LEVERAGING AI IN HEALTHCARE:

MEDICAL DOCUMENT Q&A USING A LANGUAGE MODEL AND VECTOR STORE

BY

CHRIS MANNINA

FINAL REPORT

CS467 - Capstone

Oregon State University

August 16, 2023

BACKGROUND

A large language model, or LLM, is an advanced AI model capable of understanding and generating human-like text. It's trained on a vast array of internet text, and hence it can answer questions, write essays, summarize texts, translate languages, and even perform tasks like writing Python code. As we've witnessed with ChatGPT, these models can generate diverse responses and are flexible in handling a wide array of tasks without needing task-specific training data. Instead of training a LLM, what if you could have it read and interpret a document to then use as a source of truth when asking subsequent questions.

Medical documents and guidelines are known for their vastness and complexity; they are perfect examples of documents that demand significant time and expertise for comprehension. Therefore, the thought arose that we can leverage a LLM to interpret and help answer questions by referring to the medical documents. This project explores how we can accurately provide medical literature to a LLM as a reference, and then proceed to ask questions regarding the literature.

INTRODUCTION

ChatGPT is arguably great at question answering and has an impressive knowledge base; it passed the bar, SAT, GRE, and even the US medical licensing exam. The issue being it was trained on data up to September 2021. This means GPT cannot accurately answer questions about recent events, personal documents, or updated medical guidelines.

GPT learns knowledge through 2 different ways: model weights (fine-tuning) or model inputs (knowledge inserted into an input message). Fine-tuning is actually less reliable for factual recall. One limitation of text search compared to fine-tuning is the maximum amount of text the model can read at once. Therefore, to create a system capable of answering questions using extensive text sources, we implement the Search-Ask method:

1) **Search**: Look through your library of text for relevant sections.
2) **Ask**: Insert the retrieved text sections into a message to GPT and ask your question.

In the following illustration, you can see how the application works:

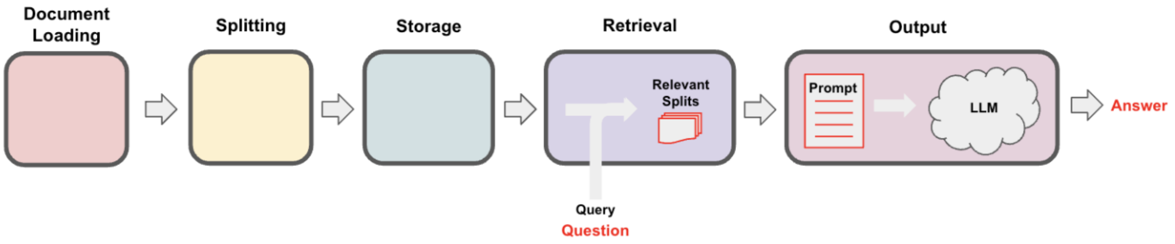


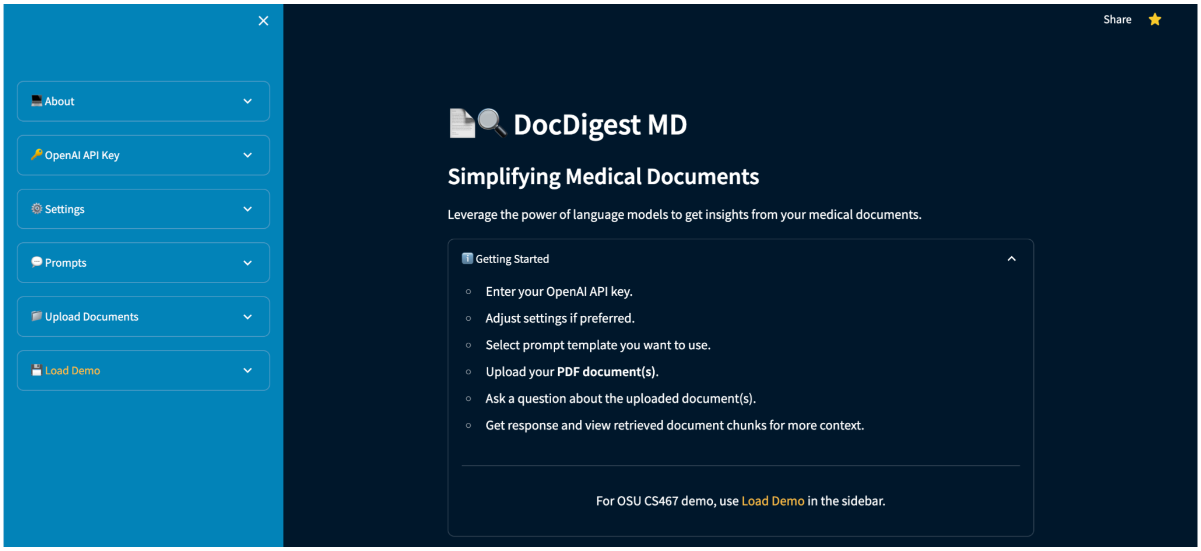
Image: https://python.langchain.com/docs/use_cases/question_answering/

