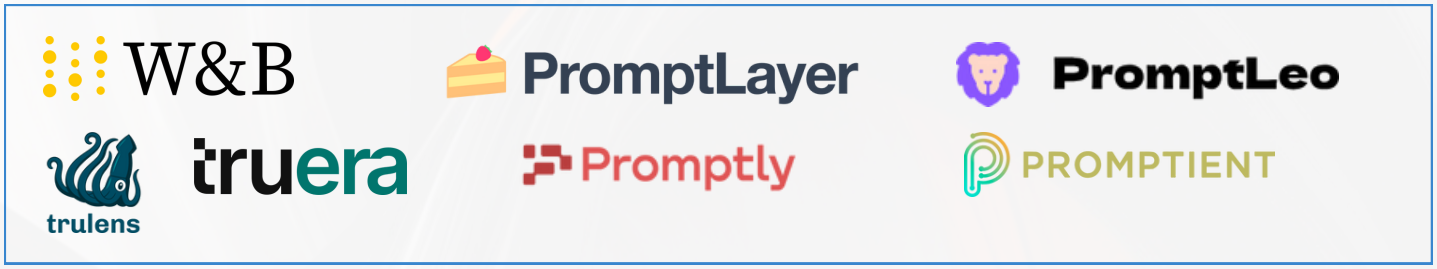
Prompt Layer

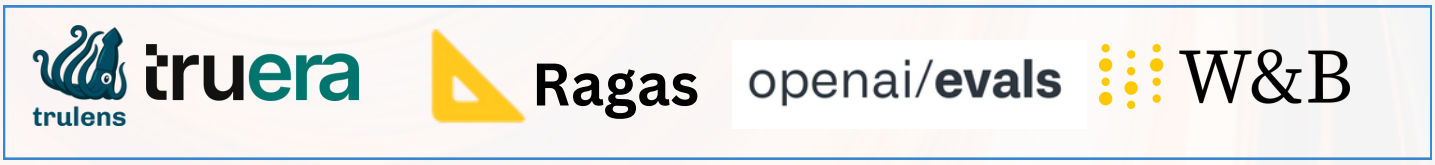
Prompt Engineering is more than writing questions in natural language. There are several prompting techniques and developers need to create prompts tailored to the use cases. This process often involves experimentation: the developer creates a prompt, observes the results and then iterates on the prompts to improve the effectiveness of the app. This requires tracking and collaboration



Popular prompt engineering platforms (Non Exhaustive)

Evaluation

It is easy to build a RAG pipeline but to get it ready for production involves robust evaluation of the performance of the pipeline. For checking hallucinations, relevance and accuracy there are several frameworks and tools that have come up.



Popular RAG evaluation frameworks and tools (Non Exhaustive)

App Orchestration

An RAG application involves interaction of multiple tools and services. To run the RAG pipeline, a solid orchestration framework is required that invokes these different processes.

Popular App orchestration frameworks (Non Exhaustive)

Deployment Layer

Deployment of the RAG application can be done on any of the available cloud providers and platforms. Some important factors to consider while deployment are also -

- Security and Governance
- Logging
- Inference costs and latency

Popular cloud providers and LLMOps platforms (Non Exhaustive)

Application Layer

The application finally needs to be hosted for the intended users or systems to interact with it. You can create your own application layer or use the available platforms.

Popular app hosting platforms (Non Exhaustive)

Monitoring

Deployed application needs to be continuously monitored for both accuracy and relevance as well as cost and latency.

Popular monitoring platforms (Non Exhaustive)

Other Considerations

LLM Cache - To reduce costs by saving responses for popular queries
LLM Guardrails - To add additional layer of scrutiny on generations

Multimodal RAG

Up until now, most AI models have been limited to a single modality (a single type of data like text or images or video). Recently, there has been significant progress in AI models being able to handle multiple modalities (majorly text and images). With the emergence of these Large Multimodal Models (LMMs) a multimodal RAG system becomes possible.

"Generate any type of output from any type of input providing any type of context"

The high-level features of multimodal RAG are -

1. Ability to **query/prompt in one or more modalities** like sending both text and image as input.
2. Ability to **search and retrieve not only text** but also images, tables, audio files related to the query
3. Ability to **generate text, image, video etc.** irrespective of the mode(s) in which the input is provided.

