# KEY EXCERPTS FROM


SCAN CODE OR CLICK HERE

## A SIMPLE GUIDE TO

# Retrieval Augmented Generation

Abhinav Kimothi

50% OFF

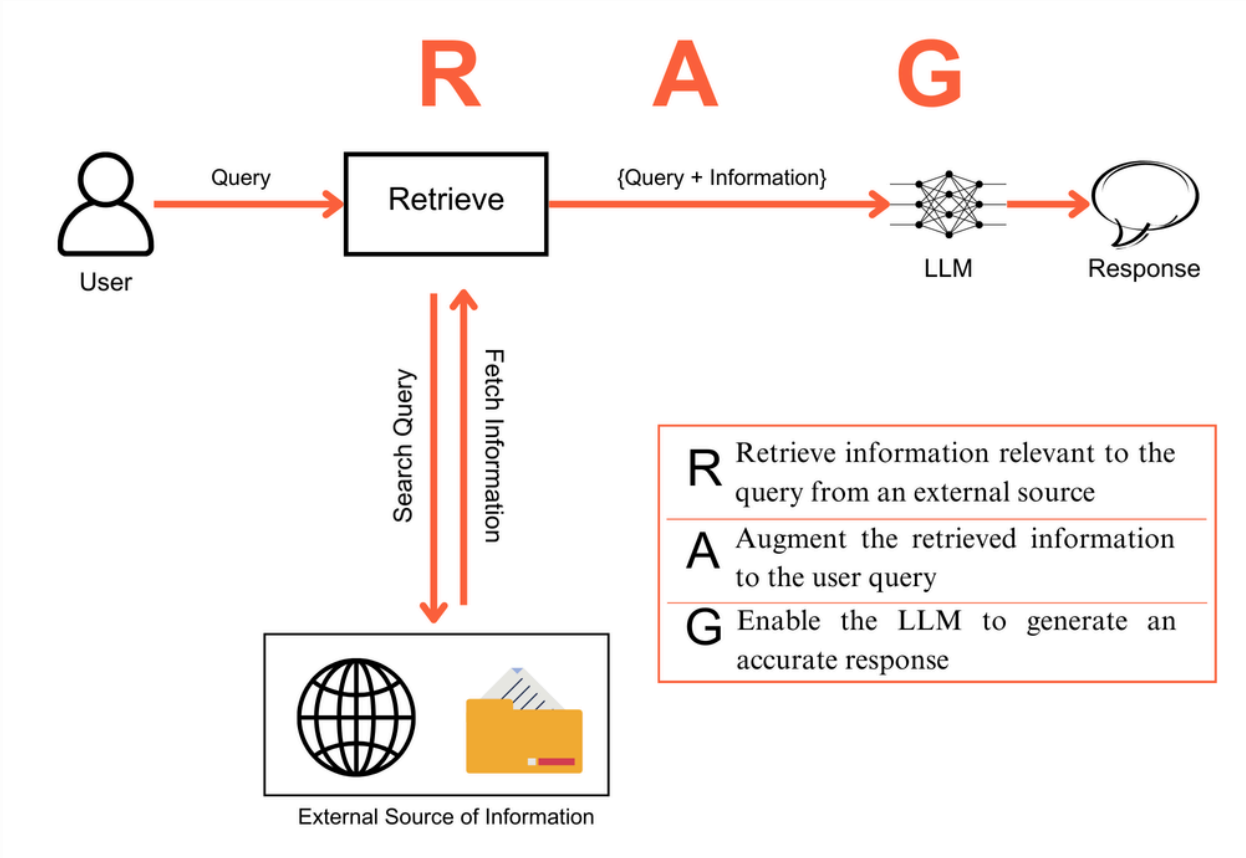

MANNING

# Table of Contents

# Ch 1 : LLMs and the Need for Retrieval Augmented Generation

This chapter covers

- Understanding the limits of LLMs & the need for RAG
- Defining Retrieval Augmented Generation (RAG)
- Popular use cases of RAG

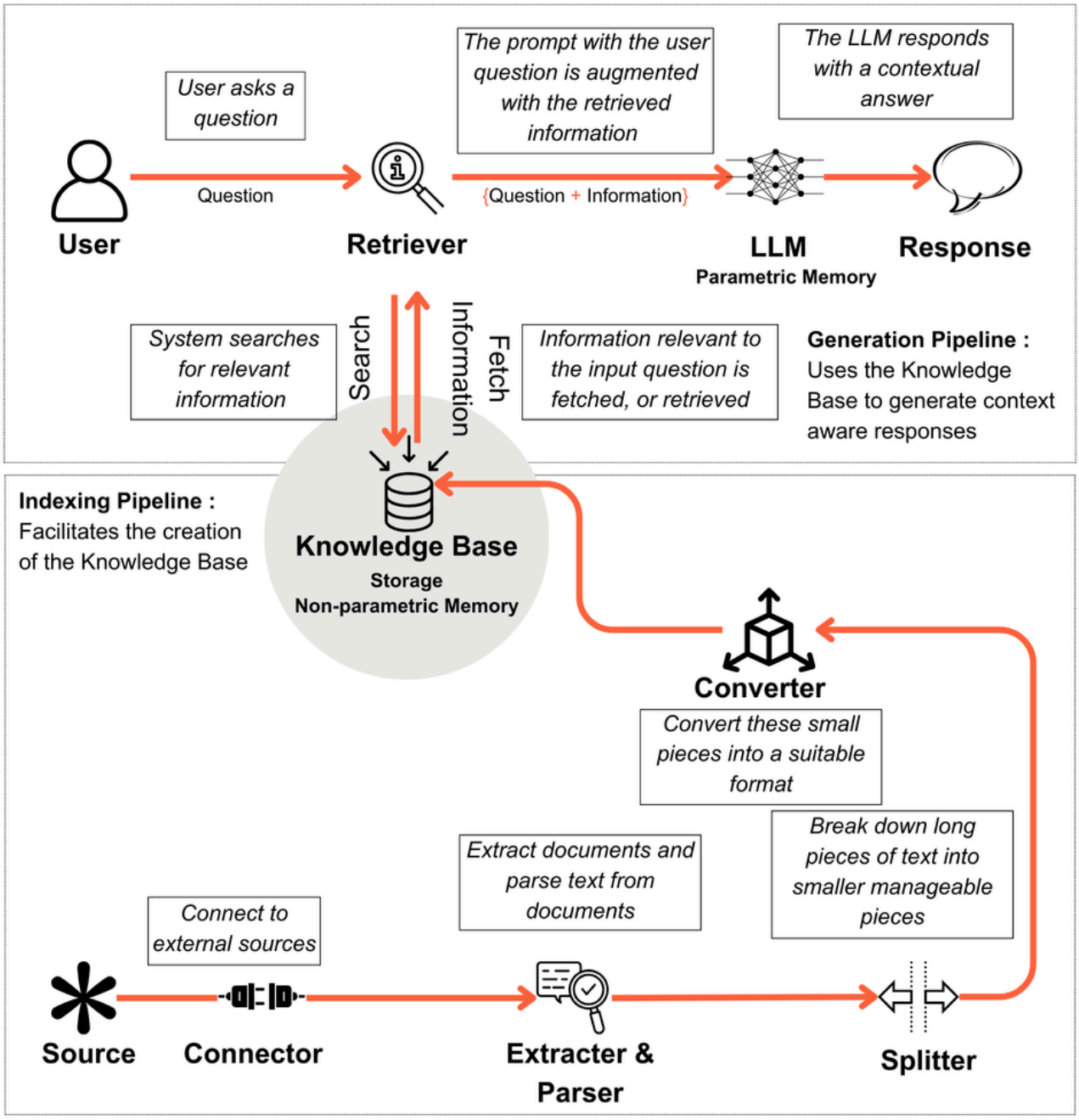



**Retrieval Augmented Generation** is the methodological approach of enhancing the parametric memory of an LLM by creating access to an explicit non-parametric memory, from which a retriever can fetch relevant information, augment that information to the prompt, pass the prompt to an LLM to enable the LLM to generate a response that is *contextual, reliable, and factually accurate*.

# Ch 2 : RAG Systems and Their Design

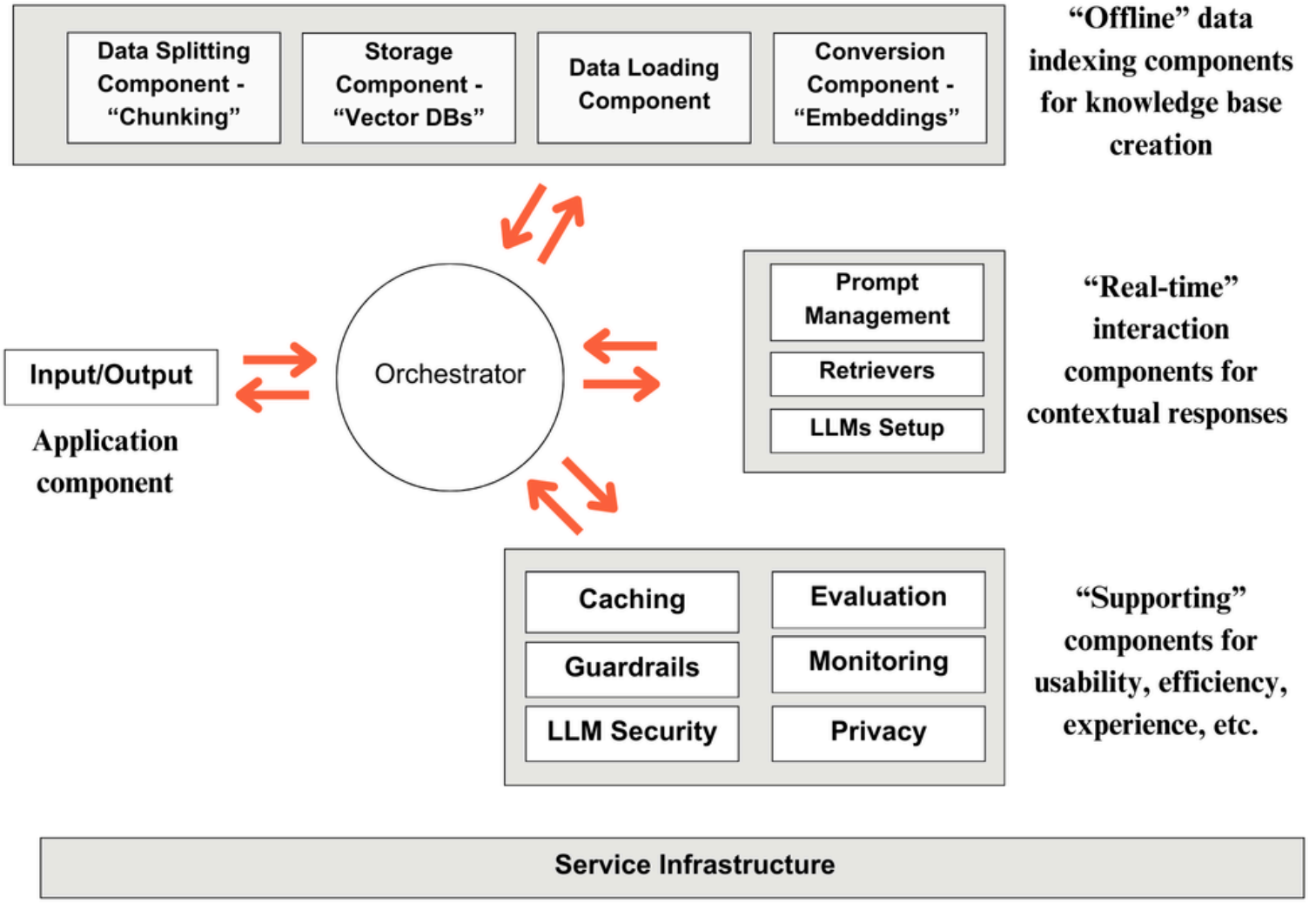
This chapter covers

- An exploration of the concept & design of RAG systems
- An Overview of the Indexing Pipeline, Generation Pipeline & their evaluation
- A high-level look at RAG Operations stack




CORE RAG PIPELINES IN 10 STEPS

# Ch 2 : RAG Systems and Their Design




## EVOLVING RAGOPS STACK

## INDEXING PIPELINE COMPONENTS

**Data loading component** is responsible for connecting to external sources, and extracting and parsing information

**Data splitting component** is responsible breaking down long pieces of text into smaller, manageable parts called "Chunks"

**Data conversion component** is responsible converting text "chunks" into numerical vectors called "embeddings"

**Data storage component** stores the "embeddings" in permanent memory using specialised databases called "Vector DBs"

## GENERATION PIPELINE COMPONENTS

**Retriever** is the main component of RAG, responsible for searching through the knowledge base and fetching the relevant information

**Prompt Management** layer is responsible for augmenting the retrieved information and constructing the final prompt for generation

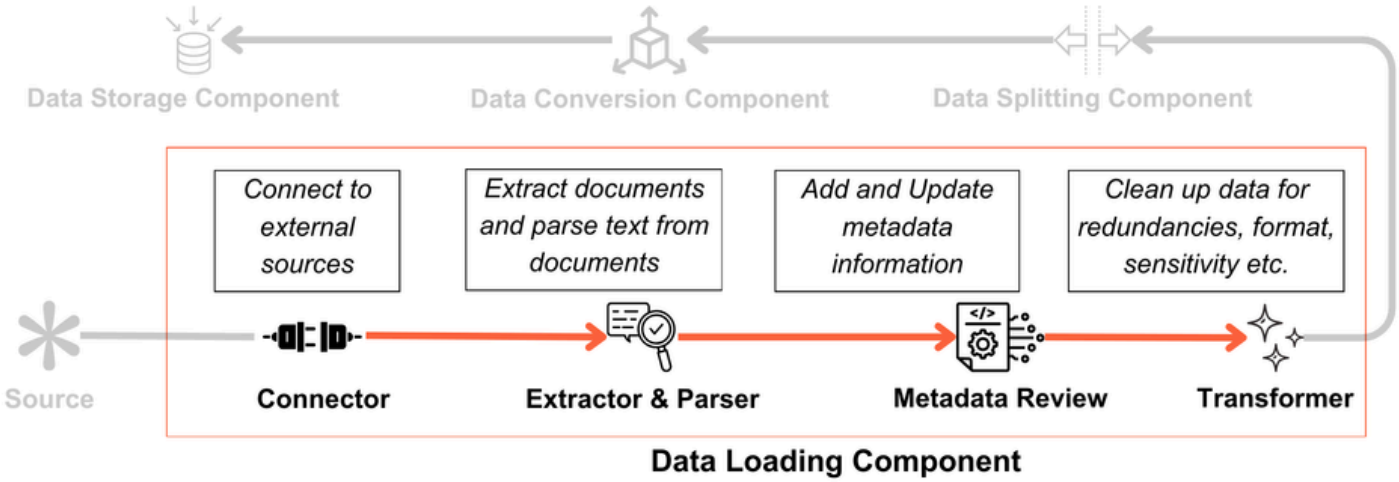
**LLM component** is responsible for generating the final response. It can comprise of several foundation or fine-tuned models that are either open or closed source