A PDF document is first loaded, and then split and divided into chunks. The document chunks are then vectorized to create an embedding, and then stored in a vector database. At this point the user is able to ask a question about the document. Before sending the question to the LLM, that question is vectorized and a similarity search is performed to find the relevant context from the vector database. Finally, a formatted input message that contains the question and relevant context that was retrieved is sent to ChatGPT.

SOFTWARE DESCRIPTION

The application is designed to allow users to ask questions related to uploaded documents, with the primary focus on medical documents and guidelines. Here's a brief breakdown of the features, along with some visual examples of the application.

- The application employs an interactive and intuitive Streamlit interface.



DocDigest MD

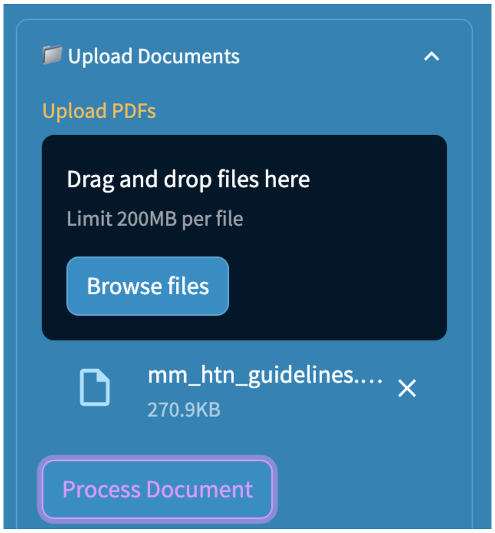
Simplifying Medical Documents

Leverage the power of language models to get insights from your medical documents.

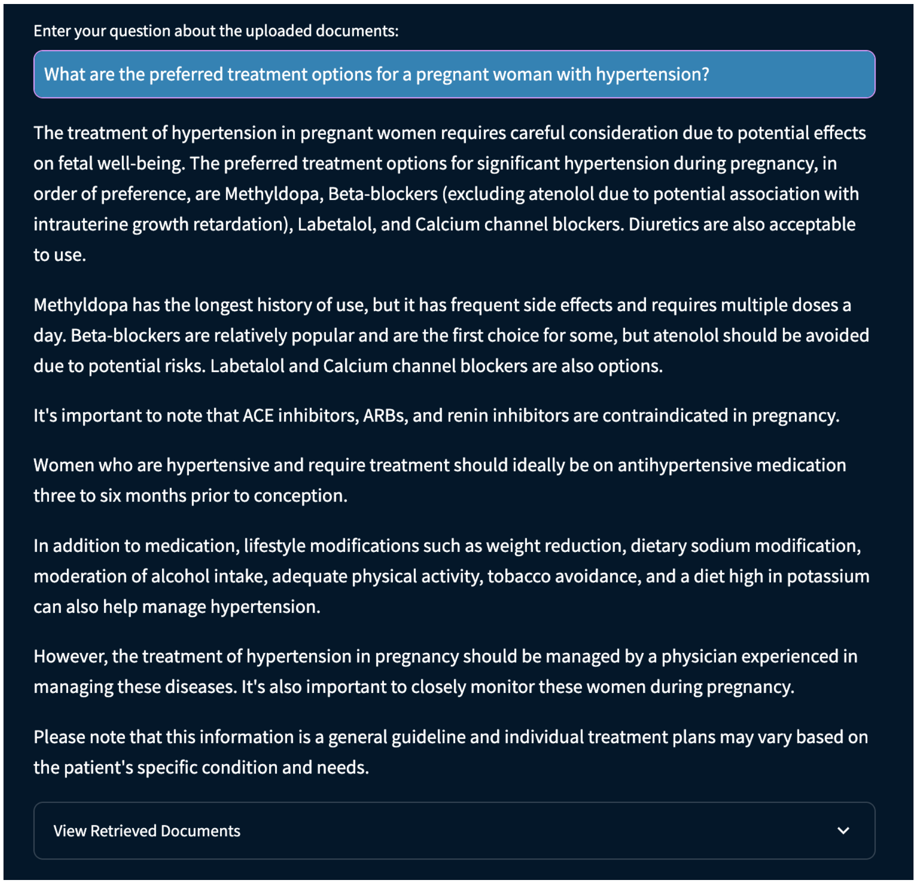
- Enter your OpenAI API key.
- Adjust settings if preferred.
- Select prompt template you want to use.
- Upload your **PDF document(s)**.
- Ask a question about the uploaded document(s).
- Get response and view retrieved document chunks for more context.

