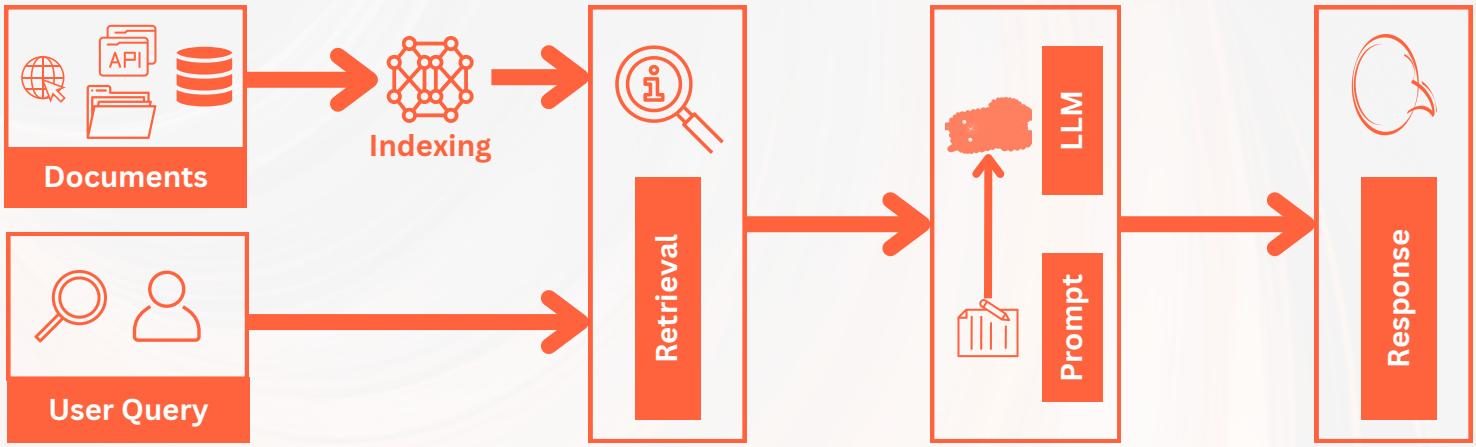
Progression of RAG Systems

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 can also be termed "Naive RAG"

Challenges in Naive RAG

Retrieval Quality

- **Low Precision** leading to Hallucinations/Mid-air drops
- **Low Recall** resulting in missing relevant info
- **Outdated information**

Augmentation

- **Redundancy and Repetition** when multiple retrieved documents have similar information
- **Context Length** challenges

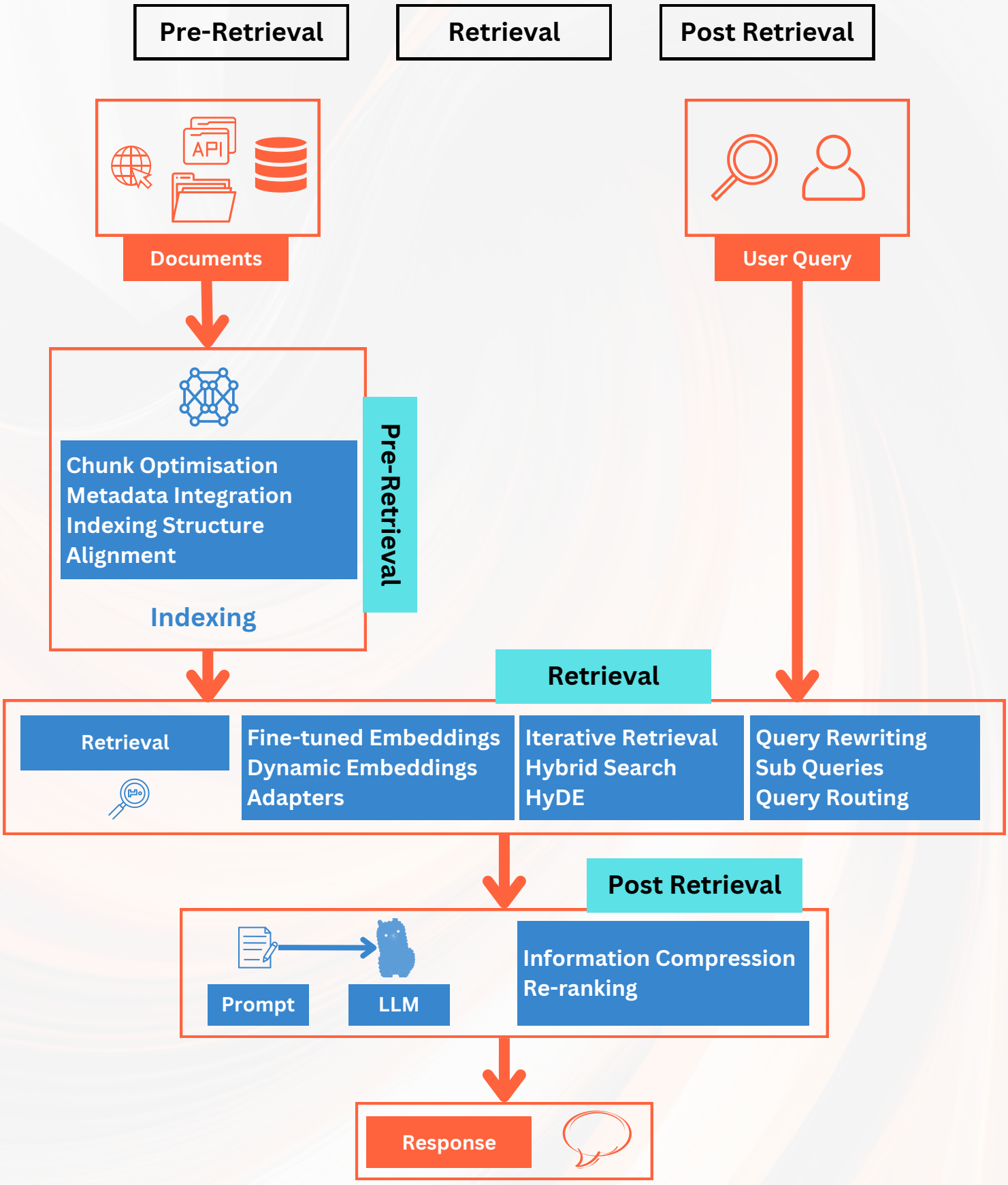
Generation Quality

- Generations are **not grounded** in the context
- Potential of **toxicity and bias** in the response
- **Excessive dependence** on augmented context

Abhinav Kimothi

Advanced RAG

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



** Indicative, non-exhaustive list*

Abhinav Kimothi

Advanced RAG Concepts

Pre-retrieval/Retrieval Stage

Chunk Optimization

When managing external documents, it's important to break them into the right-sized chunks for accurate results. The choice of how to do this depends on factors like content type, user queries, and application needs. No one-size-fits-all strategy exists, so flexibility is crucial. Current research explores techniques like sliding windows and "small2big" methods

Metadata Integration

Information like dates, purpose, chapter summaries, etc. can be embedded into chunks. This improves the retriever efficiency by not only searching the documents but also by assessing the similarity to the metadata.

Indexing Structure

Introduction of graph structures can greatly enhance retrieval by leveraging nodes and their relationships. Multi-index paths can be created aimed at increasing efficiency.

Alignment

Understanding complex data, like tables, can be tricky for RAG. One way to improve the indexing is by using counterfactual training, where we create hypothetical (what-if) questions. This increases the alignment and reduces disparity between documents.

Query Rewriting

To bring better alignment between the user query and documents, several rewriting approaches exists. LLMs are sometimes used to create pseudo documents from the query for better matching with existing documents. Sometimes, LLMs perform abstract reasoning. Multi-querying is employed to solve complex user queries.

Hybrid Search Exploration

The RAG system employs different types of searches like keyword, semantic and vector search, depending upon the user query and the type of data available.

Sub Queries

Sub querying involves breaking down a complex query into sub questions for each relevant data source, then gather all the intermediate responses and synthesize a final response.

Query Routing

A query router identifies a downstream task and decides the subsequent action that the RAG system should take. During retrieval, the query router also identifies the most appropriate data source for resolving the query.

Iterative Retrieval

Documents are collected repeatedly based on the query and the generated response to create a more comprehensive knowledge base.