Retrieval → Augmentation → Generation

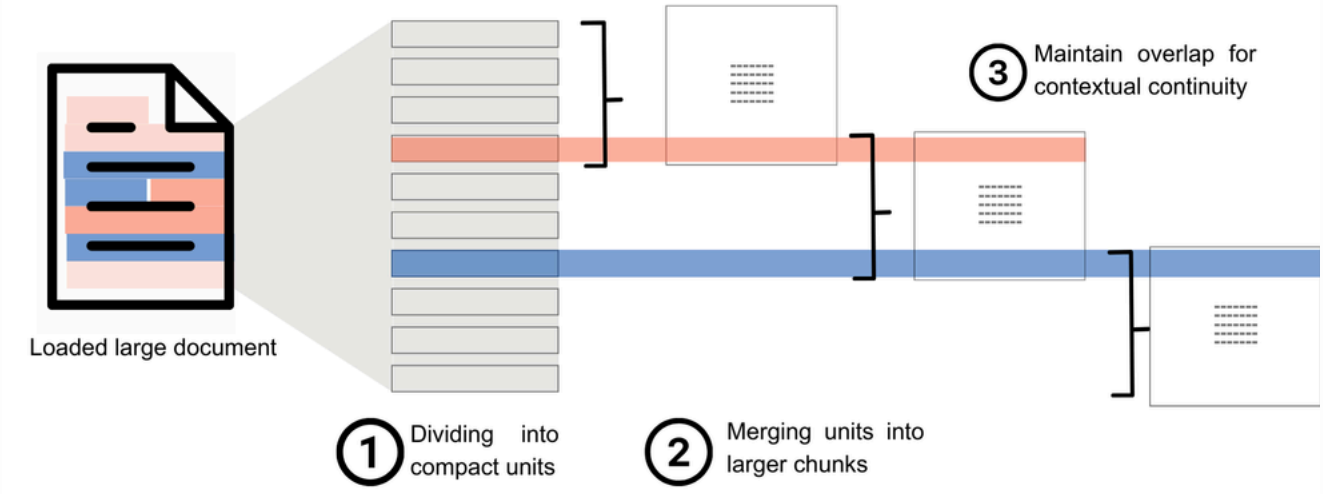
# Ch 3 : Indexing Pipeline: Creating a Knowledge Base for RAG

This chapter covers

- The four components of the Indexing Pipeline
  - Data Loading
  - Text Splitting or Chunking
  - Converting Text to Embeddings
  - Storing Embeddings in Vector Databases
- Examples in Python using LangChain.

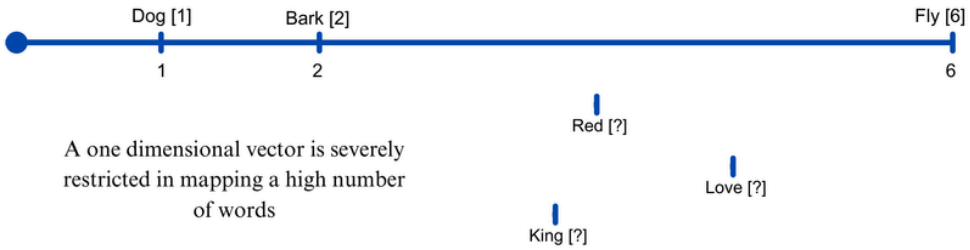



## DATA LOADING

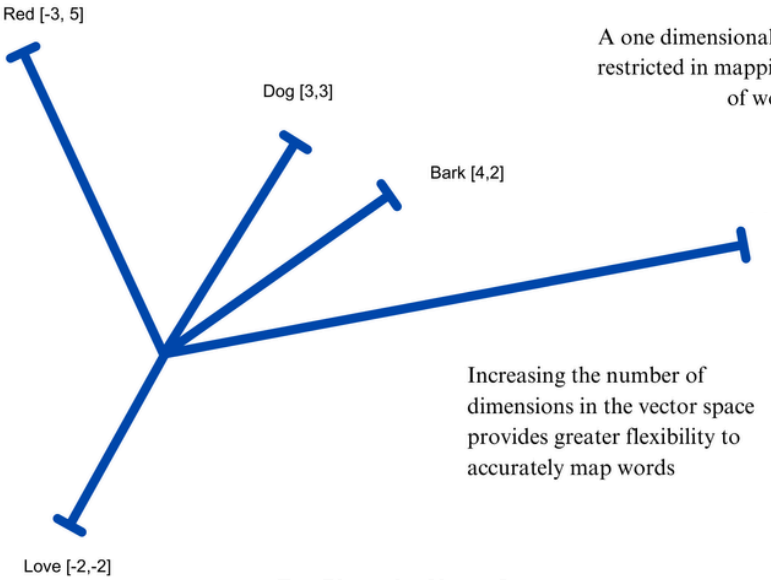



## SMALL TO BIG TEXT CHUNKING

# Ch 3 : Indexing Pipeline: Creating a Knowledge Base for RAG




A one dimensional vector is severely restricted in mapping a high number of words

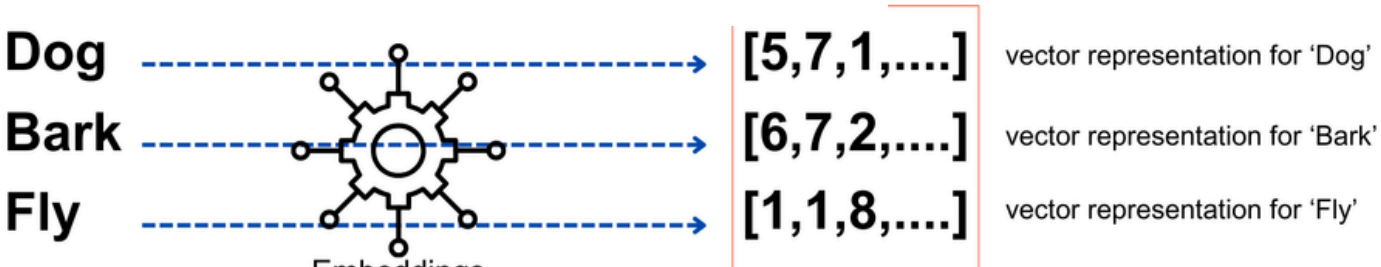



Increasing the number of dimensions in the vector space provides greater flexibility to accurately map words

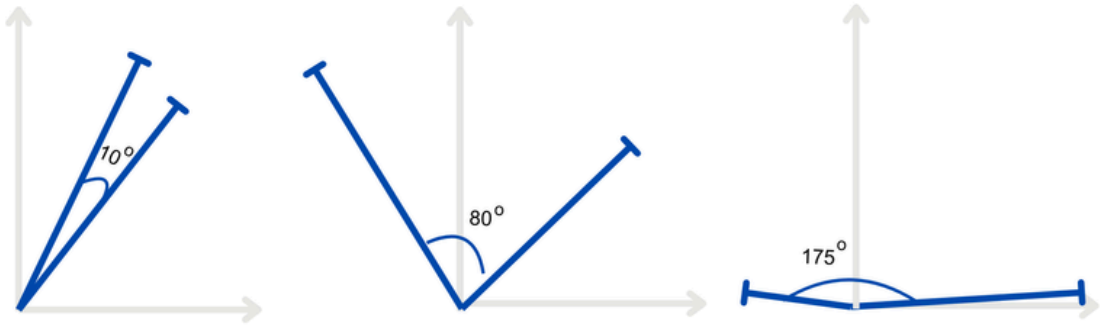
**Two Dimension Vector Space**

## EMBEDDINGS

embeddings help in establishing semantic relationship Amongst unstructured Data




The goal of an embedding model is to convert words (or sentences/paragraphs) into n-dimensional vectors




Cos 10 =0.985
Close to 1
**Very Similar**

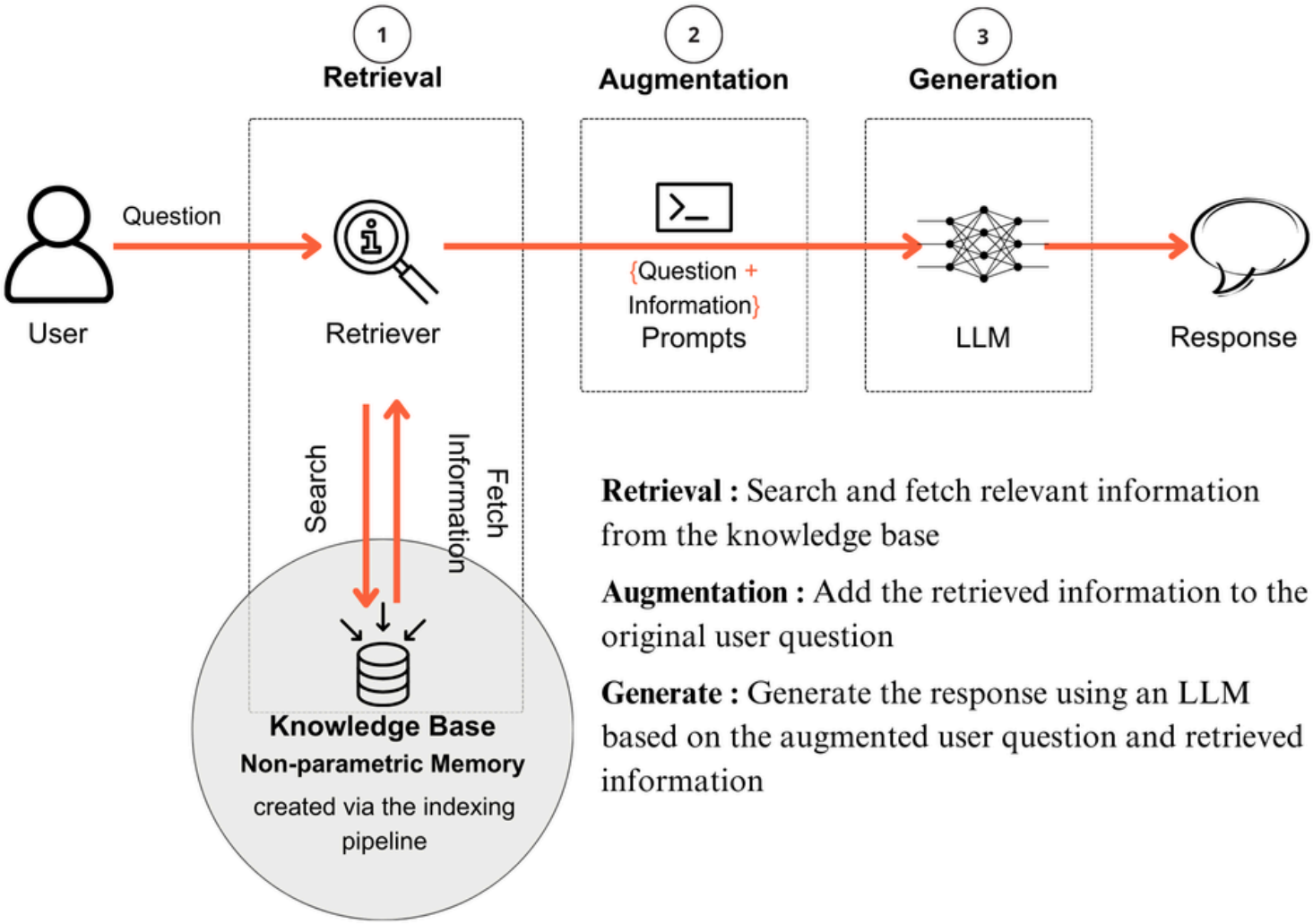
Cos 80 =0.173
Close to 0
**Unrelated**

Cos 175 =-0.996
Close to -1
**Opposite**

# Ch 4 : Generation Pipeline: Generating Contextual Responses

This chapter covers

- Retrievers and Retrieval Methodologies
- Augmentation with Prompt Engineering Techniques
- Generation using Large Language Models
- Basic implementation of the RAG pipeline in Python




## COMPONENTS OF GENERATION PIPELINE

# Ch 4 : Generation Pipeline: Generating Contextual Responses

## Components of TF-IDF

### Term Frequency (TF)

Measures how frequently a term 't' appears in a document 'd'

$$TF(t,d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}}$$

### Inverse Document Frequency (IDF)

Measures how important a term 't' is within the entire corpus 'D'

$$IDF(t,D) = \log\left(\frac{\text{Total number of documents 'D'}}{\text{Number of documents containing term 't'}}\right)$$

### TF-IDF

Product of TF & IDF

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times IDF(t,D)$$

### Documents (D)

d1 = Australia won the Cricket World Cup 2023
d2 = India and Australia played in the finals
d3 = Australia won the sixth time & last won in 2015

### Search Term

"won"

TF ("won",d1)=1/7 = **0.14**
TF ("won",d2)=0/7 = **0**
TF ("won, d3)= 2/10 = **0.2**

IDF ("won", D) = log (3/2) = **0.176**

TF - IDF ("won",d1,D)= 0.14 x 0.176 = **0.025**
TF - IDF ("won",d2,D)= 0 x 0.176 = **0**
TF - IDF ("won, d3,D)= 0.2 x 0.176 = **0.035**

**Result - d3 > d1 > d2**

## Calculating BM25

$$BM25(t,d,D) = IDF(t,D) \times \frac{TF(t,d) \times (k+1)}{TF(t,d) + (k) \times \left(1-b+b \times \frac{|d|}{avgdl}\right)}$$

- TF(t,d) is the term frequency of term '$t$' in document 'd'
- IDF(t,D) is the inverse document frequency of term in the corpus
- |d| is the length of the document
- avgdl is the average document length in the entire corpus.
- $k$ and b are free parameters

### Documents (D)

d1 = Australia won the Cricket World Cup 2023
d2 = India and Australia played in the finals
d3 = Australia won the sixth time & last won in 2015

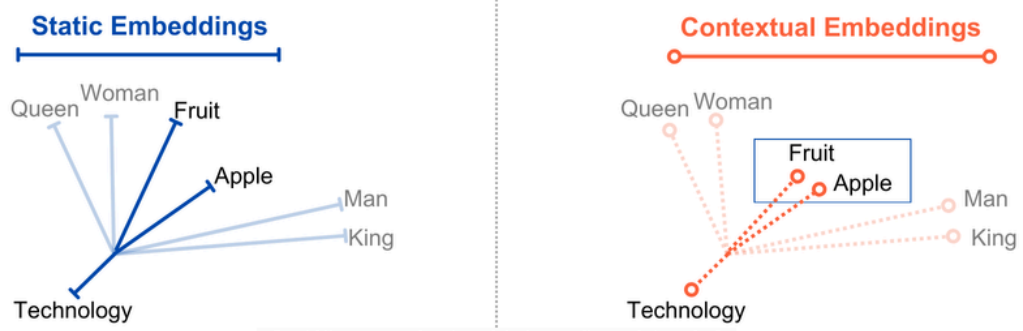
BM25("won", d1, D) = **0.193**
BM25("won", d2, D) = **0**
BM25("won", d2, D) = **0.168**

**Result - d1 > d3 > d2**

**TF-IDF AND BM25 ARE EXAMPLES OF SPARSE RETRIEVERS**

# Ch 4 : Generation Pipeline: Generating Contextual Responses

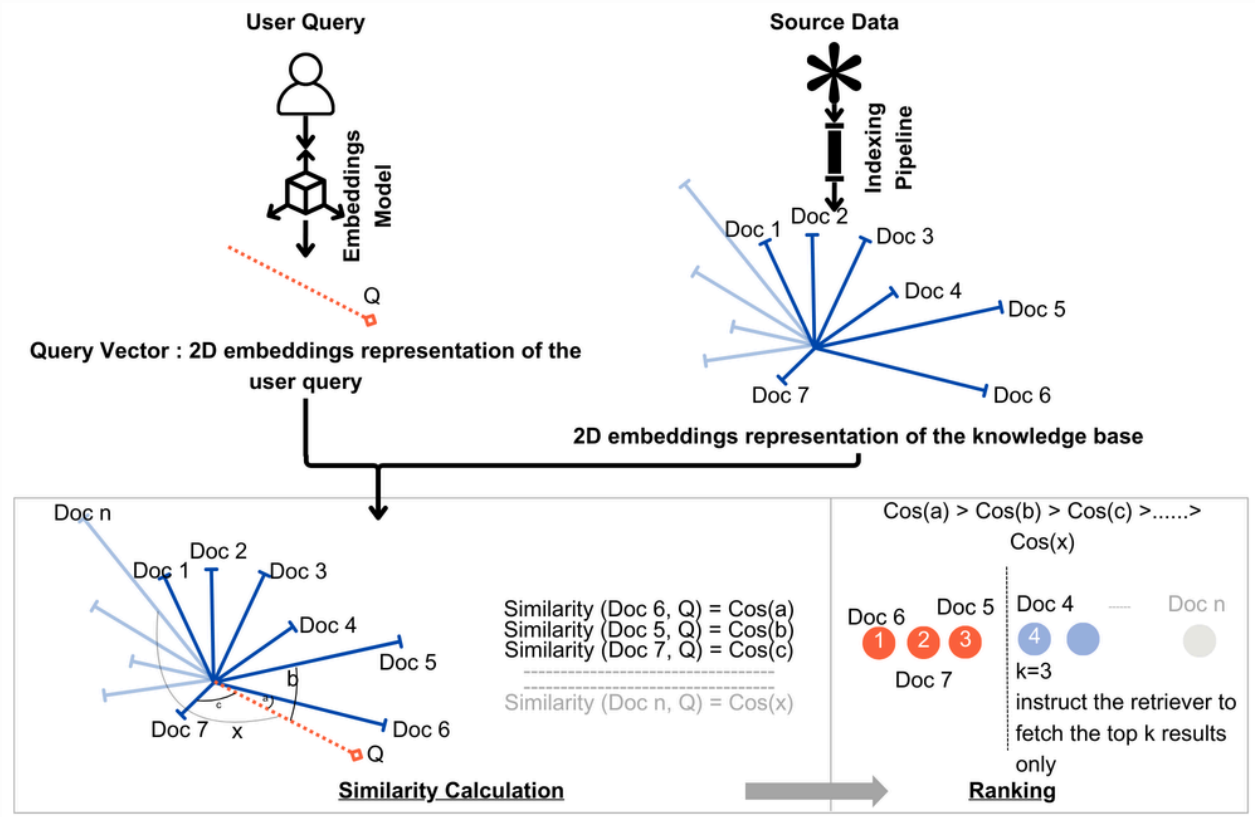
## STATIC VS CONTEXTUAL EMBEDDINGS




- Vectors do not change with the input query
- Computationally cheaper but do not work well for words that have multiple meanings

- Vectors calculated dynamically basis the input query
- Capture the context very well but are computationally intensive

## COSINE SIMILARITY BASED DOCUMENT RETRIEVAL



# THESE EXCERPTS ARE FROM

# A SIMPLE GUIDE TO RETRIEVAL AUGMENTED GENERATION



## SCAN CODE OR CLICK HERE TO GET AN EARLY ACCESS COPY

# INTERESTED IN CODING RAG PIPELINES?

## THE SOURCE CODE OF

### A SIMPLE GUIDE TO

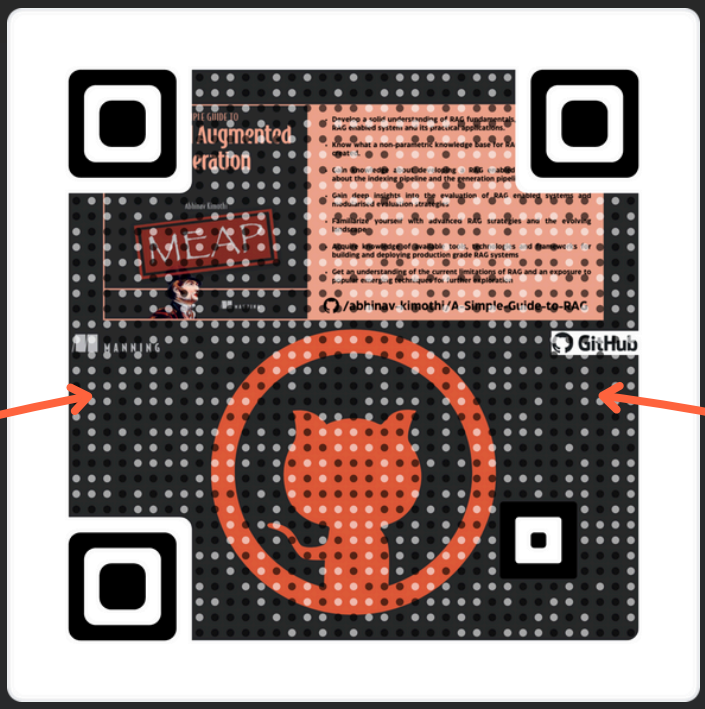
# RETRIEVAL AUGMENTED GENERATION

## IS NOW AVALIABLE FOR FREE PUBLIC ACCESS


180+ stars

MANNING

GitHub

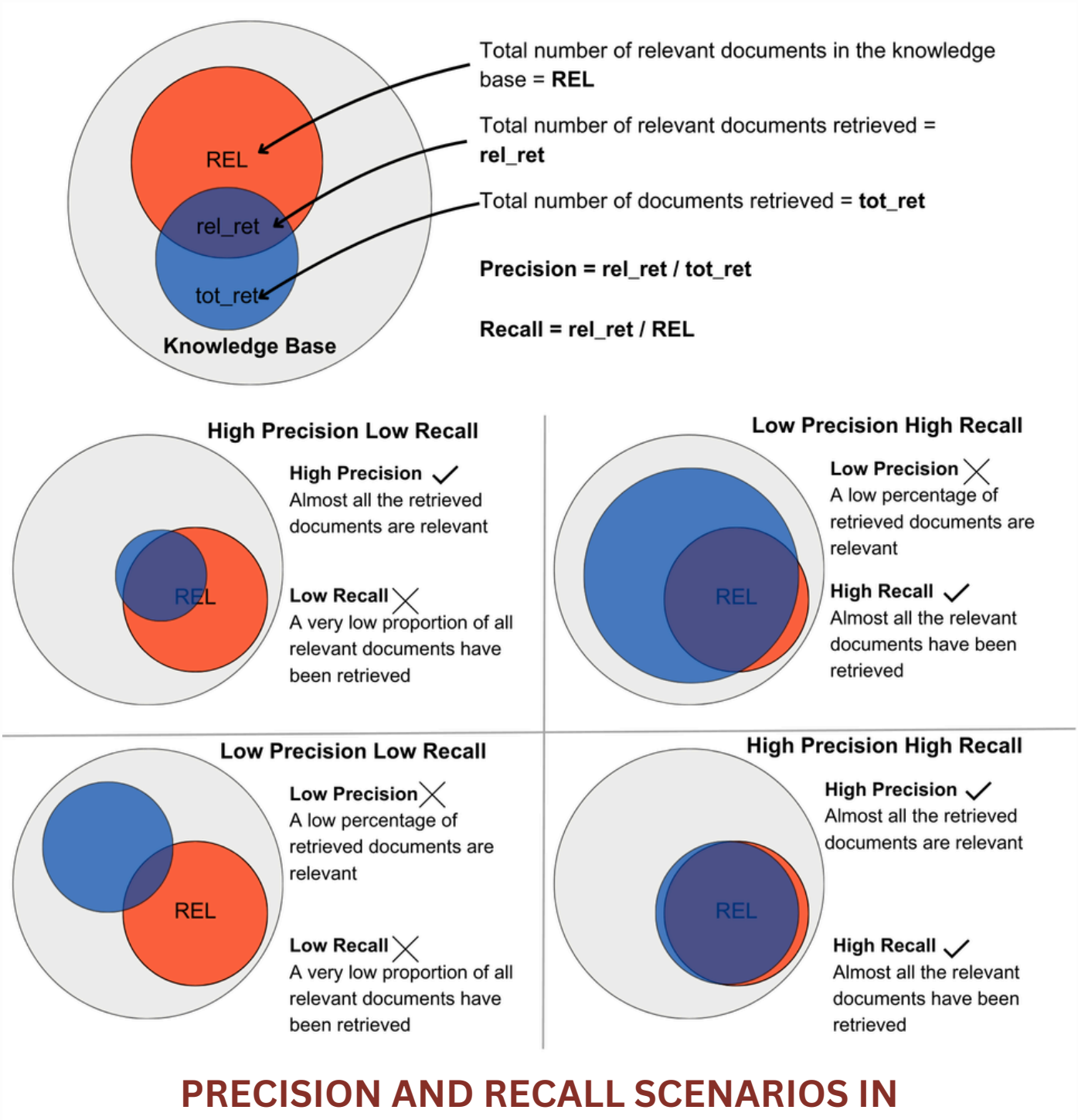


SCAN CODE OR CLICK HERE TO VIEW SOURCE CODE

# Ch 5 : RAG Evaluation: Accuracy, Relevance, Faithfulness

This chapter covers

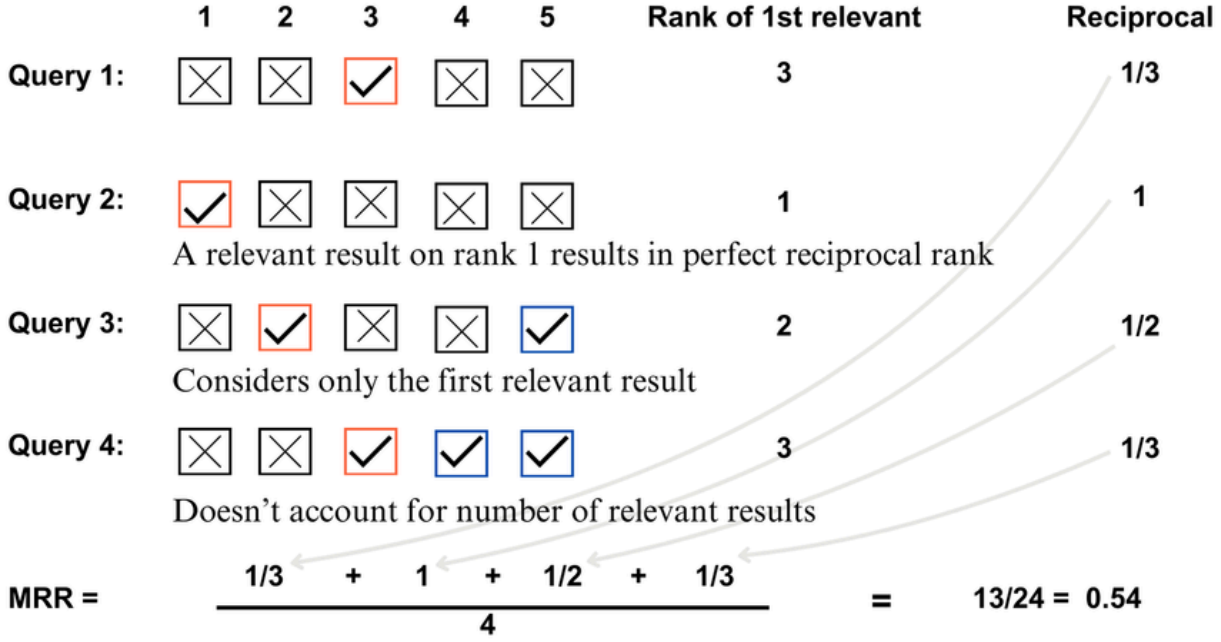
- The need and requirements for evaluating RAG pipelines
- Metrics, Frameworks, and Benchmarks for RAG Evaluation
- Current limitations and future course of RAG evaluation



