Documentación del Sistema RAG en Python

Este documento describe la implementación de un sistema **RAG (Retrieval-Augmented Generation)** desarrollado en Python. El sistema permite cargar documentos (PDF y TXT), fragmentarlos, almacenar representaciones vectoriales (embeddings) en una base de datos Chroma, recuperar información relevante y generar respuestas contextuales utilizando el modelo GPT-4 de OpenAI.

1. Requisitos previos

- Python 3.9 o superior.
- Instalar dependencias necesarias: langchain, langchain-community, langchain-chroma, langchain-openai, reportlab.
- Configurar la variable de entorno OPENAI_API_KEY con un token válido de OpenAI.
- Archivos PDF o TXT que se deseen utilizar como fuente de información.

2. Flujo del sistema

El sistema RAG implementa los siguientes pasos: 1. Carga de documentos mediante loaders especializados para PDF y TXT. 2. División del texto en fragmentos (chunks) de 1000 caracteres con un solapamiento de 200 caracteres. 3. Generación de embeddings utilizando el modelo `text-embedding-3-small` de OpenAI. 4. Almacenamiento de los embeddings en una base de datos Chroma persistente. 5. Recuperación de información relevante mediante búsqueda de similitud (top 5 fragmentos). 6. Construcción del prompt con la pregunta del usuario y los fragmentos relevantes. 7. Envío del prompt al modelo GPT-4 para obtener una respuesta contextualizada.

3. Documentación de funciones

load_documents(file_paths): Carga documentos PDF o TXT y devuelve una lista de documentos.

chunk_documents(documents): Divide los documentos en fragmentos (chunks) para facilitar la indexación.

store_in_chromadb(chunks, embeddings_model): Crea o carga una base de datos Chroma para almacenar embeddings.

retrieve_relevant_info(question, vector_store): Recupera los 5 fragmentos más relevantes de acuerdo a la pregunta.

generate_prompt(question, relevant_docs): Genera un prompt con la pregunta y los documentos relevantes.

generate_answer(prompt): Envía el prompt al modelo GPT-4 para generar la respuesta.

Pregunta_LLM(file_paths, question): Función principal que ejecuta el flujo completo del sistema RAG.

4. Ejecución del sistema

Para ejecutar el sistema desde un script Python o un notebook, siga los pasos: 1. Defina la variable de entorno con su clave de OpenAI: ```bash export OPENAI_API_KEY="su_api_key" ``` 2. Prepare los documentos PDF o TXT en el directorio de trabajo. 3. Ejecute el script principal. Ejemplo de uso:

```python if __name__ == "__main__": file_paths = ['documento1.pdf', 'documento2.txt'] question = "¿Cuál es el resumen del contenido?" answer = Pregunta_LLM(file_paths, question) print(answer.content) ``` 4. El sistema imprimirá en consola la respuesta generada por el modelo GPT-4.

5. Notas adicionales

- La base de datos Chroma se almacena localmente en la carpeta `./chroma_db`. - Si la base de datos ya existe, el sistema no recalcula embeddings, solo los reutiliza. - El modelo puede ajustarse cambiando los parámetros de temperatura o seleccionando otra variante de GPT en la función `generate_answer`.
```