Intelligent Query Resolution Using RAG with Self-Reflection

Introduction

This project focuses on building an intelligent chatbot capable of resolving user queries by extracting and utilizing information from PDF documents. The chatbot employs Retrieval-Augmented Generation (RAG) techniques with integrated self-reflection to ensure precise, contextually accurate answers while minimizing errors and providing fallback responses when necessary.

Data Preparation and Embedding Creation

Extracting Text from PDFs

The chatbot utilizes `PyMuPDF` to extract text from PDF documents. Each page is processed iteratively, and the resulting text is consolidated into a single corpus for efficient handling and analysis.

Segmenting Text into Chunks

To improve retrieval accuracy, the extracted text is divided into smaller, manageable, overlapping chunks using `RecursiveCharacterTextSplitter`. This segmentation ensures each chunk maintains logical coherence, enhancing context preservation during query resolution.

Generating Semantic Embeddings

OpenAI's embedding model is employed to generate semantic vector representations for each text chunk. These embeddings are stored in a `Chroma` vector database, which is persisted locally (e.g., `./chroma_db`) to enable rapid retrieval of relevant information.

Interactive Workflow for Query Resolution

Workflow Components

The chatbot operates through a state-based workflow, designed using `StateGraph`, and processes queries through the following key stages:

1. **Content Retrieval:** Retrieves document chunks most similar to the user query based on cosine similarity of semantic embeddings.
2. **Relevance Filtering:** Filters retrieved chunks through a `retrieval_grader`, retaining only the most pertinent ones for generating answers.
3. **Query Refinement:** Improves poorly performing queries by rephrasing them with a `question_rewriter` to enhance retrieval outcomes.
4. **Response Generation:** Combines the refined query with relevant content to generate an answer using a RAG pipeline powered by `gpt-4o-mini`.
5. **Quality Assurance:** Assesses the response for factual accuracy and relevance using grading mechanisms. Unsupported or inadequate responses trigger query refinement and regeneration.

Dynamic Adjustments Through Feedback

To ensure high-quality responses, the workflow incorporates feedback loops. These loops allow the system to revisit earlier stages, refine queries, and improve the accuracy and relevance of generated answers.

Advanced Functionalities

Error Handling and Fallback Responses

If the chatbot is unable to retrieve relevant information or produce a grounded response, it displays the fallback message:

> *"Sorry, I didn't understand your question. Do you want to connect with a live agent?"*

Integrated Self-Reflection Mechanisms

The chatbot utilizes self-reflection at multiple stages:

- **Document Relevance Grading:** Ensures retrieved content directly relates to the query.
- **Response Validation:** Verifies that generated answers are grounded in the retrieved content and free from hallucinations.
- **Query Optimization:** Dynamically rewrites suboptimal user inputs to align with intended meaning and improve retrieval effectiveness.

User Interaction and Interface

Streamlit-Based Interface

The user interface is built using `Streamlit`, providing an intuitive and interactive experience:

1. Users enter their queries via a text input field.
2. The interface displays the processing steps as the workflow progresses.
3. A final response is presented, or a fallback message is shown when the system cannot generate a valid answer.

Technological Stack

Core Tools and Libraries

- **PDF Processing:** `PyMuPDF (fitz)` for text extraction.
- **Text Segmentation:** `LangChain`'s `RecursiveCharacterTextSplitter`.
- **Vector Management:** `Chroma` for embedding storage and retrieval.
- **Language Models:** `gpt-4o-mini` for query rewriting, document filtering, and response generation.
- **Workflow Frameworks:** `LangChain` and `LangGraph` for modular state-based workflow design.
- **User Interface:** `Streamlit` for interactive application development.

Addressing Key Challenges

Improving Retrieval Accuracy

Irrelevant retrievals are mitigated by employing a rigorous filtering mechanism, leveraging self-reflection to ensure retrieved content is contextually aligned with user queries.

Avoiding Hallucinations in Responses

To prevent generating ungrounded answers, the system incorporates hallucination grading, ensuring all responses are derived from verified content.

Enhancing Ambiguous Queries

Ambiguous or poorly phrased user inputs are dynamically rewritten using a query rewriter to capture semantic intent more effectively.

Conclusion

By combining Retrieval-Augmented Generation (RAG) with self-reflection, this chatbot delivers precise, context-aware responses from PDF documents. Its modular design ensures scalability, maintainability, and robust error handling, making it a powerful tool for query resolution and knowledge extraction.