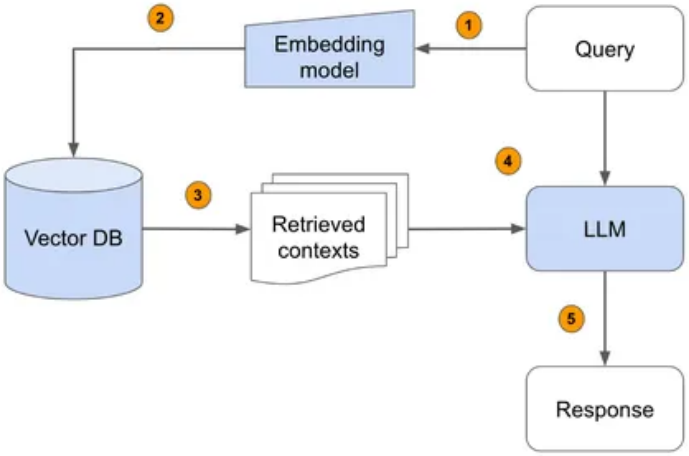
Challenge RAG con LLMs

Objetivo general

Se requiere desarrollar una solucion simple de tipo RAG (retrieved augmented generation), en donde, mediante una API se permita interactuar con un LLM con el fin de generar una respuesta (sobre un documento en particular) a la pregunta brindada por el usuario.



RAG Framework

El entregable debe contar con los siguientes componentes.

Componentes

API

Desarrollar una API en python con Flask o FastApi. La cual correra localmente y sera el medio de comunicación entre el usuario y el LLM.

El request tendra la siguiente estructura.

```
{
    "user_name": "John Doe",
    "question": "How are you today?",
}
```

LLM

Usar un LLM mediante su API para responder las preguntas del usuario.

Es libre de elegir cualquiera que acepte chat y embeddings, Cohere es una opcion gratuita, OpenAI es una opcion paga.

Puede usarse Langchain si lo desea.

EMBEDDINGS

El documento proporcionado debera dividirse en chunks (determinar cuantos y porque) y luego se tiene que hacer un encoding de esos chunks.

Dichos chunks con su correspondiente encoding deberan guardarse en alguna vectorDB, chromaDB es una opcion gratuita (https://docs.trychroma.com/)

A la hora de hacerle la pregunta del usuario al LLM, se tiene que hacer el encoding de la pregunta y buscar por similaridad el chunck mas relevante, y luego pasarlo como contexto en el prompt para que el LLM pueda responder en base a ese contexto.

HINT: de ser necesario pueden hacerse modificaciones en la estructura del documento, pero nunca de su contenido.

PROMPT

EL prompt a pasar debe incluir la pregunta del usuario, el contexto para responderla y lo que uds crean necesario para cumplir con los requisitos de la respuesta.

REQUISITOS DE LA RESPUESTA

- ante la misma pregunta siempre debe responder exactamente igual.

- que responda en solo una oracion.

- el idioma que responde debe ser el mismo que con el que se pregunta (ingles, espanol, portugues).

- que agregue emojis en la oracion que resuman el contenido de la misma.

- que responda siempre en tercera persona.

PRUEBAS

Se debera responder correctamente a las preguntas que se hagan sobre los documentos, algunos ejemplos son:

- Quien es Zara?

- What did Emma decide to do?

- What is the name of the magical flower?

ENTREGABLE FINAL

- script de Python para correr de manera local.

- requirements.txt

- repositorio en git con el progreso del proyecto

- prompt

Características técnicas

El servicio deberá tener las siguientes características:

- Se deberá usar framework de Python para crear la aplicación (FastAPI o Flask)
- Deberá implementarse sobre un ambiente virtual.
- Deberá implementarse en un repositorio GIT.
- Se requiere utilizar una base de datos vectorial (vectorDB).

Consideraciones

Algunos de los puntos que serán tenidos en cuenta son los siguientes:

- Aplicar buenas prácticas en organización del proyecto (clean architecture).
- Aplicar buenas prácticas de código y comentarios.
- El proyecto deberá estar disponible en un repositorio GIT con la documentación necesaria para poder correr localmente el mismo.
- Se valorará la Documentacion (formato Readme.md)
- Se valorará la implementación de un Dockerfile para poder levantar la api en un contenedor de docker.
- Se valorará la la construcción de una colección en PostMan (o similar) para la prueba de la API.

IMPORTANTE: Se evaluara el funcionamiento del proyecto en la instancia de devolucion del challenge utilizando las preguntas de prueba brindadas y/o alguna pregunta adicional.

Se adjunta jupyter notebook con documentacion util y documento con los temas a responder.

Cualquier duda o consulta comunicarse a la dirección donde recibiste el challenge.