Introduction to Retrieval Augmented Generation (RAG)

Yogesh Haribhau Kulkarni

YHK

Outline

Retrieval Augmented Generation (RAG)

YHK

Quiz

- How to do domain adaptation? ie
- How to build custom LLM solution? meaning
- How can LLM based chatbot answer from my data?

Answers??

YHK

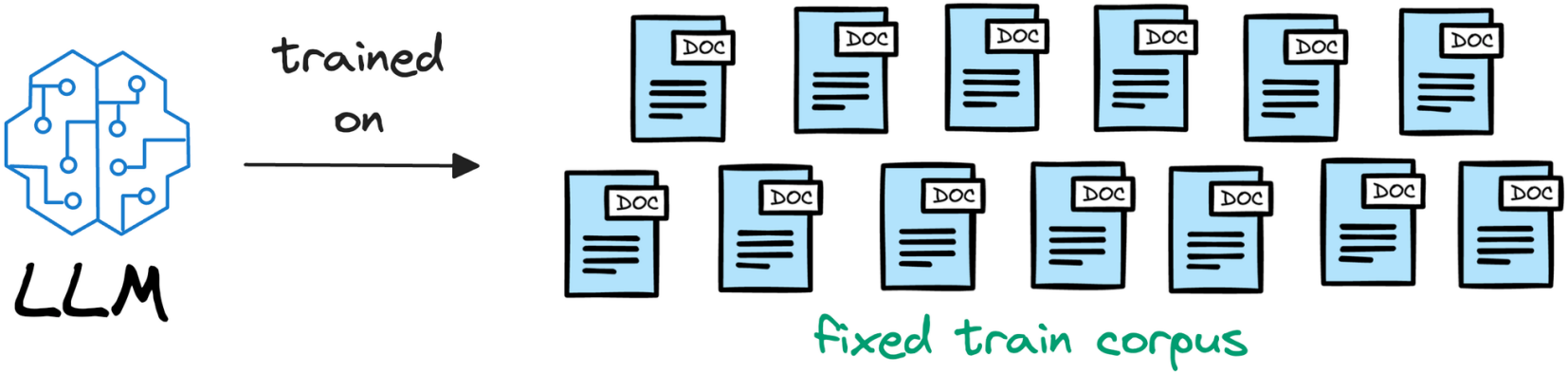
Need for Domain Adaptation

- Industrial settings prioritize cost, privacy, and reliability of solutions.
- LLMs are prone to hallucinations, factual errors, and looping.
- LLMs need to work on own/private/streaming/latest data
- Speed/latency/efficiency Concerns
- Solution: To build LLM from scratch?? needs??
- Extensive training data, high computational resources and $$$$

So, what to do?

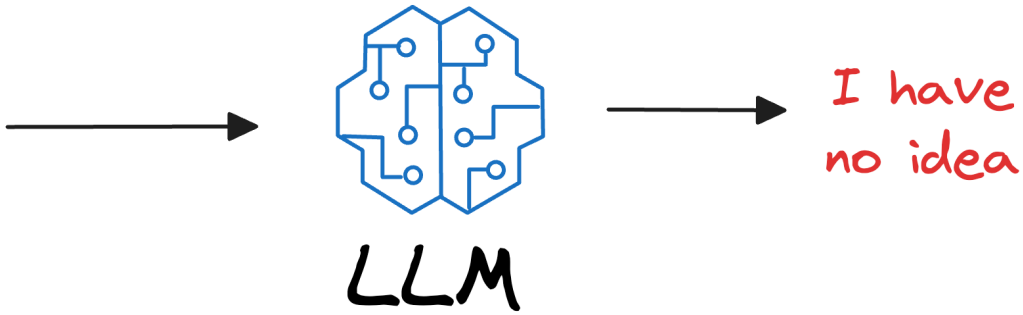
YHK

Training and Query Mismatch



Prompt

Election results were declared on 2nd Feb. What happened?



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

YHK

Solutions

- Few Shots
- RAG
- Fine-tuning

Why RAG?

- **Unlimited Knowledge:**
  - RAG enables access external sources, surpassing limitations of its data.
  - Allows exploration of proprietary documents and internet searches
- **Easier to Update/Maintain:**
  - Offers a cost-effective way to update and maintain
  - Building a knowledge base minimizes ongoing maintenance financial burden.
- **Confidence in Responses:**
  - Enhances confidence by providing extra context for more accurate responses.
  - Practical boost to overall intelligence in generating responses.
- **Source Citation:**
  - RAG provides access to sources, improving transparency in LLM responses.
  - A step towards building trust in LLM systems.
- **Reduced Hallucinations:**
  - RAG-enabled LLMs exhibit reduced creative misfires.
  - Solid foundation of information keeps models focused and grounded.

