staff or gate hardware, is logged as the initial record of the incident. Engineers are then dispatched to interpret these descriptions and recommend repairs, often requiring specific replacement parts. This process presents an opportunity to improve pre-visit diagnostics and confirm part availability in advance, helping reduce repeat visits and downtime. To address this, we propose automating replacement part identification using a large language model (LLM), thereby learning a function

$$f(\text{Customer Problem Descriptions}) \rightarrow \text{Replacement Part}$$

where f is implemented via a neural network. For example, the input "*device crashed, cold start failed, requires pc*" should yield "*PART NEED: 9504-08019-1 QTY: 1 (AFM PC)*". This setting naturally defines a multi-class classification problem over the universe of possible parts.

Key achievements

1. We design and implement a robust data preprocessing pipeline that cleans raw repair logs into high-quality training samples through de-duplication, text editing, and mapping of part codes to class indices.

2. Using the Low-Rank Adaptation (LoRA) paradigm, we efficiently fine-tune the Llama large language model for accurate predictive maintenance while training only 4.2M parameters (0.0524% of the full model).

3. The so introduced PartLlama model is systematically evaluated against baseline LLMs, tested for robustness across human- and machine-generated inputs, and analyzed for prediction accuracy.

4. The proactive predictive maintenance enabled by PartLlama promises to reduce the number of engineer on-site visits by up to 70%.

Our fine-tuned language model, PartLlama, trained on a curated dataset of maintenance logs, achieves 83.5% Top-1, 93.4% Top-3, and 95.9% Top-5 classification accuracy (see Figure 1). This proof-of-concept highlights the potential of neural network-based LLMs to automate incident diagnosis and represents a significant step toward predictive maintenance within Cubic Transportation Systems [2]. By substantially reducing the number of on-site engineer visits required for diagnostics and repairs through proactive gate maintenance, **our model is expected to reduce the number of engineer on-site visits by up to 70%** for the tested parts, offering a clear and measurable path to improved operational efficiency. Furthermore, **the compactness of our fine-tuning enables high-speed inference, which meets and exceeds Cubic's real-time throughput requirements, allowing seamless integration into existing maintenance workflows.**

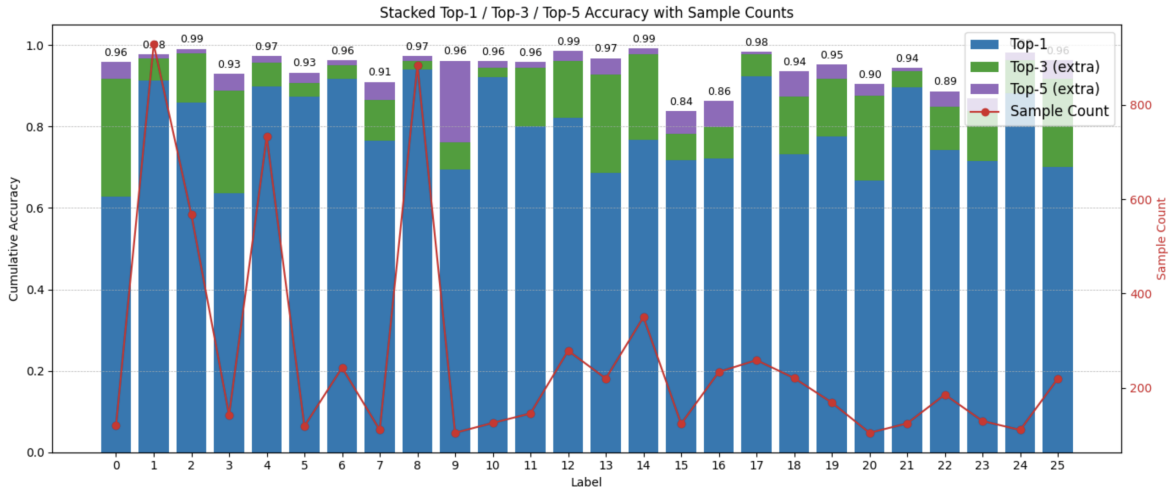

Figure 1: Stacked Top-1, Top-3, and Top-5 accuracy per part label, with sample count shown as a red line. Each bar represents cumulative accuracy (Top-1 in blue, Top-3 gains in green, and Top-5 gains in purple).

2 Introduction

Millions of passengers use fare collection gates every day, inevitably leading to failures in their mechanisms and payment terminals. Unexpected events, such as vandalism, can also cause downtime and require on-site engineer visits. Today, engineers manually examine thousands of incident reports and machine-generated error codes each month to predict which replacement part will be needed before or during a repair visit. This manual triage slows time-to-repair, increases operational costs, contributes to station crowding, and undermines passenger trust.

