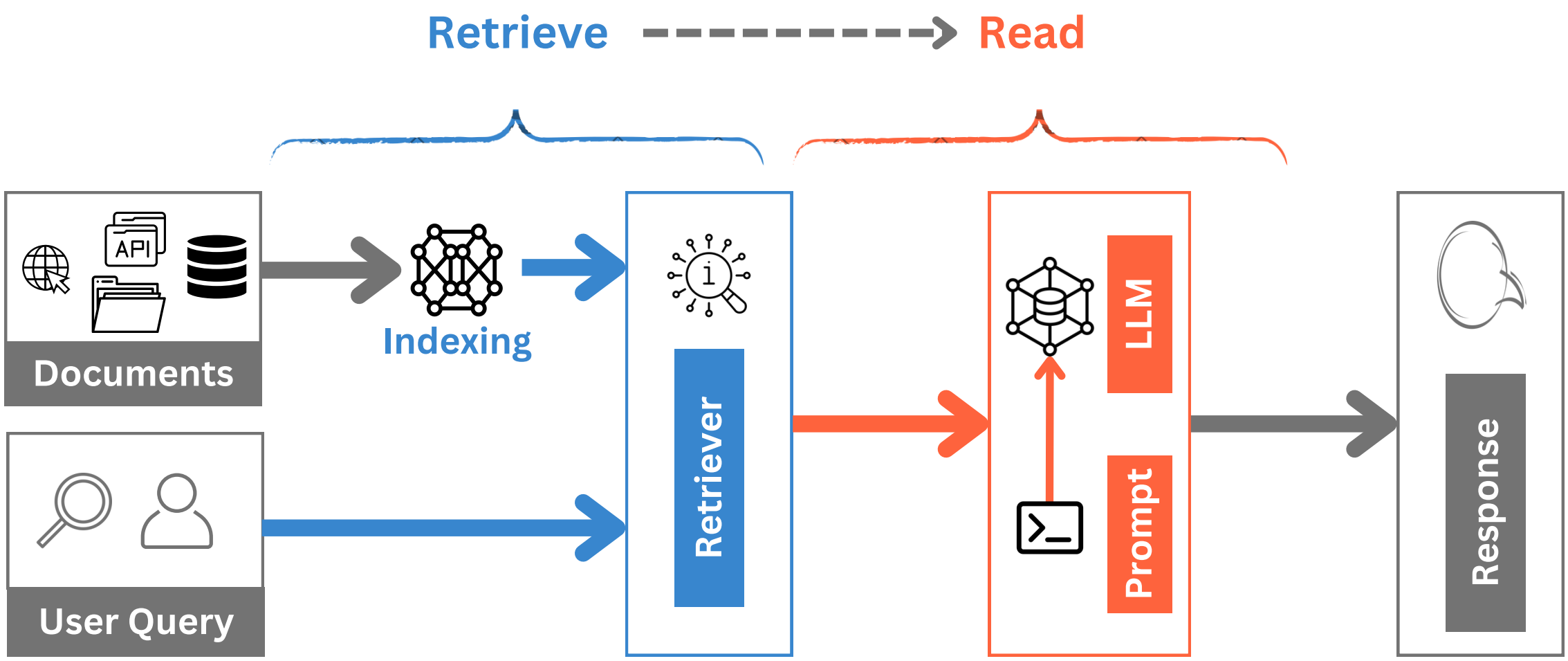
Progression of RAG Systems

Ever since its introduction in mid-2020, RAG approaches have followed a progression aiming to achieve the redressal of the hallucination problem in LLMs

Naive RAG

At its most basic, Retrieval Augmented Generation can be summarized in three steps -

1. **Indexing** of the **documents**
2. **Retrieval** of the context with respect to an input query
3. **Generation** of the **response** using the input query and retrieved context



This basic RAG approach is also termed "Naive RAG"