YHK

Important Questions

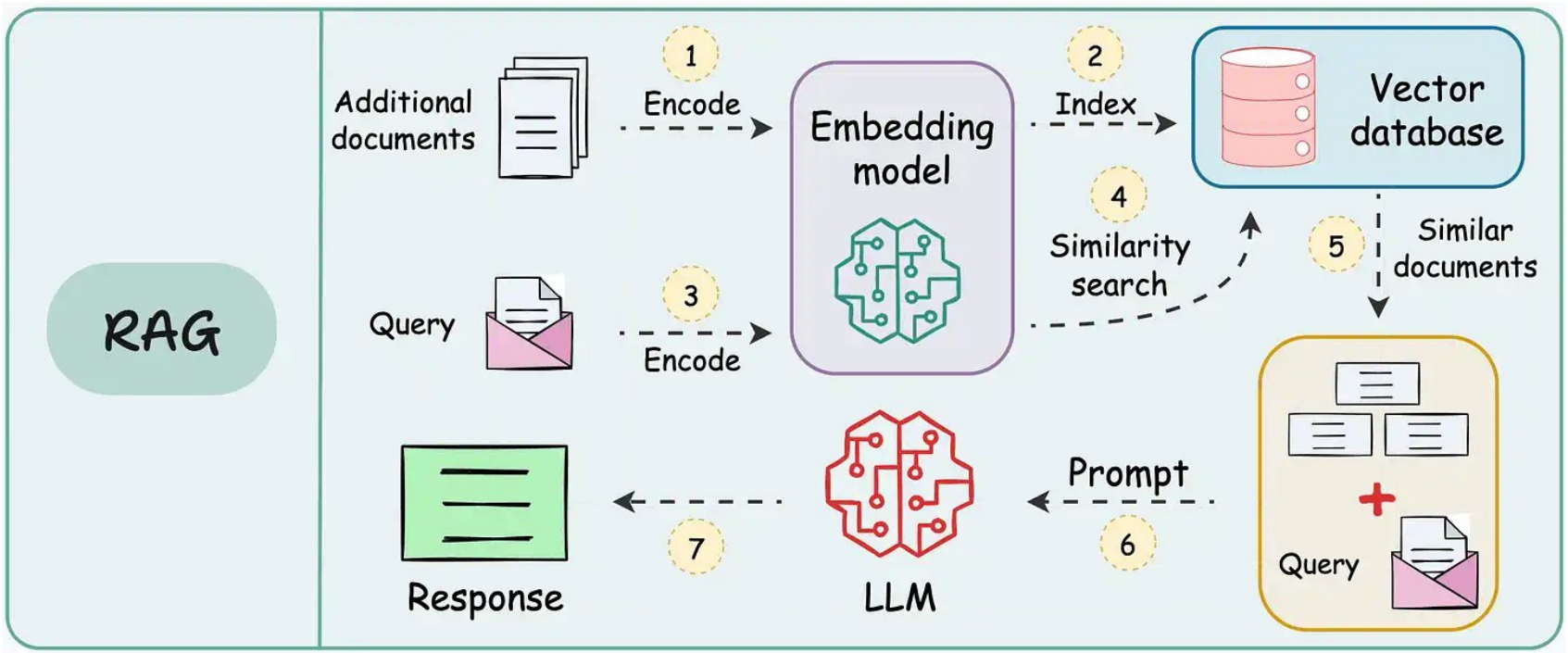
- Does the use case require external data access?
- Does the use case require changing foundation model style?
- Does the use case require addressing hallucinations?
- Is labeled training data available?
- Is citing the source of information important?
- How critical is system latency?
- What are the cost implications?
- What are the scalability requirements?
- Do we have the necessary expertise?