Recent advances in transformer-based large language models (LLMs) have dramatically elevated the state-of-the-art in textual understanding beyond statistical methods and classical machine learning techniques [3, 4, 5, 6]. However, LLMs are trained for general purpose question-answering and can not be taken off-the-shelf to automate the process of identifying the part in need given a customer problem description in this case. Furthermore, LLMs require a significant amount of compute in fine-tuning to adapt them to new downstream tasks [6, 7]. Low-Rank Adaptation (LoRA) is a newly appeared parameter-efficient fine-tuning technique which preserves the base model's general knowledge while training only a small set of specialized weights, thus achieving full fine-tune quality at substantially lower computational cost [7].

To address these challenges, we present **PartLlama,** a lightweight yet robust classification model with only 4.2 million trainable parameters (just **0.0524%** of the Llama-3-8B model) enabled via LoRA fine-tuning. PartLlama is trained to identify potential replacement parts based on customer problem descriptions. Leveraging a curated dataset of repair logs, PartLlama demonstrates strong predictive performance, achieving 83.5% exact-match accuracy, and 93.4% and 95.9% Top-3 and Top-5 accuracies, respectively, for the 26 most commonly required parts. As a result, PartLlama offers a promising solution for significantly accelerating the part-procurement-to-repair cycle in large-scale maintenance operations.

The so introduced novelties of this project are threefold:

- **Preprocessing pipeline**: We design and implement a robust data-cleaning workflow that transforms raw repair logs into high-quality training samples.
- **Parameter-efficient fine-tuning**: We apply LoRA to adapt a state-of-the-art LLM to the predictive maintenance task using only 4.2M trainable parameters, enabling high prediction accuracy with minimal computational overhead.
- **Comprehensive benchmarking**: We evaluate PartLlama across multiple settings, including comparisons with the baseline LLM, robustness to variations in input style, and detailed per-class performance analysis.

These demonstrate the potential to disrupt the current approach to predictive maintenance, moving from reactive to AI-driven proactive processes.

3 Related Work and Preliminaries

Historically, predictive maintenance systems have relied on statistical methods such as bag-of-words, n-grams, and support vector machines [8], which are notoriously ineffective at handling typos and struggle to understand nuanced semantics and contextual information. While recurrent neural networks (RNNs) offer substantial improvements in contextual modeling, their sequential nature introduces challenges in large-scale training settings [9]. The emergence of transformer-based neural network architectures has resolved these sequential bottlenecks and led to the development of popular AI chatbots such as ChatGPT [6].

Large language models (LLMs) represent the state of the art models in natural language processing and are based on the powerful attention mechanism as shown in Figure 2 [10, 11]. These large neural-network models are built on the transformer architecture and learn autoregressive token predictions by

back-propagating errors. Initially, two dominant types of LLMs existed – encoder-only and decoder-only transformer architectures [10]. Owing to the superior performance of decoder-only transformers, they have become the preferred architecture for modern LLMs. For example, ChatGPT, Llama, and DeepSeek are all built on decoder-only transformer architectures [4, 5, 12]. In this project, we fine-tune Llama to utilise its pretrained knowledge.

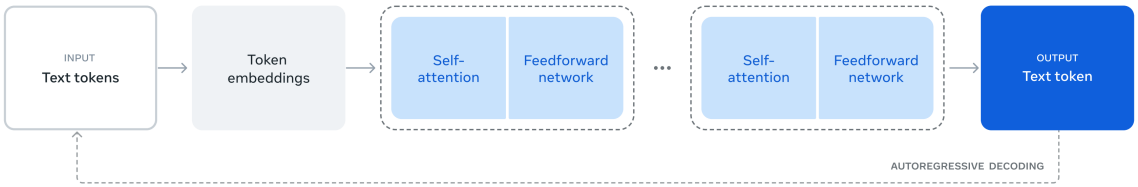

Figure 2: Large Language Models (LLMs) are built upon self-attention and feedforward neural networks. After converting the input text into embeddings, they process this representation to predict the next token in a sequence [12].

However, as LLMs are trained primarily for autoregressive prediction (see Figure 3) they require adaptation techniques such as fine-tuning to perform effectively on downstream tasks, such as the part classification problem addressed in this project. At the same time, naïve full-model fine-tuning remains prohibitively expensive for most enterprises. To this end, recent advances in parameter-efficient fine-tuning, such as instruction tuning, prompt tuning, and notably LoRA, have demonstrated strong performance in specialized domains while significantly reducing computational costs. Nevertheless, existing predictive-maintenance methods remain limited in sensor-derived prognostics, often neglecting valuable textual repair logs. Our study addresses this gap by fine-tuning LLMs with LoRA to tackle the highly varied predictive-maintenance classification problem at minimal training cost.

