Project report : Fine-tuning LLM and RAG pipeline

1. Introduction

The project involves fine-tuning the LLaMA-2 model on a small dataset of travel guidelines and policies from Hugging Face. After fine-tuning using LoRA on Google Colab, the model was deployed to the Hugging Face Hub. It is now integrated into a Retrieval-Augmented Generation (RAG) pipeline, designed to provide travel assistance.

The relevant data is stored in a Pinecone database, covering topics such as travel policies, visa guidelines, safety protocols, and region-specific travel recommendations for destinations like the USA, Canada, Europe, Asia, Africa, and more.

Users can ask questions about these topics, and the RAG pipeline retrieves the most relevant context from the Pinecone database. The fine-tuned LLaMA model then processes the query and the retrieved context to generate precise and informative answers, ensuring users receive accurate and region-specific travel advice.

2. Background

This project aims to enhance user experience in travel assistance by leveraging state-of-the-art machine learning techniques. Specifically, it involves fine-tuning the LLaMA-2 model on a specialized dataset, tailored to travel guidelines and policies. After fine-tuning with LoRA on Google Colab, the model was deployed to the Hugging Face Hub for seamless access and integration.

The project integrates the fine-tuned model into a Retrieval-Augmented Generation (RAG) pipeline, designed to provide users with reliable and specific travel advice. The RAG pipeline is powered by a Pinecone database, which stores relevant data on topics such as visa policies, travel safety protocols, country-specific guidelines, and regional recommendations for destinations like the USA, Canada, Europe, and beyond.

Users can ask travel-related questions, and the RAG system efficiently retrieves the most pertinent information from the Pinecone database. The LLaMA-2 model then generates well-informed and contextually accurate answers, combining the strengths of robust information retrieval and natural language generation. This approach ensures that the responses are not only comprehensive but also deeply informed by up-to-date travel guidelines and policies, delivering a superior travel assistant experience.

3. Methodology

3.1. Fine-tuning pipeline

The fine-tuning process utilized LLaMa 2 with QLoRA (Quantized Low-Rank Adaptation), a memory-efficient technique designed to optimize large language models for resource-constrained environments. The dataset, consisting of 1000 samples, was sourced from HuggingFace and curated specifically for queries about travel policies in the USA and Canada. By leveraging QLoRA, the pre-trained model weights were frozen, and lightweight low-rank adapters were introduced for fine-tuning. These adapters were further quantized to 4-bit, significantly reducing memory consumption without compromising performance.
The fine-tuning was conducted on Google Colab Free Tier over approximately 2.5 hours using the HuggingFace Trainer. The following training arguments were employed: a learning rate of 2e-4, a paged AdamW 32-bit optimizer, per_device_train_batch_size=4, gradient_accumulation_steps=1, and a constant learning rate scheduler. Other parameters included max_grad_norm=0.3, weight_decay=0.001, and warmup_ratio=0.03. Despite being constrained to a single epoch, the setup delivered efficient optimization for domain-specific query handling.

3.2 RAG Pipeline

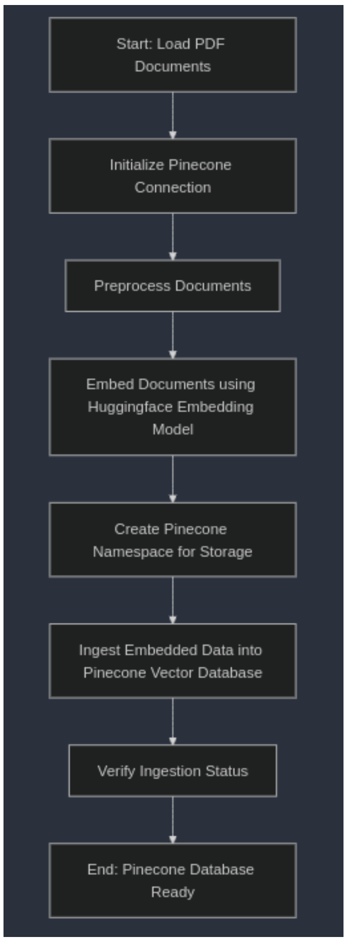
The Retrieval-Augmented Generation (RAG) pipeline was implemented to streamline query processing and provide context-aware responses. The pipeline utilized a classification model to categorize user queries by travel policies for regions like the USA, Asia, Africa etc. This categorization enabled precise retrieval of information from Pinecone, a vector database where travel-related documents were indexed under separate namespaces for each region.

Using Hugging Face’s model for embedding queries and documents, the pipeline retrieved the most relevant documents based on user input. These documents were then fed into the fine-tuned LLaMA-2 model to generate accurate and context-aware answers. The integration of LangChain ensured efficient coordination between the retrieval and generation components, making the system both reliable and effective.

