Capstone MedBot

Abdullah Siddique

LinkedIn | Huggingface | Github

Formulation of Problem Statement

The rising cost of essential medications in India, driven by inflation and high pharmaceutical profit margins, has significantly increased the financial burden on patients, limiting their access to necessary treatments. In response to this issue, the Government of India introduced the Pradhan Mantri Bhartiya Janaushadhi Pariyojana (PMBJP) in 2008, aiming to provide affordable, high-quality generic medicines through Janaushadhi Kendras. Despite these efforts, post-pandemic inflation, coupled with limited public awareness of these alternatives and the locations of the Kendras, has diminished the program's overall impact.

Possible Solutions

Three approaches were considered to address this problem:

i. Develop a large language model encompassing all relevant medical information and guidelines.
ii. Build a Retrieval-Augmented Generation (RAG) system using pre-existing large language models and integrate it with a database of medicines and locations.
iii. Fine-tune an existing pre-trained language model tailored to our specific needs.

After careful evaluation, I chose the third approach—fine-tuning a pre-trained model.
The rationale behind this decision included:

i. Cost efficiency
ii. Resource efficiency
iii. Faster development time compared to creating a new language model or implementing a complex RAG system
iv. Ease of deployment
v. Low computational cost

While this approach does not represent a radical innovation, it offers a balanced, pragmatic solution that addresses the project's objectives within the available resources and timeline.

Collection and Dataset Formation

The dataset for this project was constructed from two primary sources:

i. Generic medicine data from the Government Of India Public Portal.
ii. Branded medicine data sourced from Pharma Easy.

The objective was to link each generic medicine with five equivalent branded medicines, resulting in a potential dataset of over 10,000 rows (based on the initial 2,000 generic medicines). However, due to the manual and labor-intensive nature of the task, only a portion of the dataset was completed, covering roughly 10% of the total target size. My contribution involved gathering 400 rows, while my teammates collected an additional 400 and 200 rows, respectively. The dataset was focused on key medicine categories, including antibiotics, antivirals, and antidiabetics.

I subsequently processed this raw data into a format suitable for fine-tuning the model. Initially, the dataset included columns for context, questions, and answers, but this format was later refined to consist of simplified question-answer pairs.

The datasets created during this process were:

1. Raw Dataset
2. Initial Dataset
3. Simplified Augmented

Phase 1 Solution: Capstone Project Components

The first phase of the project comprised three major components:

1. Location Search Engine
2. Quantized Fine-Tuned Large Language Model (LLM)
3. Chatbot Function

Component 2 Progress: Quantized Fine-Tuned LLM Model

Given the constraints of limited RAM and limited CPU/GPU computational power, Google Colab's T4 was selected as the optimal platform for fine-tuning and quantization. It provided the necessary balance between efficiency and computational capacity.

This component was further divided into two main tasks:

1. Fine-Tuning
2. Quantization

Fine-Tuning Process Documentation

Initial Setup

The fine-tuning process began by selecting a few promising models from both Small Language Models (SLMs) and Large Language Models (LLMs), with a focus on models that had:

1. Good conversational capabilities
2. Open-source availability, avoiding the need for external requests and APIs

The following models were shortlisted for evaluation:

1. Llama 2
2. Llama 3.1
3. Phi-3
4. Mistral
5. Llama 3.2 (released on 25th September)

Each model was fine-tuned in successive phases as I refined the approach to identify the most suitable model for our specific requirements.

Phase 1: Fine-Tuning (Llama 2 and Phi-3)

Using the Autotrain GitHub repository and following tutorial-based guidance, I fine-tuned Llama 2 and Phi-3 models. The dataset contained 2,000 rows and took less than 20 minutes per run. However, despite successful runs, the training loss did not drop below 1.00, and issues with merging LoRA layers with the base model persisted due to size mismatches. After encountering these persistent issues, I decided to proceed without addressing the LoRA mismatch, as the model's loss remained unsatisfactory.

Phase 2: Fine-Tuning (Llama 2)

In this phase, I concentrated solely on Llama 2, again using the Autotrain Colab script for improved efficiency. However, the LoRA size mismatch issue persisted, preventing successful merging of fine-tuned layers with the base model.

Phase 3: Fine-Tuning (Llama 3.1 and Mistral)

This phase introduced Llama 3.1 and Mistral, with guidance of the tutorial, both of which proved more stable and easier to fine-tune. The loss consistently dropped below 0.09 across multiple runs. Memory efficiency improved, but quantization during initialization was absent, resulting in resource-intensive merging of fine-tuned layers.