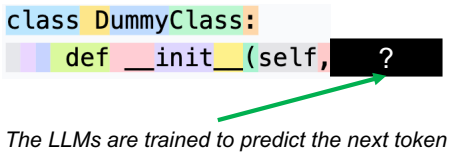

Figure 3: LLMs are trained to predict the next token, and therefore cannot be used to predict replacement parts out-of-box. Part prediction is a classification problem: finding the right part needed given an incident log.

4 Problem Definition

When an on-site issue with the fare collection gates is observed, either the on-site staff or the gate itself raises a "customer problem description", which is a description of the incident. Currently, an engineer must visit the affected gate at the station and review the customer problem description to propose a potential solution. Some solutions require a replacement part, which may not be available immediately, and/or require additional engineer visits. Therefore, in this project, we aim to automate the identification of the required replacement part by analyzing the input customer problem descriptions using an LLM:

$$f(\text{Customer Problem Descriptions}) \rightarrow \text{Replacement Part}$$

where f is the function learnt by the large language model that we will develop. An example pair is: "*device crashed, cold start failed, requires pc*" → *PART NEED: 9504-08019-1 QTY: 1 (AFM PC).*

Therefore, this problem is a classification problem, where we want to find the right part (i.e., "class") out of a potential list.

5 Dataset and Pre-Processing

5.1 Corpus Overview

Our dataset is based on 1-year of maintenance incident logs for fare collection gates collected (over 1M excel rows), comprising a blend of auto-generated firmware error codes and human-written narratives. As each incident may or may not link directly to a replacement part purchase, we filtered for the incidents that required a replacement part. In case many repair attempts were recorded, we use the part that was required in the final request, as it is likely the part that finally fixed the issue. Data preprocessing techniques, such as data cleaning, data leakage prevention, and mapping alphanumeric Part IDs into integer class labels were carried out and yielded a carefully curated dataset.

5.2 Tokenization and Part Class Assignment

Customer Problem Description Inputs. We use the original Llama tokenizer, which partitions the input natural language into different tokens which is the required input format for the LLM. Figure 4 illustrates an example for the difference between different tokenizers.

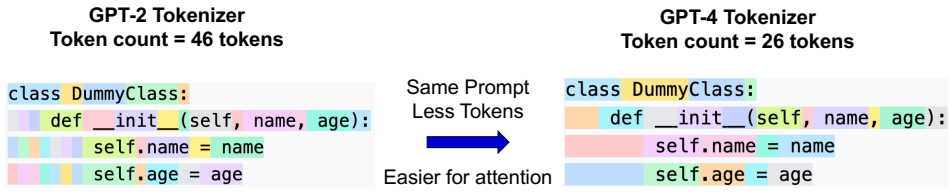

Figure 4: An illustration of the tokens different tokenizers yield after ingesting the same sentence.

Replacement Part Outputs. To convert the output from a natural language part number to different part classes, we assigned label 0 to the case when "no-part" was required, label 1 to a unique part number, label 2 to another unique part number, and so on. This in turn yields sample pairs of: "customer problem description" → "part class required". The dataset exhibited significant class (part) imbalance, with a dominant "no-part-required" class representing 85% of tickets. Ticket descriptions were truncated to 256 tokens to save compute, as this length covers 99% of maintenance narratives. The median lengths of maintenance logs were 112 tokens (IQR 97–128). Figure 5 illustrates the distribution of part occurrences in 1-month of the training logs.

6 Experimental Design

In our experiments, we utilized the Llama-3-8B model [12], an open-source LLM developed by Meta, which represents one of the current state-of-the-art architectures in the field. The model has been pre-trained on a vast corpus comprising approximately 15 trillion tokens (equivalent to around 12 trillion words), enabling it to capture rich contextual information and semantic relationships in natural language. This extensive pre-training equips the model with strong general language understanding capabilities, which we seek to leverage for our downstream task.

