Assignment Report: Development of a Custom Chatbot with LangChain
By: Anbhi thakur

---

1. Project Overview

The objective of this project is to develop an interactive chatbot using LangChain. This chatbot will extract data from Brainlox technical courses, convert the data into vector embeddings, store these embeddings in a vector database, and provide users with relevant responses via a Flask-based RESTful API.

---

2. Project Requirements

2.1 Data Collection

- Utilize LangChain's URL loaders to extract technical course information from:
  - Brainlox Courses
- Perform data cleaning and structure the extracted content for use in the chatbot.

2.2 Embedding Generation & Storage

- Use embedding models (e.g., OpenAI, Hugging Face, Sentence Transformers) to convert the extracted course content into embeddings.
- Store the generated embeddings in a vector database like FAISS, ChromaDB, Pinecone, or Weaviate.

2.3 API Development

- Create a RESTful API using Flask to handle communication between the user and the chatbot.
- The API must:
  - Accept and process user queries.
  - Retrieve the most relevant data from the vector database.
  - Return the results in a structured JSON format.

---

3. Implementation Process

3.1 Environment Setup

- Install the required dependencies for the project:

```
bash
CopyEdit
pip install langchain flask faiss-cpu openai chromadb requests
beautifulsoup4
```

3.2 Data Extraction with LangChain

- Utilize `langchain.document_loaders.WebBaseLoader` to collect data from the Brainlox website.
- Clean and organize the extracted information to prepare it for embedding generation.

3.3 Embedding Generation and Vector Storage

- Select an embedding model (e.g., OpenAI, Hugging Face, Sentence Transformers) to generate embeddings for the course data.
- Store these embeddings in a vector store like FAISS or ChromaDB for efficient retrieval.

3.4 Flask RESTful API Development

- Design an API with the following endpoint:
  - `/query`: Accepts user input, queries the vector database, and returns the relevant information in JSON format.

3.5 Testing and Deployment

- Test the chatbot with different sample queries to ensure the accuracy and responsiveness of the model.
- Deploy the solution using Flask or FastAPI, with Gunicorn for scalability if necessary.

---

4. Submission Requirements

- Upload the completed code to a public GitHub repository and share the link.
- Submissions via GitHub will be the only accepted format.

---

5. Helpful Resources

- LangChain Documentation for VectorStores
- LangChain Quickstart Guide

---

6. Conclusion

This project successfully integrates LangChain for document retrieval and vector storage, enabling a fast and reliable user experience through the Flask RESTful API. The chatbot is capable of efficiently retrieving relevant data from the Brainlox technical courses, providing users with valuable information in real-time.