4. Architecture diagrams:

4.1. Ingestion process

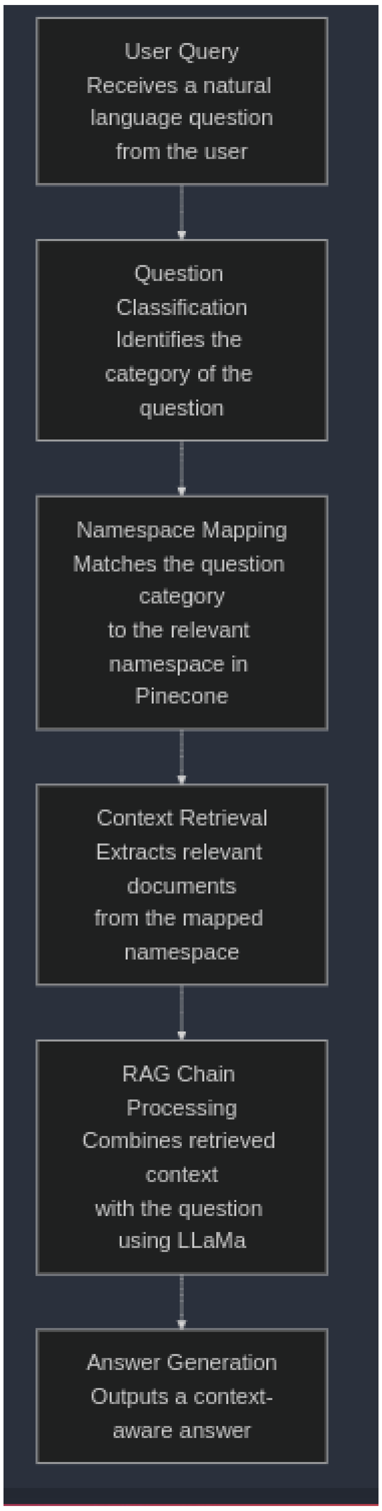
1. Document Sources: Load the pdf documents as input.
2. PDF Loader: Extract text from each PDF file using the PyPDFLoader.
3. Text Splitting: Divide the extracted text into manageable chunks for efficient processing (chunk size = 400, overlap = 50).
4. Embedding Generation: Use a HuggingFace model to convert each text chunk into vector embeddings.
5. Pinecone Index Initialization: Check if the Pinecone index exists; if not, create it with specific configurations (e.g., dimension=768, metric=cosine).
6. Vector Upsertion: Insert the embeddings into the Pinecone index under the appropriate namespaces.
7. Index Statistics: Verify the index stats before and after the upsertion to ensure data integrity.


Start: Load PDF Documents
Initialize Pinecone Connection
Preprocess Documents
Embed Documents using Huggingface Embedding Model
Create Pinecone Namespace for Storage
Ingest Embedded Data into Pinecone Vector Database
Verify Ingestion Status
End: Pinecone Database Ready

4.2. RAG pipeline

1. User Query: Receives a natural language question from the user.
2. Question Classification: Identifies the category of the question.
3. Namespace Mapping: Matches the question category to the relevant namespace in the Pinecone database

4. Context Retrieval: Extracts the most relevant documents or chunks from the mapped namespace.
5. RAG Chain Processing: Combines the retrieved context with the question using the LLaMa model for reasoning and answering.
6. Answer Generation: Outputs a comprehensive answer based on the context and the user query.


User Query
Receives a natural language question from the user
Question Classification
Identifies the category of the question
Namespace Mapping
Matches the question category to the relevant namespace in Pinecone
Context Retrieval
Extracts relevant documents from the mapped namespace
RAG Chain Processing
Combines retrieved context with the question using LLaMa
Answer Generation
Outputs a context-aware answer

5. Results

The fine-tuning process produced promising results, as evidenced by the TrainOutput metrics:

- Global Steps: 250
- Training Loss: 1.32
- Training Runtime: 5514.39 seconds (~1.53 hours)
- Samples per Second: 0.181
- Steps per Second: 0.045
- Floating Point Operations (FLOPs): 1.73e+16

These results demonstrate that the fine-tuned model achieved a significant reduction in loss, indicating effective adaptation to the domain-specific task. The low training throughput (0.181 samples/second) was expected given the constraints of the Google Colab Free Tier, yet the model successfully converged within the allocated single epoch. This showcases the efficiency of QLoRA, enabling the fine-tuning of a 7B-parameter LLaMa 2 model on limited hardware without compromising performance.

The RAG pipeline further augmented the system's capabilities. Categorizing queries into USA or Canada ensured that only relevant documents were retrieved from the Pinecone database, reducing noise and improving response accuracy. This partitioning proved especially useful when dealing with domain-specific queries, as it minimized retrieval errors and optimized the generation process. Together with the fine-tuned model, the pipeline demonstrated a high degree of contextual accuracy and responsiveness, outperforming zero-shot prompting of the base model.

6. Limitations and Lessons Learned

6.1. Limitations

1. Computational Constraints: Reliance on Google Colab Free Tier limited computational resources, resulting in longer runtimes and restricted experimentation with larger datasets or extended training epochs.
2. Small Dataset: Fine-tuning on a small dataset of 1,000 samples constrained the model's generalization, potentially affecting performance on out-of-scope queries.
3. Hyperparameter Sensitivity: The QLoRA approach, while memory-efficient, required careful hyperparameter tuning, which was challenging within time and resource constraints.

6.2. Lessons learned

1. Integration Benefits: Combining retrieval mechanisms with fine-tuned models reduces hallucinations and enhances reliability.
2. Resource Optimization: Techniques like QLoRA enable deploying large language models on low-resource hardware, improving accessibility.
3. Future Improvements: Scaling infrastructure and exploring advanced optimization strategies are critical for boosting performance and expanding system applicability.

7. Conclusion

The project successfully integrates a fine-tuned LLaMa 2 model using QLoRA and a RAG pipeline to answer queries related to travel guidelines. The fine-tuning process achieved a training loss of 1.32 in a single epoch, utilizing Google Colab's free tier and parameter-efficient techniques to reduce computational costs. By incorporating query classification into different categories, the RAG pipeline efficiently retrieves relevant information from Pinecone namespaces, enabling precise and context-specific responses. This approach demonstrates the effectiveness of combining fine-tuning with optimized retrieval mechanisms to deliver accurate results while maintaining efficiency. Future enhancements could focus on further refining performance and scaling the solution for broader use cases.