Prompt Engine Instruction File: Retrieval-Augmented Generation (RAG)

1. Core RAG Paradigms

The implementation of RAG can be categorized into three main paradigms, each evolving from the last:

2.1. Naive RAG

[cite_start]This is the most straightforward implementation of RAG, following a simple "Retrieve-Read" framework[cite: 178].

- **Indexing:** Documents are cleaned, extracted, and segmented into smaller chunks. [cite_start]These chunks are then converted into vector embeddings and stored in a vector database[cite: 179, 180, 181].
- **Retrieval:** When a user submits a query, it's converted into a vector. [cite_start]The system then searches the vector database for the `top-K` most similar document chunks[cite: 183, 184, 185].
- [cite_start]**Generation:** The retrieved chunks and the original query are combined into a prompt that is fed to the LLM to generate an answer[cite: 187].

2.2. Advanced RAG

[cite_start]This paradigm introduces optimizations to the Naive RAG process to improve retrieval quality[cite: 200, 201].

- **Pre-retrieval:** This stage focuses on optimizing the indexing process and the user query itself. [cite_start]Techniques include enhancing data granularity, adding metadata to chunks, and query rewriting or expansion[cite: 265, 266, 267, 268, 269].
- [cite_start]**Post-retrieval:** After retrieving documents, this stage involves re-ranking the chunks to place the most relevant information at the beginning and end of the prompt (to counter the "lost in the middle" problem) and compressing the context to remove noise and irrelevant information[cite: 270, 271, 272, 274, 275].

2.3. Modular RAG

[cite_start]The most flexible and adaptable paradigm, Modular RAG allows for the addition of specialized modules and the reconfiguration of the RAG pipeline[cite: 277].

- [cite_start]**New Modules:** This can include a `Search` module for direct access to various data sources, a `Memory` module that uses the LLM's memory to guide retrieval,

and a `Routing` module to select the best data source for a given query[cite: 283, 285, 288].
- [cite_start]**New Patterns:** Instead of a fixed "Retrieve-Read" sequence, Modular RAG can employ more complex patterns like `Rewrite-Retrieve-Read` or `Generate-Read`[cite: 294, 295]. [cite_start]It also allows for adaptive retrieval, where the model decides when and what to retrieve[cite: 300, 301].

---

3. Key Components & Optimization Techniques

3.1. Retrieval

The quality of the retrieval process is crucial for the success of any RAG system.

- [cite_start]**Chunking Strategy:** Instead of fixed-size chunks, consider recursive splitting or a "small2big" approach where smaller, more precise chunks are retrieved, but the surrounding context is provided to the LLM[cite: 404, 406].
- **Query Optimization:**
    - [cite_start]**Expansion:** Expand a single query into multiple, more specific queries to cover different aspects of the user's intent[cite: 429, 430].
    - **Transformation:** Rewrite the user's query to be more suitable for retrieval. [cite_start]Techniques like HyDE (Hypothetical Document Embeddings) generate a hypothetical answer to the query and use its embedding for retrieval[cite: 438, 439, 444].
    - [cite_start]**Routing:** Use a router to direct the query to the most appropriate data source or RAG pipeline based on its content or metadata[cite: 448, 449, 450].
- **Embedding:**
    - [cite_start]**Fine-tuning:** For domain-specific applications, fine-tune the embedding model on your own dataset to improve its understanding of specialized jargon[cite: 466].
    - [cite_start]**Hybrid Retrieval:** Combine sparse retrieval methods (like BM25) with dense retrieval to leverage the strengths of both[cite: 460].

3.2. Generation

Simply feeding all retrieved information to the LLM is not optimal.

- **Context Curation:**
    - [cite_start]**Reranking:** Reorder the retrieved chunks to place the most relevant information at the beginning and end of the context[cite: 495].
    - [cite_start]**Compression:** Use a smaller LLM to compress the retrieved context by removing unimportant tokens, making it more digestible for the main generator LLM[cite: 499].

- [cite_start]**LLM Fine-tuning:** Fine-tune the generator LLM on domain-specific data to improve its ability to understand the retrieved context and generate responses in a specific style or format[cite: 512, 514, 517].

3.3. Augmentation Process

The interaction between retrieval and generation can be optimized.

- **Iterative Retrieval:** The LLM generates a response, and then another retrieval step is performed based on the generated text to gather more information. [cite_start]This process can be repeated multiple times[cite: 530].
- **Recursive Retrieval:** Break down a complex query into a series of sub-queries. [cite_start]The results from each sub-query are used to inform the next, creating a chain of reasoning[cite: 571, 572, 573].
- **Adaptive Retrieval:** Allow the LLM to decide when it needs to retrieve information. [cite_start]This can be achieved by using special tokens that trigger the retrieval process when the LLM's generation confidence is low[cite: 583, 589, 592].

---

4. Evaluating RAG Systems

Evaluating a RAG system goes beyond measuring the final answer's accuracy.

- **Evaluation Targets:**
    - [cite_start]**Retrieval Quality:** Measured by metrics like Hit Rate, MRR, and NDCG[cite: 611, 614].
    - [cite_start]**Generation Quality:** Assessed based on faithfulness (does the answer contradict the source?), relevance, and non-harmfulness[cite: 615, 617].
- **Required Abilities:**
    - [cite_start]**Noise Robustness:** Can the model handle irrelevant or noisy documents in the retrieved context[cite: 629]?
    - [cite_start]**Negative Rejection:** Does the model know when to say "I don't know" if the answer is not in the retrieved documents[cite: 630]?
    - [cite_start]**Information Integration:** How well can the model synthesize information from multiple sources[cite: 631]?
    - [cite_start]**Counterfactual Robustness:** Can the model identify and ignore inaccuracies in the source documents[cite: 632]?

[cite_start]Several benchmarks and tools, such as RAGAS, ARES, and TruLens, can be used for a more systematic evaluation of RAG models[cite: 648].

5. References

- RAG Example