However, customer problem descriptions of fare collection gates often contain nuanced, domain-specific language that may not be well represented in the pre-training corpus. This linguistic complexity can lead to a degradation in performance when using the pre-trained model out-of-the-box. Moreover, our problem is framed as a classification task, in contrast to the next-token prediction objective used

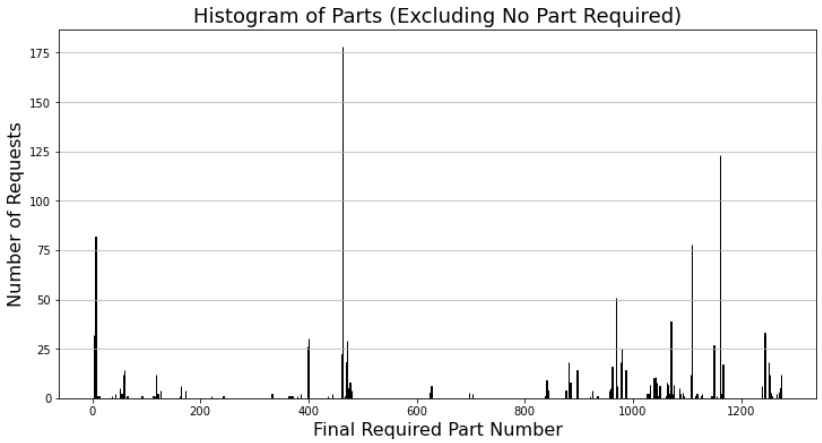

Figure 5: The distribution of part occurrences in 1-month of the incident logs.

during the model's pre-training. As a result, it is necessary to adapt the model to better suit our specific task through fine-tuning.

Given the limited size of our labeled dataset, and the computational cost associated with fine-tuning all 8 billion parameters of the model, we adopted a Parameter-Efficient Fine-Tuning (PEFT) approach. Specifically, we applied Low-Rank Adaptation (LoRA), which introduces trainable rank-decomposition matrices into the transformer architecture. This method allows us to update only a small subset of parameters while freezing the majority of the model weights, significantly reducing memory usage and training time, without compromising downstream performance.

6.1 Parameter-Efficient Fine-Tuning with Low-Rank Adaptation (LoRA)

The most conventional approach to adapting LLMs for downstream tasks is Supervised Fine-Tuning (SFT), where all or most of the model's parameters are updated using a small, task-specific labeled dataset. However, fine-tuning all parameters of a large-scale model, such as the Llama-3-8B, presents significant computational and memory challenges. Moreover, full fine-tuning on limited data risks *overfitting* or *catastrophic forgetting* of the model's pre-trained knowledge.

To address these limitations, we employed PEFT, a paradigm that adapts large pre-trained models by updating only a small number of additional parameters while keeping the vast majority of the original weights frozen. PEFT not only reduces computational requirements and memory usage but also improves generalization in low-resource scenarios by constraining the optimization space, thereby acting as an implicit regularizer.

Among various PEFT techniques, LoRA has gained widespread popularity due to its simplicity, effectiveness, and minimal overhead [7]. LoRA modifies the fine-tuning process by reparameterizing weight updates in the neural network as low-rank matrices. Instead of updating the full weight matrix $\mathbf{W}_0 \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA introduces a low-rank update $\Delta\mathbf{W}$ expressed as the product of two trainable matrices:

$$\mathbf{W} = \mathbf{W}_0 + \Delta\mathbf{W} = \mathbf{W}_0 + \mathbf{BA} \tag{1}$$

Here,

- $\mathbf{A} \in \mathbb{R}^{r \times d_{\text{in}}}$ and $\mathbf{B} \in \mathbb{R}^{d_{\text{out}} \times r}$ are trainable matrices,
- $r \ll \min(d_{\text{in}}, d_{\text{out}})$ is the rank of the adaptation,
- $\mathbf{W}_0$ is kept frozen during fine-tuning.

This formulation ensures that the number of trainable parameters is reduced from $d_{\text{in}} \cdot d_{\text{out}}$ (as in full fine-tuning) to $r \cdot (d_{\text{in}} + d_{\text{out}})$, a significant reduction when $r \ll \min(d_{\text{in}}, d_{\text{out}})$.

In practice, LoRA is typically applied to the attention mechanism of transformer-based models. Each attention head involves multiple linear projections — particularly the *query* $\mathbf{W}_q$, *key* $\mathbf{W}_k$, *value* $\mathbf{W}_v$, and *output* $\mathbf{W}_o$ weight matrices. For natural language processing tasks such as ours, the most common implementation of LoRA is to apply it to the *query* and *value* projections of the attention mechanism. In our implementation, we adopted this configuration, inserting LoRA adapters into the corresponding projection layers. This modification is illustrated in Figure 6, where LoRA adapters are inserted into the attention mechanism without altering the original architecture. During training,