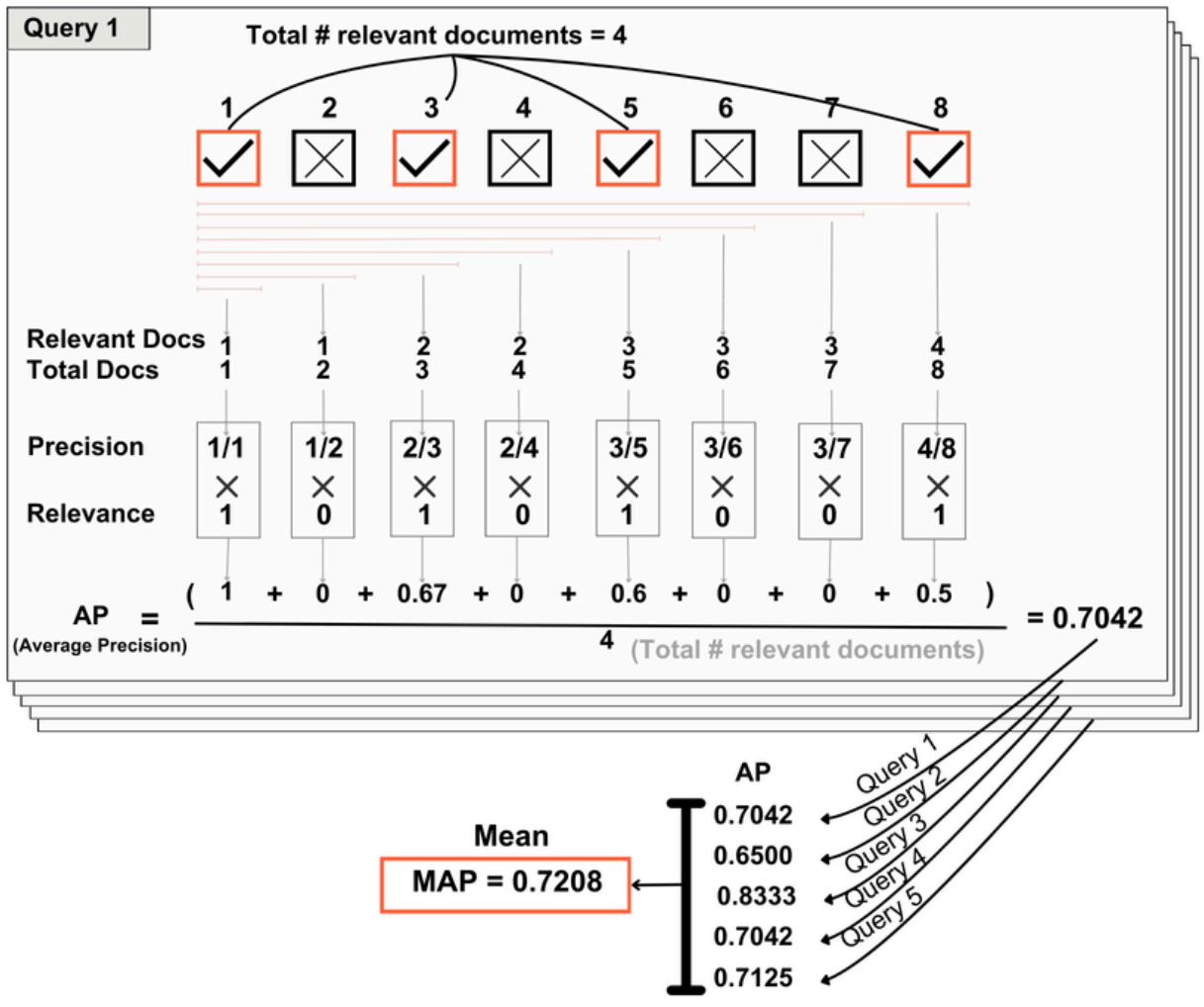

PRECISION AND RECALL SCENARIOS IN DOCUMENT RETRIEVAL

# Ch 5 : RAG Evaluation: Accuracy, Relevance, Faithfulness

## MEAN RECIPROCAL RANK




## MEAN AVERAGE PRECISION



# Ch 5 : RAG Evaluation: Accuracy, Relevance, Faithfulness

**Query** : Who won the 2023 ODI Cricket World Cup and when?

**Context 1 : High Context Relevance**

*The 2023 Cricket World Cup, concluded on 19 November 2023, with Australia winning the tournament. The tournament took place in ten different stadiums, in ten cities across the country.*

**Total sentences = 2**

**Relevant sentences = 1**

**Context Relevance = 0.5 or 50%**

**Context 2 : Low Context Relevance**

*The 2023 Cricket World Cup was the 13th edition of the Cricket World Cup. It was the first Cricket World Cup which India hosted solely. The tournament took place in ten different stadiums.In the first semi-final India beat New Zealand, and in the second semi-final Australia beat South Africa.*

**Total sentences = 4**

**Relevant sentences = 0**

**Context Relevance = 0**

## CONTEXT RELEVANCE

**Query** : Who won the 2023 ODI Cricket World Cup and when?

**Context** : The 2023 ODI Cricket World Cup concluded on 19 November 2023, with Australia winning the tournament.

**Response 1 : High Faithfulness**

*[Australia] won on [19 November 2023]*

**Number of claims generated = 2**

**Number of claims in context = 2**

**Answer Faithfulness = 1 or 100%**

**Response 2 : Low Faithfulness**

*[Australia] won on [15 October 2023] by [defeating India]*

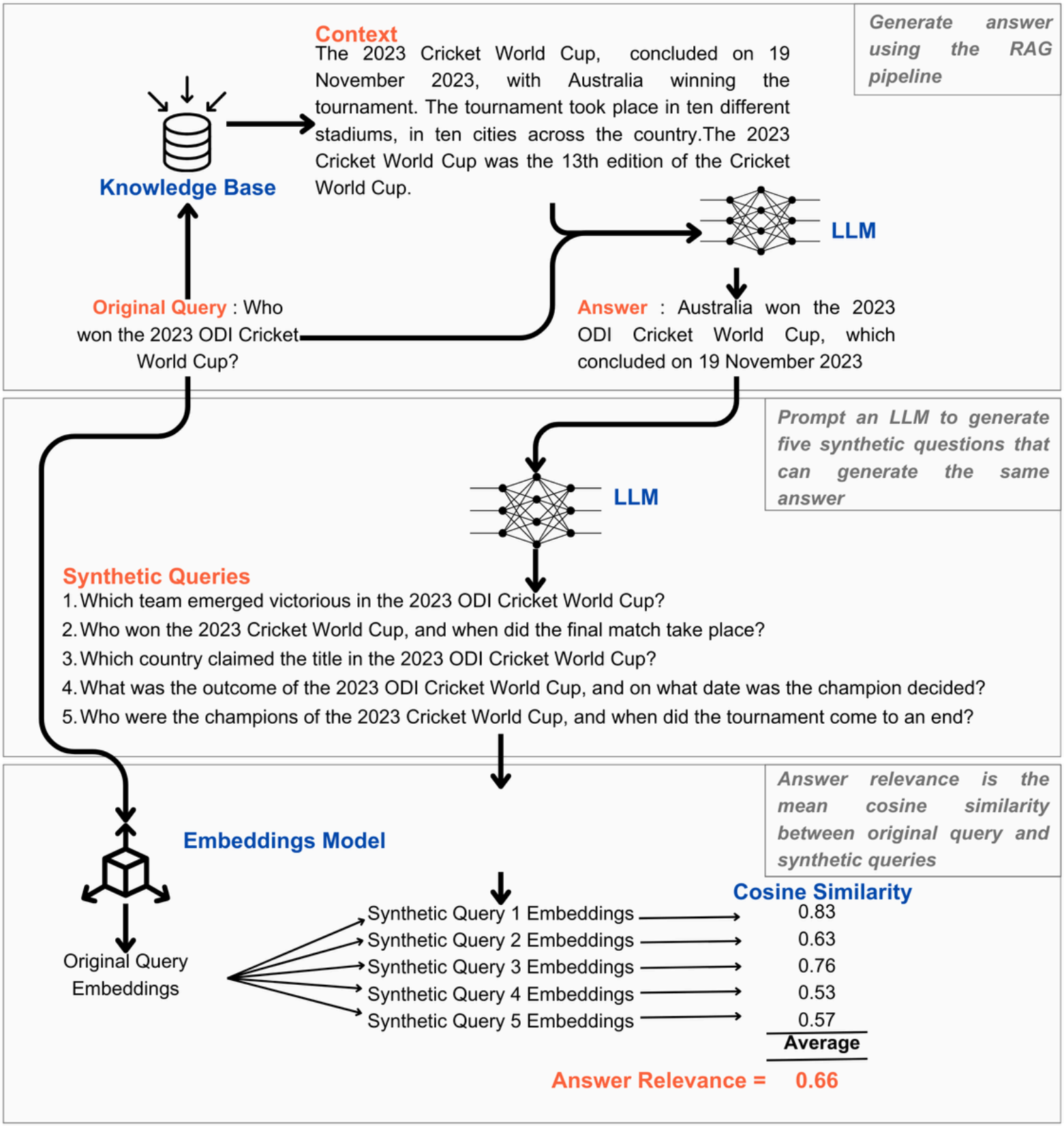
**Number of claims generated = 3**

**Number of claims in context = 1**

**Answer Faithfulness = 0.33 or 33%**

## FAITHFULNESS OR GROUNDEDNESS

# Ch 5 : RAG Evaluation: Accuracy, Relevance, Faithfulness

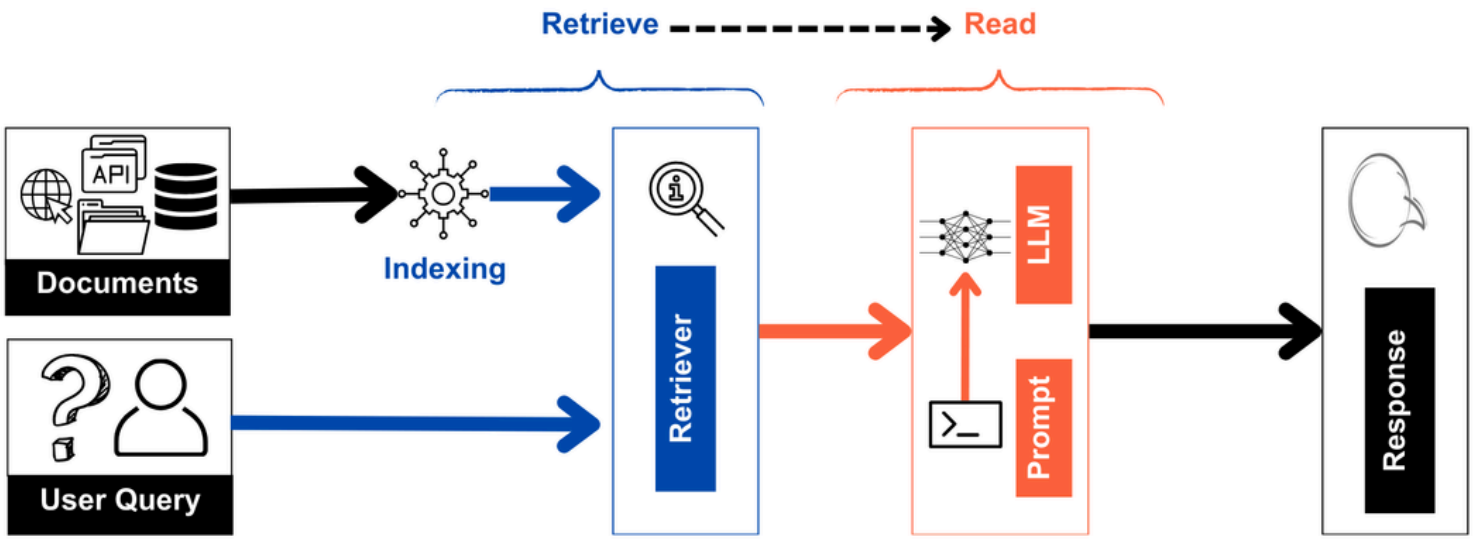


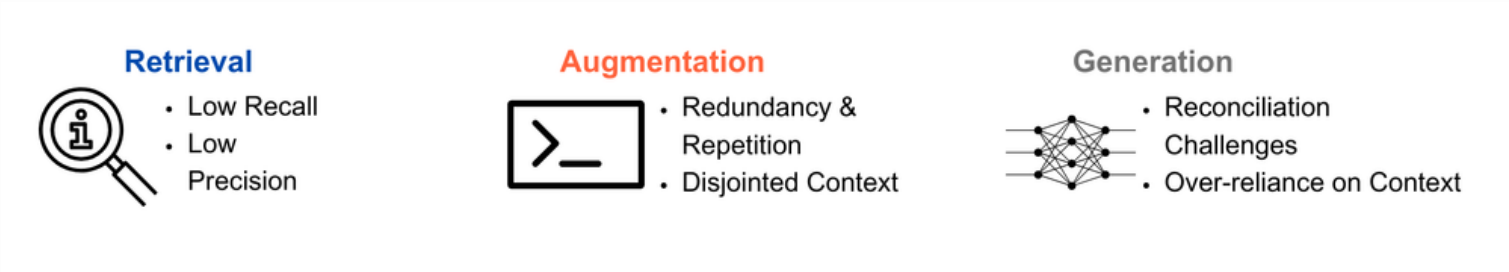

## ANSWER RELEVANCE

# Ch 6 : RAG Systemd Progression: Naïve, Advanced, & Modular RAG

This chapter covers

- Limitations of the Naïve RAG approach
- Advanced RAG strategies and techniques
- Modular patterns in RAG

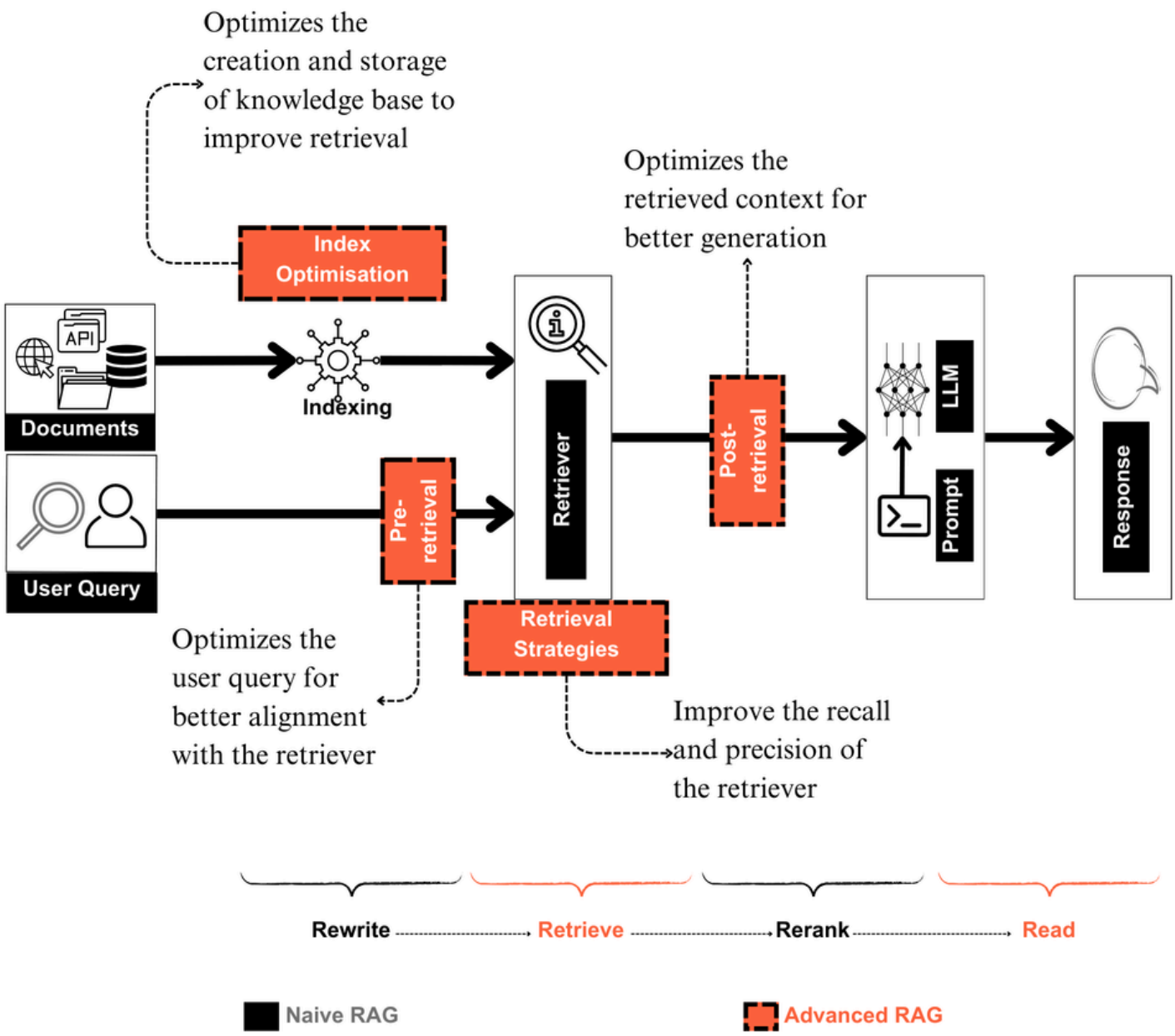



## LIMITATIONS OF NAIVE RAG

The Naïve RAG approach falls in a “retrieve then read” framework - which means that there’s a retriever that is retrieving information and then there’s an LLM that is reading this information to generate the results. This process is marred with drawbacks at each of the three stages of retrieval, augmentation and generation

