CHATBOT DEVELOPMENT SYSTEM

OBJECTIVE

objective is to build a chat system using the Python-Flask framework. This system will employ the embeddings built from specified data sources and stored in Vector DB, as well as OpenAI's language model and prompt engineering through LangChain, to understand user inputs and generate relevant responses.

INTRODUCTION OF LANGCHAIN

LangChain is a blockchain-based platform that leverages AI and Machine Learning to translate and verify documents, aiming to remove language barriers. Its decentralized nature ensures transparency and reliability. OpenAI, on the other hand, is a leading AI research lab known for developing cutting-edge AI models like GPT-4. OpenAI focuses on creating safe, beneficial AI and ensuring its benefits are widely distributed. Its innovations have significant implications in language understanding and generation tasks. Both organizations strive to harness AI's power, with LangChain focusing on translation and OpenAI on a broader range of applications.

TECHNOLOGY USED

- LangChain
- OpenAI
- Weaviate
- GIT
- Flask
- Tkinter

1. LangChain

LangChain is a versatile framework designed for building applications that leverage language models. It aims to integrate language models with data sources and facilitate interaction with their environment. LangChain provides modular components necessary for language model operation, making them accessible regardless of your use of the overall framework. It also offers "Use-Case Specific Chains" that assemble these components for specific applications, allowing for easy startup and customization. It effectively bridges language models and application development, promoting powerful and differentiated applications with high customization potential. For language-specific application details, LangChain offers extensive documentation.

2. OpenAI

OpenAI is an artificial intelligence research lab comprised of both for-profit and non-profit entities. Founded in December 2015, it has a mission to ensure that artificial general intelligence (AGI) benefits all of humanity. OpenAI aims to build safe and beneficial AGI directly, but is also committed to aiding others in achieving this outcome. While striving to lead in AI capabilities, OpenAI adheres to a cooperative orientation, actively cooperating with other research and policy institutions. It's known for developing sophisticated AI models, like GPT-3 and GPT-4, which have significantly advanced the field of natural language processing.

3. Weaviate

Weaviat is an open-source, real-time, scalable knowledge graph that allows users to create, manipulate, and query information in a knowledge graph structure. It uses machine learning models for semantic search and analysis, enabling users to draw complex inferences and connections between data points. Weaviat's automatic classification functionality allows for linking data sets and enhancing the richness of existing data. It is highly scalable and comes with RESTful APIs for easy integration. It also supports vector search, making it suitable for advanced data exploration and semantic search use cases. It's an innovative tool for managing and exploring complex data structures.

4. **GIT**

   It is a widely-used distributed version control system that allows developers to track changes in their code over time. It was developed by Linus Torvalds, the creator of Linux, in 2005. Git facilitates collaboration, allowing multiple developers to work on a project simultaneously without overwriting each other's changes. It keeps a complete history of every change made to a project, enabling developers to revert to any previous version. Git supports branching and merging, allowing developers to create separate branches for feature development or bug fixes and later merge these changes back into the main code. It is an essential tool in modern software development.

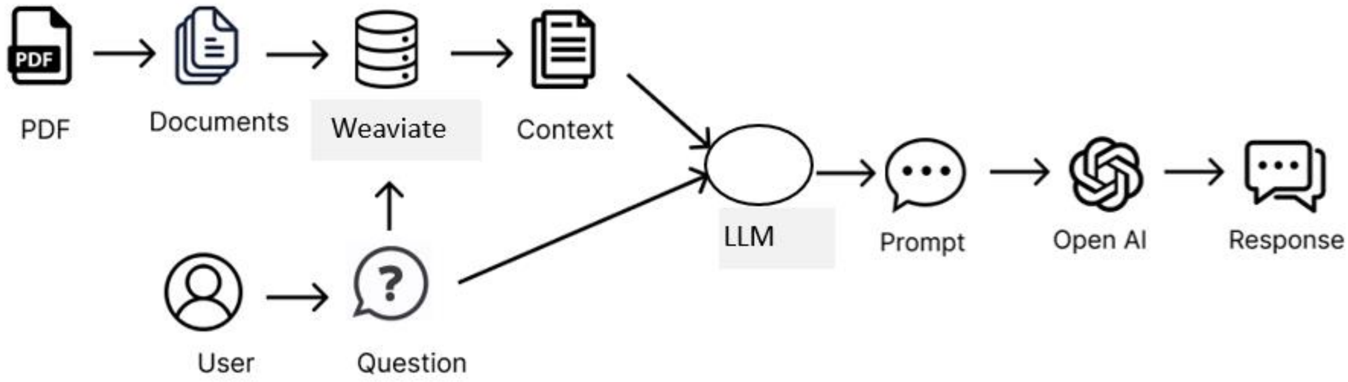
5. **Flask**

   It is a lightweight and highly flexible web framework for Python. It's designed to make getting started quick and easy, with the ability to scale up to complex applications. It's often referred to as a micro-framework because it does not require particular tools or libraries and has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, Flask supports extensions that can add application features as if they were implemented in Flask itself, including support for form validation, upload handling, various open authentication technologies, and more. It's suitable for both small and large projects.

