Retrieval-Augmented Generation (RAG) Chatbot

A Mental Health Assistant

Fares Hazem (ID: 20221443356)
Ali Ashraf (ID: 2103106)
Ahmed Dawood (ID: 20221454408)
Ahmed Yousri (ID: 2103108)
Khalid Mansour (ID: 20221320444)

May 5, 2025

Project Overview

This project implements a Retrieval-Augmented Generation (RAG) chatbot designed to answer user questions using real-world information from websites and documents related to **mental health**. The chatbot combines document retrieval with a generative language model to provide accurate and context-aware responses. Unlike existing frameworks such as LangChain, this project was developed from scratch to demonstrate a deeper understanding of RAG architecture.

Project Goal

The main objective of this project is to develop a chatbot that:

- Collects and understands domain-specific content.
- Retrieves relevant information for user questions.
- Generates informative responses using a language model.

Chosen Topic and Use Case

Domain: Mental Health

Use Case: The chatbot answers questions about mental health issues such as anxiety, depression, and treatment options. It aims to raise awareness and provide reliable information to users.

Project Phases

Phase 1: Choose a Domain and Use Case

- Domain: Mental Health
- Use Case: Answer user queries on mental health topics.

Phase 2: Collect Data

- Collected over 20,000 words of mental health content using `trafilatura`.
- Output: `mental_health_data.csv`

Phase 3: Preprocess and Chunk the Text

- Cleaned and split content into chunks (200–500 words).
- Output: Cleaned and chunked dataset.

Phase 4: Embed the Chunks

- Used `all-MiniLM-L6-v2` model to generate sentence embeddings.
- Output: Embeddings saved as `embeddings.npy`.

Phase 5: Create a Vector Store

- Stored embeddings in a FAISS index for fast retrieval.
- Output: `mental_health_index.faiss`

Phase 6: Build the RAG System

- Embedded user queries and retrieved top 3–5 relevant chunks.
- Used a language model to generate answers.

Phase 7: Build a Chat Interface (Bonus)

- Created a user-friendly interface using `Streamlit`.
- Allows users to type questions and receive responses.

Project Structure

```
data/
    chunk_metadata.csv
    cleaned_chunk_metadata.csv
    cleaned_chunked_data.csv
    mental_health_data.csv
Embeddings/
    embeddings.npy
models/
    mental_health_index.faiss
scripts/
    1) Collect_Data.py
    2) Preprocess_and_Chunk_the_Text.py
    3) Embed_the_Chunks.py
    4) Create_a_Vector_Store.py
    5) Clean_and_Save_the_Data.py
    6) Build_the_RAG_System.py
    7) Build_Chat_Interface.py
```

How to Run the Project

1. Install dependencies: `pip install -r requirements.txt`
2. Run scripts in order from `scripts/` to build the chatbot.
3. Launch the interface using: `streamlit run "scripts/7) Build_Chat_Interface.py"`

Key Features

- Custom RAG implementation (no external frameworks).
- Accurate, domain-specific responses.
- FAISS for efficient semantic search.
- Streamlit interface for user interaction.

Future Enhancements

- Expand dataset with more mental health topics.
- Integrate advanced language models.
- Support multiple languages.

Test Report

A comprehensive `Test_Report.md` is included, showcasing chatbot responses to various user queries to demonstrate its capabilities and reliability.