Approaches

Using MultiModal Embeddings

Using LMMs Only

Large MultiModel Models

Google DeepMind Flamingo

salesforce BLIP

Microsoft KOSMOS-1

Tencent Macaw-LLM

OpenAI GPT4

Google Gemini

LlaVA

LAVIN

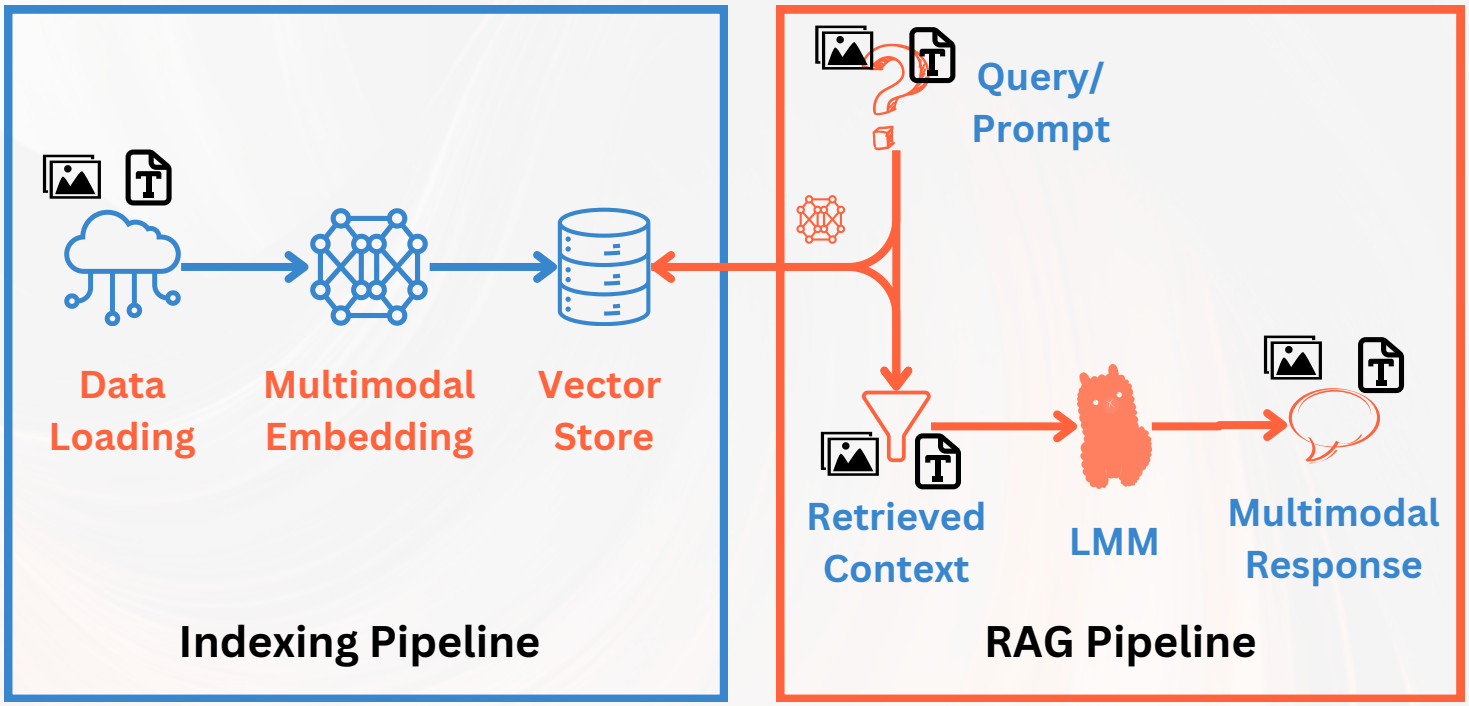
LLaMA - Adapter

FUYU

Multimodal RAG Approaches

Using MultiModal Embeddings

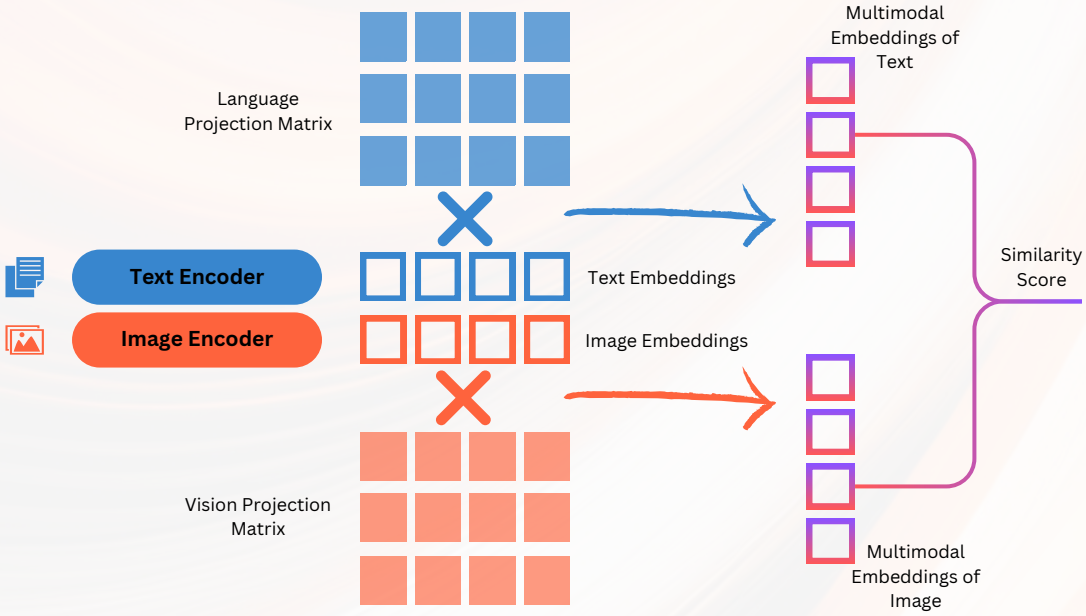
- Multimodal embeddings (like **CLIP**) are used to embed images and text
- User Query is used to retrieve context which can be image and/or text
- The image and/or text context is passed to an LMM with the prompt.
- The LMM generates the final response based on the prompt



Multimodal RAG using Multimodal Embeddings

CLIP : Contrastive Language-Image Pre-training

Mapping data of different modalities into a shared embedding space



CLIP is an example of training multimodal embeddings

OpenAI's CLIP (Contrastive Language-Image Pre-training), maps both images and text into the same semantic embedding space. This allows CLIP to "understand" the relationship between texts and images for powerful applications

