Retrieval Augmented Generation (RAG)

Dani Siaj / Carlos Rodríguez

Project Overview:

This project is based on the construction of a Retrieval Augmented Generation (RAG) model, where the user can upload a pdf document, ask something and the integrated LLM will generate a response that is coherent, complete and relevant to the user's question.

The goal is to create a RAG langchain model where the user's query is used to generated a dynamic prompt with instructions, context related to the user's query and restrictions, to create a very specific response.

***Content*:**

The uploaded pdf document is a customized collection of information about food allergies, symptoms and management, retrieved from the American College of Allergy, Asthma and Immunology (ACAAI).

***Acknowledge*:**

The uploaded pdf is a small document with only 9 pages of information. No tables or images are present in the document.

Model Architecture

Model Selection:

We chose the OpenAIEmbeddings API as the text transformer of this model. We will use the langchain and the Chroma DB libraries to implement text extraction and build our vector store.

- Document Loader: PyPDFLoader
- Embeddings: OpenAIEmbeddings
- Text extraction: Langchain RecursiveTextCharacterSplitter & ChromaDB.

Chain Architecture:

- Retrieval of information:
    - Use the user’s query to retrieve a set of k number of documents (k=3, in our code) from the chroma DB vector store, using similary_search() from chromaDB.
- Prompt engineering:
    - Build a context based on those 3 documents that will be feed as a variable to the prompt generating function
    - Third, build the dynamic prompt with a specific context based on the user’s query.
- LLM model implementation:
    - Through OpenAI API, our model we feed the prompts to the LLM to generate the desired responses.

Model Evaluation:

- LLM model implementation:
    - Through prompt engineering, build a prompt for our LLM to be trained as a judge and evaluate our model’s responses based on certain criteria:
        - Relevance
        - Accuracy
        - Completeness
        - Clarity

StreamLit APP:

Deploy the model on Streamlit platform, creating a nice user interface where the user can prompt their question and the application returns a response in markdown format, followed by the evaluation from our LLM for better user experience.

Conclusions

Conclusion 1:

The responses from the RAG model show a high efficiency in time, relevance and all the required criteria.

Conclusion 2:

The PDF document limited further evaluation of the model, due to its small size. Future evaluations will be needed with larger files.