Phase 4: Fine-Tuning (Llama 3.1 8B)

At this stage, I utilized the Unsloth Google Colab script optimized for single GPUs, like Colab's T4. This method allowed me to fine-tune Llama 3.1 using our medicine-related dataset. A key change here was the inclusion of quantization, enabling the model to be saved in the .gguf format, and successfully merge fine-tuned layers with the base model.

Phase 5: Dataset Refinement and Prompt Templates

During this phase, I shifted focus to improving the dataset and refining the prompt templates used for fine-tuning. The dataset was simplified into question-answer pairs, with corresponding templates designed for the Llama models.

Phase 6: Simplification and Fine-Tuning (Llama 3.1)

The dataset was further simplified, and input prompts were structured around Llama 3.1's Q&A format. This allowed for a significant improvement in generalization, enabling the fine-tuned model to generate more flexible and adaptable responses.

Phase 7: Augmented Dataset and Final Fine-Tuning

I expanded the dataset to over 7,000 rows and continued training. This resulted in a model with decent performance and a well-generalized knowledge base, marking a significant milestone in the project.

Phase 8: Hyperparameter Tuning

Several hyperparameters were adjusted to optimize performance, including:

1. Learning rates: Optimal between 2e-4 to 5e-5
2. Gradient accumulation: 2 was found to be optimal
3. Batch sizes: Increased as gradient accumulation decreased
4. LoRA Alpha: Values above 10 led to poor generalization
5. Rank (r): 128 was necessary for accurate medicine name and dosage handling

These efforts culminated in the creation of 23 different models, with one achieving the best overall performance.

Phase 9: Troubleshooting and Limitations

From late September, models began to encounter saving issues, with tokenizer-related warnings disrupting the fine-tuning process. Despite troubleshooting, these issues persisted, prompting me to revert to the last successfully saved model configuration.

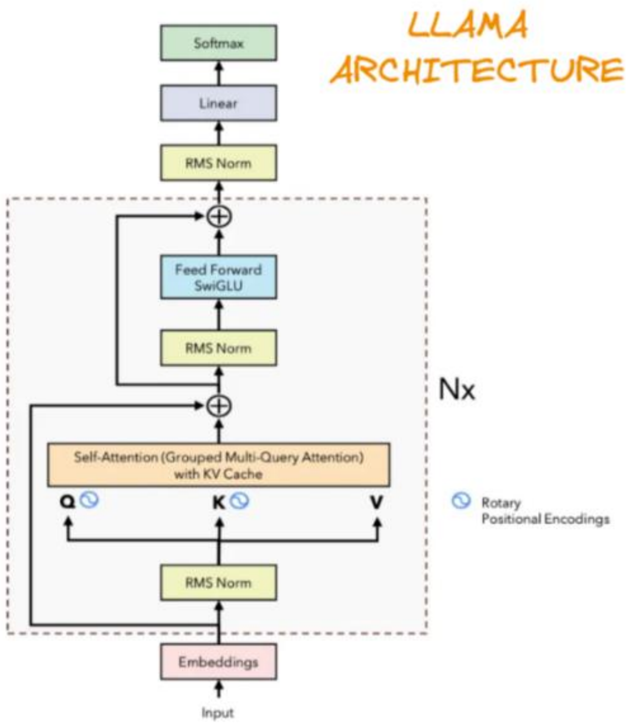
Phase 10: Introduction of Llama 3.2

On 25th September, Meta released Llama 3.2, a more distilled and efficient version of its predecessor. Unfortunately, the same saving issues persisted, though I was able to use the LoRA layers in a trial chatbot code.

Phase 11: A Final Breakthrough

The solution involved fine-tuning as usual but using the base model's tokenizer rather than the combined tokenizer. This approach succeeded, resulting in two fully functional models:

1. Llama 3.1 (8-bit quantized)
2. Llama 3.2 (unquantized)



I conducted Evaluation Of the Models. As Both models underwent head-to-head evaluation, each tested on 1,000 questions with consistent chatbot script parameters. The results revealed performance from both models; Model 1 Being High in accuracy while being high in latency as well, while Model 2 with much lower latency.

Concluding Model 2, even though not having as generalised knowledge base with 1B parameters compared to 8B parameter of Model 1, was much faster in response without being quantised. While Model 1 required quantisation while hindered the accuracy.

Following the success of Model 2, and further improvement were in trial phase, marked the completion of Phase One.

Phase 2 Solution: Hybrid Multimodal Approach (In Progress)

The next phase of the project will involve a hybrid solution, combining:

1. Fine-tuned RAG Approach
2. Medicinal Details for Drugs
3. Prescription Reader