For OSU CS467 demo, use Load Demo in the sidebar.

- Users can upload and process PDF documents, which serve as the knowledge base.



- Users then pose questions related to the uploaded documents.



Enter your question about the uploaded documents:

What are the preferred treatment options for a pregnant woman with hypertension?

The treatment of hypertension in pregnant women requires careful consideration due to potential effects on fetal well-being. The preferred treatment options for significant hypertension during pregnancy, in order of preference, are Methyldopa, Beta-blockers (excluding atenolol due to potential association with intrauterine growth retardation), Labetalol, and Calcium channel blockers. Diuretics are also acceptable to use.

Methyldopa has the longest history of use, but it has frequent side effects and requires multiple doses a day. Beta-blockers are relatively popular and are the first choice for some, but atenolol should be avoided due to potential risks. Labetalol and Calcium channel blockers are also options.

It's important to note that ACE inhibitors, ARBs, and renin inhibitors are contraindicated in pregnancy.

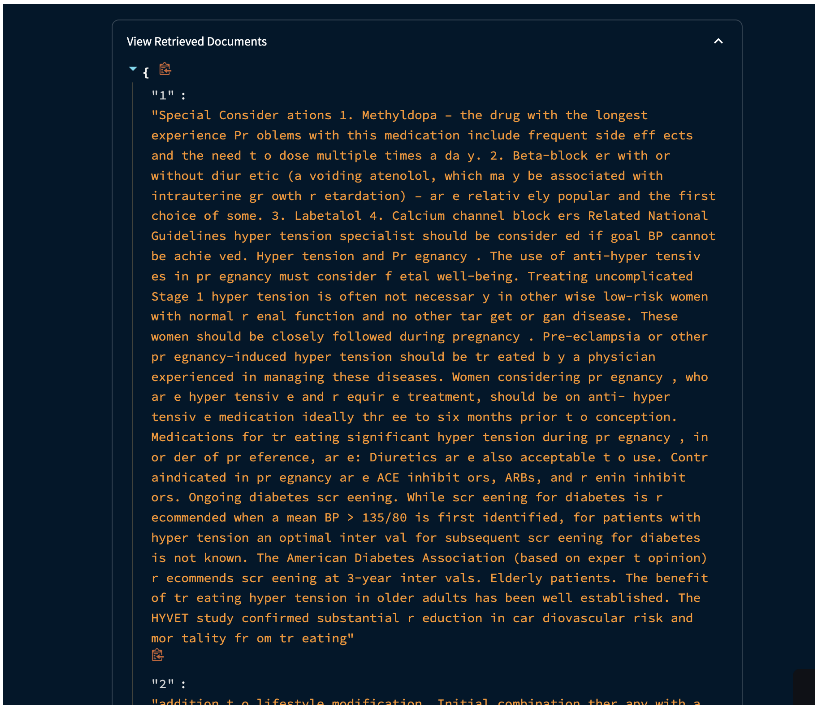
Women who are hypertensive and require treatment should ideally be on antihypertensive medication three to six months prior to conception.