YHK

Retrieval Augmented Generation (RAG) Workflow

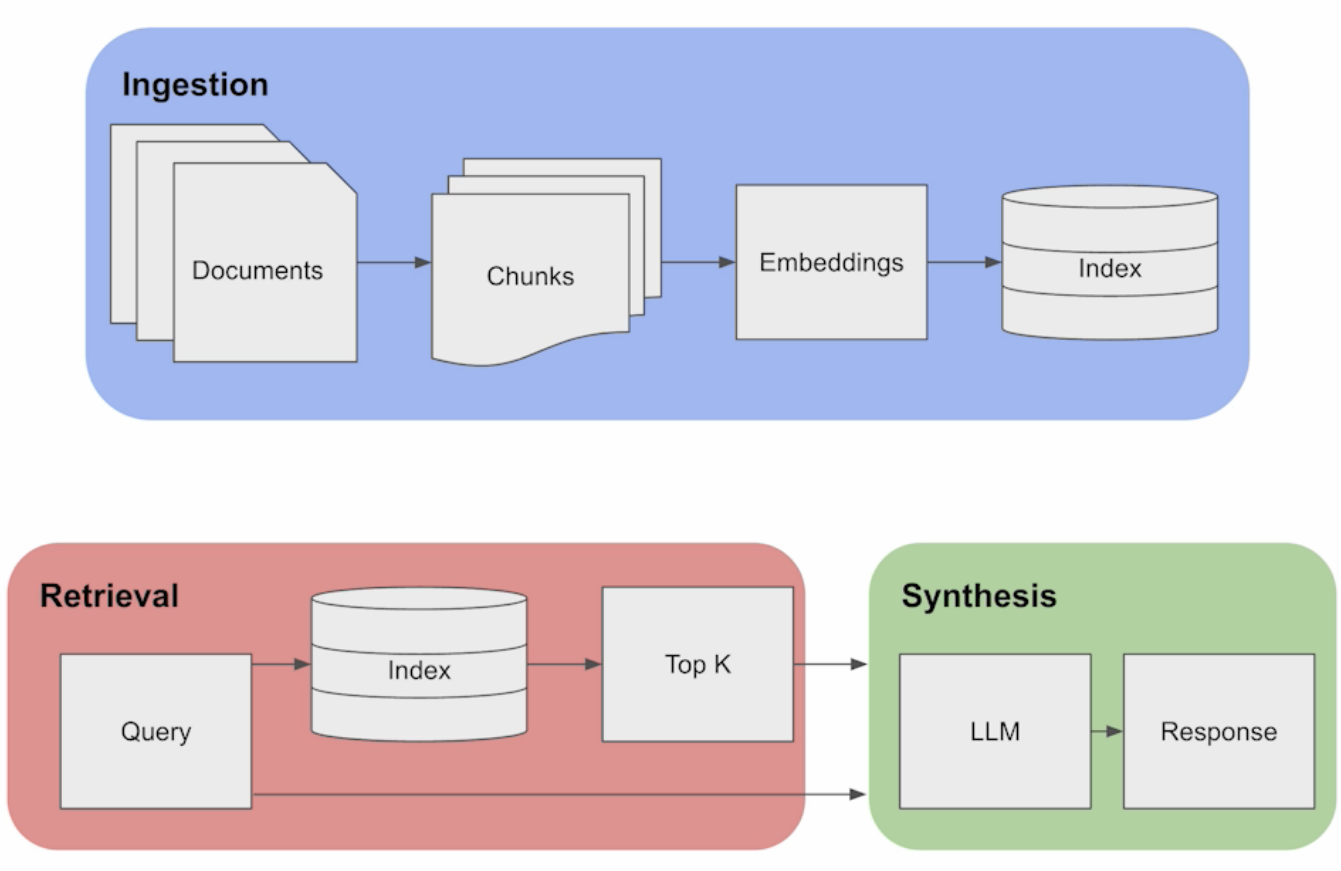
YHK

RAG Workflow



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

RAG Components



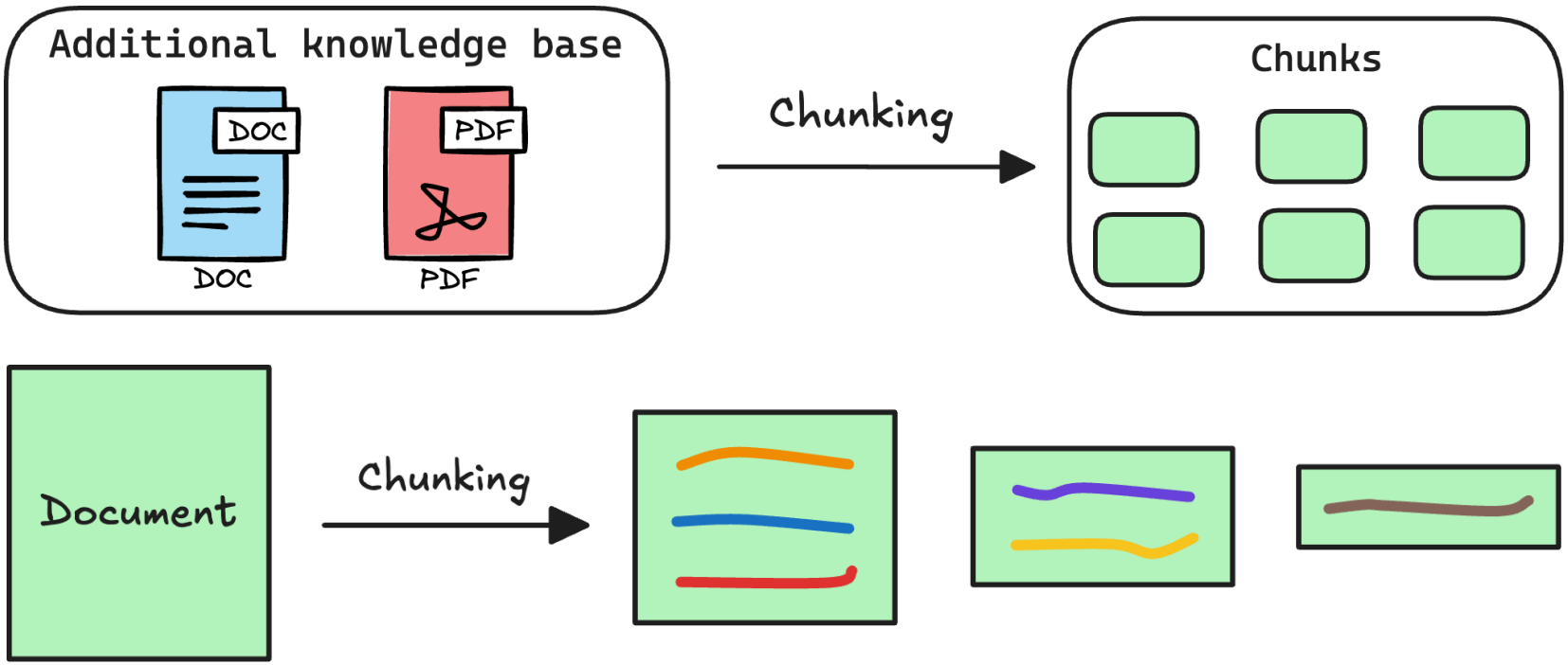
(Ref: [Week 4] Retrieval Augmented Generation -Aishwarya Naresh Reganti)

RAG Components

- Ingestion:
  - Documents undergo segmentation into chunks, and embeddings are generated from these chunks, subsequently stored in an index.
  - Chunks are essential for pinpointing the relevant information in response to a given query, resembling a standard retrieval approach.
- Retrieval:
  - Leveraging the index of embeddings, the system retrieves the top-k documents when a query is received, based on the similarity of embeddings.
- Synthesis:
  - Examining the chunks as contextual information, the LLM utilizes this knowledge to formulate accurate responses.