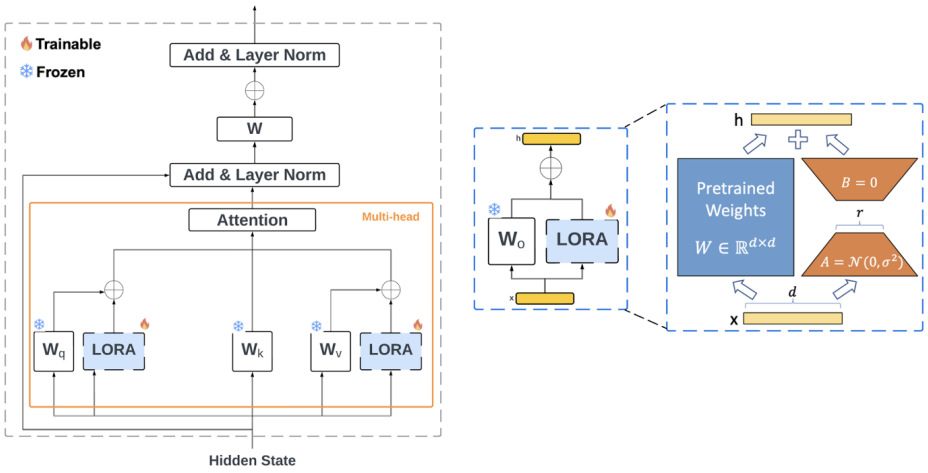

Figure 6: Integration of LoRA adapters into the query and value projection layers of the attention mechanism. Only the LoRA parameters are trainable; all other weights remain frozen [7].

only the LoRA parameters $\mathbf{A}, \mathbf{B}$ are updated, while the rest of the model remains unchanged. At inference time, the low-rank updates can be merged with the original weights for efficiency or kept as a separate computation. This approach enables task-specific adaptation with a minimal number of trainable parameters and no degradation in the model's general language capabilities.

Overall, LoRA allowed us to fine-tune the Llama-3-8B model on our classification task with minimal compute resources, while still achieving competitive performance to full fine-tuning.

6.2 Model Training

Our proposed model was initialized with the pre-trained Llama-3-8B model and subsequently fine-tuned on our customer description dataset. The data was partitioned using an 80/20 split for training and testing, respectively, resulting in 5,565 training samples (of which 835 were used for validation) and 1,352 testing samples. A LoRA configuration with a rank $r = 8$, scaling factor $\alpha = 16$, and a dropout probability of 0.05 was employed. Only these LoRA-specific parameters were updated during training, with all other pre-trained weights kept frozen. This resulted in a total of **4.2 million** trainable parameters, which corresponds to just **0.0524%** of the full model's parameters.

The fine-tuning process was conducted using the AdamW optimizer, which decouples weight decay from gradient updates to improve generalization. A batch size of 16 was achieved through micro-batching with a micro-batch size of 4 and gradient accumulation steps of 4. We trained the model for 33 epochs using a cosine learning rate scheduler with an initial learning rate of 3×10^{-4}. Weight decay was set to 0.01 to prevent overfitting. Mixed-precision training was enabled via bfloat16 (bf16) to reduce memory usage and accelerate training. Logging and evaluation were performed at each epoch using Weights & Biases. The best-performing model checkpoint was retained based on validation performance.

Training was conducted on an AWS cluster instance of type `ml.p4de.24xlarge`, equipped with 8 NVIDIA A100 GPUs, each with 40GB of memory. The complete fine-tuning process over 33 epochs required approximately 50 hours to complete, resulting in a total compute cost of around $2,000. This cost is significantly lower compared to the pre-training of Llama-3, which is estimated to have required tens to hundreds of millions of dollars in compute resources.

7 Results

To assess the performance and robustness of our fine-tuned model, multiple evaluations were conducted. First, the fine-tuned model was compared against its corresponding base (out-of-the-box) model to evaluate the effectiveness of the fine-tuning process. Second, the model's performance on human-written versus machine-generated descriptions was analyzed to assess its robustness to variations in input style and instruction type. Finally, the top-k accuracy was computed for each individual part to determine whether the model exhibited consistent performance across the full range of categories.

7.1 Performance Comparison: Base Model vs. Our Fine-Tuned Model

To evaluate the effectiveness of our fine-tuning strategy, we compared the performance of our fine-tuned model against the base (out-of-the-box) Llama-3 model using three widely adopted classification metrics: Top-1, Top-3, and Top-5 accuracy. These metrics provide insight into both the precision of the model's most confident prediction and its ability to include the correct label within a shortlist of top candidates.

- **Top-1 Accuracy**: Measures how often the model's most confident prediction exactly matches the ground truth label (part).
- **Top-3 Accuracy**: Measures whether the ground truth label (part) is included among the model's top 3 predicted parts.
- **Top-5 Accuracy**: Measures whether the ground truth label is included among the model's top 5 predicted parts.