6. **Tkinter**

   Tkinter is the standard Python interface to the Tk GUI (Graphical User Interface) toolkit. It is one of the most commonly used and easy-to-use libraries for developing desktop applications in Python. With Tkinter, developers can create windows, labels, buttons, menus, textboxes, and other widgets. It's a powerful tool for creating interactive applications, with event-driven programming support, and it's included with most standard Python installations. Despite its age and somewhat dated look, Tkinter remains popular for its simplicity, extensive functionality, and the fact it doesn't require any additional installation if you're using a standard Python distribution.

BLOCK DIAGRAM



STEP BY STEP EXPLANATION:

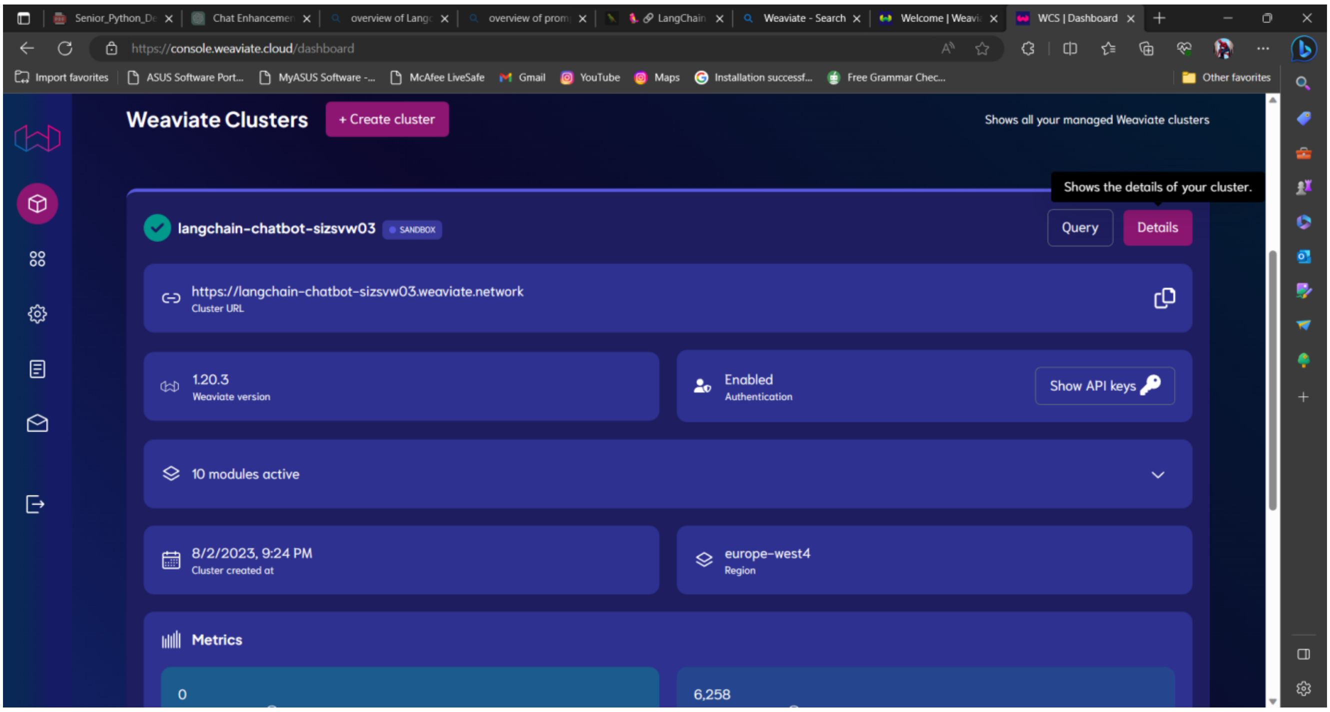
Step 1

Create an OpenAI and weaviate account, login with login credentials

Step 2

Once logged in get a security Key

- For API=> enter into API -> Personal(top right corner) -> View API key -> create new secret key give a name for that. It is a KEY to access the OpenAI
- For Weaviate=> enter weaviate account -> create cluster with a cluster name -> take URL and also API key



langchain-chatbot-sizsvw03 SANDBOX

https://langchain-chatbot-sizsvw03.weaviate.network
Cluster URL

1.20.3
Weaviate version

Enabled
Authentication

10 modules active

8/2/2023, 9:24 PM
Cluster created at

europe-west4
Region

Metrics

0

6,258

Step 3

Create a .env file in the project folder save all secret key in the .env file access the key through

```
from dotenv import load_dotenv
load_dotenv()
```

STEP 4

Create an api server and send the request to the flask post method to post a user input from GUI through form data and also to get a response

