DSCI-560: Data Science Practicum Laboratory Assignment 9

Custom Q&A Chatbot

Instructor: Young H. Cho, Ph.D.

You will gain experience building a Question-and-answer (Q&A) chatbot using Large Language Models (LLMs) and embedding models. You will start by preparing the data and creating embeddings for text.

You will work with your team on pdf text extraction in this lab. Additionally, you'll preprocess the data to create word embeddings and text chunks that will be used to train your LLMs and spend time getting familiar with different LLM models.

1. Description

a) Resources

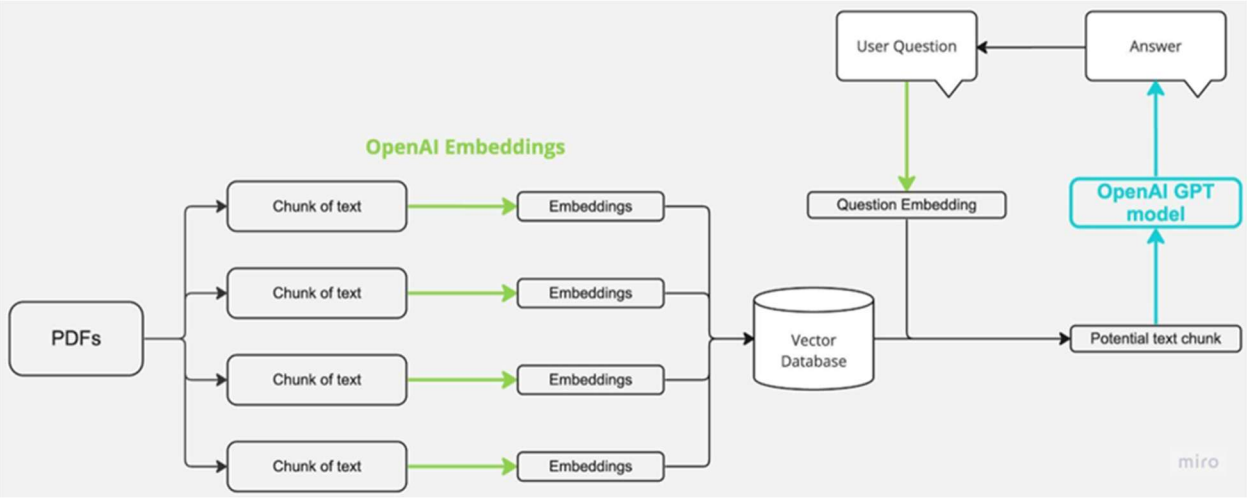
This drive folder contains suggested files for the Lab.
https://drive.google.com/drive/folders/1hdUoDvtQoFkIbJyUr8kwDQghLNTptoCQ

b) Chatbot Model Structure

At the end of both Parts of the Lab, the System should have the above structure. The model should read a pdf of the textbook and installation instructions in the Drive link, convert them into Embeddings using OpenAI's embeddings library, and store them in a vector database. The LLM model should be trained on this database, and then a user input question should be taken. It should identify potential text chunks from the database corresponding to the answer, format the desired answer, and return it to the User.

Here is a more detailed explanation of how each system component works.

Prepare search data (once per document)



- Collect: Convert PDF into Text document suitable for embeddings and GPT
- Chunk: The text document will need to be split into short, mostly self-contained sections to be embedded
- Embed: Each section is embedded with the OpenAI API

- Store: Embeddings are saved (for large datasets, use a vector database)

Search (once per query)

- Given a user question, generate an embedding for the query from the OpenAI API
- Using the embeddings, rank the text sections by relevance to the query.

Ask (once per query)

- Insert the question and the most relevant sections into a message to GPT.
- Return GPT's answer.

c) Tools / Libraries

Use all the tools you used in the previous labs, including the conversion tools and database, for this assignment.

For the assignment, you will need to use Python scripts and other useful tools in the Linux environment. ***(Make sure to document any setup steps/requirements for running your scripts in the document you submit)***

The **Drive folder** has a **Readme.md** file with all the setup instructions and a list of libraries and their versions and installation steps required for Parts 1 and 2 of Lab 6. Use the skeleton file ***App_p1.py*** for this lab. You should add the actual code into the file to make things work.

2. Data Collection / Processing / Storage

For this set of assignments, we will process data related to Advanced Digital Systems and some installation guides related to the circuit design tool used in the textbook.

a) PDF Text Extraction

Download a copy of the folder to your local machine. Your task is to write a Python script to iterate over the PDF in the folder, extract the information from the PDF, and store it in your database tables.

b) Get Text Chunks

Process the extracted data into chunks of size 500. Use the ***CharacterTextSplitter*** provided by langChain to split the text. Read about the different parameters the function takes and set appropriate values for those parameters to split the extracted data into chunks efficiently.

c) Create Vector Datastore

Write a function that would take the chunks generated in the above step as input and use the OpenAI Embeddings to create word embeddings for those chunks and store them in a vector database.

d) Create a Conversation Chain

Write a function that would take an LLM model, and the vector store created above as input and create a conversation chain to understand user questions and generate responses based on its knowledge from the vector database.

(Take a look at the ConversationMemoryBuffer function provided by langchain for writing this function)

e) **Driver function**
Write a driver function that calls the above functions and then gets questions from the user, runs it through the conversation chain to get a response, prints the response on the screen, and then asks the user for the next question. The user must type in 'exit' to quit the program. Use the OpenAI to create conversation chains.

3. Using Open-Source Resources

You may have noticed that OpenAI charges fees for using their tools and models. This development path may not be the best for developing and deploying domain-specific chatbots, especially because functionally equivalent solutions with acceptable performances can be built with open-source tools and models. Moreover, developers may not want ChatGPT to analyze their original proprietary data that contains domain-specific answers. In some cases, the solution may not have access to the Internet.

Therefore, your task in this section is to replace the OpenAI resource with free tools and models. You may use any open-source local embedding models and LLMs if they can produce output similar to that of OpenAI embeddings. Many open-source text embedding models are available at https://huggingface.co/models

Additional guidance and information may be found at https://sbert.net/, and the following video gives a good summary of open-source text embedding models and how to use them https://www.youtube.com/watch?v=QdDoFfkVkcw

4. Web Interface Design

You will focus on designing a user-friendly chatbot interface using HTML and CSS. Implement JavaScript to handle user interactions and communicate with the chatbot.

a. **PDF Upload**
Provide a sidebar option where the user can upload single or multiple PDF documents for analysis.

b. **Analyzing PDFs**
Store the PDFs input by the user and execute your Python script to analyze the input PDFs, extract text, create chunks, generate word embeddings, and store it in the vector database.

c. **Input Field and Chat Window**
Create a chat window with an input field for the user to ask questions based on the pdf and display all the messages as conversations. Store the questions and messages in chat history for later use.
Refer to the ***html.py*** for some starter code on displaying user and bot messages.

5. Team Discussions

Your team is expected to meet in person / virtually each day of the week and discuss the assignment progress & next steps. Document and compile minutes of all meetings in a separate file called ***'meeting_notes_L9_<team_name>.pdf'***

Submission

Make one submission per team. Each team must submit all the code files for the working solution, a readme document containing information for running the code in PDF format, and a document that outlines the minutes of all team meetings in PDF format. You must include a detailed GitHub history describing what each team member submitted.

Provide a video per team that demonstrates the entire working solution. The video should explain how the data tables were loaded, demonstrate query results, and discuss the design decisions and reasoning. Also, include details about how your team preprocessed the data. Please include the team's name and the names of all team members in the video.

There will be a 50% penalty for all late submissions.