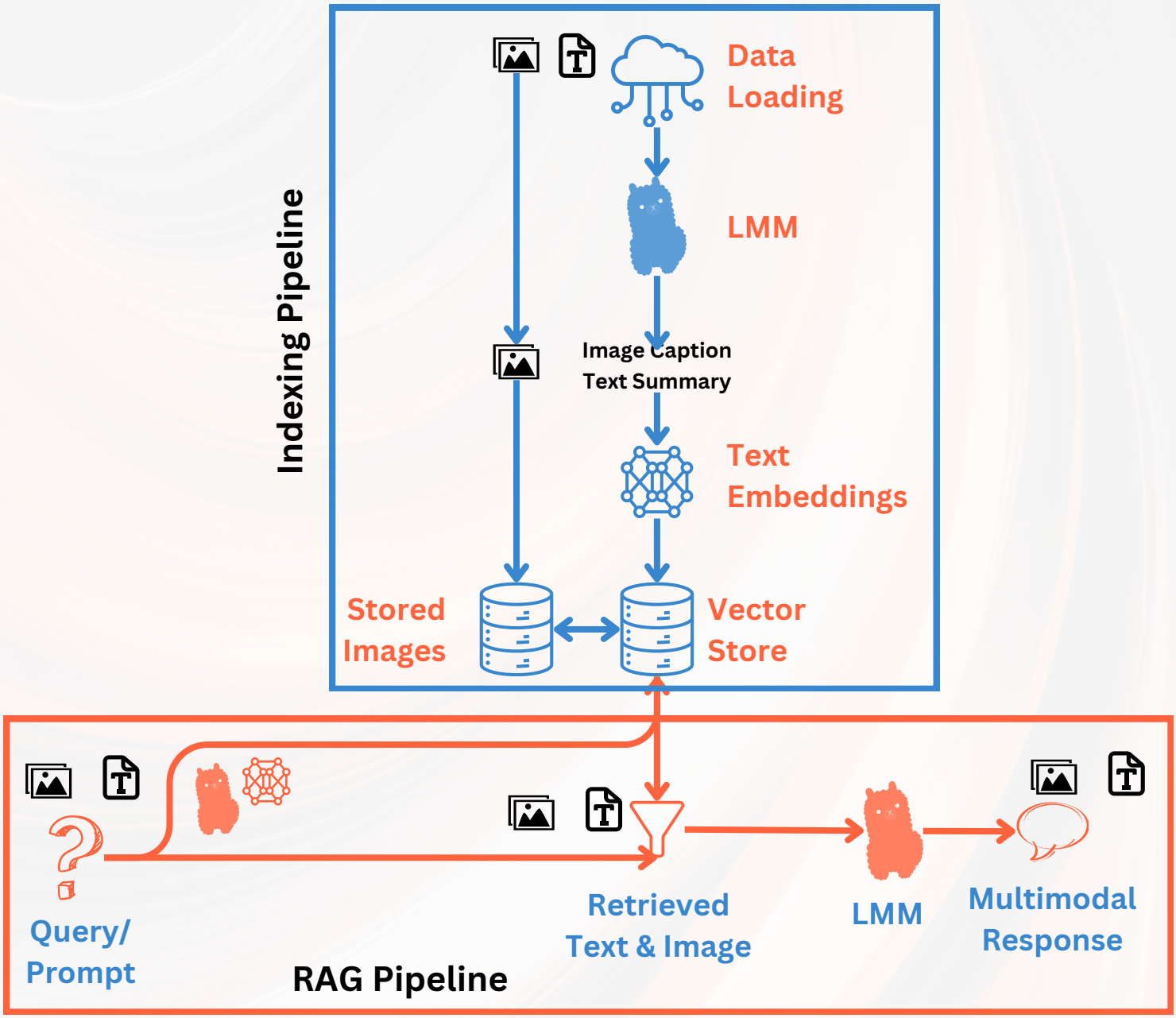
Using LMMs to produce text summaries from images

Indexing

- An LLM is used to generate captions for images in the data
- The image captions and text summaries are stored as text embeddings in a vector database
- A mapping is maintained from the image captions to the image files

Generation

- User enters a query (with text and image)
- Image captions are generated using an LLM and embeddings are generated
- Text summaries and image captions are searched. Images are retrieved based on the relevant image captions.
- Retrieved text summaries, captions and images are passed to the LMM with the prompt. The LMM generates a multimodal response