While Top-1 accuracy reflects strict correctness, Top-3 and Top-5 accuracy are particularly useful in human-in-the-loop systems, where a human can choose among several model-suggested options.

Metrics	Our Model	Out-of-the-Box LLaMA Model	Accuracy Improvement
Top-1 Accuracy	0.835	0.0194	4204%
Top-3 Accuracy	0.934	0.1121	733%
Top-5 Accuracy	0.959	0.195	392%

Table 1: Accuracy comparison between our fine-tuned model and the base LLaMA-3 model.

As shown in Table 1, our fine-tuned model significantly outperformed the base model across all three metrics. Notably, the Top-1 accuracy improved from 1.94% to 83.5%, a relative improvement of over 43-fold. Top-3 and Top-5 accuracies also increased substantially, suggesting that our model not only provides more accurate predictions but also consistently ranks the correct part among its top suggestions. This is a critical property for practical deployments, especially in systems where predictions can be reviewed or validated by human users.

7.2 Robustness to Input Style: Human-Written vs. Machine-Generated Descriptions

To evaluate the robustness and generalization ability of our fine-tuned model, we assessed its performance separately on human-written and machine-generated customer descriptions. This comparison

helps determine whether the model can maintain predictive accuracy across variations in natural versus synthetic language inputs.

Metrics	Human-Written Descriptions	Machine-Generated Descriptions
Top-1 Accuracy	0.838	0.813
Top-3 Accuracy	0.939	0.896
Top-5 Accuracy	0.963	0.935

Table 2: Model accuracy comparison on human-written vs. machine-generated customer descriptions.

As shown in Table 2, the model performs consistently well across both input types, achieving comparable accuracy scores for human- and machine-generated descriptions. The slight drop in performance on machine-generated inputs is expected and likely attributable to a smaller sample size, rather than a limitation in model robustness.

These results suggest that the model generalizes effectively to diverse linguistic styles and is reliable for deployment in both manually-curated and machine-generated instruction environments. This robustness is particularly valuable for applications such as ours, where input variability is high and model outputs are used in downstream automated systems.

7.3 Per-Class Top-k Accuracy Analysis

To further evaluate the consistency and reliability of our fine-tuned model across different part labels, we analyzed Top-3 and Top-5 accuracy metrics on a per-part basis. These metrics are particularly relevant in applications where a human operator may select the correct part from among the model's top suggestions.

The model demonstrates strong overall performance, achieving over 84% Top-3 accuracy and over 87% Top-5 accuracy in 23 out of 26 classes. This indicates that even when the model's top prediction is incorrect, the true part often appears within the top-3 or top-5 predictions, underscoring the model's practical value in human-in-the-loop systems.

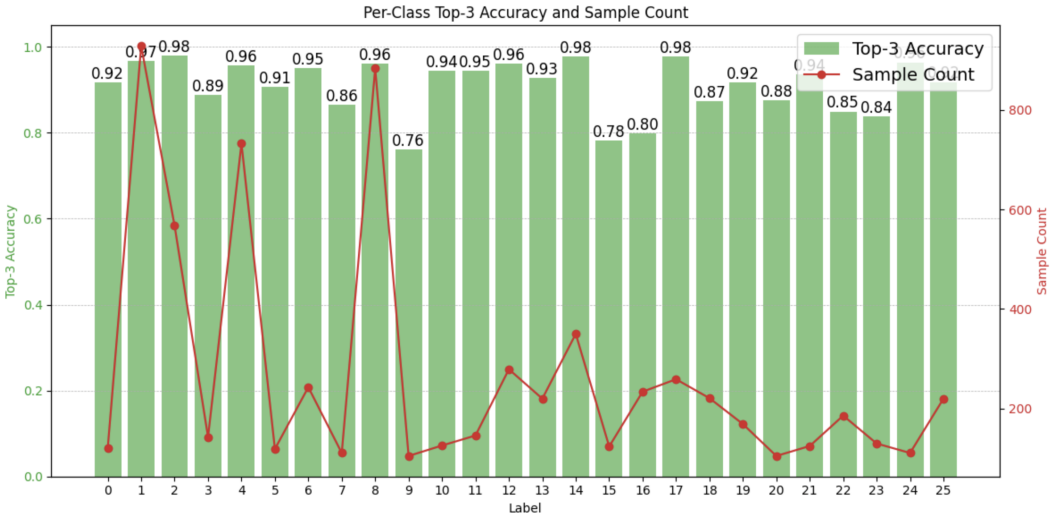

Figure 7: Per-part Top-3 accuracy across the 26 part labels along with the corresponding sample count.

