Project Report: RAG-based COVID-19 Travel Guidelines Chatbot

Arjun Rajesh Kulkarni

1. Introduction

The project involves building a chatbot application that provides accurate and up-to-date information related to COVID-19 travel guidelines for air travel. The system uses a Retrieval-Augmented Generation (RAG) approach with LangChain and OpenAI's GPT-3.5 Turbo model for natural language processing and response generation. The chatbot is designed to extract text from a PDF document containing COVID-19 travel guidelines, process it, and return helpful answers to user queries through a simple Gradio interface.

1.1 Objectives

- To create a chatbot that answers queries related to COVID-19 air travel guidelines.
- To efficiently extract text from PDFs, including images, using OCR and process the text for querying.
- To implement RAG using LangChain, OpenAI's GPT model, and Faiss for similarity search and vector embedding storage.
- To provide a user-friendly interface for interacting with the chatbot.

1.2 Scope

- The system supports queries related to COVID-19 guidelines from air travel documents in PDF format.
- It provides real-time responses to users using a pre-trained GPT-3.5 Turbo model.
- The system uses chunking, OCR, and vector embeddings for efficient querying and response generation.

2. System Design and Architecture

2.1 Overview

The system architecture consists of several components, which include:

1. **PDF Text Extraction**: Extracts text, including OCR for images, from a provided PDF document.
2. **Text Preprocessing**: Cleans and processes the extracted text by removing unwanted characters like `\n` and `|`.
3. **Chunking and Embedding**: The processed text is divided into chunks and stored in a vector database (Faiss) with embeddings.
4. **Retrieval and Question Answering**: LangChain is used to retrieve the most relevant chunk based on user queries, process it with GPT-3.5, and generate a well-formatted answer.
5. **Gradio Interface**: A simple web interface for users to interact with the chatbot.

2.2 Tools and Technologies

- **LangChain**: Used for managing the RAG process (text chunking, retrieval, and prompt management).
- **OpenAI GPT-3.5 Turbo**: Used for language modeling and generating answers from the retrieved text.
- **Faiss**: A library for efficient similarity search and storing vector embeddings locally in memory.
- **OCR (Optical Character Recognition) using unstructured.io**: Applied to extract text from images within the PDF.
- **Gradio**: Used for the user interface to allow easy querying of the chatbot.

3. Implementation Details

3.1 PDF Text Extraction and OCR

- The input document (COVID-19 air travel guidelines) is in PDF format, which may contain images. We utilize an OCR library (Tesseract) to extract text from these images.
- The text is then processed to remove unwanted characters such as `\n` and `|`, which may appear during the extraction.

3.2 Text Preprocessing

- Once the text is extracted, we perform the following preprocessing:
  - **Removal of unwanted characters**: Characters such as `\n`, `|`, and other non-text elements are cleaned using Python's built-in string functions.
  - **Formatting**: We ensure that the text is correctly formatted and ready for chunking.

3.3 Chunking with LangChain

- The cleaned text is divided into manageable chunks using LangChain's `RecursiveCharacterTextSplitter`. The chunking configuration is set with:
  - **Chunk size**: 1000 characters.
  - **Overlap**: 100 characters, ensuring the chunks are contextually connected.

3.4 Embeddings with OpenAI

- For each text chunk, an embedding vector is generated using OpenAI's GPT-3.5 Turbo model. These embeddings represent the semantic content of the chunks in a form that can be easily searched.
- The embeddings are stored locally in a Faiss vectorstore, which enables fast similarity searches.

