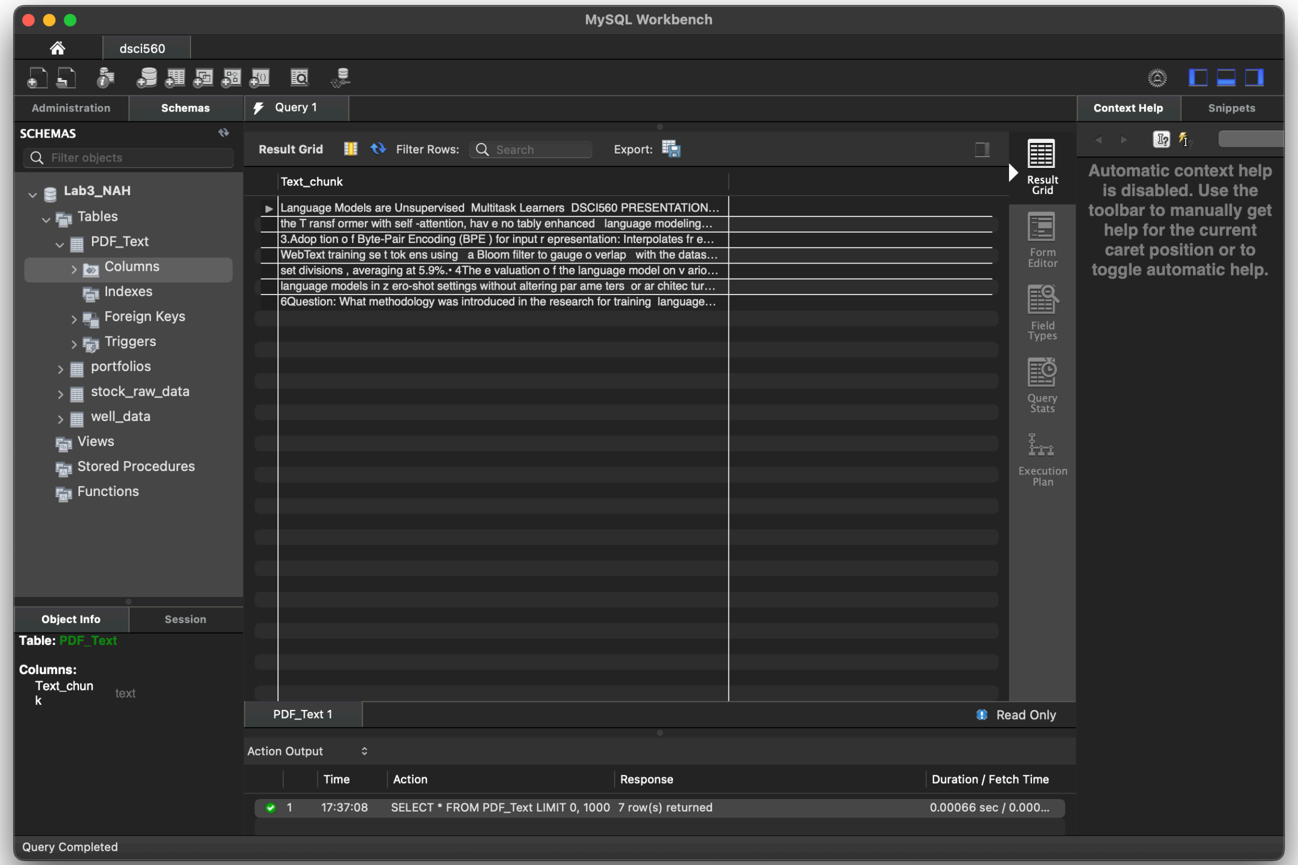
PDFs Chatbot using Langchain, GPT 3.5

This is a Python gui application that demonstrates how to build a custom PDF chatbot using LangChain and GPT 3.5.

Here is the difference we make

- Database Integration: MySQL is incorporated to store text chunks extracted from PDFs.



Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

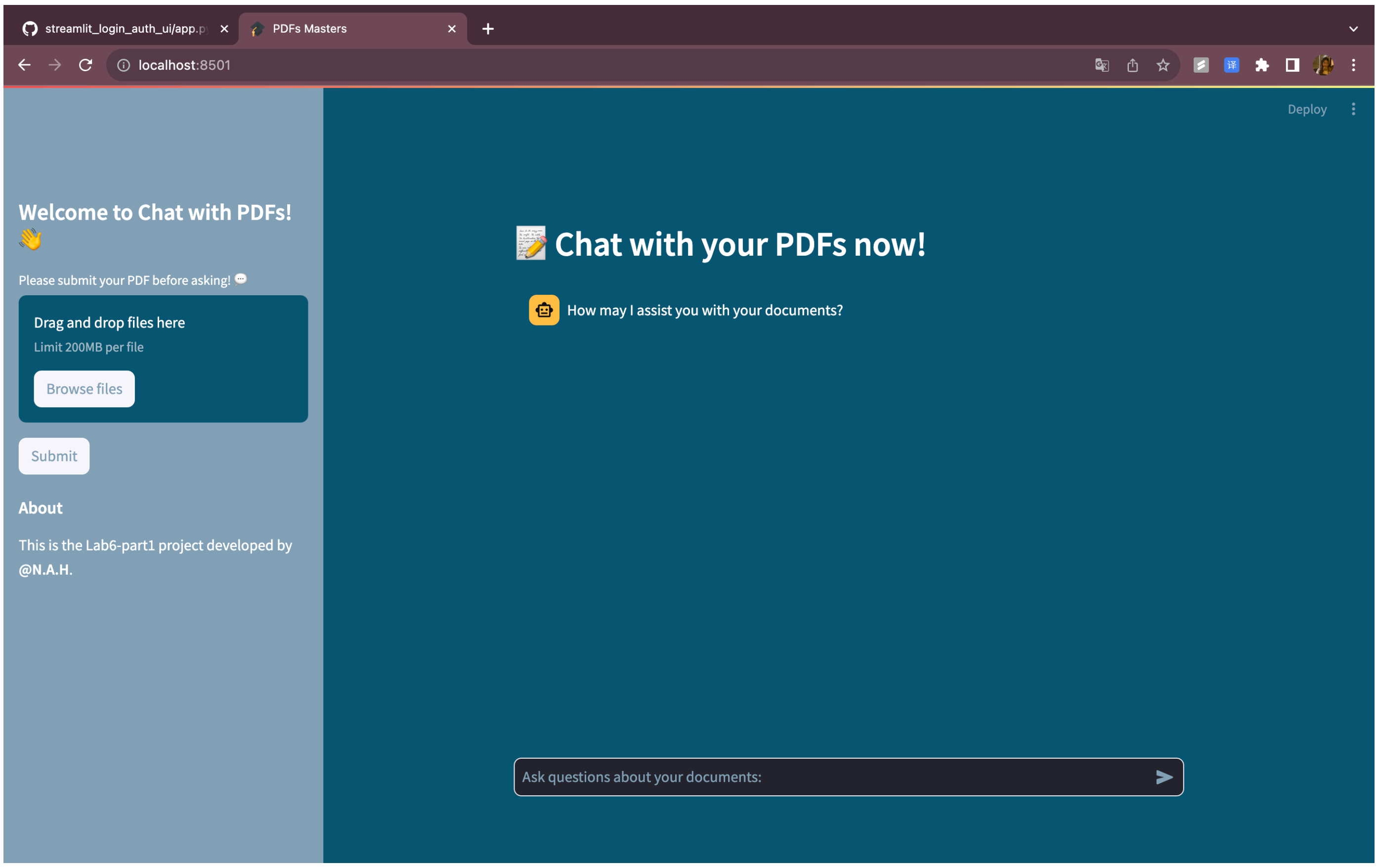
Table: PDF_Text

Columns:
Text_chunk text

Text_chunk
Language Models are Unsupervised Multitask Learners DSCI560 PRESENTATION…
the T ransf ormer with self -attention, hav e no tably enhanced   language modeling…
3.Adop tion o f Byte-Pair Encoding (BPE) for input r epresentation: Interpolates fr e…
WebText training se t tok ens using   a Bloom filter to gauge o verlap   with the datas…
set divisions , averaging at 5.9%.• 4The e valuation o f the language model on v ario…
language models in z ero-shot settings without altering par ame ters or ar chitec tur…
6Question: What methodology was introduced in the research for training language…

		Time	Action	Response	Duration / Fetch Time
✓	1	17:37:08	SELECT * FROM PDF_Text LIMIT 0, 1000	7 row(s) returned	0.00066 sec / 0.000...

- User Authentication: The code signifies an intention to implement user authentication using `streamlit_login_auth_ui.widgets`. (Some errors were encountered while using this function)
- User Interface: The application boasts an enhanced UI design using advanced CSS stylings, distinct from the reference code which employs basic styles.
- Chat Interface: My code uses native Streamlit chat UI functionalities while the reference code leans on custom HTML templates.



Welcome to Chat with PDFs! 👋

Please submit your PDF before asking! 💬

Drag and drop files here
Limit 200MB per file

Browse files

Submit

About

This is the Lab6-part1 project developed by @N.A.H.

📝 Chat with your PDFs now!

How may I assist you with your documents?

Ask questions about your documents:

> The following content comes from the ReadMe file provided by the teacher。

How it works

1. The application gui is built using streamlit
2. The application reads text from PDF files, splits it into chunks
3. Uses OpenAI Embedding API to generate embedding vectors used to find the most relevant content to a user's question
4. Build a conversational retrieval chain using Langchain
5. Use OpenAI GPT API to generate respond based on content in PDF

Requirements

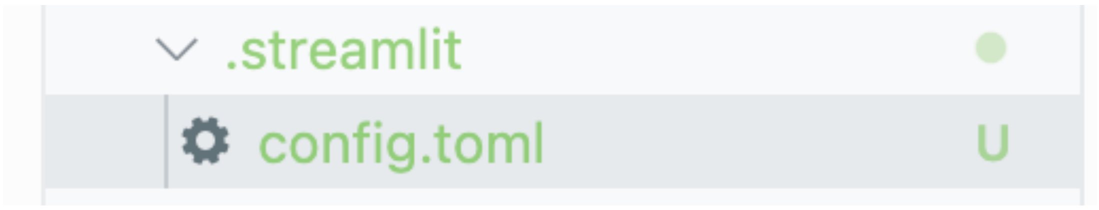
1. Install the following Python packages:

```
pip install streamlit pypdf2 langchain python-dotenv faiss-cpu openai sentence_transformers
```

2. Create a `.env` file in the root directory of the project and add the following environment variables:

```
OPENAI_API_KEY= # Your OpenAI API key
```

3. Create a folder named `.streamlit` in your root:



4. In `.streamlit`, create `config.toml`:

```
[theme]
primaryColor="#29b5e8"
backgroundColor="#115675"
textColor="#FFFFFF"
```

Code Structure

The code is structured as follows:

- `app.py`: The main application file that defines the Streamlit gui app and the user interface.
  - get_pdf_text function: reads text from PDF files
  - get_text_chunks function: splits text into chunks
  - store_db function: store the text chunks to MySQL database.
  - get_vectorstore function: creates a FAISS vectorstore from text chunks and their embeddings
  - get_conversation_chain function: creates a retrieval chain from vectorstore
  - handle_userinput function: generates response from OpenAI GPT API
- `htmlTemplates.py`: A module that defines HTML templates for the user interface.

How to run

```
streamlit run app_p1.py
```