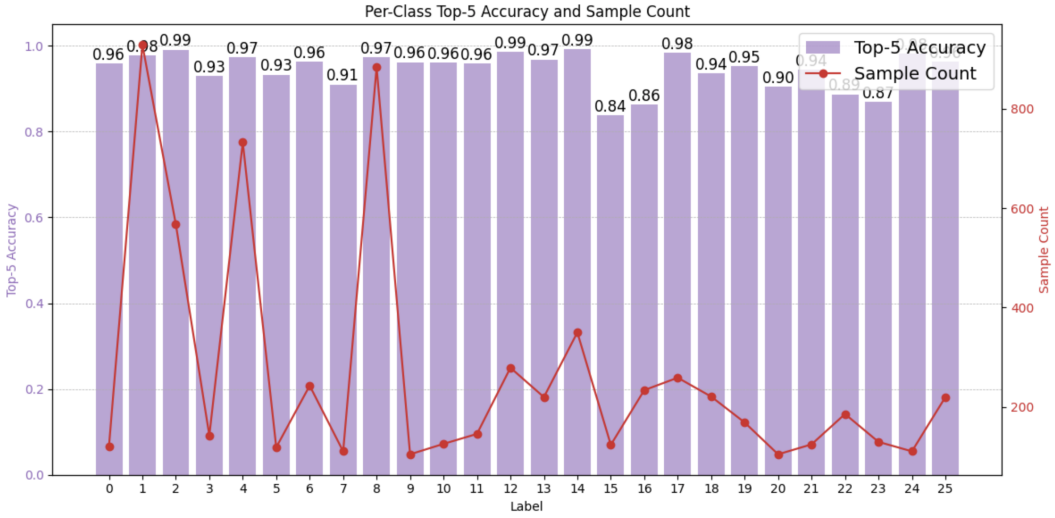

Figure 8: Per-part Top-5 accuracy across the 26 part labels along with the corresponding sample count.

7.4 Cumulative Accuracy and Coverage Analysis Across Part Labels

To assess both the model's *precision* (how often it predicts the correct part as the top-1 guess) and *coverage* (how often the correct part is included among its top few prediction), we plotted a stacked bar chart of Top-1, Top-3, and Top-5 accuracy across all 26 part labels in our filtered dataset, as shown in Figure 9.

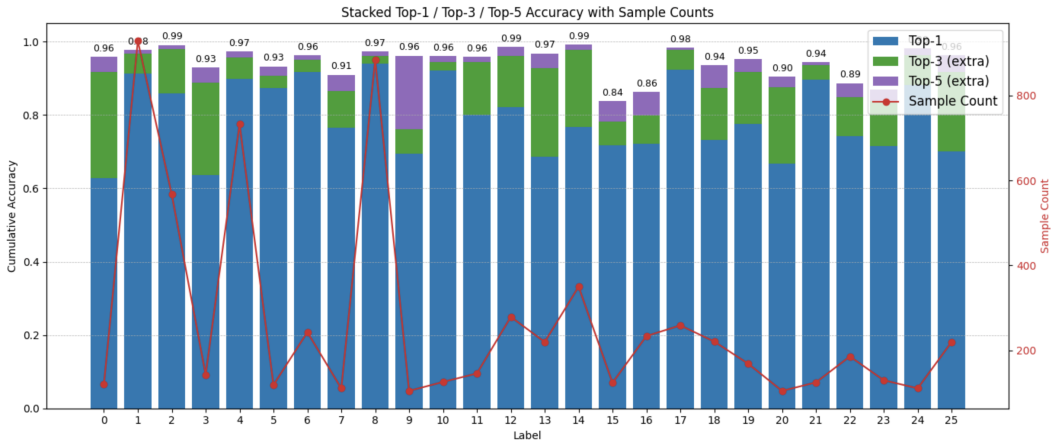

Figure 9: Stacked Top-1, Top-3, and Top-5 accuracy per part, with sample count shown as a red line. Each bar represents cumulative accuracy (Top-1 in blue, Top-3 gains in green, and Top-5 gains in purple), while the red line indicates the number of samples available for each part.

The stacked chart confirms that the model is not only highly accurate but also reliable: even when the top prediction is incorrect, the correct part is often included within the model's top-3 or top-5 prediction. This property is particularly valuable in real-world applications where predictions are reviewed by a human-in-the-loop or where downstream components benefit from having a ranked list of likely outcomes.

