Documentación del Proyecto

Descripción General

Este proyecto es una aplicación web de chat llamada **"Diario del Cazador"**, que utiliza modelos de lenguaje y herramientas de búsqueda para proporcionar información sobre el juego "Hollow Knight". La aplicación está construida con Python, Streamlit y LangChain. Utiliza una base de datos de documentos y un modelo de lenguaje para responder preguntas del usuario sobre el juego y buscar videos relevantes en YouTube.

Estructura del Proyecto

Archivos

1. `**main.py**`

   - Contiene la lógica principal del chatbot, incluyendo la configuración del modelo de lenguaje, la integración con la base de datos de documentos, y la definición de herramientas y memoria para el agente de chat.

2. `**app.py**`

   - Implementa la interfaz de usuario con Streamlit. Permite a los usuarios interactuar con el chatbot y ver las respuestas en una interfaz web.

3. `**embeddings.ipynb**`

   - Un cuaderno Jupyter para cargar un PDF y crear embeddings, que luego se guardan en una base de datos Chroma para su uso en el chatbot.

4. `**imagen.png**`

   - Imagen utilizada en la interfaz de usuario de la aplicación web.

5. `**docs/**`

- Carpeta que contiene:

- `**Chroma/**`: Base de datos persistente para embeddings.

- `**Absolute_Radiance.pdf**`: Documento PDF cargado y procesado para crear embeddings.

6. `Documentacion/`

-Carpeta que contiene:

- `**Documentacion.pdf**`: Este archivo.

- `**Presentacion.pptx**`: Una presentación (.ppt) resumida del asistente y las decisiones que se tomaron en su desarrollo

Dependencias

Asegúrate de tener instaladas las siguientes librerías:

- `dotenv`
- `langchain`
- `langchain_chroma`
- `langchain_openai`
- `langchain_community`
- `langchain_core`
- `streamlit`
- `PIL` (Pillow)
- `time`

Puedes instalar estas dependencias utilizando pip:

pip install python-dotenv langchain langchain_chroma langchain_openai langchain_community langchain_core streamlit pillow

Instrucciones para Ejecutar

1. **Configuración del Entorno**

   Asegúrate de tener un archivo `.env` en el directorio raíz del proyecto con las credenciales necesarias, como la clave API para OpenAI.

2. **Generación de Embeddings**

   Ejecuta el cuaderno Jupyter `embeddings.ipynb` para cargar el PDF y generar los embeddings necesarios para la base de datos. Asegúrate de que el PDF y la base de datos Chroma estén en las rutas correctas.

3. **Ejecución del Servidor de Streamlit**

   Para iniciar la aplicación web, ejecuta el siguiente comando:

   ```bash
   streamlit run app.py
   ```

Detalles del Código

`main.py`

- **Configuración de Embeddings:**

Utiliza `OpenAIEmbeddings` para crear una función de embeddings que se usa para indexar y buscar en la base de datos Chroma.

- **Instanciación del Modelo:**

Se crea una instancia de `ChatOpenAI` con el modelo `gpt-3.5-turbo`.

- **Base de Datos:**

Se carga la base de datos Chroma desde el directorio especificado.

- **Herramientas de Búsqueda:**

Define dos herramientas:

- `busqueda_Hollow`: Busca información en una lista de hipervínculos.
- `busqueda_videos_youtube`: Busca videos en YouTube.

- **Agente:**

Se crea un agente utilizando `create_openai_tools_agent`, que combina el modelo de lenguaje y las herramientas definidas.

- **Función de Chatbot:**

La función `chatbot` procesa la consulta del usuario y devuelve la respuesta del agente.

`app.py`

- **Interfaz de Usuario:**

Utiliza Streamlit para crear una interfaz web donde los usuarios pueden ingresar consultas y recibir respuestas del chatbot.

- **Historial de Chat:**

Mantiene el historial de chat en `st.session_state` y permite borrarlo si es necesario.

- **Interacción con el Chatbot:**

Envía las consultas del usuario a la función `chatbot` y muestra las respuestas en la interfaz.

`embeddings.ipynb`

- **Carga de PDF y Creación de Embeddings:**

Utiliza `PyPDFLoader` para cargar el PDF y `Chroma` para crear y guardar los embeddings.

Consideraciones Adicionales

- **Seguridad:**

Asegúrate de que las credenciales en el archivo `.env` estén seguras y no se incluyan en el control de versiones.

- **Rendimiento:**

La creación de embeddings y la consulta a la base de datos pueden requerir tiempo y recursos. Asegúrate de optimizar según sea necesario.

- **Mantenimiento:**

Actualiza las dependencias y el código regularmente para garantizar la compatibilidad y seguridad.

Diagrama de Arquitectura: