Docs and Notes

Todo

☐ Check how the code works in detail ☐ Read through doc: https://techcommunity.microsoft.com/t5/ai-azure-ai-services-blog/azure-ai-search-outperforming-vector-search-with-hybrid/ba-p/3929167 ☐ Go through Search Approach: Understand what Hybrid Retrieval means ☐ Find out how expensive it was to build the App and how the cost was produced. (e.g. for Upload / Chunking for documents with 350 mb in total it was rougly 15 euros)

A) RAG Basics

Why RAG and not just GPT (an LLM)?

LLMs are good at Language but not at Reasoning. RAG is a combination of both. It is a hybrid approach.

- **Knowledge Cutoff**: There is a always a cut-off time for the training data of LLMs. So from this day on, the knowledge available is already outdated.
- **Only public knowledge**: LLMs are trained on public knowledge. All sources which are interal to a company or behind a paywall are not included in the training data.

How can you incorporate your own knowledge?

- **Prompt Engineering**: You can give the LLM a bit of context but it only works if it has the knowledge inside it. Often it also halicinates knowledge which looks correct but if you are a domain expert you can see that it is wrong. So prompt engineering Can be helpful but normally not enough. => Example with Market Salad GPT and "Indian food"
- **Fine tuning**: You can fine tune the LLM on your own data. But this is very expensive and you need a lot of data. This is a valid option when your use case is very specialisd, you have a lot of data and you need very high accuracy, then this is probably the way to go. For most of company use cases it's not a good option economically.
- **Retrieval Augmented Generation (RAG)**: You can use a retrieval system to find the most relevant documents and then use the LLM to generate the answer. This is the approach we are using in this project.

How does RAG work?

- The user asks a question, then you take it and search for fitting documents in a knowledge base.
- Afterwards you take the orginal user question together with the chunks from the knowledge base and feed it into the LLM to generate the answer.
- Typical for a RAG based system is that the user will get the sources of the answer as well, so he can evalute the answer himself.

TODO: add example prompts / queries of the chat here as code

Explain typical RAG components

- **Retriver**: A knowledge base which is used to find the most relevant documents for a given question. This can be a search engine or a database which support vector search. (Azure Ai Search, CosmosDB, Postgres (https://github.com/pgvector/pgvector), Weaviate, Qdrant, Pineconce...)
- **LLM** A Model which can answers the questions based on the provided sources and can include citations (GPT3.5 / 4 Models, etc.)
- **Integration Layer ("Glue" in MS Slides)**: Optional Middleware which helps to connect the Retriver and the LLM. It can also be used to cache the results of the Retriver to speed up the process. It can be also done in pure Python but there are libaries which can help you with that. (Langchain, LLammaindex, Semantic Kernel, etc.)
- **Additional Features**: You can add additional features to your chatbot like chat history, Feedback buttons, Text to Speech, User Login, File Upload, etc.

What kind of skillset is needed to build a RAG based chatbot?

- **No Code**: For easy applications using Copilot Studio of Azure or OpenApi GPT Builder. This might be enough for simple use cases.
- **Low Code**: UIs which help you to build more complex cases but within a UI (e.g. Azure Studio - On Your Data). There you can add hardware compontents (Retrievers as Azuer Ai Search, differen LLMs, Features as User Authentication, Chat History persistace.)
- **Code**: For Code base there are a lot of Azure Examples or for other suppliers as well. An example is the Azure RAG Chatbot which is used in this project (https://github.com/Azure-Samples/azure-search-openai-demo).

Learnings

1. What is an optimal chunk size and a optimal overlap?