Retrieval-Augmented Generation (RAG) System Report

Document Ingestion, Indexing, and Retrieval Approach

Initially, implemented a straightforward but effective pipeline to quickly build a functional Retrieval-Augmented Generation (RAG) system. This approach began with ingesting raw, unstructured text data. Using regular expressions, I extracted essential elements such as document IDs, titles, and descriptions, converting unstructured content into structured records suitable for downstream tasks.

For representing these documents in a meaningful semantic space, I chose the "BAAI/bge-small-en-v1.5" model from the SentenceTransformers library. Each document was transformed into a vector embedding, enabling semantic similarity comparisons. To facilitate rapid and efficient retrieval, I used FAISS (specifically, the basic IndexFlatL2), allowing quick nearest-neighbor searches based on the vector embeddings. Queries were processed similarly—converted into embeddings to find relevant documents.

While this initial setup works well for smaller-scale, simple applications, I noticed limitations when scaling to larger datasets or needing frequent updates. To address these issues, I thought o developing a more robust and scalable production-level pipeline, incorporating several improvements:

- **Streaming Data Ingestion:** Rather than loading the entire dataset into memory, documents are processed sequentially (streamed), significantly reducing memory usage and improving scalability.
- **Batch Embedding Strategy:** Documents are embedded in manageable batches, enhancing memory efficiency and computational performance.
- **Advanced FAISS Index:** Transitioning to the more sophisticated FAISS IndexHNSWFlat significantly improved the speed and scalability of similarity searches. Additionally, saving the index to disk ensures quick recovery and stability. Other ways to make it more efficient and scalable are if we create an index in cloud for ex: Pinecone Vector DB Index, Azure AI Search Index, MongoDB Atlas Vector DB Index etc.
- **Persistent Metadata Management:** Using SQLite for metadata storage allows persistent, efficient, and reliable management of document details, ensuring consistency and ease of maintenance.

Prompt Design and Experimentation

Prompt engineering was a critical component of my RAG system, significantly influencing the quality of the responses generated by the language model. I explored several types of prompts:

- **Standard Prompt:** Simple and direct context-question pairs that effectively handle straightforward queries.
- **Chain-of-Thought Prompt:** Encourages step-by-step reasoning, leading to clearer, more logically coherent responses.

- **Role-Based Prompt:** Positions the model as a specific expert (e.g., forensic analyst), producing nuanced, detailed insights particularly valuable in specialized analytical tasks.
- **Bullet Points Prompt:** Formats responses into concise lists, greatly improving readability and accessibility.
- **ChatML Prompt:** Designed specifically for conversational models (like OpenAI's ChatML), facilitating detailed and engaging dialogue.

Each prompt type was dynamically generated, incorporating context-specific retrieved content, allowing me to adjust easily according to the task requirements and observe the resulting effects.

Observations on Prompt Variations and Performance

Through experimentation, I observed how variations in prompt styles directly influenced model performance and response quality:

- **Standard Prompts** were quick to set up and ideal for basic queries.
- **Chain-of-Thought Prompts** consistently improved logical reasoning and the clarity of responses, especially beneficial when complex reasoning was required.
- **Role-Based Prompts** significantly increased specificity and detail, particularly effective for in-depth analytical tasks such as risk assessment.
- **Bullet Points Prompts** improved readability, making the responses easy to digest quickly, which was particularly useful for summarizing information.
- **ChatML Prompts** enhanced user engagement through detailed, conversationally structured responses, ideal for interactive scenarios.

Overall, deliberate experimentation with prompt structures provided valuable insights into optimizing system performance and highlighted the importance of tailoring prompts to specific analytical goals.