RETRIVAL AUGMENTED GENERATION(RAG)

Terminologies

Gen AI, short for Generative Artificial Intelligence, refers to a type of AI that can create new content, ideas, or data based on existing data. Generates new content such as text, images, videos, and code.

Large Language Model(LLM) is a type of AI that excels at understanding, processing and generating human language.

Prompt-Engineering refers to carefully designing and refining the inputs(prompts) to a large language model, to guide them towards generating desired outputs.

Large Language Models Examples are:GPT-4, Deepseek, Antropic's Claude and llama.

RAG

This is a technique used in Natural Language Processing that combines:

a) **Retrieval**: Finding relevant information from a large collection. LLM's have a knowledge cut of date and also don't interact with private data. Collection can be a database, PDFs or web scraping a web site.
b) **Generation**: Using LLMs to generate answers based on the query(user input) and the retrieved data from our collection.

Importance of RAG

- Increases the accuracy of large language models.
- Because of the large information chatbots like ChatGPT have they tend to be inaccurate as they hallucinate information. RAG reduces this as it provides more context to the large language model.

Frameworks to perform RAG: Langchain, llama_index, Haystack, Pydantic ai, Hugging Face, Agent SDK(open AI). For training your own model from scratch uses TensorFlow and PyTorch which are deep learning frameworks.

Tutorial

Let us do a tutorial on RAGs using llama_index framework.

RAG processes

1. *Load Data*- We extract text from our data sources example for this tutorial we will use PDFs. Llama_index has a function for this called **SimpleDirectoryReader**.
2. *Splitting Text*- We then split the text into chunks and this is done by the SimpleDirectoryReader function when we call it. This is significant in passing it into our LLM and also searching through the information for similar chunks to the users query.

3. *Embeddings*- This refers to the numerical representation of texts. As machines understand binary. We convert our chunks into embeddings using a function llama_index called **HuggingFaceEmbedding** which is open source.
4. *Vector Store*- After converting texts into embeddings we store them in a vector database such as **Chroma DB** or **FAISS**.
5. *Last Step carry two tasks:*
   Retrieve- Given user input, relevant splits are retrieved from storage using a retriever. In llama_index we will use a function called **as_query_engine**.
   Generate- We call our LLM to produce an answer using a prompt that includes the question and the retrieved data.