Project Submission: LangChain & NLP Task

Title: Question Answering System Using LangChain and GPT-2 with Memory

Submitted By: Davan C Reddy

---

1. Project Overview

This project implements a **question-answering system** integrated with **LangChain** and an **open-source GPT-2 model**. The system processes user queries and provides responses based on a provided context using document retrieval. The system also supports **conversation memory** using LangChain's memory buffer, enabling it to retain and utilize previous interactions across multiple questions.

The system is designed to:

1. Retrieve the most relevant context for a question.
2. Use GPT-2 to generate an appropriate answer.
3. Store conversation history to provide context-aware responses for subsequent questions.

2. Environment Setup

To run the project, follow the steps below to set up the environment and install the necessary dependencies.

Dependencies:

- **Python 3.x**
- **Hugging Face Transformers**: For GPT-2 model and tokenizer.
- **LangChain**: For document retrieval and memory buffer.
- **Sentence Transformers**: For embeddings used in document retrieval.
- **FAISS**: For fast document search using vector embeddings.
- **OpenAI**: For future integrations if needed.

Installation Instructions:

1. Install the necessary Python packages by running the following command:

!pip install transformers langchain datasets sentence-transformers faiss-cpu openai

2. Ensure that you have a GPU available if you wish to speed up text generation by the GPT-2 model (optional).

3. Code Structure

Main Components:

1. **GPT-2 Model Integration**:
    - The GPT-2 model from Hugging Face's transformers library is used to generate answers based on the question and retrieved context.
2. **Document Retrieval**:
    - FAISS, along with LangChain's retriever class, is used to fetch relevant documents or sections that relate to the input question.
3. **Memory Integration**:
    - LangChain's ConversationBufferMemory is employed to store and manage the conversation history across multiple user queries.

Key Files:

- **main.py**: This script contains the implementation of the question-answering system, including document retrieval, GPT-2 text generation, and memory management.
- **requirements.txt**: Contains the list of dependencies required for the project.
- **README.md**: Project documentation .

4. Usage Instructions

Follow these steps to run the QA system:

1. **Load the GPT-2 Model**:
    - The GPT-2 model and tokenizer are loaded using Hugging Face's transformers. The model generates text in response to the user's question based on the retrieved context.
2. **Document Retrieval**:

- Add the documents or paragraphs you wish to search from. These documents will be embedded and stored using FAISS, allowing for efficient retrieval.

3. **Running the QA System**:
    - You can interact with the system by asking questions, and it will provide answers based on the context it retrieves.

Example Usage:

```
# Initialize QA system
qa_system_with_memory = CustomQAWithMemory(retriever, memory)

# Ask a question
response1 = qa_system_with_memory.run("What is LangChain?")
print("Response 1:", response1)

# Ask a follow-up question
response2 = qa_system_with_memory.run("Who developed GPT-2?")
print("Response 2:", response2)
```

This will output context-based answers to the questions you provide. The system can handle follow-up questions by utilizing the memory feature to maintain the conversation context.

5. Memory and Context Management

The memory buffer stores each question and answer, allowing the system to maintain a conversation across multiple queries. This feature enhances the relevance of the answers provided by considering previous interactions.

6. Repository Link

The complete code for this project has been uploaded to GitHub. You can access it via the following link:

https://github.com/Davan57/Langchain-QA-System

7. Future Work

- **Tool Integration**: Adding external tools to enhance the context the model uses for answering questions.
- **Fine-Tuning**: Training the GPT-2 model further on domain-specific data to improve the quality of the answers in specialized contexts.

8. Conclusion

This project successfully demonstrates the integration of LangChain with an open-source LLM like GPT-2. The question-answering system efficiently handles document retrieval, and with the inclusion of memory, it provides coherent and context-aware responses across multiple interactions.