We also observe that part labels with lower Top-1 accuracy, such as labels 0, 3 and 20, often have relatively fewer training examples, as reflected by the sample count line. To further understand the relationship between sample count and predictive performance, we calculated the Pearson correlation

between the number of samples per part and the model's Top-k accuracy. The results indicate a positive correlation across all three accuracy metrics with Top-1 correlation = 0.5062 ($p = 8.32 \times 10^{-3}$), Top-3 correlation = 0.4607 ($p = 1.79 \times 10^{-2}$), and Top-5 correlation = 0.4078 ($p = 3.86 \times 10^{-2}$). All correlations are statistically significant at the 0.05 level. These findings suggest that the model performs better on classes with more training data, highlighting the importance of data balancing or augmentation strategies, especially for under-represented classes.

8 Conclusion

In this work, we introduced **PartLlama**, a parameter-efficient LoRA-based adaptation of the Llama-3-8B model for automated replacement-part prediction from customer problem descriptions of fare collection gates in large-scale transit maintenance. By fine-tuning only 0.0524% of model parameters on a curated corpus of maintenance incident logs, PartLlama achieves 83.5% Top-1, 93.4% Top-3, and 95.9% Top-5 accuracy, representing a dramatic improvement over the out-of-the-box baseline. Our preprocessing pipeline effectively addresses noisy, imbalanced textual data and maps alphanumeric part identifiers into a tractable classification space. Through systematic evaluation, we demonstrate that the model generalizes across human- and machine-generated inputs and maintains high reliability across the 26 most common part classes, with performance positively correlated to class sample size. Importantly, the compact LoRA adapter enables deployment on a single GPU or CPU, facilitating real-time inference and cost-effective integration into existing maintenance workflows. Overall, PartLlama exemplifies how parameter-efficient fine-tuning of large language models can unlock practical, high-accuracy predictive maintenance solutions with minimal computational overhead, supporting a shift from reactive repairs to proactive gate maintenance that reduces downtime and improves service reliability.

9 Limitations and Future Works

A limitation of the current model is that it is trained on a subset of parts. This is caused by the unbalance in part classes, as some parts may be requested very often, while others were never requested during the training period. Consequently, the model has no exposure to these rare or unseen parts, which can lead to a fall in prediction accuracy during practical deployment due to those parts. In addition, the current training data is static and limited to one year of historical incidents, meaning that the model may not adapt well to evolving operational conditions, newly introduced parts, or changes in terminology over time. Another limitation is the lack of *explainability* owing to the black-box nature of LLMs. While the model can output ranked part predictions, it does not currently provide insight into which parts of the input customer description influenced the decision. To effectively address these limitations, we propose to:

- Productionize the model by building a deployment pipeline which ingests real-time logs to generate ranked predictions in real time, ensuring cost-efficient inferences;
- Extend the current model to all existing classes beyond the 26 most frequently occuring ones;
- Develop an instruction-tuning model for this task, enabling the incorporation of advanced accuracy-improvement techniques;
- Implement periodic re-training on newly ingested logs to maintain performance against evolving patterns of failures and parts usage;
- Add explainability features that highlight the words, phrases, or error codes in the input that most strongly influenced the predictions of the model, enabling engineers to understand and verify model outputs more effectively.

References

[1] Transport for London. *Transport for London Quarterly Performance Report, Quarter 4 2024/25*. Tech. rep. Quarter 4 2024/25. Transport for London, June 2025.

[2] *Cubic Transportation Systems*. URL: https://www.cubic.com/transportation.

[3] Qiang Wang et al. "Learning Deep Transformer Models for Machine Translation". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. July 2019, pp. 1810–1822. DOI: 10.18653/v1/P19-1176. URL: https://aclanthology.org/P19-1176/.

[4] Hugo Touvron et al. "Llama 2: Open foundation and fine-tuned chat models". In: *arXiv preprint arXiv:2307.09288* (2023).

[5] Hugo Touvron et al. "LLaMA: Open and efficient foundation language models". In: *arXiv preprint arXiv:2302.13971* (2023).

[6] Alec Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI blog* 1.8 (2019), p. 9.

[7] Edward J Hu et al. "Lora: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3.

[8] Adailton Araujo et al. "From bag-of-words to pre-trained neural language models: Improving automatic classification of app reviews for requirements engineering". In: *Encontro Nacional de Inteligência Artificial e Computacional (ENIAC)*. SBC. 2020, pp. 378–389.

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks". In: *Advances in neural information processing systems* 27 (2014).

[10] Kevin Clark et al. "What Does BERT Look at? An Analysis of BERT's Attention". In: *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Aug. 2019, pp. 276–286. DOI: 10.18653/v1/W19-4828. URL: https://aclanthology.org/W19-4828/.

[11] Ashish Vaswani et al. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[12] Aaron Grattafiori et al. *The Llama 3 Herd of Models*. 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.