Prueba Técnica: AI Engineer (LLM)

Front: https://medium.com/@kachari.bikram42/making-a-chatbot-using-openai-gpt4-langchain-and-fastapi-4dd4e09b92ff

Objetivo:

Construir una aplicación sencilla utilizando LangChain que permita a los usuarios interactuar con un modelo de lenguaje, recordar el contexto de las interacciones previas y realizar una tarea específica mediante la integración con una API externa.

Descripción de la Tarea:

Desarrollar una aplicación de chat para:

1. Interactuar con un modelo de lenguaje grande (como GPT-4).
2. Mantener un historial de conversación para recordar el contexto de las interacciones.
3. Consultar una API externa para obtener información adicional cuando sea necesario.
4. Implementar un agente que pueda decidir cuándo llamar a la API externa basada en la entrada del usuario.
5. Implementar la técnica RAG (Retrieval-Augmented Generation) para mejorar las respuestas del modelo de lenguaje mediante la recuperación de información relevante de una base de datos de documentos.
6. Usar FastAPI para levantar los endpoints necesarios para interactuar con la aplicación.
7. Crear el Dockerfile y docker-compose adecuados para levantar la aplicación en Docker.

Requisitos Específicos:

1. **Interacción con el Modelo de Lenguaje:**
    - Utilizar LangChain para integrar un modelo de lenguaje (puede ser GPT-4 o cualquier otro LLM accesible).
    - El usuario debe poder iniciar una conversación y recibir respuestas coherentes del modelo.
2. **RAG:**
    - Implementar la técnica RAG para mejorar las respuestas del modelo de lenguaje.
    - Configurar una base de datos vectorial (puede ser un conjunto de artículos, documentos PDF, etc.).
    - El agente debe ser capaz de recuperar información relevante de esta base de datos en respuesta a las consultas del usuario.

    - Integrar el sistema de recuperación de información con el modelo de lenguaje para generar respuestas mejoradas.
3. **Memoria del Contexto:**
    - Implementar un sistema de memoria que almacene el historial de la conversación para mantener el contexto a lo largo de las interacciones.
    - La memoria debe ser persistente durante la sesión del usuario.
4. **Consulta a API Externa:**
    - Integrar una API externa (puede ser una API pública como la API de OpenWeatherMap para obtener información meteorológica).
    - El agente debe decidir cuándo llamar a esta API basada en la entrada del usuario (por ejemplo, si el usuario pregunta sobre el clima).
5. **Agente de Decisión:**
    - Implementar un agente que analice la entrada del usuario y determine si es necesario realizar una consulta a la API externa.
    - El agente debe ser capaz de manejar la lógica de la decisión y devolver la respuesta apropiada al usuario.
6. **FastAPI:**
    - Utilizar FastAPI para levantar los endpoints necesarios que permitan la interacción con la aplicación. (Basta con definir el endpoint de entrada de query del usuario)
    - No se requiere autenticación
    - No es necesario implementar endpoints para la subida de documentos del RAG. El conjunto de documentos a usar puede ser procesado por defecto al levantar la aplicación
7. **Docker:**
    - Crear un Dockerfile para contenerizar la aplicación.
    - Crear un archivo docker-compose.yml para levantar la aplicación en Docker, asegurando que todos los servicios necesarios estén correctamente configurados.

Entregables:

1. Código fuente completo de la aplicación.
2. Instrucciones claras sobre cómo configurar y ejecutar la aplicación.
3. Un archivo README.md que describa el proyecto, incluyendo la arquitectura de la solución, dependencias, y cualquier otra información relevante.
4. Dockerfile y archivo docker-compose.yml para contenerizar y levantar la aplicación.
5. (Opcional) Un breve video o demo mostrando la aplicación en funcionamiento.

Criterios de Evaluación:

1. **Funcionalidad:** La aplicación debe cumplir con todos los requisitos especificados.
2. **Calidad del Código:** El código debe ser limpio, bien comentado y seguir buenas prácticas de desarrollo.
3. **Uso de LangChain:** Evaluaremos cómo se utilizan las funcionalidades de LangChain, incluyendo la integración del modelo de lenguaje, la memoria y los agentes.

4. **Documentación:** La documentación debe ser clara y suficiente para que cualquier desarrollador pueda configurar y ejecutar la aplicación fácilmente.
5. **Uso de FastAPI y Docker:** La implementación de FastAPI y Docker debe ser correcta y funcional.

Tiempo Estimado: Para la comodidad del aspirante y para facilitar la conciliación con el resto de sus actividades, se dispone de un plazo de 7 días para completar la prueba.