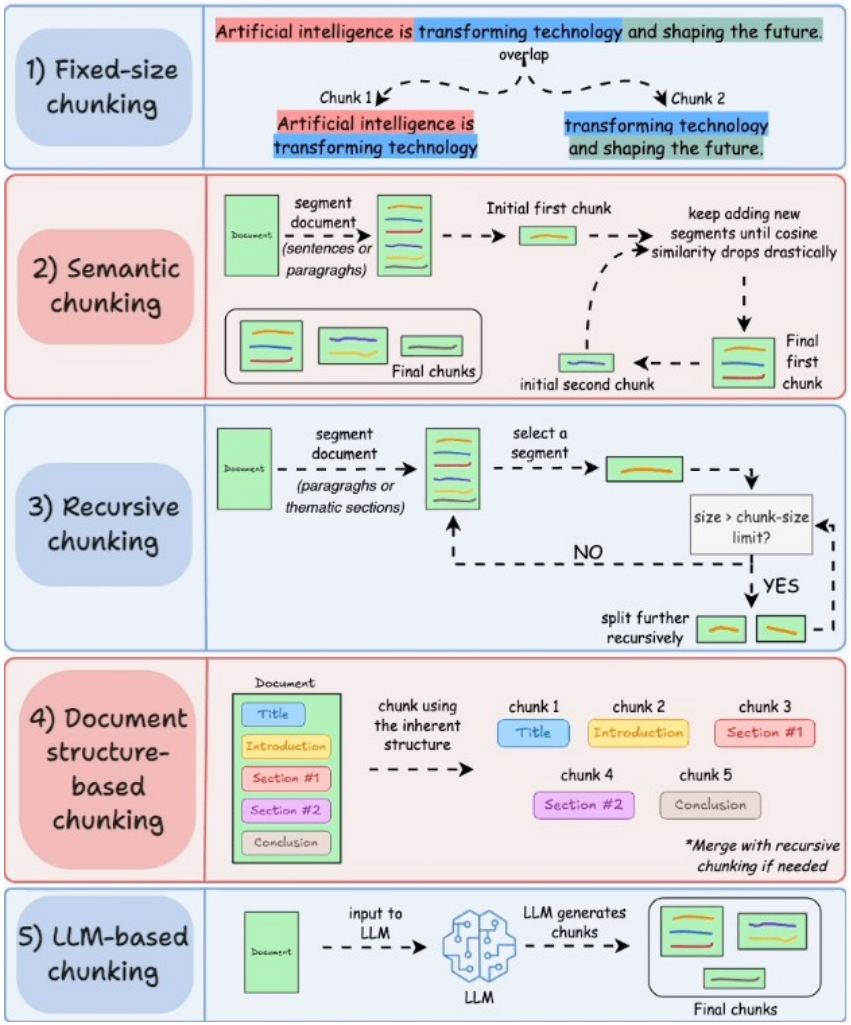
YHK

Create chunks



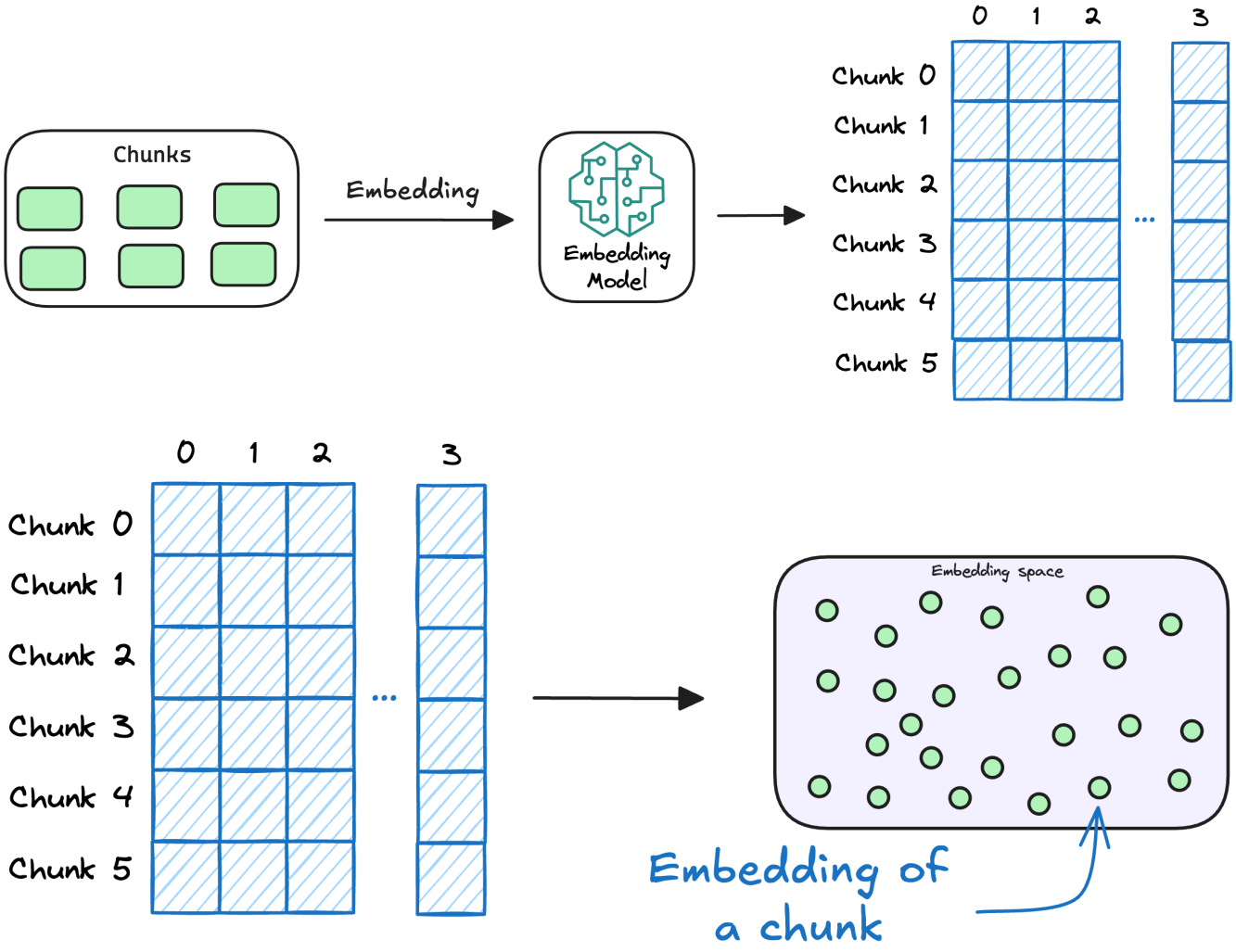
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

5 Chunking Strategies For RAG



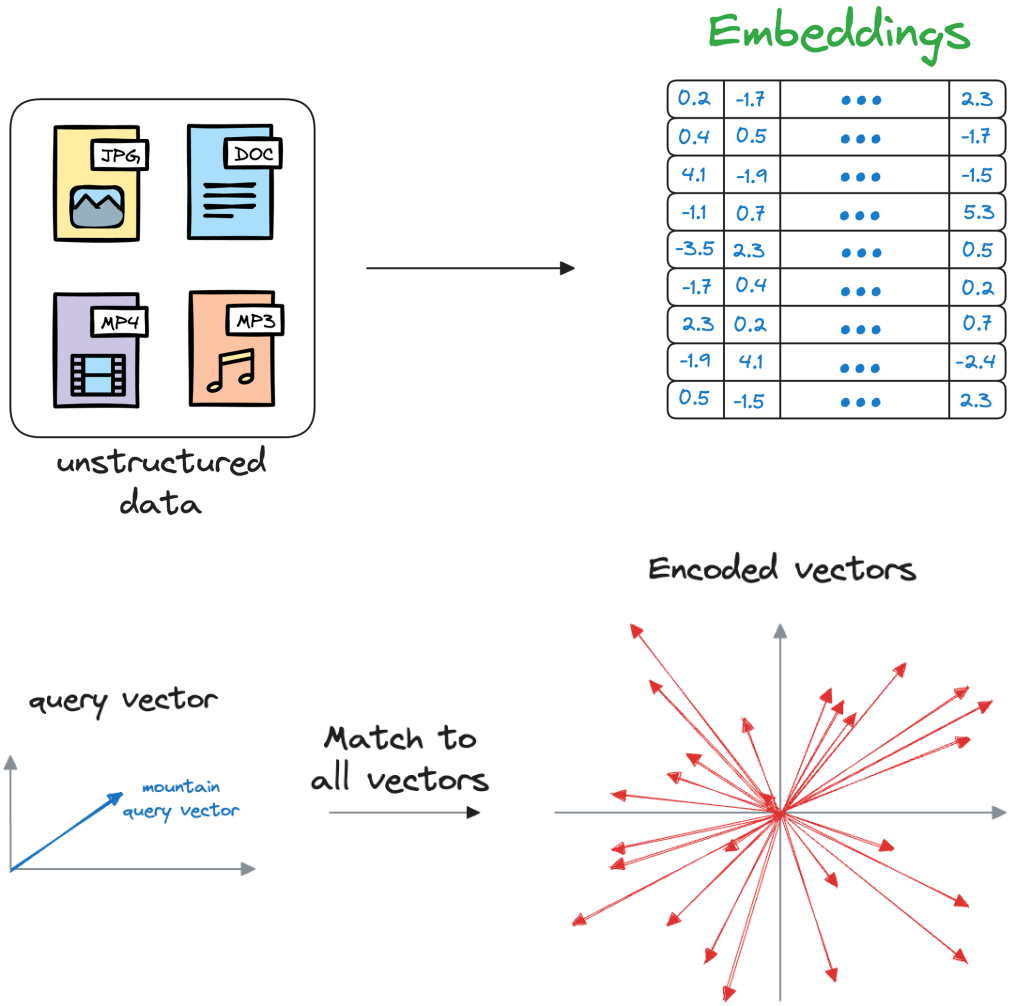
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