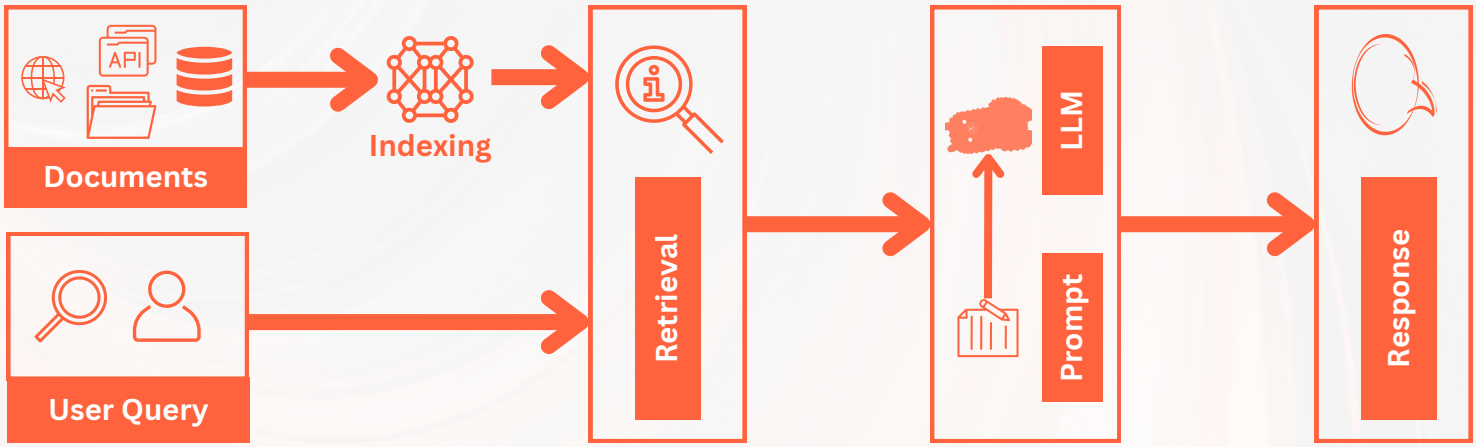
Progression of RAG Systems

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

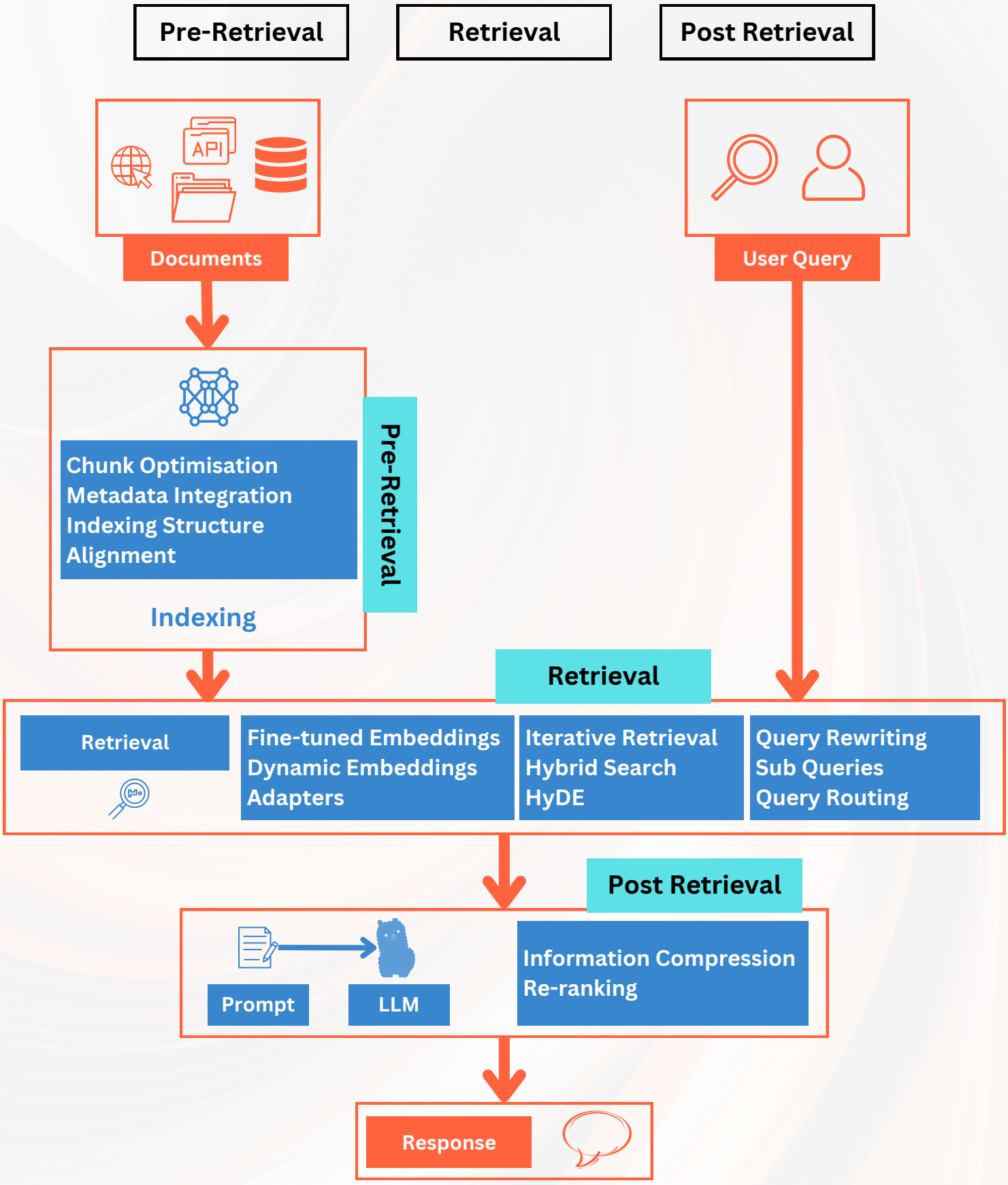
- **Redundancy and Repetition** when multiple retrieved documents have similar information
- **Context Length** challenges

