Disfluent Question Correction Using LLMs

By: Sepideh Bahadoripour

# Problem Statement

# In conversational workflows, users often input questions that contain errors due to typing mistakes or discontinuities in their thought process. Despite these input errors, it is crucial to accurately extract the intended question to provide correct answers and predict the query's nature or intent. This project aims to design and develop a Large Language Model (LLM) solution to correct disfluent questions, ensuring accurate understanding and response.

# Proposed Solution

To address the issue of erroneous user inputs, we propose fine-tuning a Large Language Model (LLM). Based on research and problem specifications, Llama 3.1 8b is selected as the base model. Llama, an open-source model pre-trained by Meta, is known for its ability to follow instructions effectively.

# Given the GPU memory limitations (e.g., a 24GB GPU), loading the full precision model for fine-tuning was not feasible. To overcome this, we employed Low-Rank Adaptation (LORA) to reduce the number of trainable parameters and Quantized Low-Ranked Adaptation (QLORA) to further quantize these parameters, representing them with fewer bits. These methods allowed efficient fine-tuning without compromising the model's performance.

# Data Pre-processing

After loading the data, the Llama chat template was applied, which structures the data into three roles: system, user, and assistant. This template aligns with how the model will be used in deployment, ensuring that the training data closely resembles real-world usage. Figure 1 shows an example of the chat template applied to a sample data row.

A computer code with black text

Description automatically generated

Figure - Applying LLama 3.1 chat template on a sample

# Model Fine-tuning

Based on the problem statement, the training data was split into training and validation sets, with 80% used for training and 20% for validation. Llama 3.1 was loaded in 4-bit quantization using the bitsandbytes library. To avoid fine-tuning all parameters, LORA with r=16 and alpha=32 was used, which made fine-tuning faster by only training 0.52% of parameters (41,943,040 trainable parameters instead of 8,072,204,288). The model was fine-tuned with a batch size of four for five epochs. Figure 2 shows the loss and validation loss during the fine-tuning process. As illustrated, the validation loss increased after the second epoch, indicating overfitting. Therefore, the model parameters at the second epoch were chosen as the final model.

The fine-tuning process takes ~3 hours on AWS Sagemaker using single L4 GPU with 24 GB of GRAM. The final model and tokenizer were saved to be used in the future.

Figure - plot of loss vs. validation loss for training epochs.

# Model Evaluation

The proposed solution was evaluated using valid data containing 1000 samples. Three evaluation metrics were chosen to assess the model:

**Exact Match**: This metric checks if the model’s output is exactly the same as the expected answer, without any variations. It is a strict measure of accuracy, where even minor differences in wording or punctuation can result in a non-match.

**String Distance**: Also known as edit distance, this metric measures how many edits (insertions, deletions, substitutions) are needed to change one string into another. It evaluates how close the model’s output is to the expected answer by quantifying the differences between the two strings.

**Embedding Distance**: This metric measures the distance between vectors in an embedding space that represents words or sentences. It evaluates how semantically close the model’s output is to the expected answer by comparing their positions in this high-dimensional space. This method captures the meaning and context of the text rather than just the exact wording.

Table 1 shows the results of these three metrics. The initial dataset compared disfluent and original questions, while the fine-tuned model compared model predictions with original questions.

Table - Evaluation results of the proposed solution on valid set

|  |  |  |  |
| --- | --- | --- | --- |
|  | Exact Match | String Distance | Embedding Distance |
| Initial Dataset | 0% | 0.17 | 0.11 |
| Fine-tuned Model | 79% | 0.025 | 0.016 |

As illustrated in Table 1, the fine-tuned model could convert 79% of disfluent inputs to the exact requested response and significantly decreased string and embedding distances. Moreover, some model responses had similar meanings to the requested response but with different wordings, which caused the embedding distance to be small. Table 2 shows some of these samples.

Table - Some samples of the similar but not exact responses.

|  |  |
| --- | --- |
| Model output | Original response |
| What is the current status of Haensch's study? | What is the current status of the Haensch study? |
| What does connection orientation require? | What does connection orientation require |
| What is an example of a non-specific immune response? | What immune response is not antigen-specific? |
| Pharmacy technicians are limited to what responsibilities? | What responsibilities are pharmacy technicians limited to? |
| How can a knot be distinctively indicated? | How can any knot be distinctively indicated? |

# Conclusion

This project successfully developed a Large Language Model capable of correcting disfluent user inputs by employing techniques like Low-Rank Adaptation (LORA) and Quantized Low-Ranked Adaptation (QLORA). The fine-tuned model demonstrated substantial improvements over the initial dataset across all evaluation metrics—Exact Match increased to 79%, while String Distance and Embedding Distance decreased significantly—indicating enhanced accuracy in understanding user intent despite input errors.

# Future Works

Future work could focus on further optimizing model efficiency without compromising accuracy or exploring alternative adaptation techniques that may yield better results with fewer computational resources. Additionally, expanding the dataset to include more varied forms of disfluencies could improve robustness. Implementing real-time feedback mechanisms for continuous learning from user interactions can also be considered to maintain relevance with evolving language use patterns.