Retrieval-Augmented Generation (RAG) System Report

Document Ingestion, Indexing, and Retrieval Strategy

As a data scientist building a functional RAG system, my first priority was to establish a robust pipeline for ingesting and structuring unstructured text data. I began by designing a preprocessing routine that parsed raw text using regular expressions, extracting document IDs, titles, and descriptions. This allowed for clean and consistent input downstream.

To represent textual content in a semantically meaningful way, I used the "BAAI/bge-small-en-v1.5" model from the SentenceTransformers library. Each document was embedded into a high-dimensional vector space. I then built a FAISS index (specifically using `IndexFlatL2`) to enable fast and scalable similarity search. This approach ensured that even large datasets could be queried efficiently using vector-based retrieval.

For each user query, I followed a symmetric encoding approach: the query was transformed into a vector using the same embedding model, and FAISS was used to retrieve the top-k most relevant documents based on L2 distance. These documents formed the contextual backbone for the next stage—language model prompting.

Prompt Design and Experimentation

A critical component of the RAG system was prompt engineering. I experimented with several prompt templates, each tailored for different types of analytical tasks and response styles:

- **Standard Prompt**: A straightforward format that combined context with a question, prompting a direct answer.
- **Chain-of-Thought Prompt**: Encouraged the LLM to reason step-by-step, which often led to more detailed and accurate outputs.
- **Role-Based Prompt**: Positioned the LLM as a forensic expert, which helped in surfacing nuanced insights from the retrieved documents.
- **Bullet Points Prompt**: Requested findings in a list format to promote clarity and quick scanning.
- **ChatML Prompt**: Structured for LLMs that require system/user prompt formatting (e.g., OpenAI's ChatML).

Each prompt was dynamically generated based on retrieved content and the task at hand, allowing for flexible experimentation.

Impact of Prompt Variations on System Performance

The impact of prompt design on LLM output was significant:

- **Standard Prompts** were fast and easy to implement, with clear instructions.
- **Chain-of-Thought Prompts** consistently improved logical coherence and were particularly useful when interpretability was a priority.
- **Role-Based Prompts** elevated the specificity about the potential risks analysis.

- **Bullet Points Prompts** improved readability and conciseness, making them ideal for summarization tasks.

- **ChatML Prompts**. Nicely explained and thorough answers of queries.

In conclusion, thoughtful prompt engineering played a pivotal role in shaping the quality and usefulness of model outputs. The ability to test multiple prompting strategies gave me insight into the nuanced interplay between context, task framing, and model behavior—ultimately improving the performance and reliability of the RAG pipeline.