Our final product: A RAG (Retrieval-Augmented Generation) Chatbot

What is an LLM?: https://www.youtube.com/watch?v=LPZh9BOjkQs&t=83s (7 mins)

Embedding/Vectorization Reel:
https://youtube.com/shorts/FJtFZwbvkI4?si=ux7qM8zzAQcHPfGn

Vector Databases (we did not watch this in meeting, but I recommend it!):
https://www.youtube.com/watch?v=klTvEwg3oJ4

Main problem statement: How do we properly take these various forms of data and store them (with necessary metadata) in a Vector Store?

Data Types, from Simplest to Toughest:

- Text data from high quality PDFs:
    a. LLMs only work with text (aka, natural language), so, the text data from PDFs will be the most natural to deal with.
    b. Fortunately, not too hard.
        - Most hands on: Build some vectorization, embedding model from scratch. IMO: Not worth it! We are busy enough and won't do anything better than what's out there
        - Happy middle: Use library like LangChain to take in these high quality PDFs and embed them into a vector store.
            - Gives us A LOT of hands on work, optimizing the embeddings and chunking and all that good stuff. Still will be a lot of work, but will allow us to focus on what is important.
            - It is easy to do this poorly, and harder to do it well, but overall is not insanely challenging.
        - Least Hands on: OpenAI has APIs for a LOT of stuff. I have built a chatbot with them before where users can upload documents, the model embeds them, you can ask the chatbot for stuff. Only around ~100 lines of code, but it costs more $$ than langchain and is not very personable

    c. INPUT/OUTPUT FLOW FOR TEXT DATA:
        - User enters the query "What is the wing length of Bombus californicus?"
        - Our RAG tool searches available data (embedded PDF text) from our vector store, finds relevant section
        - User query PLUS relevant data chunk is passed our Chatbot
        - Great response is generated "The wing length of Bombus californicus is around 10 – 23 mm"

- Images and tables from PDFs:
    - This is trickier, as LLMs only deal with natural language, not with graphs and tables
    - Using OpenAI API:
        - This would be easy but give crappy results. For my Chatbot I mentioned above, I tried asking it questions about a graph, and it didn't really know what to do with it.
    - Possible Solution:
        - Get all images and graphs from our database and caption them. We can use ChatGPT4o to create detailed captions. Using LangChain, we should be able to embed these captions with meta data which points to the graph itself.
        - So: User enters query -> Chatbot does it's thing and finds caption in vector store most closely matching user query -> we build tool which renders graph/table from metadata

    - INPUT/OUTPUT FLOW FOR GRAPHS/TABLES:
        - User enters the query "What is the wing length of Bombus californicus?"
        - Our RAG tool searches available data (embedded graph/table captions) from our vector store, retrieves most relevant caption
        - Our Graph/Table Tool we build gets the metadata of this selected caption, including the link to the graph, and renders the graph.

- CSV files:
    - This is hard, and I need to do more research. I have seen implementations of this on Youtube but have not deep dived.
    - Possible Solution:
        - Use Pandas to turn CSVs into Dataframes
        - We caption all of the Dataframes we have (similarly to graphs/tables), giving in depth information on what is in the Dataframe
        - Again, like graphs/tables, we take in user query, return caption most similar based off of search, use metadata to return proper dataset
        - We can fine-tune another model to take in user query ("What is Bombus Californicus' wing span") into a formal query

    - INPUT/OUTPUT FLOW FOR CSV DATA:
        - User enters the query "What is the wing length of Bombus Californicus?"
        - Our RAG tool searches available data (embedded CSV captions) from our vector store, retrieves most relevant caption
        - Our CSV Tool we build translates natural language query into actual data frame query (In NAME EQUALS BOMBUS_CAL select WING_LENGTH")

Other concerns:

- Large Vector DB Size
    - Solution: Use Cases?
- Successful integration of “tools” mentioned above (for gathering various data types):
    - When should Chatbot use CSV tool instead of just referencing text? When should it make a table from the textbook appear?

Afterthought: Possible fine tune embedding model because we don’t need 3D, this is probably enhancement, afterthought to other stuff but check it out ← better semantic meaning, think flower/pollination example