In addition to medication, lifestyle modifications such as weight reduction, dietary sodium modification, moderation of alcohol intake, adequate physical activity, tobacco avoidance, and a diet high in potassium can also help manage hypertension.

However, the treatment of hypertension in pregnancy should be managed by a physician experienced in managing these diseases. It's also important to closely monitor these women during pregnancy.

Please note that this information is a general guideline and individual treatment plans may vary based on the patient's specific condition and needs.

- The system returns answers accompanied by relevant document references allowing the user to inspect source context.



```
{
  "1" :
  "Special Consider ations 1. Methyldopa – the drug with the longest
  experience Pr oblems with this medication include frequent side eff ects
  and the need t o dose multiple times a da y. 2. Beta-block er with or
  without diur etic (a voiding atenolol, which ma y be associated with
  intrauterine gr owth r etardation) – ar e relativ ely popular and the first
  choice of some. 3. Labetalol 4. Calcium channel block ers Related National
  Guidelines hyper tension specialist should be consider ed if goal BP cannot
  be achie ved. Hyper tension and Pr egnancy . The use of anti-hyper tensiv
  es in pr egnancy must consider f etal well-being. Treating uncomplicated
  Stage 1 hyper tension is often not necessar y in other wise low-risk women
  with normal r enal function and no other tar get or gan disease. These
  women should be closely followed during pregnancy . Pre-eclampsia or other
  pr egnancy-induced hyper tension should be tr eated b y a physician
  experienced in managing these diseases. Women considering pr egnancy , who
  ar e hyper tensiv e and r equir e treatment, should be on anti- hyper
  tensiv e medication ideally thr ee to six months prior t o conception.
  Medications for tr eating significant hyper tension during pr egnancy , in
  or der of pr eference, ar e: Diuretics ar e also acceptable t o use. Contr
  aindicated in pr egnancy ar e ACE inhibit ors, ARBs, and r enin inhibit
  ors. Ongoing diabetes scr eening. While scr eening for diabetes is r
  ecommended when a mean BP > 135/80 is first identified, for patients with
  hyper tension an optimal inter val for subsequent scr eening for diabetes
  is not known. The American Diabetes Association (based on exper t opinion)
  r ecommends scr eening at 3-year inter vals. Elderly patients. The benefit
  of tr eating hyper tension in older adults has been well established. The
  HYVET study confirmed substantial r eduction in car diovascular risk and
  mor tality fr om tr eating"

  "2" :
  "addition t o lifestyle modification. Initial combination ther apy with a
```

- Settings are available for the user to switch between ChatGPT 3.5 or 4, model temperature, document chunk size and overlap.
- Users choose which prompt they would like to use. Various prompts to choose from. There are 3 available categories:
  - **Default** - various default prompts.
  - **Medical** - medical related prompts.
  - **Humor** - fun and entertaining prompts (e.g. Eminem, Master Yoda, Captain Kirk, Sherlock Holmes, and others.
- Once a document has been loaded, a “Reset App” button will appear to allow users to delete the previously uploaded documents, upload new documents, and create a new chat.
- Demo mode for OSU CS467. API key is provided and two documents have already been prepared, the CS467 syllabus and schedule. *Professor and TA can use this to grade and quickly test.*

DEVELOPMENT EFFORTS VS PROJECT PLAN

In the course of developing the document Q&A application, I encountered several challenges and deviations from the initial project plan, primarily driven by the intricacies of medical guidelines and the time constraints. Here are the key deviations and their rationale:

Vector Databases: While the initial plan incorporated the use of Chroma and Pinecone, I pivoted to FAISS. This decision was influenced by its user-friendly management, cost-effectiveness, and efficient similarity search capabilities. Additionally, Pinecone's paid nature made it less appealing for our use case.

LangChain Integration: My experience with LangChain was marred by a rapidly changing landscape and challenges in documentation. A specific instance involved the deprecation of a class after a week's worth of integration effort. While I think LangChain is great for quickly building LLM applications, I don’t believe it is ready as a production-grade solution. The architecture of this application was largely influenced by the plan to eventually move away from LangChain in the future.