3.5 Setting up the Retrieval QA Chain

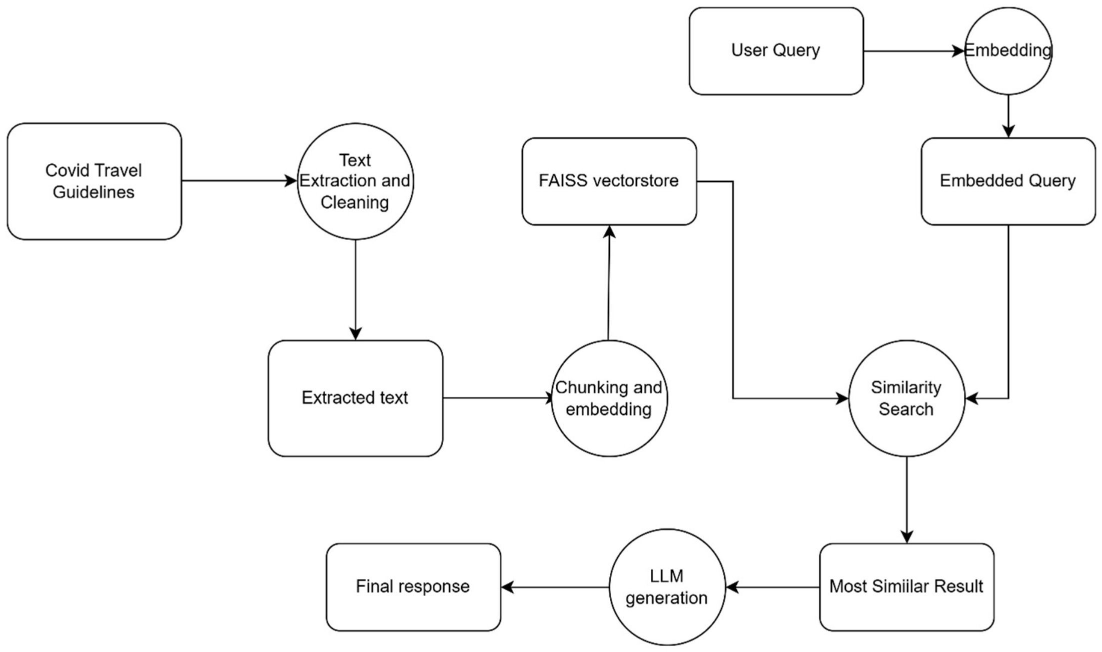
- A vectorstore retriever is initialized in LangChain to perform similarity searches on the Faiss vectorstore.

- The user’s query is passed into the retrieval QA chain, where LangChain retrieves the most relevant chunk based on the query’s similarity to the embeddings.
- The relevant chunk is then processed by the GPT-3.5 model to generate an accurate and formatted response for the user.

3.6 User Interface with Gradio

- A Gradio interface is set up to facilitate interaction with the chatbot. It consists of:
    - **Input Box**: The user can input their query regarding COVID-19 air travel guidelines.
    - **Output Box**: The chatbot returns the answer generated by the model, providing a seamless user experience.

4. Workflow Diagram of Chatbot



5. Conclusion

This chatbot effectively processes a PDF containing COVID-19 travel guidelines using OCR, chunking, embeddings, and retrieval-augmented generation (RAG) with OpenAI's GPT-3.5 Turbo. The final system is deployed with a Gradio interface, allowing easy interaction for users to ask relevant questions and receive detailed, accurate answers based on the latest guidelines.

6. Future Improvements

While the current system is functional, several improvements could be explored in the future:

1. **Real-Time Document Updates**: Implement a system to allow for real-time updates to the PDF, ensuring that the chatbot stays up to date with the latest travel guidelines.
2. **Multi-Language Support**: Extend the chatbot's capabilities to handle multiple languages, allowing users from different regions to interact with the system.
3. **Large-Scale Deployment**: Scale the chatbot for handling large volumes of queries, potentially by moving the vectorstore to a cloud database for better performance and persistence.
4. **Advanced Error Handling**: Implement better error handling for edge cases, such as no relevant data found or OCR extraction issues.
5. **Integration with Other Travel Guidelines**: Extend the scope of the chatbot to handle other travel-related guidelines, such as health check procedures, flight booking, and customs regulations.