Challenges in Naive RAG

Retrieval

- Low Recall
- Low Precision

Augmentation

- Redundancy & Repetition
- Disjointed Context

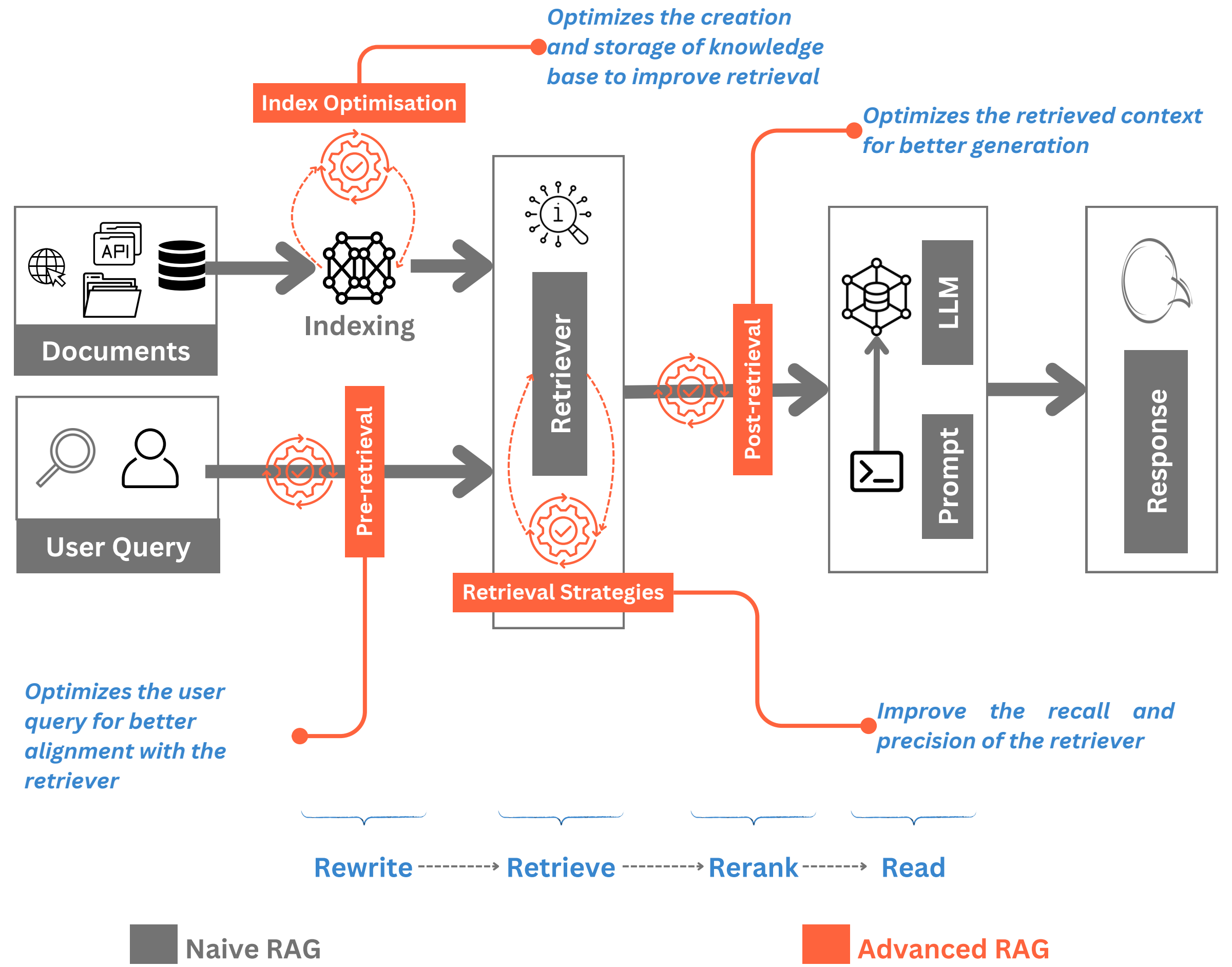
Generation

- Reconciliation Challenges
- Over-reliance on Context

Progression of RAG Systems

Advanced RAG

To address the inefficiencies of the Naive RAG approach, Advanced RAG approaches implement strategies focussed on these processes -



