README: LAB 6

Chat with PDF App

This project is a web application that allows users to upload PDF documents and interact with them using an AI-powered chatbot interface. The app is built using `Streamlit`, `LangChain`, `FAISS`, and `Hugging Face` for natural language processing and retrieval tasks. It also leverages `Llama-2` for local language model inference.

Features

- **Upload PDF documents:** Upload multiple PDFs and process them for chat interactions.
- **Text chunking and embedding:** The app splits PDF content into manageable chunks, embeds the text using Hugging Face models, and stores the embeddings in a FAISS vector store.
- **Conversational chatbot:** Engage in a conversation with your PDF content using Llama-2 as the underlying language model.
- **Conversational retrieval chain:** The app utilizes a conversational retrieval chain to answer multi-turn questions from the uploaded PDFs.

Requirements

To run the application, you'll need the following dependencies:

- Python 3.9+
- `streamlit`
- `PyPDF2`
- `langchain`
- `faiss-cpu`
- `huggingface_hub`
- `dotenv`
- `LlamaCpp` for Llama-2 inference

Installation

1. **Install the required libraries:**
   You can install the necessary dependencies using `pip`:

   ```bash
   pip install streamlit PyPDF2 langchain faiss-cpu huggingface_hub python-dotenv

2. Download and Configure the Llama-2 Model:
   - Download the llama-2-7b-chat.Q3_K_S.gguf model and place it on your local machine.
   - Update the model_path in the app.py file to point to the correct location of the downloaded model:

llm = llamacpp.LlamaCpp(model_path="path_to_your_model/llama-2-7b-chat.Q3_K_S.gguf", n_ctx=1024, n_batch=512)

3. Set up Hugging Face API token:
   - Get your Hugging Face API token from Hugging Face.
   - Create a .env file in the project root and add the token as follows:

     **touch .env**

     Inside the .env file:
     **HF_TOKEN=your_huggingface_token_here**

     Running the App
     Once everything is set up, run the app using the following command:

     **streamlit run app.py**
   ```

Code Explanation

1. app.py

- PDF Processing:
- The get_pdf_text() function extracts text from uploaded PDFs using the PyPDF2 library.
- Text Chunking:
- The get_text_chunks() function splits the extracted text into smaller chunks using CharacterTextSplitter for better processing and embedding.
- Embeddings and Vector Store:
- The get_vectorstore() function uses Hugging Face embeddings (sentence-transformers/all-MiniLM-L6-v2) to generate text embeddings and stores them in a FAISS vector store for efficient similarity searches.
- Conversational Retrieval Chain:
- The get_conversation_chain() function sets up the conversational retrieval chain using LlamaCpp, a wrapper for running Llama models locally. It retrieves relevant information from the FAISS vector store based on user queries.
- Handling User Input:
- User input is processed, and the chatbot responds using multi-turn question answering, displaying the interaction using HTML templates.

2. htmlparser.py

- Provides the HTML and CSS templates for rendering the chatbot messages with user-friendly formatting. The bot_template and user_template handle the display of chatbot and user messages, respectively.

3. Using Llama-2 Model (llama-2-7b-chat.Q3_K_S.gguf)

Llama-2 is a high-performance language model designed for conversational tasks. The .gguf model format is a quantized version, optimized for efficient inference on consumer hardware using llamacpp.

- Model Path: Ensure that the model_path in the code correctly points to where the model is stored on your machine.
- Running LlamaCpp: The model is loaded via the llamacpp interface, allowing it to run locally on your hardware. Parameters such as n_ctx (context window size) and n_batch (batch size) can be adjusted based on your system capabilities.