Generate embeddings



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

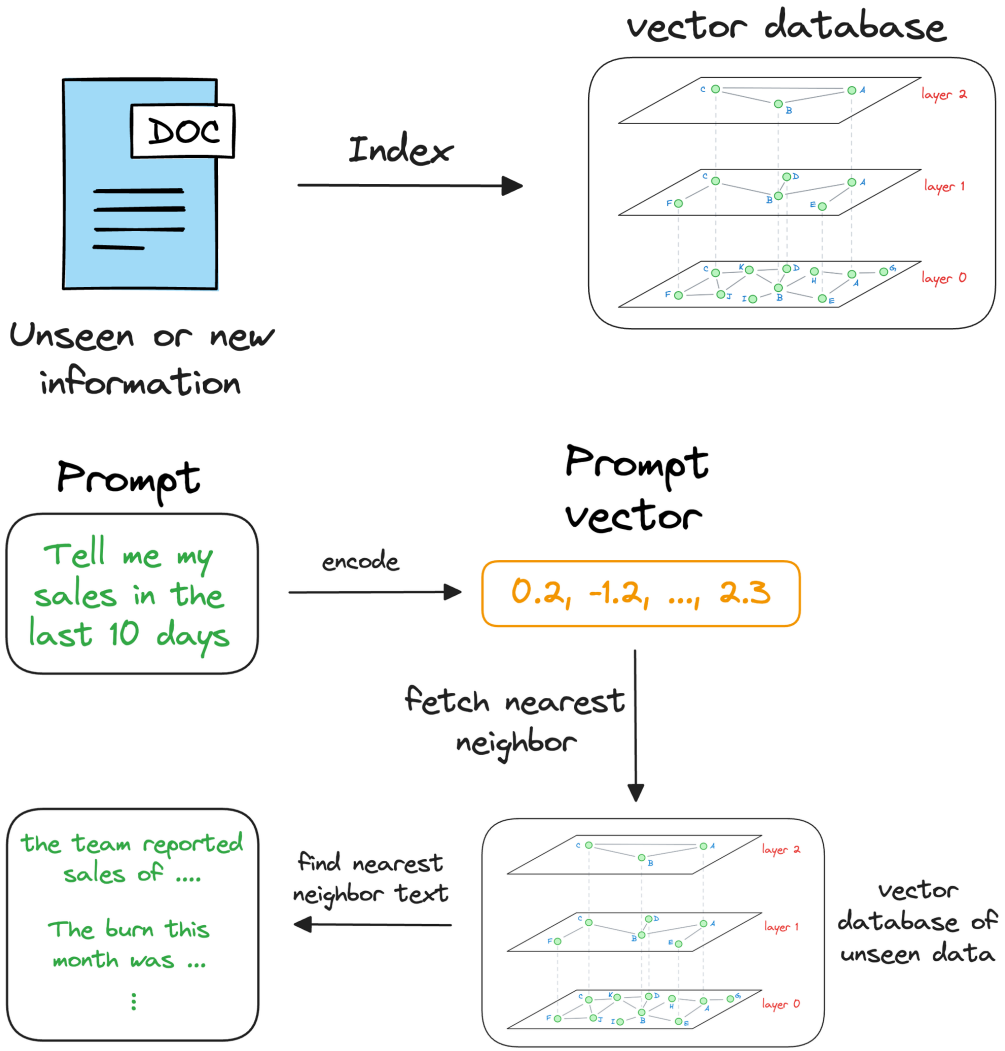
Vector Databases



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

YHK

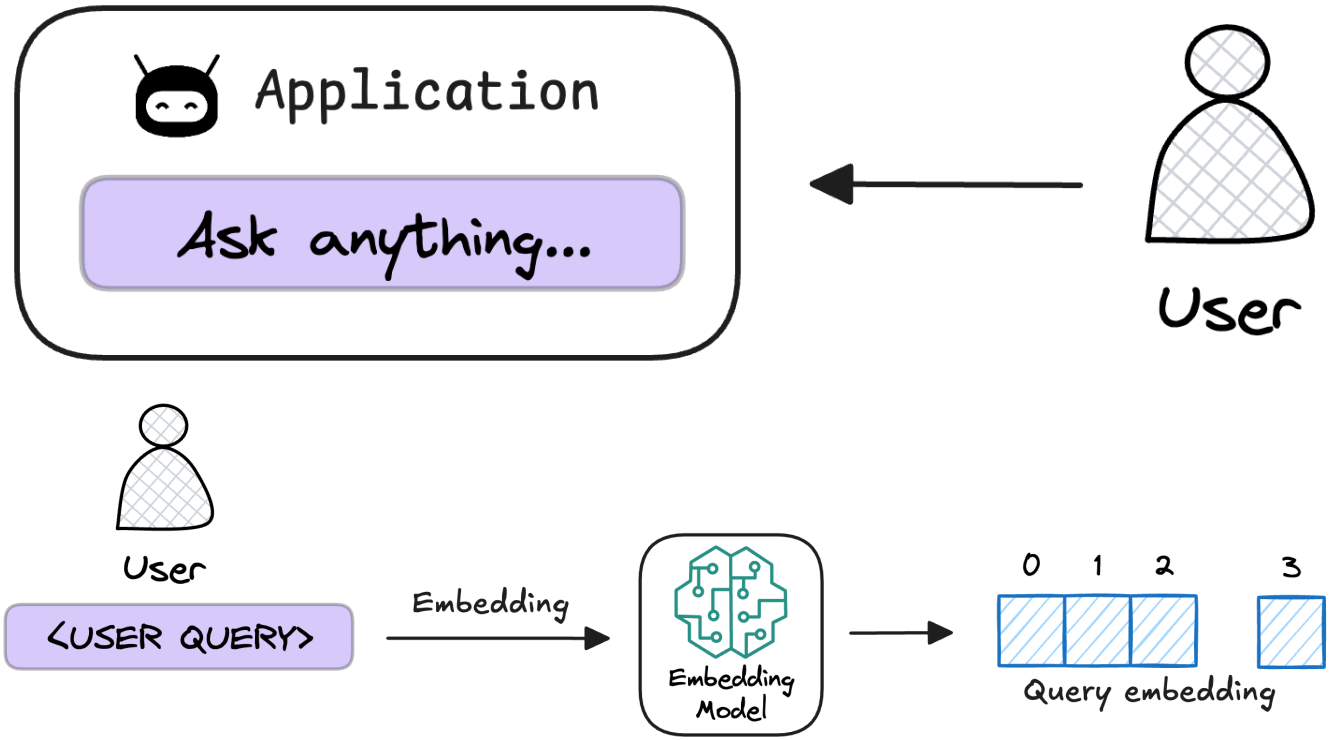
Vector Databases



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

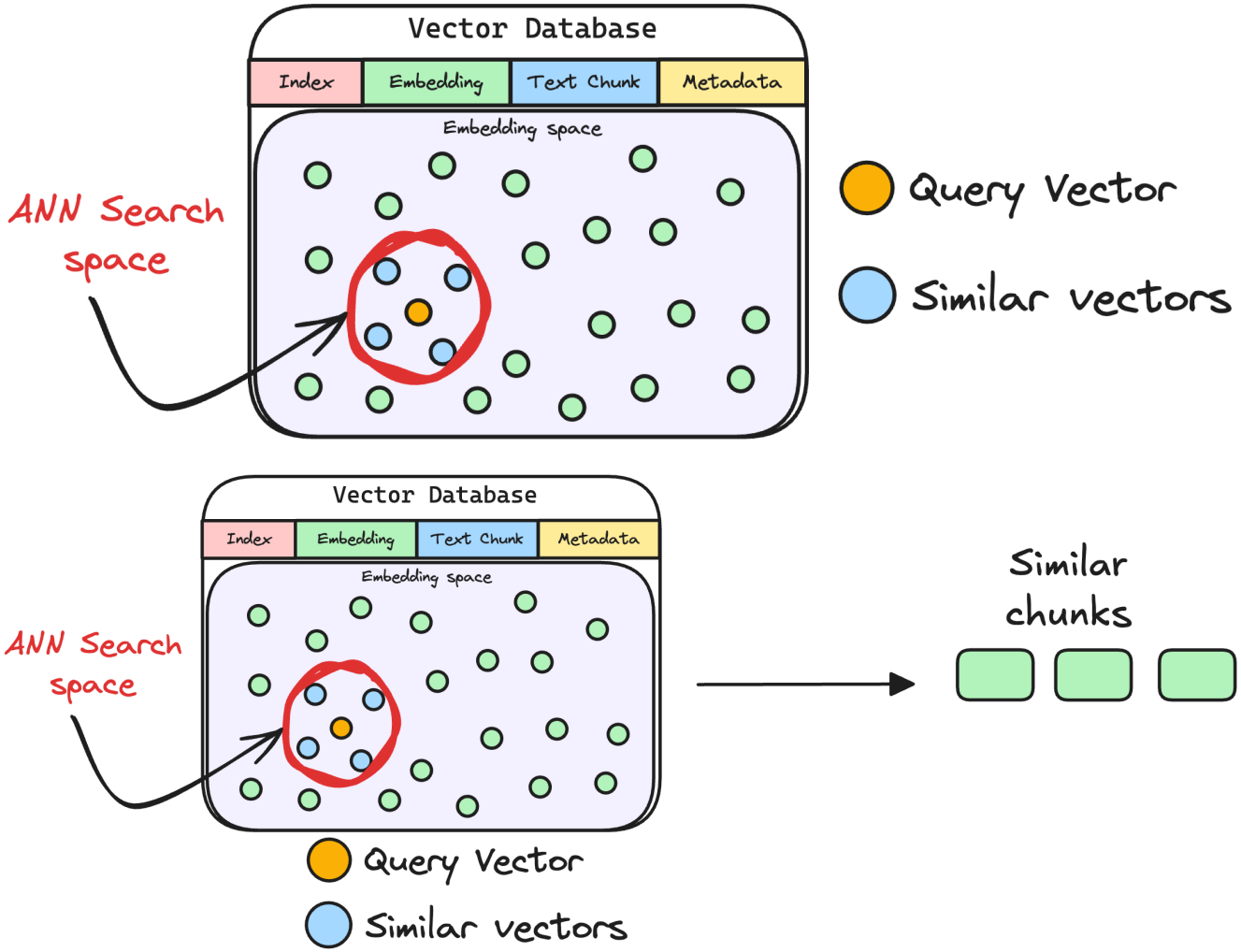
YHK

Query



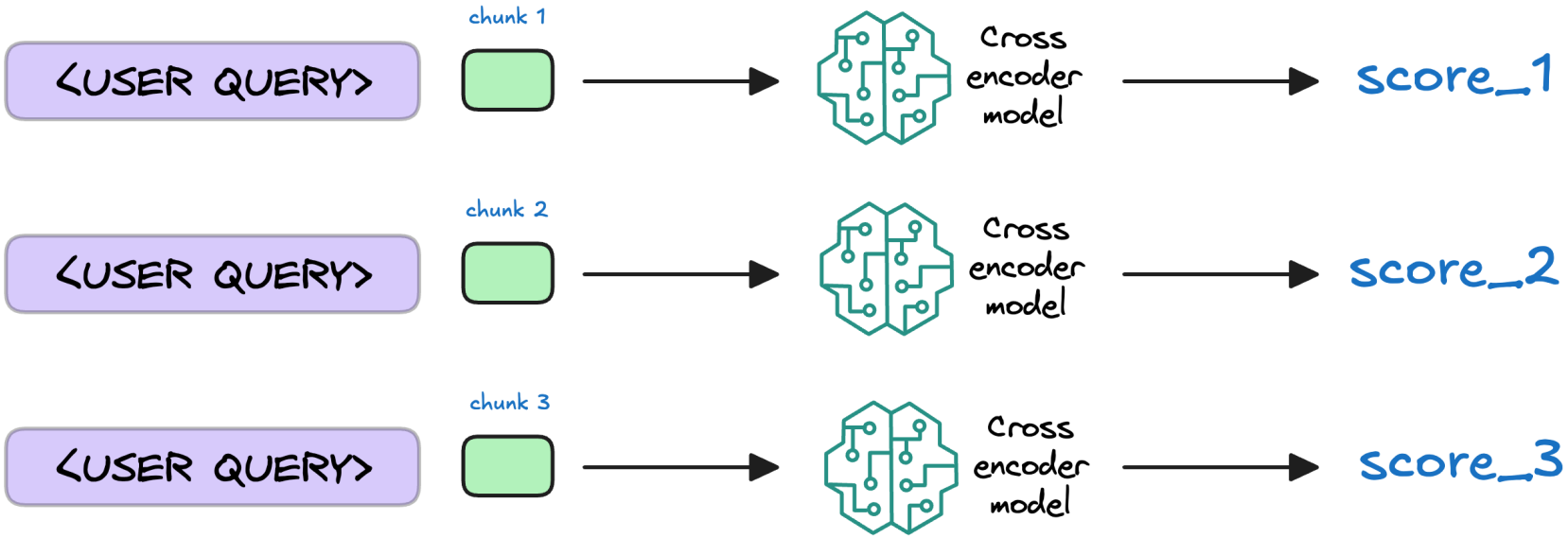
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

Retrieve similar chunks



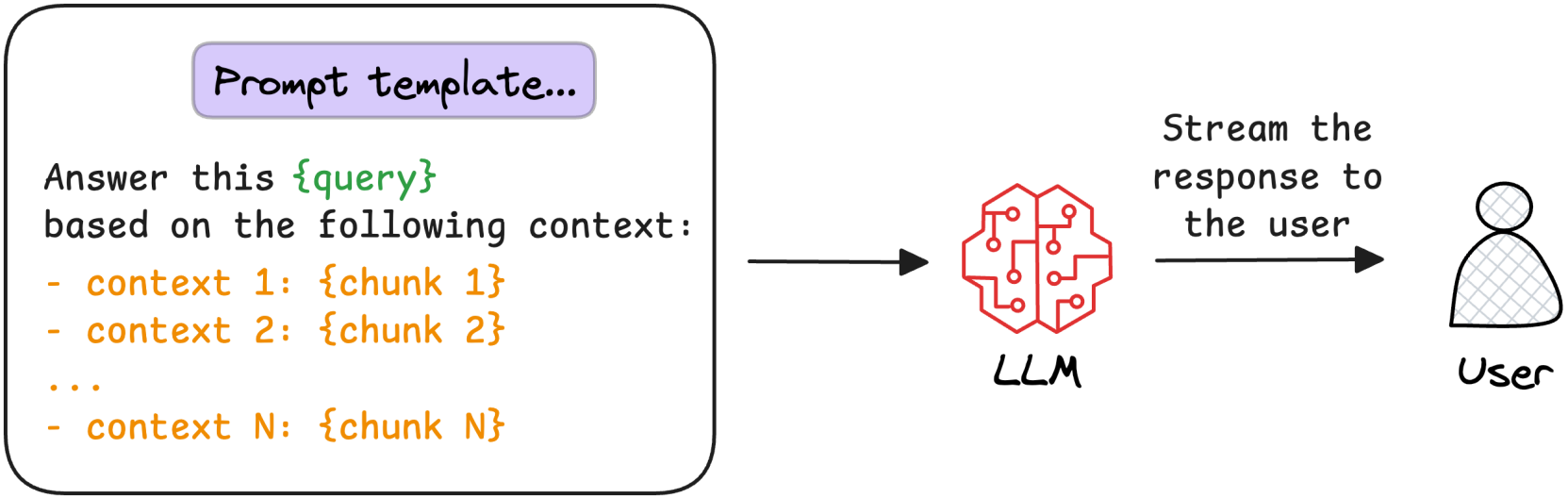
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

Re-rank the chunks



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

Generate the final response



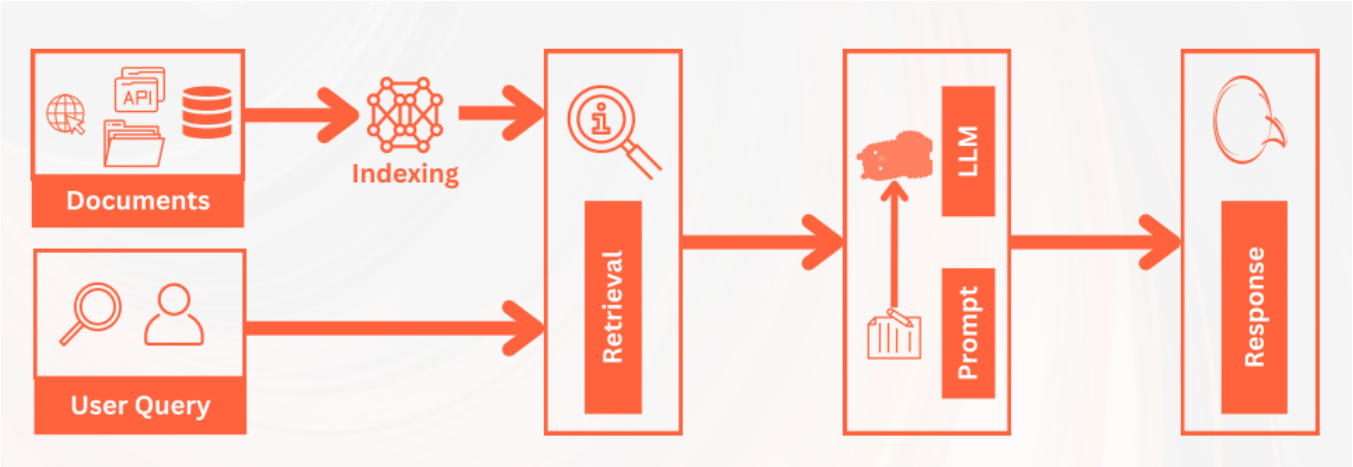
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

YHK

Naive RAG Approach

Simply retrieve texts and provide to language model as additional context during generation.