Recursive Retrieval

Recursive retrieval also iteratively retrieves documents. However, it also refines the search queries depending on the results obtained from the previous retrieval. It is like a continuous learning process.

Adaptive Retrieval

Enhance the RAG framework by empowering Language Models (LLMs) to proactively identify the most suitable moments and content for retrieval. This refinement aims to improve the efficiency and relevance of the information obtained, allowing the models to dynamically choose when and what to retrieve, leading to more precise and effective results

Hypothetical Document Embeddings (HyDE)

Using the Language Model (LLM), HyDE forms a hypothetical document (answer) in response to a query, embeds it, and then retrieves real documents similar to this hypothetical one. Instead of relying on embedding similarity based on the query, it emphasizes the similarity between embeddings of different answers.

Fine-tuned Embeddings

This process involves tailoring embedding models to improve retrieval accuracy, particularly in specialized domains dealing with uncommon or evolving terms. The fine-tuning process utilizes training data generated with language models where questions grounded in document chunks are generated.

Post Retrieval Stage

Information Compression

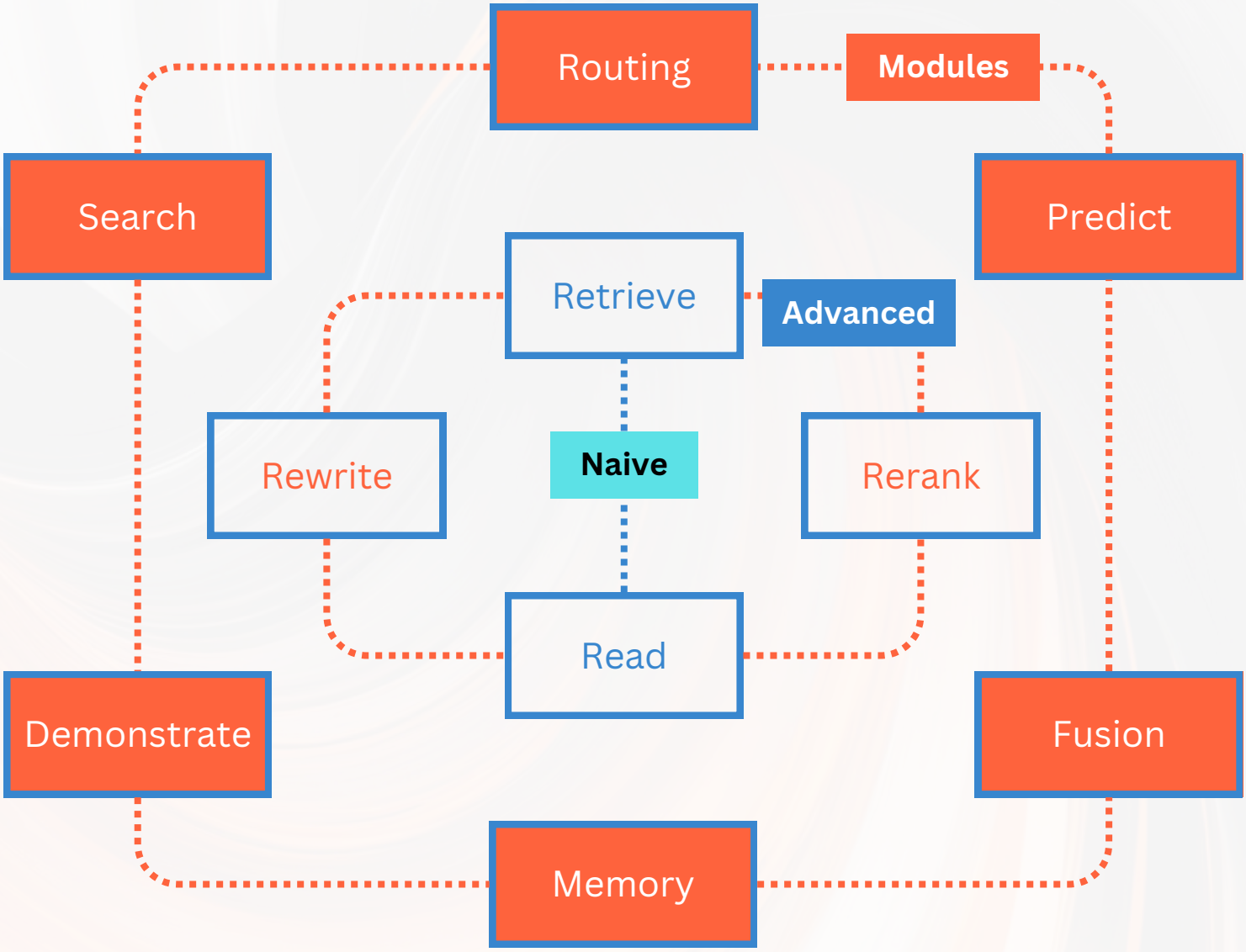
While the retriever is proficient in extracting relevant information from extensive knowledge bases, managing the vast amount of information within retrieval documents poses a challenge. The retrieved information is compressed to extract the most relevant points before passing it to the LLM.