Generation Quality

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

Advanced RAG

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



** Indicative, non-exhaustive list*

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

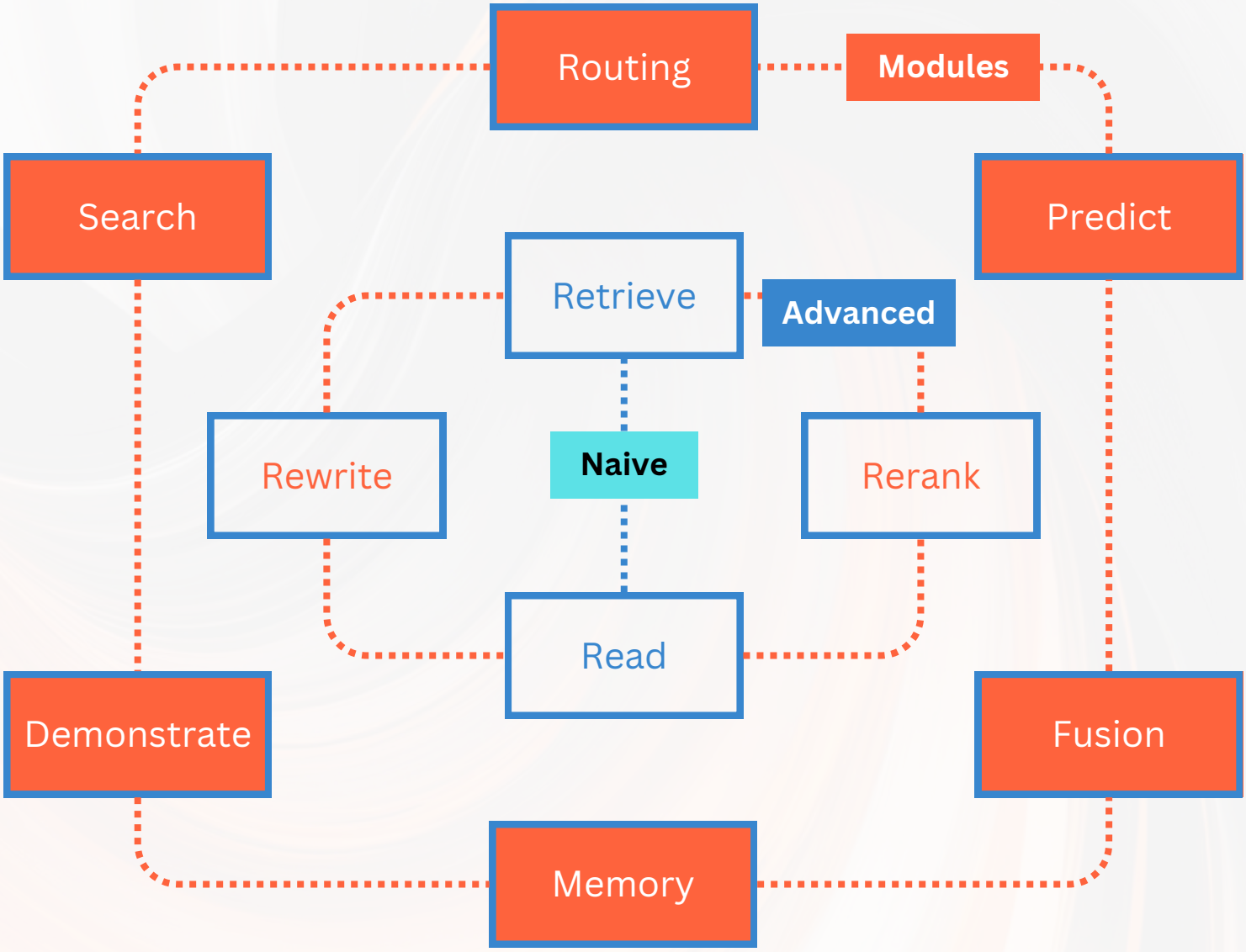
While the retriever is proficient in extracting relevant information from extensive knowledge bases, managing the vast amount of information within retrieval documents poses a challenge. The retrieved information is compressed to extract the most relevant points before passing it to the LLM.