Progression of RAG Systems

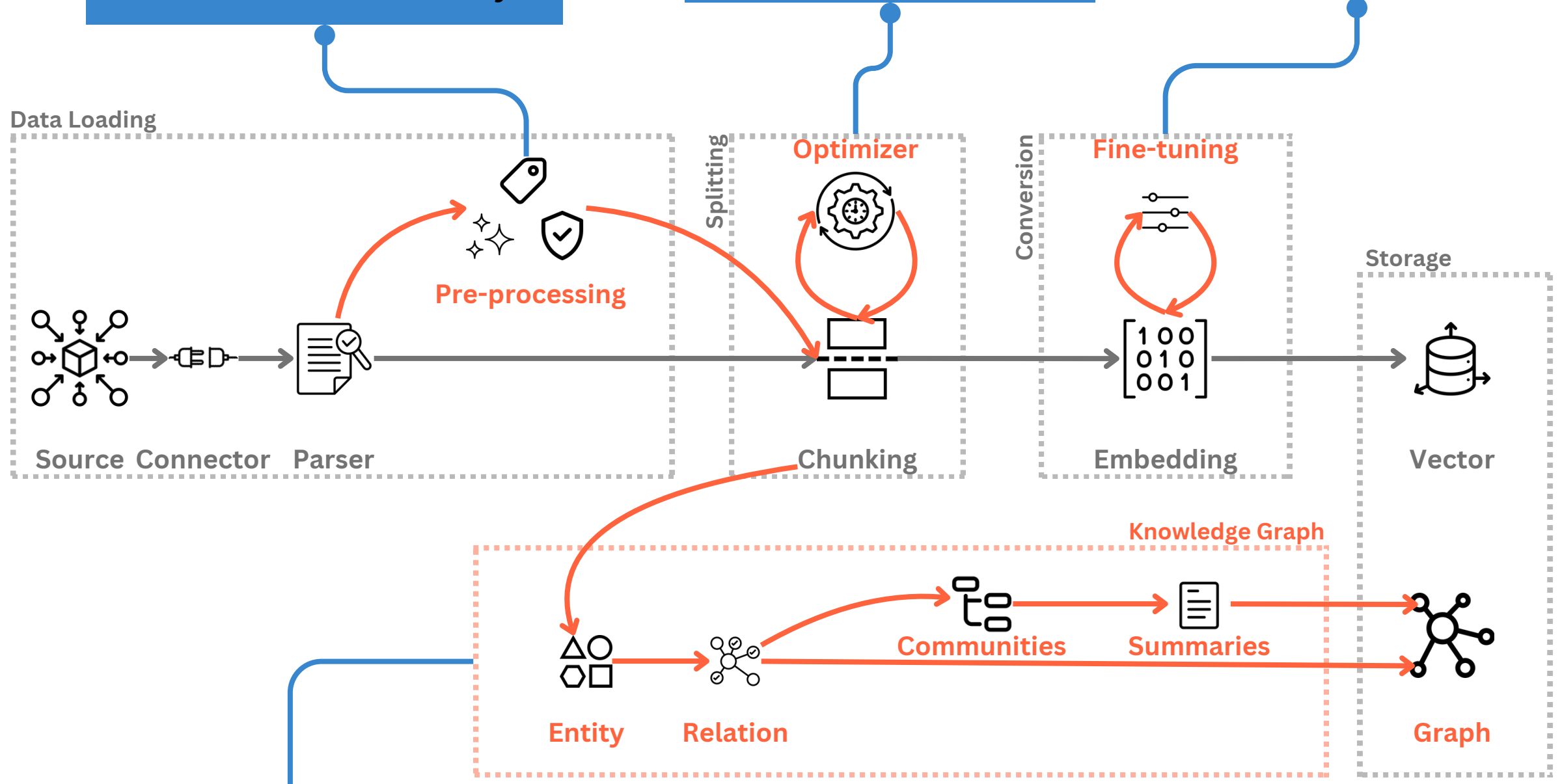
Index Optimization

The objective of index optimization is to set up the knowledge base for better retrieval.

Pre-processing steps like cleaning, metadata enhancement while loading data enhance the searcheability

Chunk Size optimization and enriched context enhances retrieval and contextual generation

Embeddings model can be fine-tuned for domain adaptability



Knowledge Graph Indexing results in deeper context by establishing entity relationships and creating higher order community sub-graph summaries

Naive RAG

Advanced RAG

Progression of RAG Systems

Query Optimization

The objective of query optimization is to optimize the input user query in a manner that makes it better suited for the retrieval tasks

Query Expansion

In query expansion, the original user query is enriched with the aim of retrieving more relevant information. This helps in increasing the recall of the system and overcomes the challenge of incomplete or very brief user queries. Some of the techniques that expand user queries are -

- Multi-query expansion
- Sub-query expansion
- Step back expansion

Query Transformation

Compared to query expansion, in query transformation, instead of the original user query retrieval happens on a transformed query which is more suitable for the retriever.

- Query rewriting
- Hypothetical Document Embedding (Hyde)

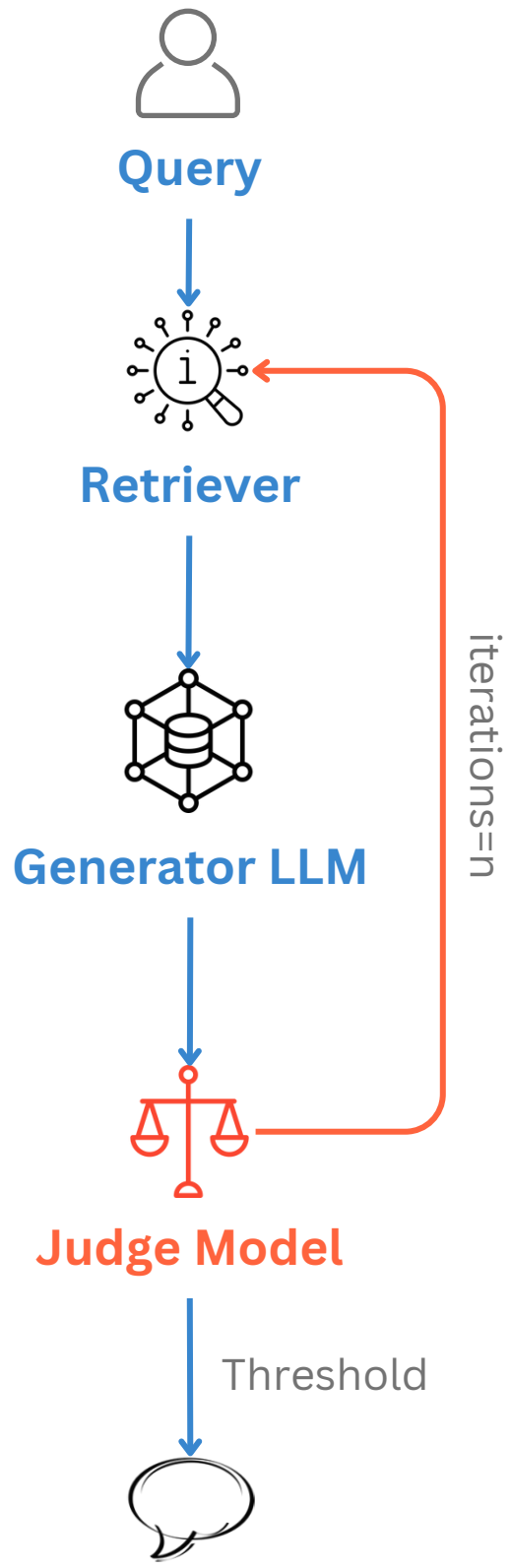
Query Routing

Different queries can demand different retrieval methods. Based on criteria like intent, domain, language, complexity, source of information etc., queries are needed to be classified so that they can follow the appropriate retrieval method. This is the idea behind optimizing the user query by routing it to the appropriate workflow. Types of routing techniques include –

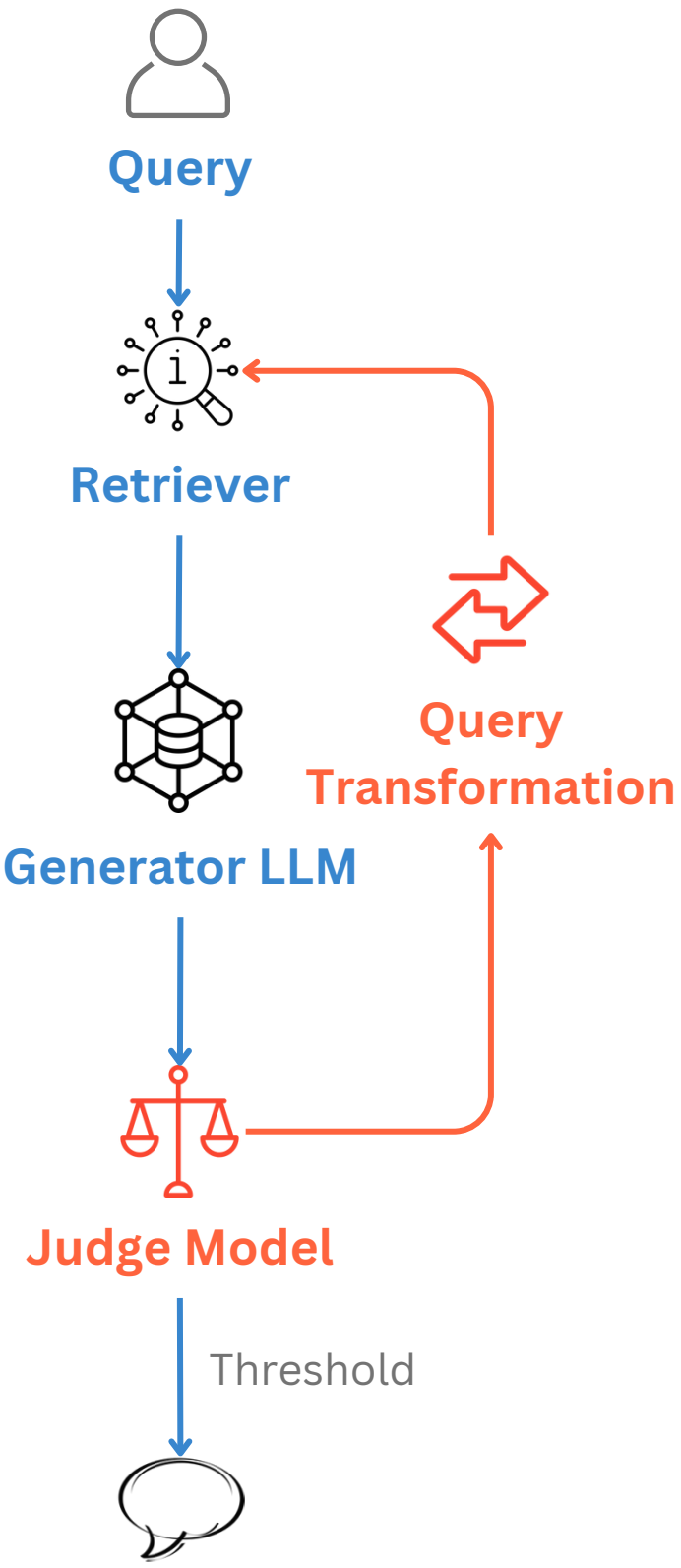
- Intent Classification
- Metadata Routing
- Semantic Routing

