Retrieval-Augmented Generation (RAG) Documentation

Overview of RAG

Retrieval-Augmented Generation (RAG) combines document retrieval and response generation to produce contextually relevant answers. It uses a knowledge base to retrieve relevant documents and a large language model to generate responses.

Key Components in the Code

1. **Vector Store (Pinecone):** Stores embeddings of the knowledge base documents for efficient similarity searches.
2. **Embeddings (OpenAI):** Converts text into high-dimensional vectors for similarity matching.
3. **Document Chain:** Combines multiple retrieved documents into a coherent response.
4. **Retrieval Chain:** Executes the retrieval and response generation pipeline.

Workflow

1. **User Query:** The chatbot receives input via WebSocket.
2. **Document Retrieval:** Retrieves the top relevant documents from Pinecone.
3. **Response Generation:** The LLM generates contextually accurate responses.
4. **Response Delivery:** Sends the generated response or profile to the user.

Features

1. Top-k Document Retrieval for relevance.

2. Context Awareness through chat history.

3. Dynamic Response Generation guided by retrieved data.

Limitations

1. Dependency on the knowledge base for accuracy.

2. Scalability concerns with in-memory chat history.

Future Enhancements

1. Incorporate hierarchical retrieval strategies.

2. Use a database for persistent chat history.

3. Explore fine-tuned embeddings for improved accuracy.