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.

Acknowledgements

Retrieval Augmented Generation continues to be a pivotal approach for any Generative AI led application and it is only going to grow. There are several individuals and organisations that have provided learning resources and made understanding RAG fun.

I'd like to thank -

- My team at **Yarnit.app** for taking a bet on RAG and helping me explore and execute RAG pipelines for content generation
- **Andrew Ng** and the good folks at **deeplearning.ai** for their short courses allowing everyone access to generative AI
- **OpenAI** and **HuggingFace** for all that they do
- **Harrison Chase** and all the folks at **LangChain** for not only building the framework but also making it easy to execute
- **Jerry Liu** and others at **LlamaIndex** for their perspectives and tutorials on RAG
- **TruEra** for demystifying observability and the tech stack for LLMOps
- **PineCone** for their amazing documentation and the learning center
- The team at **Exploding Gradients** for creating **Ragas** and explaining RAG evaluation in detail
- **TruLens** for their triad of RAG evaluations
- **Aman Chadha** for his curation of all thing AI, ML and Data Science
- Above all, to my **colleagues and friends**, who endeavour to learn, discover and apply technology everyday in their effort to make the world a better place.

With lots of love,

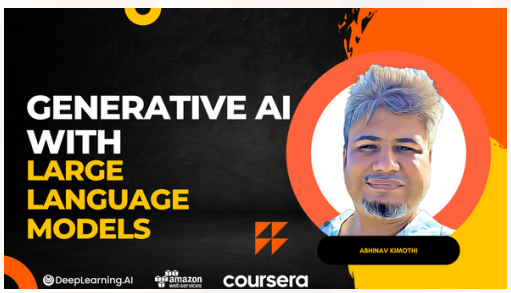
Abhinav

I talk about :

#AI #MachineLearning #DataScience #GenerativeAI #Analytics #LLMs #Technology #RAG #EthicalAI

If you like'd what you read

Detailed Notes from **Generative AI with Large Language Models** Course by **Deeplearning.ai** and **AWS**.



let's connect...

Buy me a coffee ... please

DOWNLOAD FREE EBOOK

Resources

Official Documentations

LangChain
Python Documentation

LlamaIndex
Python Documentation

Pinecone
Learning Center

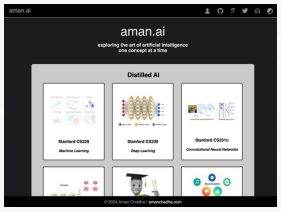
trulens
Documentation

Ragas
Documentation

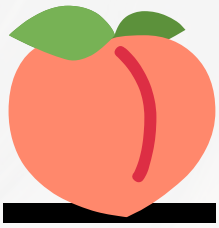
Hugging Face
Documentation

OpenAI
Documentation

Thought Leaders and Influencers



Aman Chadha's Blog



Lillian Weng's Log

Leonie Monigatti's Blogroll

Chip Huyen Blogs

Research Papers

Retrieval-Augmented Generation for Large Language Models: A Survey (Gao, et al, 2023)

Retrieval-Augmented Multimodal Language Modeling (Yasunaga, et al, 2023)

KG-Augmented Language Models for Knowledge-Grounded Dialogue (Kang, et al, 2023)

Learning Resources and Tutorials

DeepLearning.AI
Short 1-hour Courses

LangChain
Python Cookbook

LlamaIndex
Tutorials & Webinars

Hello!
I'm Abhinav...
A data science and AI professional with over 15 years in the industry. Passionate about AI advancements, I constantly explore emerging technologies to push the boundaries and create positive impacts in the world. Let's build the future, together!

Please share your feedback on these notes with me

LinkedIn Github Medium Insta email X Linktree Gumroad

Talk to me

Checkout Yarnit Magic

Newsletter

$$ Contribute $$

Buy me a coffee

Subscribe

Don't miss a post!
Get an email notification whenever I publish!

Follow on LinkedIn