Progression of RAG Systems

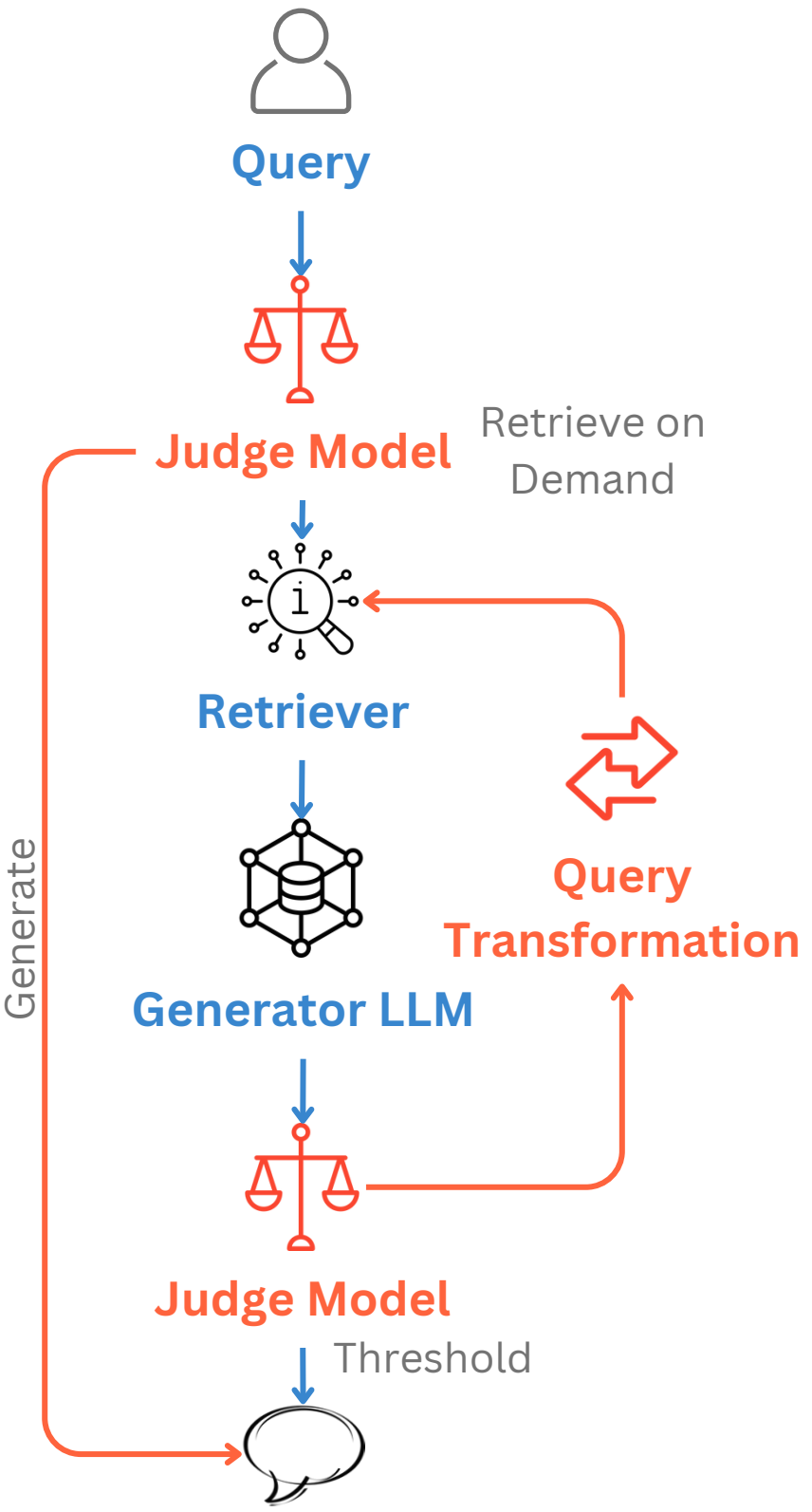
Retrieval Strategies



Iterative Retrieval



Recursive Retrieval



Adaptive Retrieval

Progression of RAG Systems

Post Retrieval Strategies

Even if the retrieval of the chunks happens in an expected manner, there is still a point of failure that remains. The LLM might not be able to process all the information. This may be due to redundancies or disjointed nature of the context amongst many other reasons. At the post-retrieval stage the approaches of reranking and compression help in providing better context to the LLM for generation.