# Ch 6 : RAG Systemd Progression: Naïve, Advanced, & Modular RAG




## ADVANCED RAG

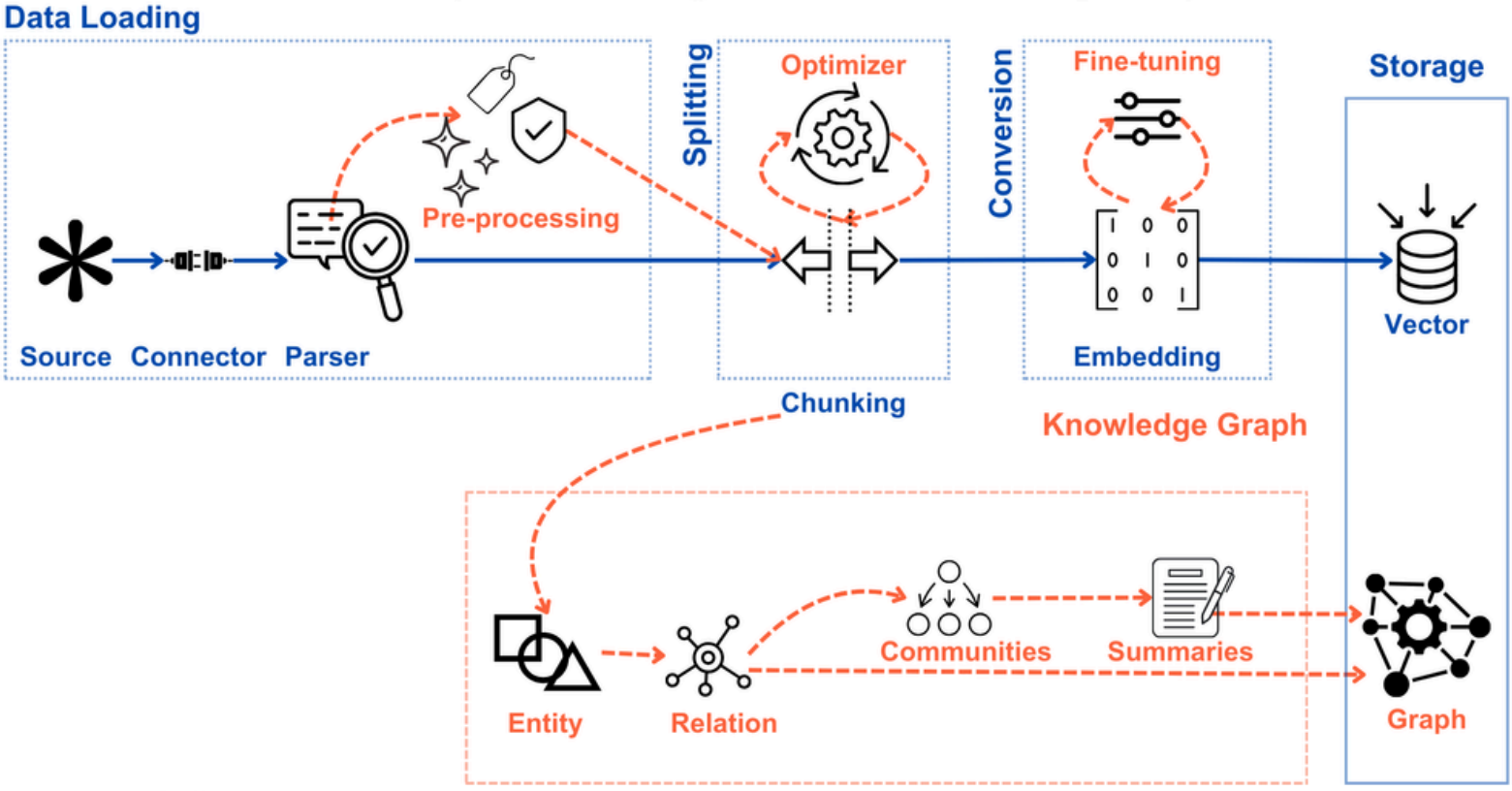
With techniques employed at the three stages, the Advanced RAG process follows a 'Rewrite then Retrieve then Re-rank then Read' frameworks. Two additional components of Rewrite and Re-rank are added, and the Retrieve component is enhanced in comparison with Naïve RAG.

# Ch 6 : RAG Systemd Progression: Naïve, Advanced, & Modular RAG

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

Chunk Size optimization and enriched context enhances retrieval and contextual generation

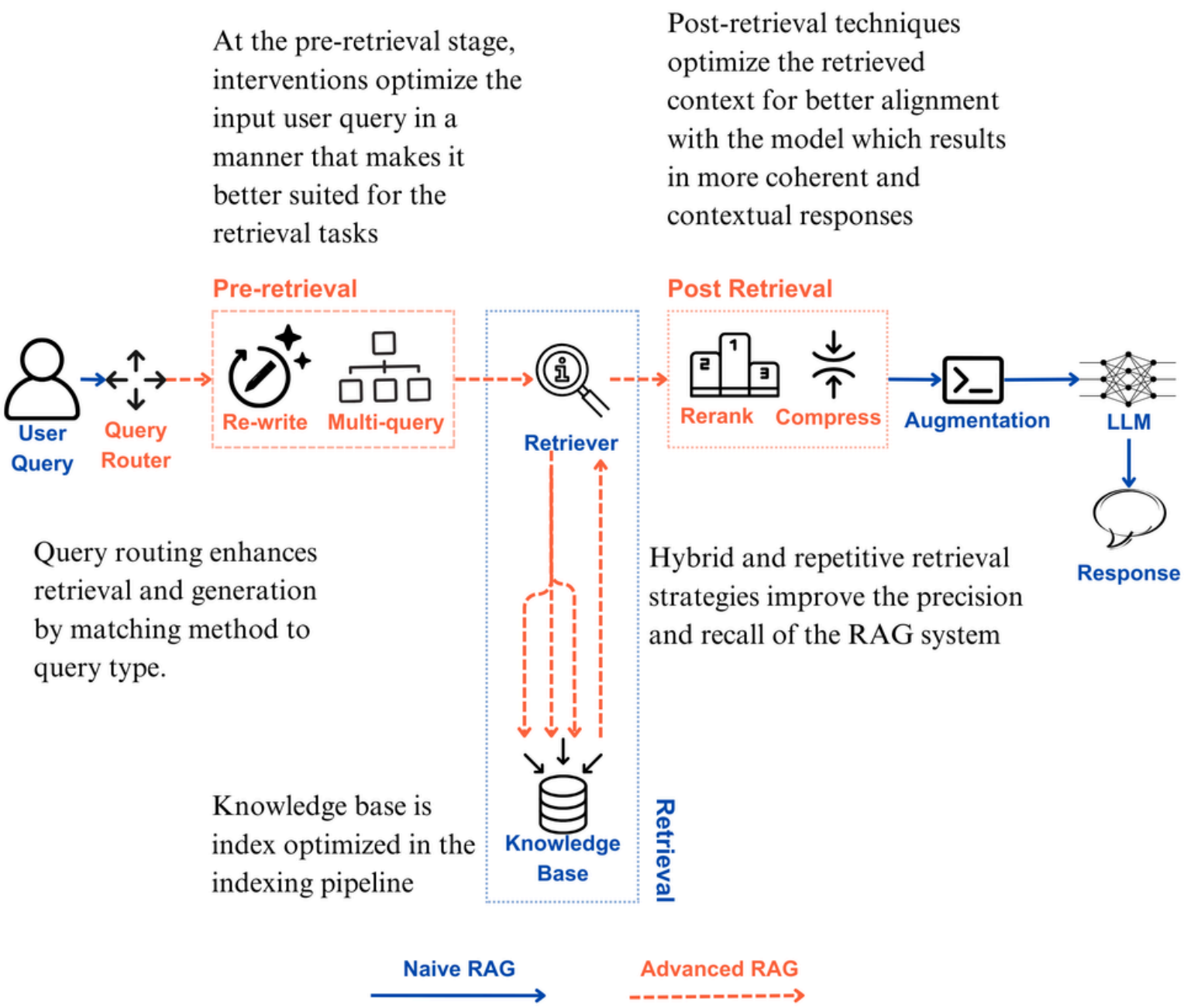
Embeddings model can be fine-tuned for domain adaptability




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

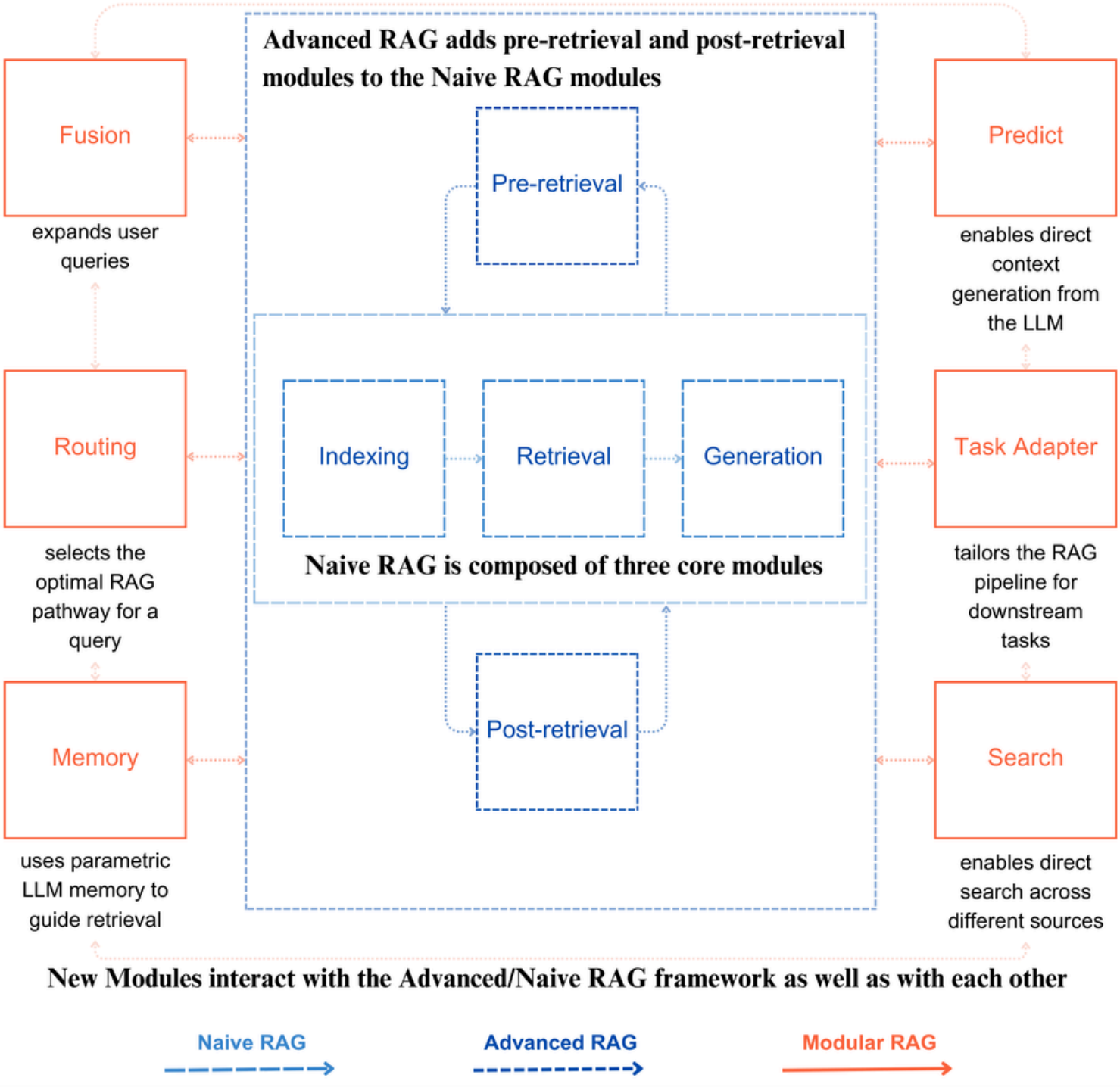
## INDEXING PIPELINE MODIFIED FOR ADVANCED RAG