Step 5

Process the user input with the help of source pdf and OpenAI

- To process the source pdf need to load the pdf through PyPDFium2Loader and make it as document and split the text using charactertextspliter
- To process the chat memory, I used conversationalretrievalchain and conversational Buffermemory from LangChain
- Finally embed the OpenAI, pdf, and conversational memory data to get a answer

Step 6

To create a GUI chat window, here I used tkinter. In GUI the user can give a question and get a response

Step 7

Finally, the end user can access the application either through GUI or Flask

CHALLENGES

- LangChain is a latest technology so some features are not supporting for the older versions of python
- LangChain is a new technology so the understanding is much tricky because there is very less number of tutorials available. So, I used trial and error method to achieve the requirements.
- I completed all requirements while I'm working in enhancement, I tried to deploy the application in cloud with the help of render because it accepts free application deployment but it's not supporting python 3.10.11 because LangChain is a recent technology.

FUTURE ENHANCEMENT

- Voice Integration
  For a more interactive experience, consider adding voice recognition and synthesis capabilities. Users could then interact with the chat system using speech instead of text.
- User-Specific Context
  Implement the ability for the system to maintain context across different sessions with the same user. This would allow the system to "remember" past interactions and provide more contextually relevant responses.
- User authentication and personalization:
  Implement user authentication to allow personalized interactions. Users could have their own profiles, and the chat system can use this information to tailor responses to individual preferences or previous interactions.
- Feedback collection:
  Gather user feedback on the responses generated by the chat system. Use this feedback to continuously improve the model and ensure that it delivers accurate and relevant responses.

PROMPT ENGINEERING

Prompt engineering is recommended in large language models (LLMs) like GPT-4 because it significantly influences the model's output and can help produce more accurate, useful, and targeted responses. Here are several reasons why prompt engineering is crucial.

- Guide the Output
  LLMs generate responses based on the input they receive. A well-crafted prompt helps guide the model to generate a specific kind of output.
- Better Accuracy
  With careful prompt design, you can increase the likelihood of the model generating accurate and helpful responses.
- Control Over Tone and Format
  Through prompt engineering, you can guide not just the content of the model's output, but also its tone (formal, casual, etc.) and format (list, paragraph, dialogue, etc.).

- Reduction in Ambiguity
  A well-engineered prompt can reduce ambiguity by providing the model with clear instructions and context.
- Efficient Use of Resources
  Instead of spending computational resources and time on multiple attempts to get the output you want, a good prompt can get it right in the first go.

Prompt engineering is essentially the practice of refining your input to get the most useful output possible. It requires understanding the model's behaviour and the task at hand, and iterative testing and refinement of your prompts. It's an important part of leveraging the full potential of LLMs.

USER GUIDE

Steps to run the application:

- Install Python 3.10.11: This is the Python version that the chatbot application has been designed to run on. It's important to ensure this version is installed on your machine to ensure compatibility. You can download and install this version from the official Python website.
- Clone the chatbot from Github: This involves getting the chatbot's code onto your local machine. You'll need Git installed on your computer. Once that's done, you can clone the repository using the command `git clone [repo url] in your command line or terminal, where "[repo url]" is the URL of the repository you're cloning.
- pip install -r requirements.txt: `requirements.txt` is a file that lists all Python libraries that the project depends on. The command `pip install -r requirements.txt` will install these dependencies. You need to run this command in the directory where the `requirements.txt` file is located.
- To use a private key, create a .env file and to install the necessary .env package, execute the following command: pip install python-dotenv.
- py run_server.py: This command is used to start the server for the chatbot application. `run_server.py` is a Python script which starts the server.
- Open another terminal to run the GUI: The Graphical User Interface (GUI) is what users interact with. You need to open a new terminal or command prompt window to run this while the server is still running in the other terminal.
- cd ui: This command changes your current directory in the terminal to the "ui" directory. Make sure that the "ui" directory exists in the current path before running this command.
- py ui.py: This command runs the Python script `ui.py`, which starts the GUI. Once this script is running, you should be able to interact with the chatbot through the GUI.

Note:

- repo url => https://github.com/thirunavukarasu152000/ChatBot?search=1
- The .env file contains three keys
  - OPENAI_API_KEY=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
  - WEAVIATE_API_URL=https://langchain-chatbot-sizsvw03.weaviate.network
  - WEAVIATE_API_KEY=4fDExmhStbYHzCZJdrsiBPZB92t1cyB8QMJB

Remember to ensure that all the above commands are run in the correct directories, especially the commands related to Python files and the `requirements.txt` file.

CONCLUSION

The task involves constructing an advanced chat system using Python-Flask, integrating Vector DB to handle embeddings from specific data sources. OpenAI's language model and LangChain for prompt engineering form a core component, allowing the system to interpret user inputs and produce contextually appropriate responses. This blend of technology showcases the convergence of machine learning, artificial intelligence, and web development to create powerful, intelligent, and responsive dialogue systems. It represents a significant step forward in AI-driven communication tools, providing a blueprint for future developments in conversational AI applications.