Compression

Excessively long context has the potential of introducing noise into the system. This diminishes the LLM's capability to process information. As a result, hallucinations and irrelevant responses to the query may persist. In prompt compression, language models are used to detect and remove unimportant and irrelevant tokens. Apart from making the context more relevant, prompt compression also has a positive influence on the cost and efficiency. Another advantage of prompt compression is to be able to reduce the size of the prompt so that it can fit into the context window of the LLM. COCOM is a context compression method which compresses contexts into a small number of context embeddings. Similarly, xRAG is a method that uses document embeddings as features.

Reranking

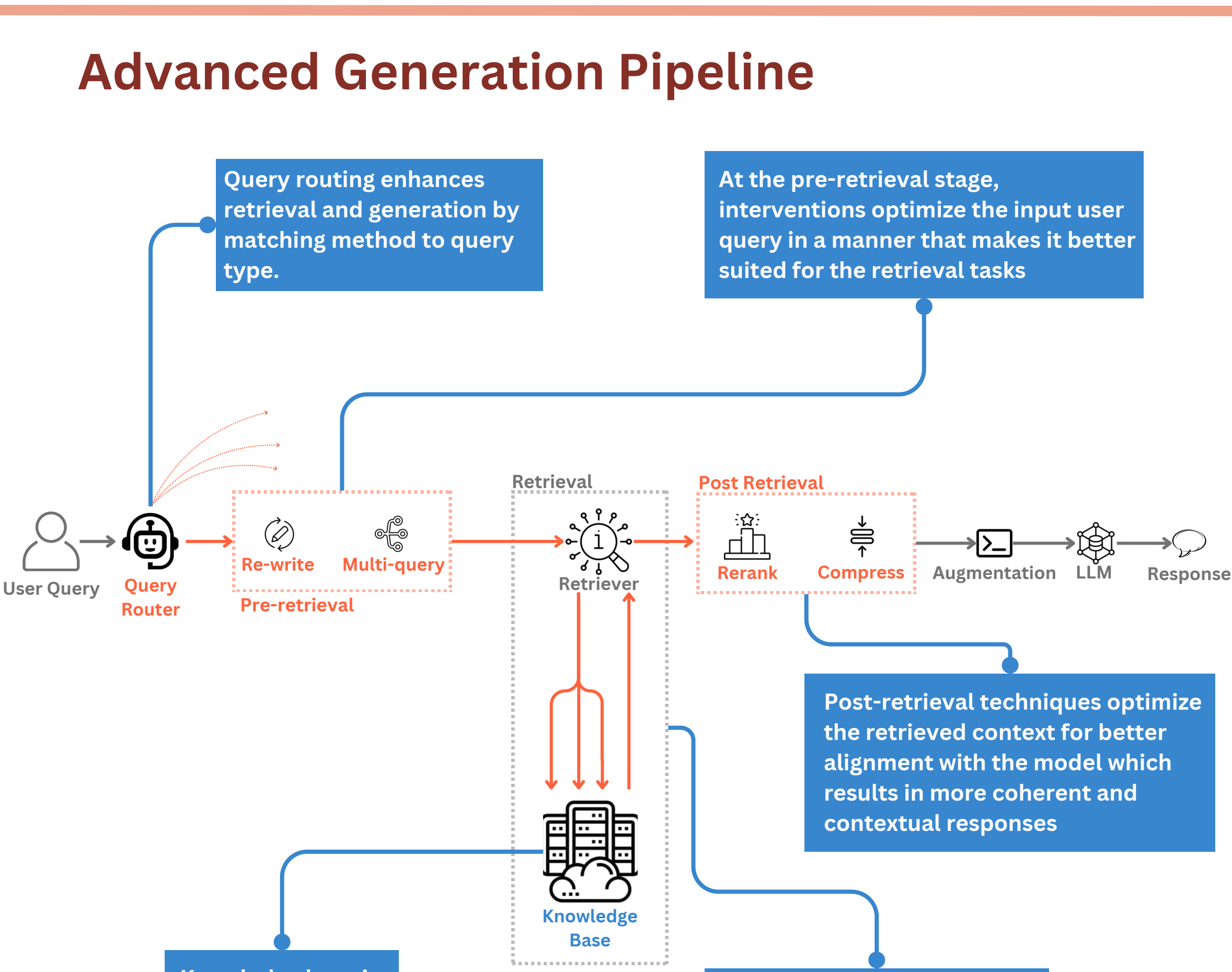
Reordering all the retrieved documents ensures that the most relevant information is prioritized for the generation step. It refines retrieval results by prioritizing documents that are more contextually appropriate for the query, improving the overall quality and accuracy of information used for generation. This also addresses the question of prioritization when a hybrid approach to retrieval is employed and improves the overall response quality. There are commonly available rerankers like multi-vector, Learning to Rank (LTR), BERT based and even hybrid rerankers that can be employed. Specialized APIs like Cohere Rerank offer pre-trained models for efficient reranking integration.

