LLM Question-Answering Application Documentation

Overview

The **LLM Question-Answering Application** allows users to upload various document formats (PDF, TXT, DOCX) and interact with the contents through a natural language interface. The application employs large language models (LLMs) to generate answers based on the provided documents. This document provides an overview of the application's architecture, functionality, and instructions for local deployment.

Features

- **Document Uploading:** Users can upload documents in PDF, TXT, or DOCX formats.
- **Chunking and Embedding:** Uploaded documents are processed into smaller chunks for efficient querying.
- **Question-Answering:** Users can ask questions related to the uploaded documents, and the LLM will generate relevant answers.
- **Chat History:** The application keeps track of the question-and-answer history for user reference.

Application Architecture

The application is structured into two main modules:

1. **Frontend (Streamlit)**: This module handles user interactions, including file uploads, question inputs, and displaying answers.
2. **Backend (LLM and Document Processing)**: This module manages document loading, chunking, embedding, and interfacing with the LLM for question-answering.

Technologies Used

- **Streamlit**: A Python library for creating web applications.
- **Langchain**: A framework for building applications using LLMs.
- **Ollama**: A library for leveraging LLMs in a streamlined manner.
- **Chroma**: A vector database for storing document embeddings.

Installation Instructions

To set up the application on your local machine, follow these steps:

Prerequisites

- **Python 3.11 or later**: Ensure you have Python installed on your system. You can download it from python.org.
- **Docker (optional)**: If you prefer to run the application in a Docker container, make sure Docker is installed. You can download it from docker.com.

Step 1: Clone the Repository

Clone the repository containing the application code:

bash

```
git clone https://github.com/yourusername/llm-qa-app.git cd llm-qa-app
```

Step 2: Create a Virtual Environment (Optional)

It's recommended to use a virtual environment to manage dependencies:

bash

```
python -m venv venv source venv/bin/activate # On Windows, use `venv\Scripts\activate`
```

Step 3: Install Required Packages

Install the required packages using pip:

bash

```
pip install -r requirements.txt
```

Step 4: Run the Application

You can run the application directly using Streamlit:

bash

```
streamlit run frontend.py
```

Alternatively, if you prefer to use Docker, build and run the Docker container:

1. **Build the Docker image**:

   bash

   ```
   docker build -t streamlit-app .
   ```

2. **Run the Docker container**:

   bash

```
docker run -p 8501:8501 streamlit-app
```

Step 5: Access the Application

Open your web browser and navigate to `http://localhost:8501` to access the application.

Application Pipeline

1. Document Uploading

- Users can upload documents using the file uploader in the sidebar.
- Supported formats: PDF, TXT, DOCX.

2. Document Processing

Once a document is uploaded:

- The application determines the file type and uses the appropriate loader (`PyPDFLoader`, `TextLoader`, or `Docx2txtLoader`) to extract text from the document.

3. Chunking and Embedding

- The text is split into manageable chunks using the `RecursiveCharacterTextSplitter`.
- Each chunk is converted into embeddings using `OllamaEmbeddings`.

4. Question-Answering

- Users can enter questions about the document content.
- The application retrieves relevant chunks based on the input question and generates an answer using the `RetrievalQA` chain with the `ChatOllama` model.

5. Chat History

- The application maintains a history of questions and answers, displayed in the chat history section.

Example Usage

1. **Upload a document** (PDF, TXT, or DOCX).
2. **Enter a question** related to the document content.
3. **View the generated answer** and the chat history.

Troubleshooting

- **Common Errors**: If you encounter issues, check that all dependencies are installed correctly and that you're using compatible versions of Python and libraries.
- **Logs and Debugging**: Check the terminal output for any error messages. This can help identify issues with document loading or embedding.

Conclusion

This LLM Question-Answering Application provides a powerful way to interact with documents using natural language. By following the deployment instructions, users can easily set up the application on their local machines and begin asking questions about their documents.