# Ch 6 : RAG Systemd Progression: Naïve, Advanced, & Modular RAG




GENERATION PIPELINE MODIFIED FOR ADVANCED RAG

# Ch 6 : RAG Systemd Progression: Naïve, Advanced, & Modular RAG



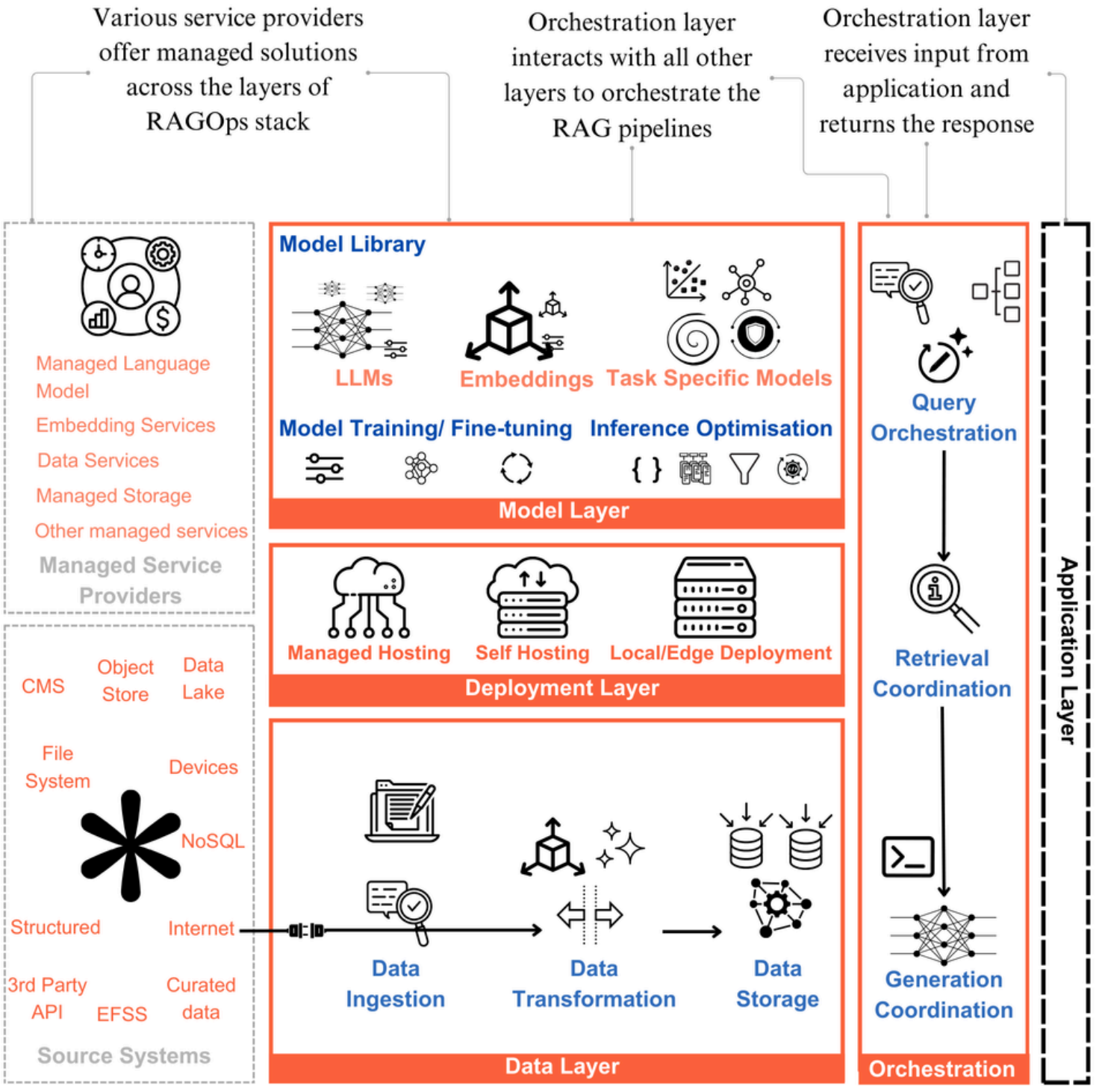

## GENERATION PIPELINE MODIFIED FOR ADVANCED RAG

Naïve, Advanced, and Modular approaches to RAG are progressive. Naïve RAG is a sub-component of Advanced RAG which is a sub-component of Modular RAG.

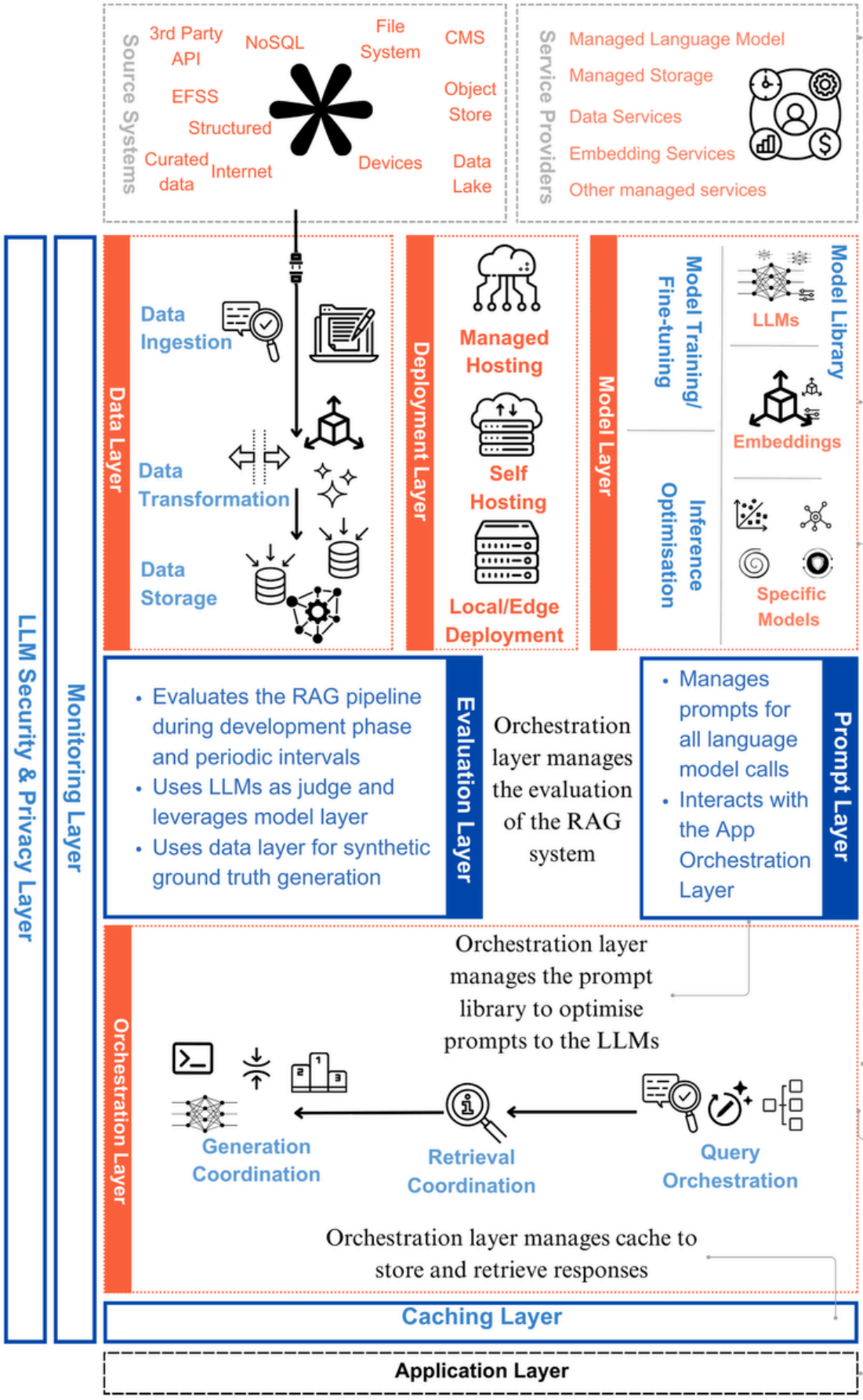
# Ch 7 : Evolving RAGOps Stack: Making RAG possible

This chapter covers

- The design of RAG systems
- Available tools and technologies that enable a RAG system
- Production best practices for RAG systems