Progression of RAG Systems

Advanced Generation Pipeline

Query routing enhances retrieval and generation by matching method to query type.

At the pre-retrieval stage, interventions optimize the input user query in a manner that makes it better suited for the retrieval tasks



Post-retrieval techniques optimize the retrieved context for better alignment with the model which results in more coherent and contextual responses

Knowledge base is index optimized in the indexing pipeline

Hybrid and repetitive retrieval strategies improve the precision and recall of the RAG system

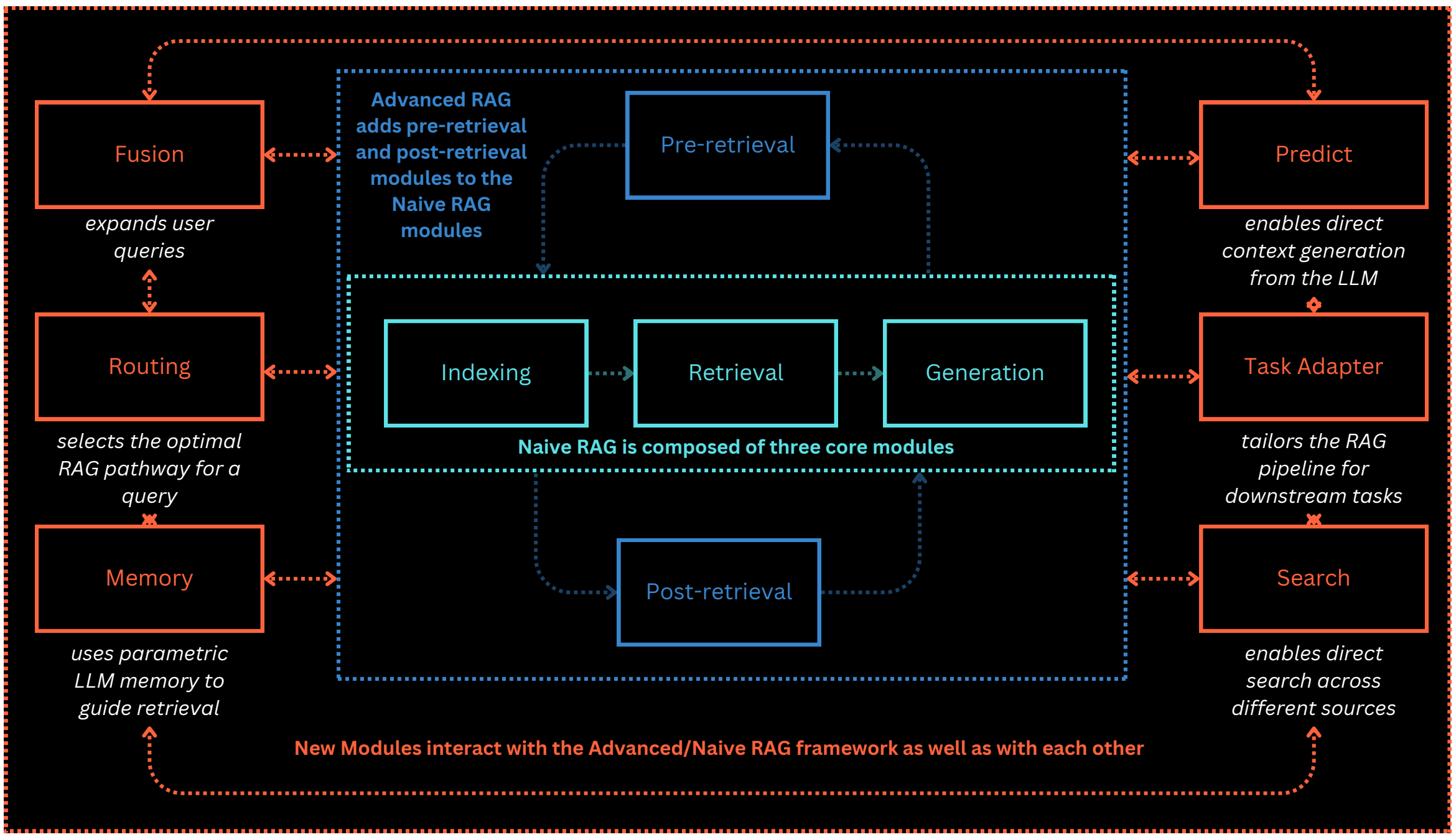
Naive RAG

Advanced RAG

Progression of RAG Systems

Modular RAG

AI systems are becoming increasingly complex and demand more customizable, flexible, and scalable RAG architectures. The emergence of Modular RAG is a leap forward in the evolution of RAG systems. Modular RAG breaks down the traditional monolithic RAG structure into interchangeable components. This allows for tailoring of the system to specific use cases. Modular approach brings modularity to RAG components like retrievers, indexing, and generation, while also adding more modules like search, memory, and fusion.