Reranking

The re-ranking model plays a crucial role in optimizing the document set retrieved by the retriever. The main idea is to rearrange document records to prioritize the most relevant ones at the top, effectively managing the total number of documents. This not only resolves challenges related to context window expansion during retrieval but also improves efficiency and responsiveness.

Modular RAG

The SOTA in Retrieval Augmented Generation is a modular approach which allows components like search, memory, and reranking modules to be configured



Naive RAG is essentially a **Retrieve -> Read** approach which focusses on retrieving information and comprehending it.

Advanced RAG is adds to the **Retrieve -> Read** approach by adding it into a **Rewrite** and **Rerank** components to improve relevance and groundedness.

Modular RAG takes everything a notch ahead by providing **flexibility** and adding modules like **Search**, **Routing**, **etc**.

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

Some RAG Modules

Search

The search module is aimed at performing search on different data sources. It is customised to different data sources and aimed at increasing the source data for better response generation

Memory

This module leverages the parametric memory capabilities of the Language Model (LLM) to guide retrieval. The module may use a retrieval-enhanced generator to create an unbounded memory pool iteratively, combining the "original question" and "dual question." By employing a retrieval-enhanced generative model that improves itself using its own outputs, the text becomes more aligned with the data distribution during the reasoning process.

Fusion

RAG-Fusion improves traditional search systems by overcoming their limitations through a multi-query approach. It expands user queries into multiple diverse perspectives using a Language Model (LLM). This strategy goes beyond capturing explicit information and delves into uncovering deeper, transformative knowledge. The fusion process involves conducting parallel vector searches for both the original and expanded queries, intelligently re-ranking to optimize results, and pairing the best outcomes with new queries.

Extra Generation

Rather than directly fetching information from a data source, this module employs the Language Model (LLM) to generate the required context. The content produced by the LLM is more likely to contain pertinent information, addressing issues related to repetition and irrelevant details in the retrieved content.

Task Adaptable Module

This module makes RAG adaptable to various downstream tasks allowing the development of task-specific end-to-end retrievers with minimal examples, demonstrating flexibility in handling different tasks.

Other Blogs on RAG

Getting the Most from LLMs: Building a Knowledge Brain for Retrieval Augmented Generation

The advancements in the LLM space have been mind-boggling. However, when it comes to using LLMs in real scenarios, we...

medium.com

Evaluation of RAG Pipelines for more reliable LLM applications

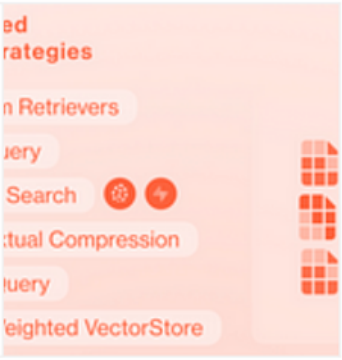
Building a PoC RAG pipeline is not overtly complex. LangChain and LlamaIndex have made it quite simple. Developing...

medium.com

RAG Value Chain: Retrieval Strategies in Information Augmentation for Large Language Models

Perhaps, the most critical step in the entire RAG value chain is searching and retrieving the relevant pieces of...

medium.com



Creating Impact: A Spotlight on 6 Practical Retrieval Augmented Generation Use Cases

In 2023, RAG has become one of the most used technique in the domain of Large Language Models. In fact, one can assume...

medium.com