Medical Guidelines Focus: Our primary objective was to enable the application to accurately answer medical queries based on established guidelines. Our trial with guidelines on Acute Myeloid Leukemia (AML) highlighted the challenges in this endeavor. These guidelines were extensive, spanning hundreds of pages, and the application's current iteration struggled with consistency. Recognizing this, I shifted my focus and time towards enhancing the user interface for a more holistic project presentation, deferring the refinement of the backend for future iterations. It was during this phase that I realized the application might be better suited for standard documents rather than intricate medical guidelines. In response to this insight, I diversified the prompt construction to include both regular and humorous prompts, offering users a broader spectrum of interaction options with various document types.

Configuration Management: The initial project plan did not account for a configuration file. However, recognizing the delicate nature of the application and the need for flexible settings adjustments, I introduced this feature.

User Interface Evolution: A GUI was always a stretch goal, however after pivoting from backend development trying to get more accurate answers from complex guidelines, I decided a UI would be integral for users to enjoy this application. The UI journey began with Gradio, but I transitioned to Streamlit. I found the latter offered a more customizable interface, while still being easy to work with and develop.

Document Type Limitation: To ensure quality and minimize potential issues, I restricted the application to only handle PDF documents. This decision was influenced by the 'noisy' nature of PDF data, which often required cleanup from extraneous characters and spaces. Documents that contain diagrams, charts, and images still cause issues. Future versions aim to support a broader range of document types.

Containerization: A stretch goal involved containerizing the application using Docker. However, due to time constraints and prioritizing other enhancements, this was deferred.

In hindsight, these deviations reflect an adaptive approach, ensuring the project's relevance and functionality despite unforeseen challenges.

SOFTWARE LIBRARIES & TOOLS

Language: Python was the primary language for development, ensuring versatility and compatibility with various libraries.
OpenAI: Leveraged for chat models and embeddings, providing the foundation for the Q&A functionality. ChatGPT was the language model used. The embedding model used was text-embedding-ada-002.
Streamlit: Used for the user interface, offering an interactive and user-friendly experience.
FAISS (Facebook AI Similarity Search): A library that provides an index and enables efficient similarity search. This is the vector store to which vectors can be added and searched. Given a set of vectors, we use FAISS to index them. Then using another vector (the query vector), we search for the most similar vectors within the index.
PyPDFLoader: Utilized for loading documents, whether they are local PDFs or online resources.
Development Tools: GitHub, VS Code

TEAM CONTRIBUTION

This was a solo project. I, Chris Mannina, led the project's complete design and development.

PROJECT CONTINUANCE

Current Release Challenges: Extracted text from PDFs often contains noise and lacks structure, leading to suboptimal embeddings and potential semantic loss. This dirty data can also strain ChatGPT's interpretation capabilities.
Next Phase Priorities: Future directions involve refining text extraction, optimizing system configurations, prompt engineering, and potentially transitioning from LangChain. Arguably most important would be implementing an evaluation framework. A solution would involve predefined questions and answers, testing the system's accuracy across various configurations. Furthermore, the vast amount of combinations related to chunk size, overlap, and text splitting methods necessitates a robust evaluation mechanism. Additionally, the dynamic nature of prompts requires rigorous tuning.

CONCLUSION

The application serves as a valuable tool for users seeking precise answers from their documents. By leveraging an embedding model and vector store, the system efficiently retrieves relevant information, minimizing the need for manual document perusal. The inclusion of a Streamlit interface ensures that users, even those unfamiliar with technology, can interact with ease. Looking forward, the potential for enhancing the system and expanding its capabilities is vast. As I anticipate a surge in similar applications, I believe that honing in on an evaluation framework and ensuring accuracy, especially for medical documents, will be worthwhile for my continued research and development.

REFERENCES

[1] "LangChain API Reference," [Online]. https://api.python.langchain.com/en/latest/api_reference.html
[2] "Streamlit API Reference," [Online]. https://docs.streamlit.io/library/api-reference
[3] "OpenAI Platform API Reference," [Online]. https://platform.openai.com/docs/api-reference
[4] "FAISS Wiki on Github," [Online]. https://github.com/facebookresearch/faiss/wiki