Naive, Advanced & Modular RAGs are not exclusive approaches but a progression. Naive RAG is a special case of Advanced which, in turn, is a special case of Modular RAG

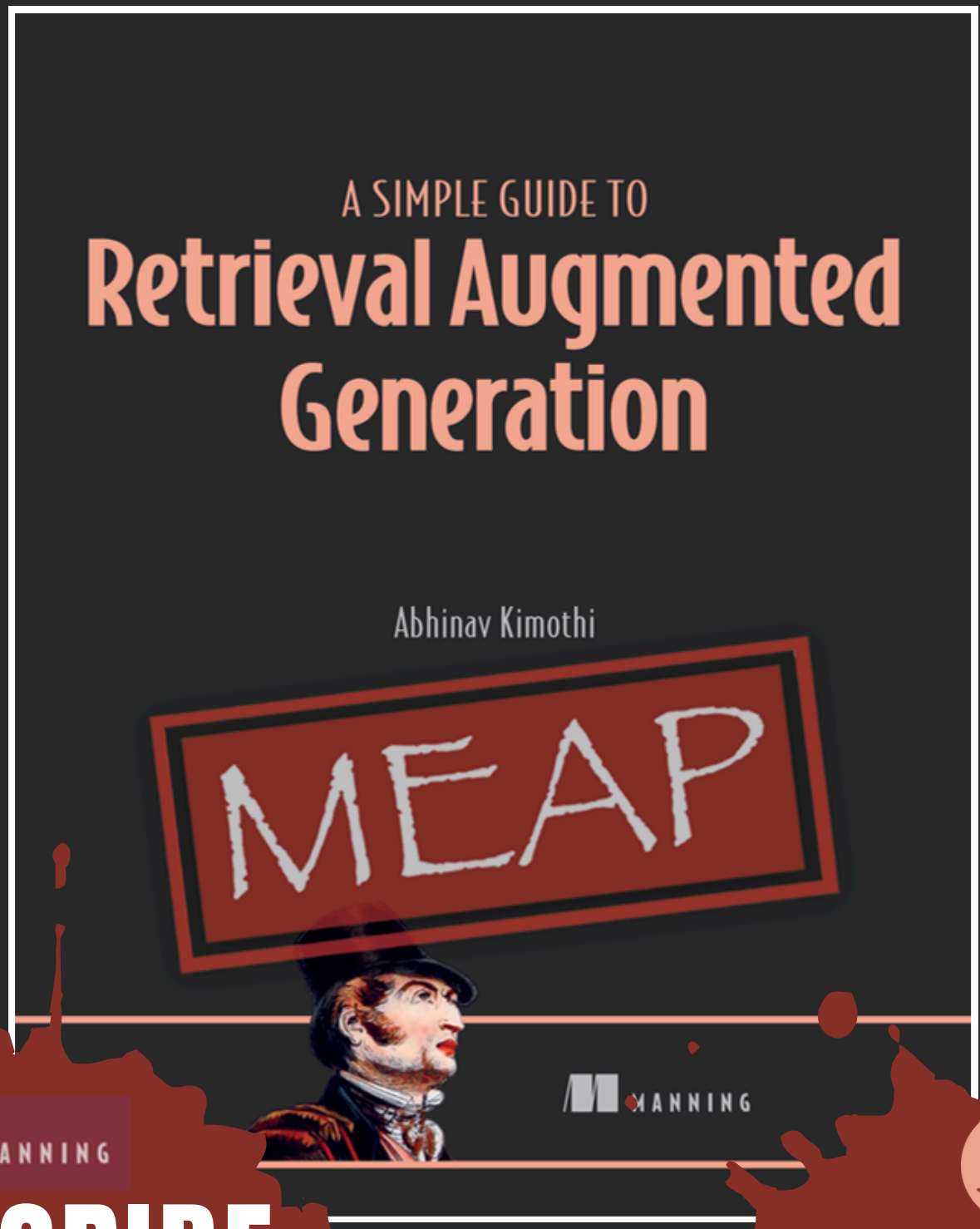
ARE YOU INTERESTED IN LEARNING MORE ABOUT RAG SYSTEMS?

SIX of NINE CHAPTERS OF

A SIMPLE GUIDE TO

RETRIEVAL AUGMENTED GENERATION

ARE NOW AVALIABLE FOR EARLY ACCESS



MANNING

SUBSCRIBE NOW

(Link in post)

SOURCE IS NOW CODE PUBLICLY AVAILABLE

(Check out on Github)