# Ch 7 : Evolving RAGOps Stack: Making RAG possible

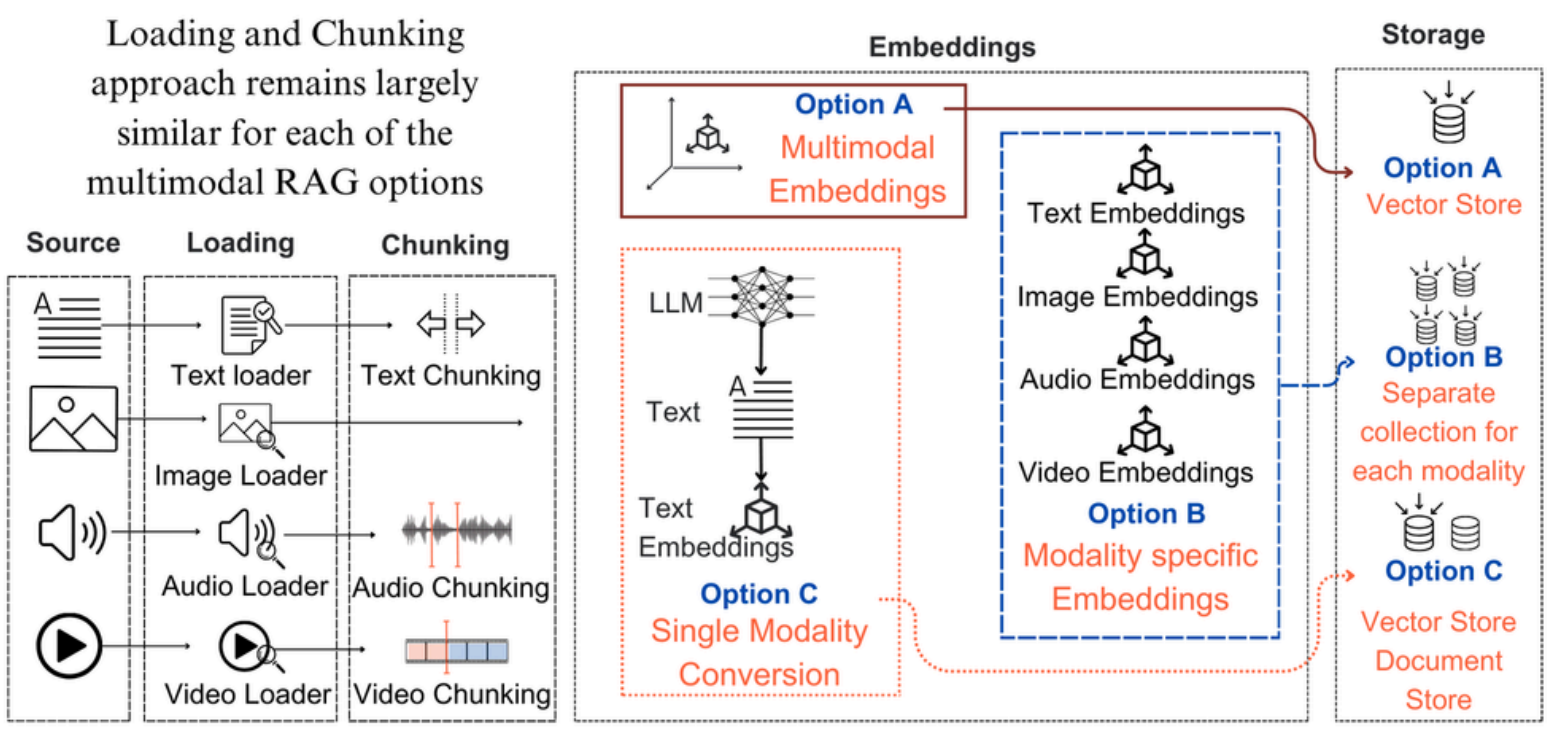


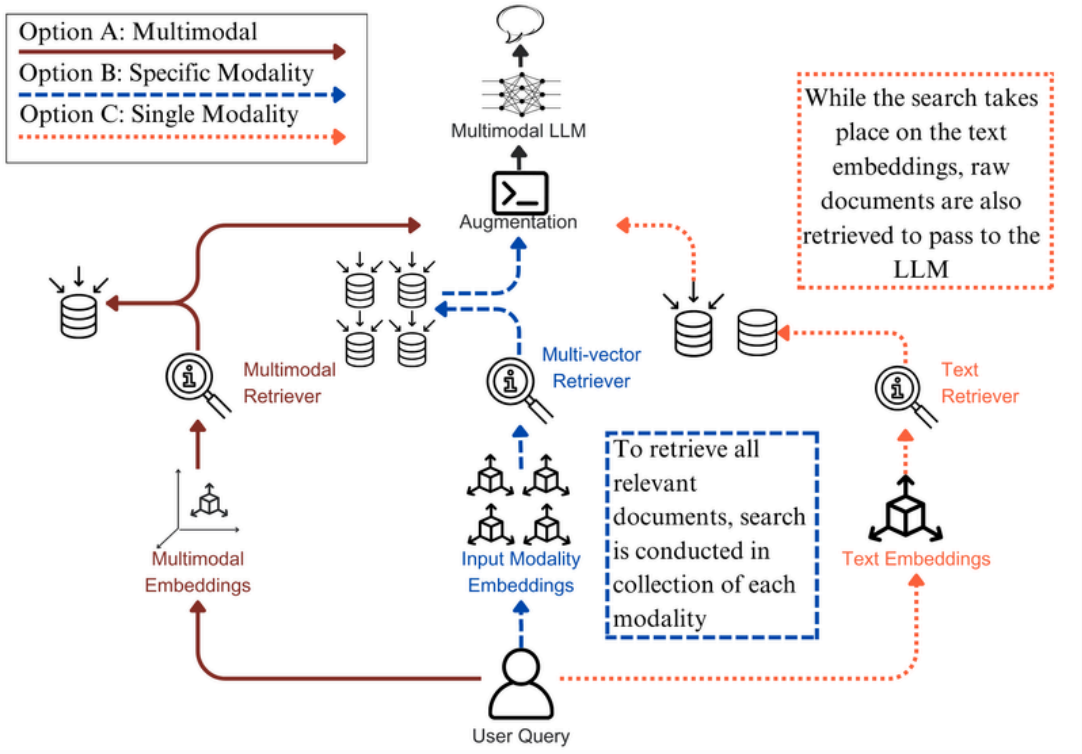

CORE & ESSENTAIL LAYERS OF A RAGOPS STACK

# Ch 8 : RAG variants: Multimodal, agentic, graph and other RAGs

This chapter covers

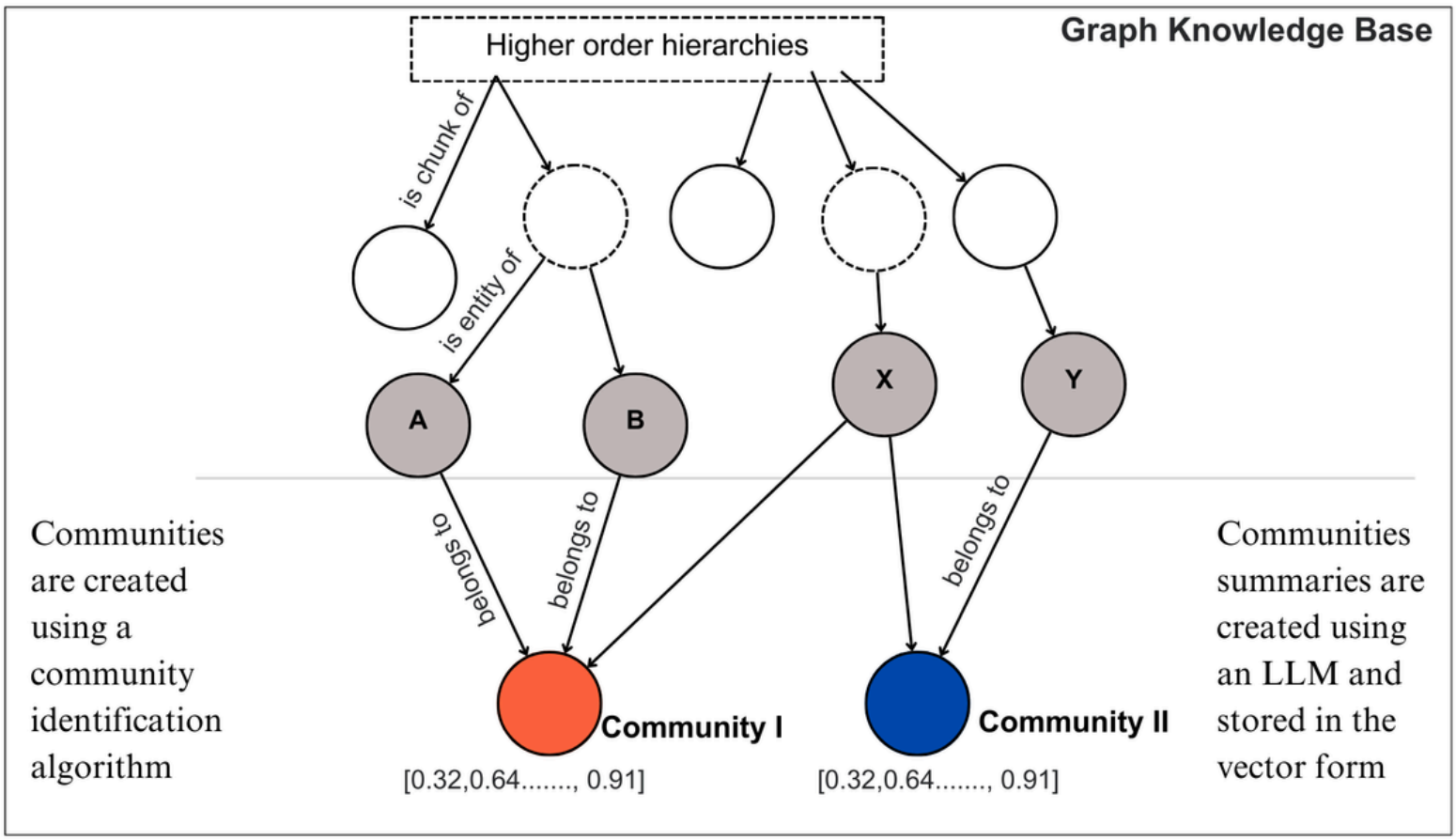
- Introduction to RAG Variants
- Knowledge Graph RAG, Multimodal RAG, Agentic RAG
- Other RAG Variants

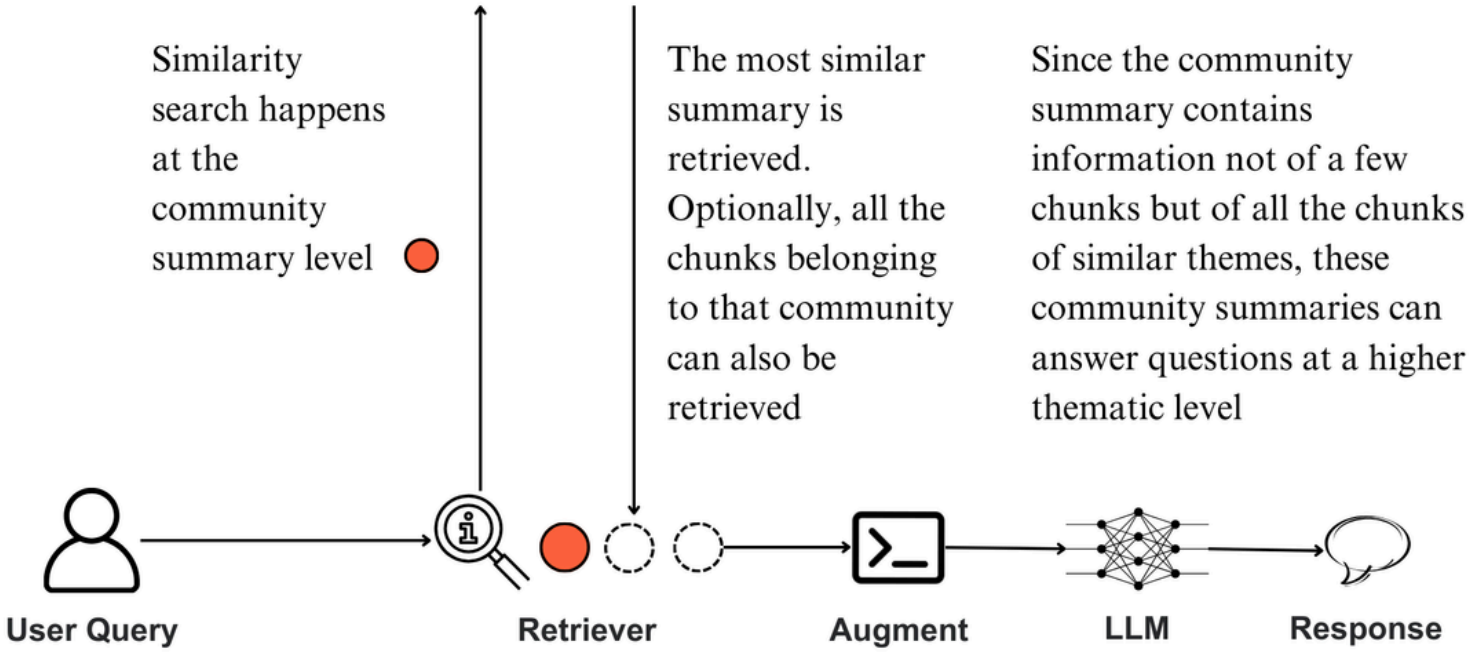



## INDEXING AND GENERATION PIPELINES FOR MULTIMODAL RAG

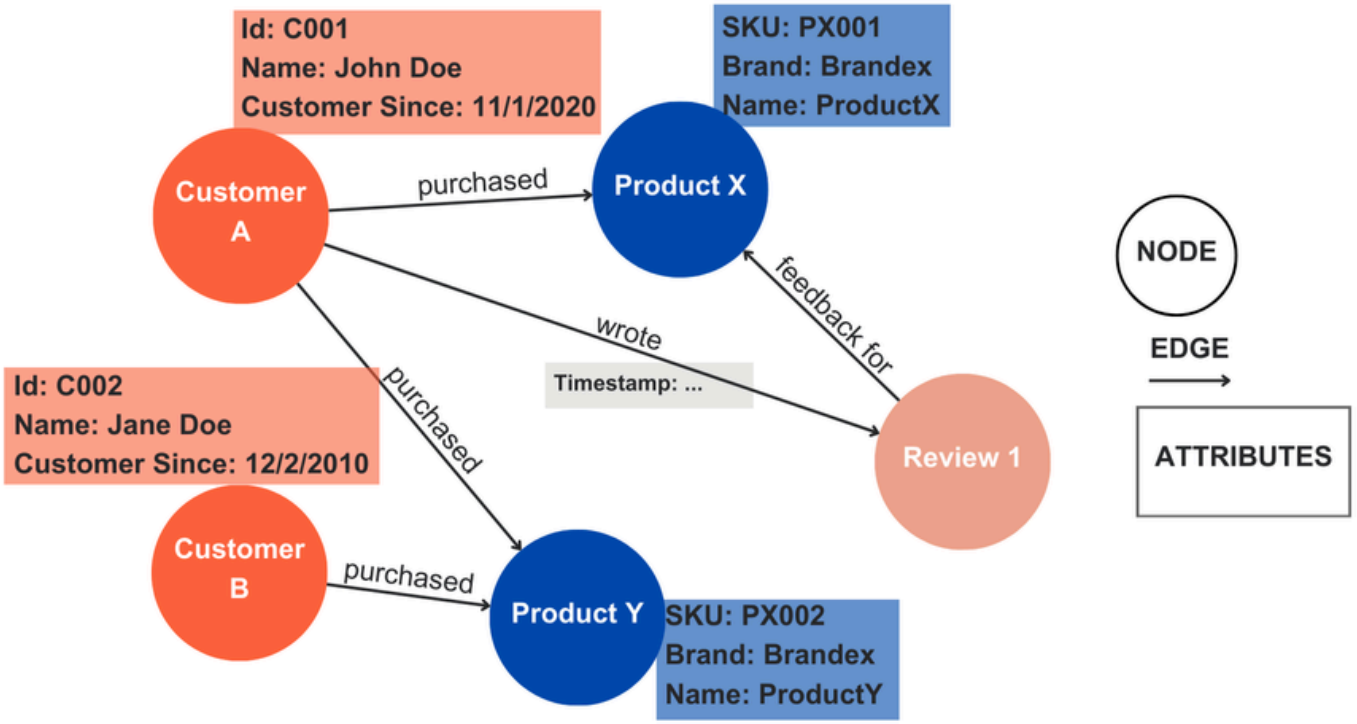
# Ch 8 : RAG variants: Multimodal, agentic, graph and other RAGs

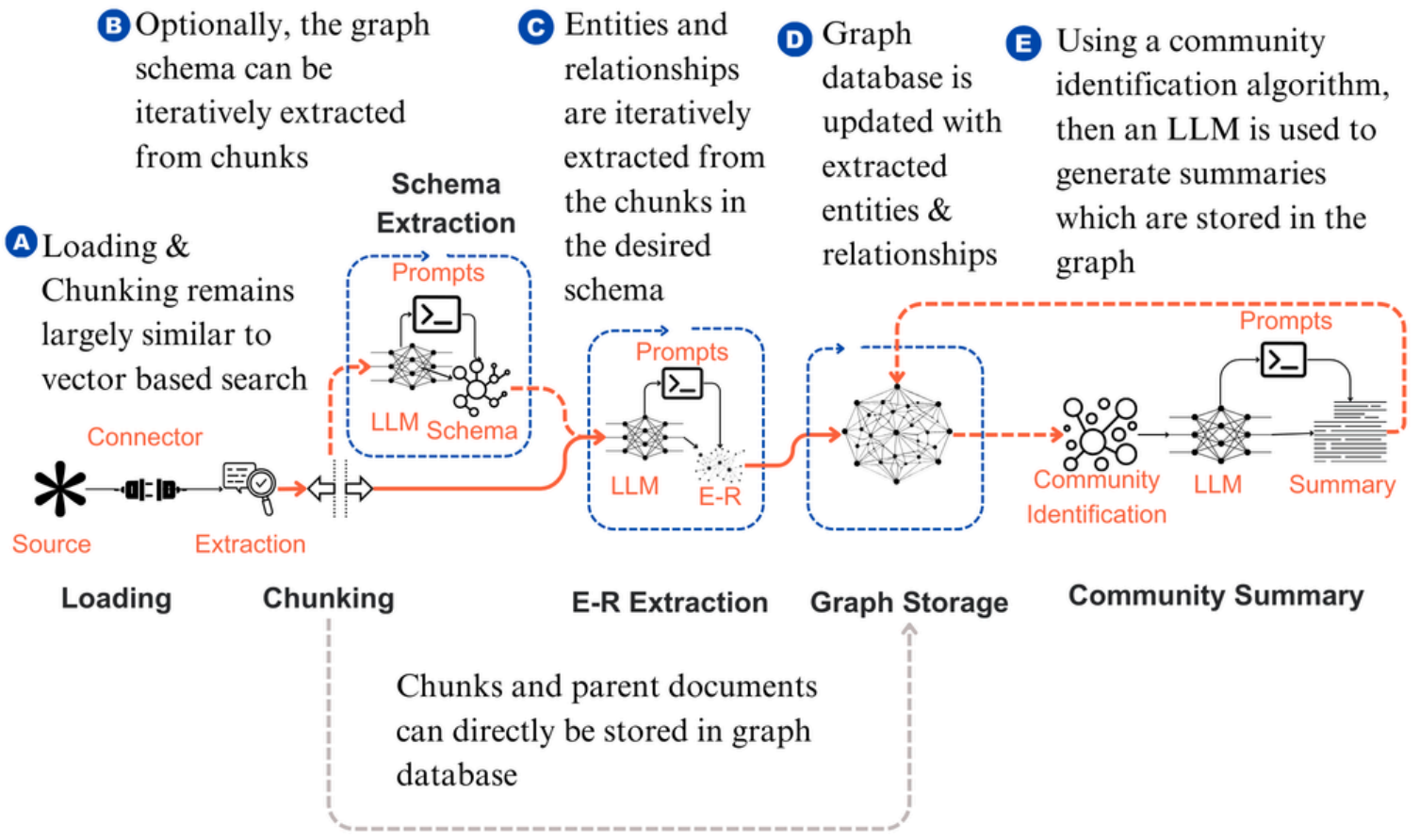



## GRAPH COMMUNITIES AND COMMUNITY SUMMARIES ENHANCED RAG

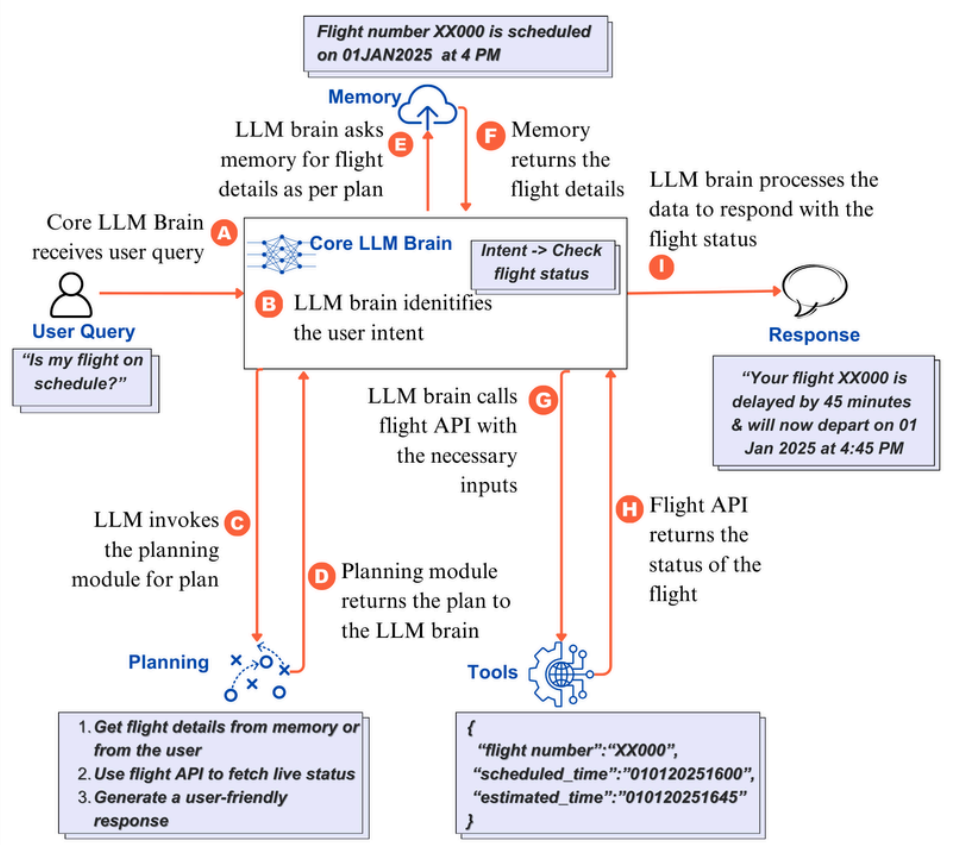
# Ch 8 : RAG variants: Multimodal, agentic, graph and other RAGs



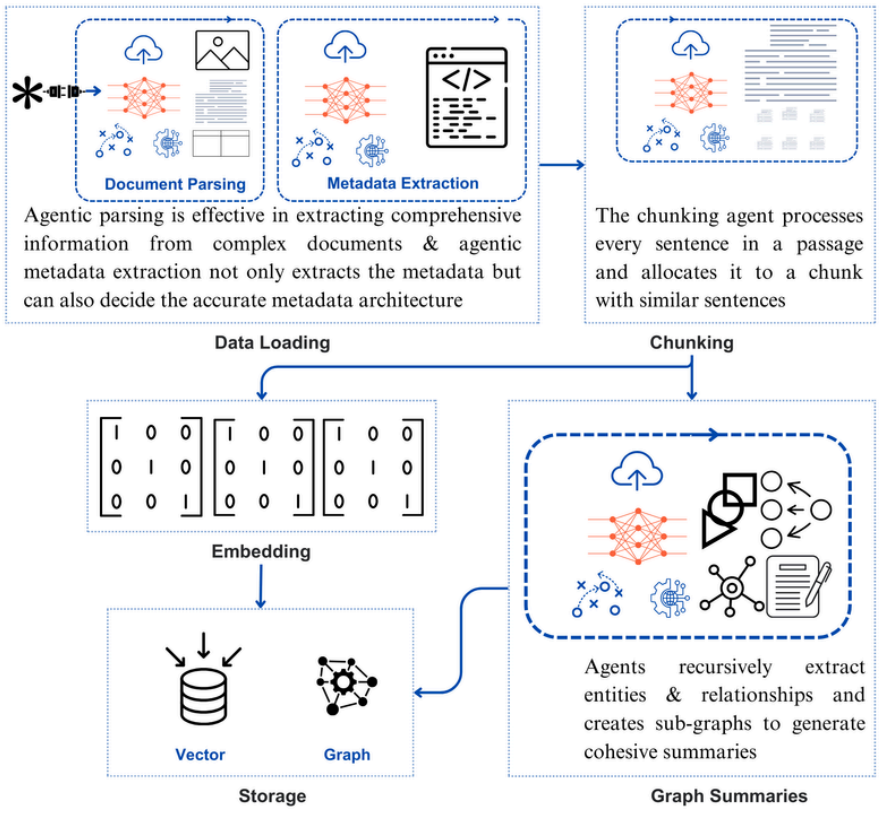

## INDEXING PIPELINE FOR KNOWLEDGE GRAPH RAG

# Ch 8 : RAG variants: Multimodal, agentic, graph and other RAGs




## AGENTIC RAG EXAMPLE: AN AIRLINE BOOKING AND ENQUIRY SYSTEM

## AGENTIC INTERVENTIONS ACROSS THE INDEXING PIPELINE STAGES



# Ch 9 : RAG Development Framework & Further Exploration

This chapter covers

- A Recap of the concepts covered in this book using a six-stage RAG development framework
- Areas for further exploration in RAG

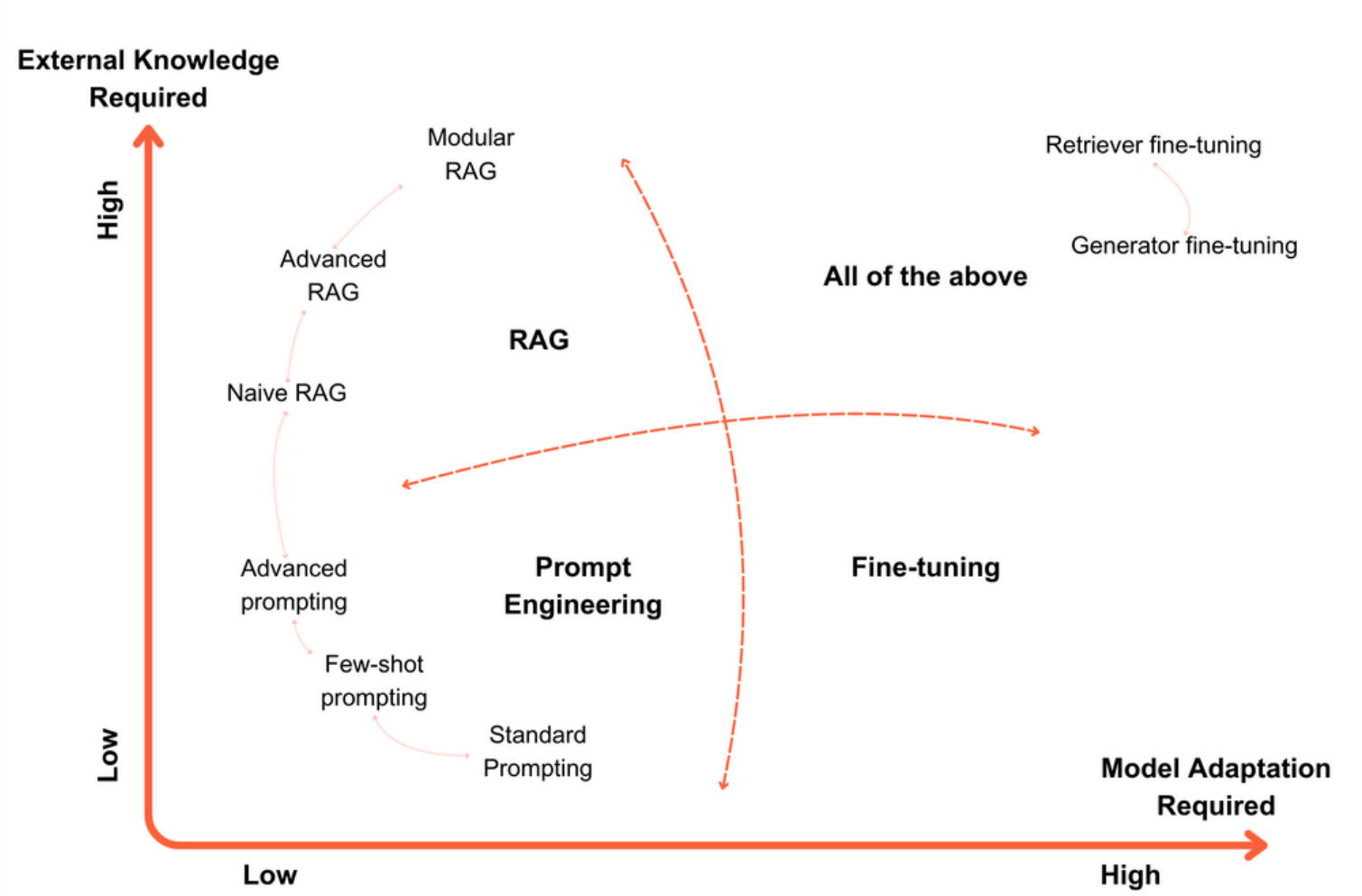

SIX STAGES OF RAG DEVELOPMENT FRAMEWORK

# Ch 9 : RAG Development Framework & Further Exploration

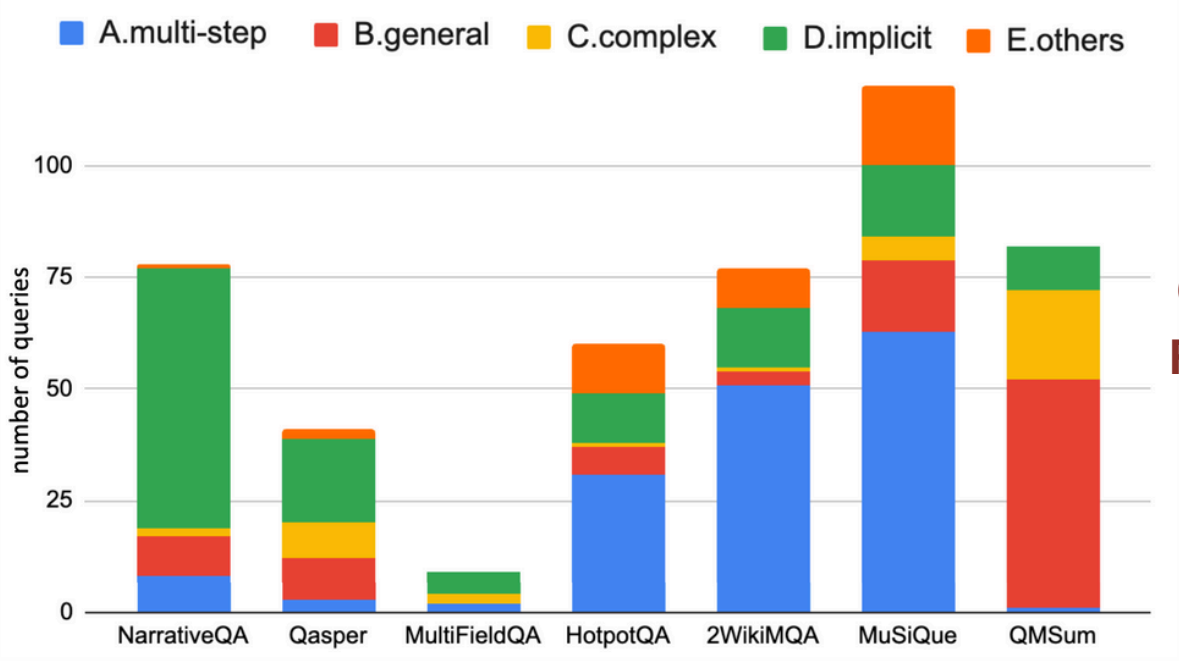
| Use Case | Use Case Evaluation Questions | | | | A RAG system is required? |
|---|---|---|---|---|---|
| | System requires data that may not be present in training set? | System requires data that is current or updates frequently? | System generates facts? | Are users looking for sources? | |
| Creative Writing Assistance | No - LLMs do not need any additional data for creative writing | No - LLMs do not need any current information for creative writing | Maybe - Creative writing may not necessarily need generated facts | No - User expectations from creative writing does not need any source citation | ✗ No |
| Customer Support Bot | Yes - Product/Company specific information may not be present in LLM training data | Yes - Product information, inventory levels, order information changes frequently | Yes - All generated information is factual | Maybe - Sources may enhance customer experience | ✓ Yes |
| Language Translation | No - LLMs do not need any additional data for language translation | No - LLMs do not need any current information for language translation | No - Facts, if any, will be the same as provided in the prompt | No - The source of information will always be the prompt | ✗ No |
| Spelling & Grammar Correction | No - LLMs do not need any additional data for checks | No - LLMs do not need any current information for checks | No - No additional facts need to be generated | No - The source of information will always be the prompt | ✗ No |

## EVALUATING USE CASES FOR THE NEED OF A RAG SYSTEM

# Ch 9 : RAG Development Framework & Further Exploration




RAG VS FINETUNING




DIFFICULT QUERIES FOR RAG SYSTEMS

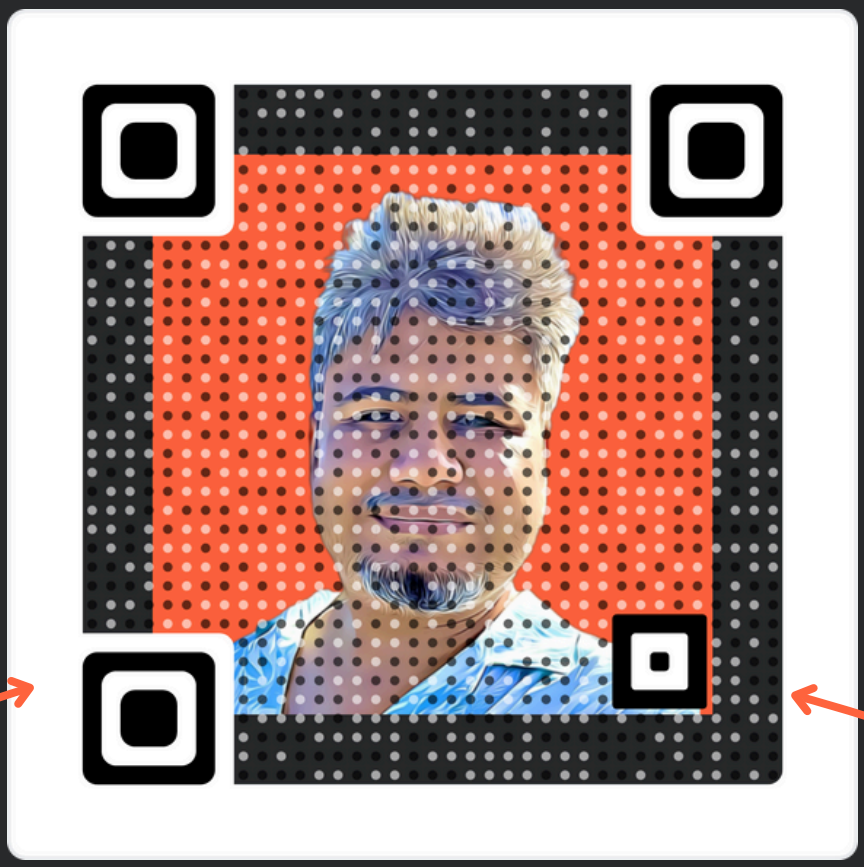
# THESE EXCERPTS ARE FROM

# A SIMPLE GUIDE TO RETRIEVAL AUGMENTED GENERATION



## SCAN CODE OR CLICK HERE TO GET AN EARLY ACCESS COPY

*Hi! My name is Abhinav, and I talk about ML, RAG, LLMs & AI Agents. If our interests align, I'd love to stay connected*



**SCAN CODE OR CLICK HERE TO CONNECT**

linktr.ee/abhinavkimothi

/in/Abhinav-Kimothi

@akaiworks

@abhinav_kimothi

@abhinavkimothi