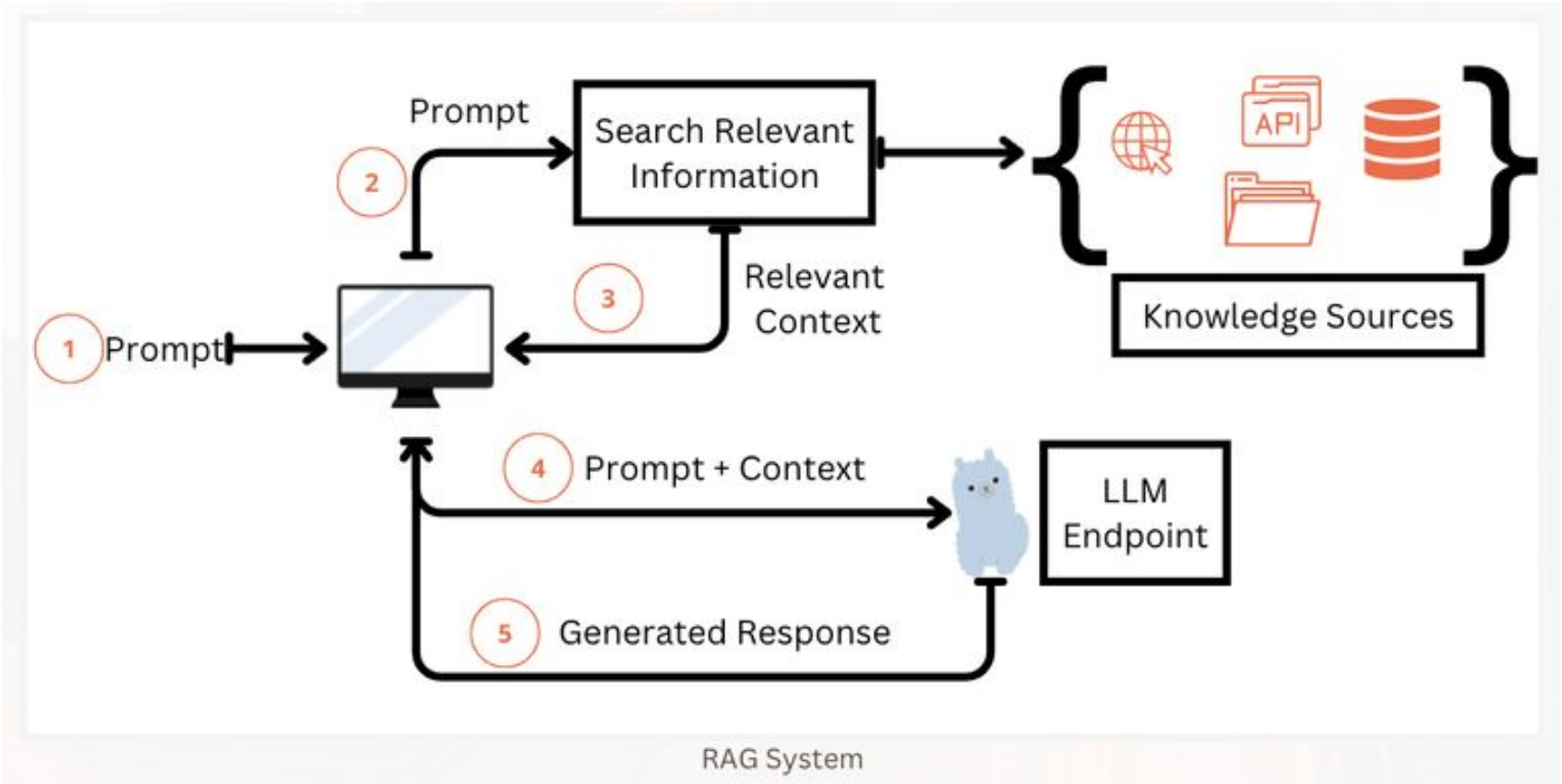
- Concept: Retrieve relevant documents from an external corpus before generation.
- Steps: 1. Input query/topic. 2. Retrieve documents. 3. Generate summary/response based on retrieved documents.
- Pros: Simple and effective for factual tasks.
- Cons: May lack fluency and originality due to heavy reliance on retrieved text.



(Ref: Progression of RAG Systems - Abhinav Kimothi)

YHK

RAG Architecture



(Ref: RAG Architecture -Abhinav Kimothi)

RAG Workflow

1. User writes a prompt or a query that is passed to an orchestrator
2. Orchestrator sends a search query to the retriever
3. Retriever fetches the relevant information from the knowledge sources and sends back
4. Orchestrator augments the prompt with the context and sends to the LLM
5. LLM responds with the generated text which is displayed to the user via the orchestrator

Two pipelines become important in setting up the RAG system. The first one being setting up the knowledge sources for efficient search and retrieval and the second one being the five steps of the generation.

Indexing Pipeline

Data for the knowledge is ingested from the source and indexed. This involves steps like splitting, creation of embeddings and storage of data.

Generation Pipeline

This involves the actual RAG process which takes the user query at run time and retrieves the relevant data from the index, then passes that to the model

(Ref: RAG Architecture -Abhinav Kimothi)

YHK

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

(Ref: Progression of RAG Systems - Abhinav Kimothi)

YHK

RAG vs SFT (Supervised Fine - Tuning)

RAG & SFT: Complementary Techniques

RAG Features

- Connects to dynamic external data sources.
- Reduces hallucinations.
- Increases transparency in source of information.
- Works well with very large foundation models.
- Does not impact style, tone, vocabulary.

SFT Features

- Changes style, vocabulary, tone of foundation model.
- Can reduce model size.
- Useful for deep domain expertise.
- May not address hallucinations.
- No improvement in transparency (black box models).

YHK

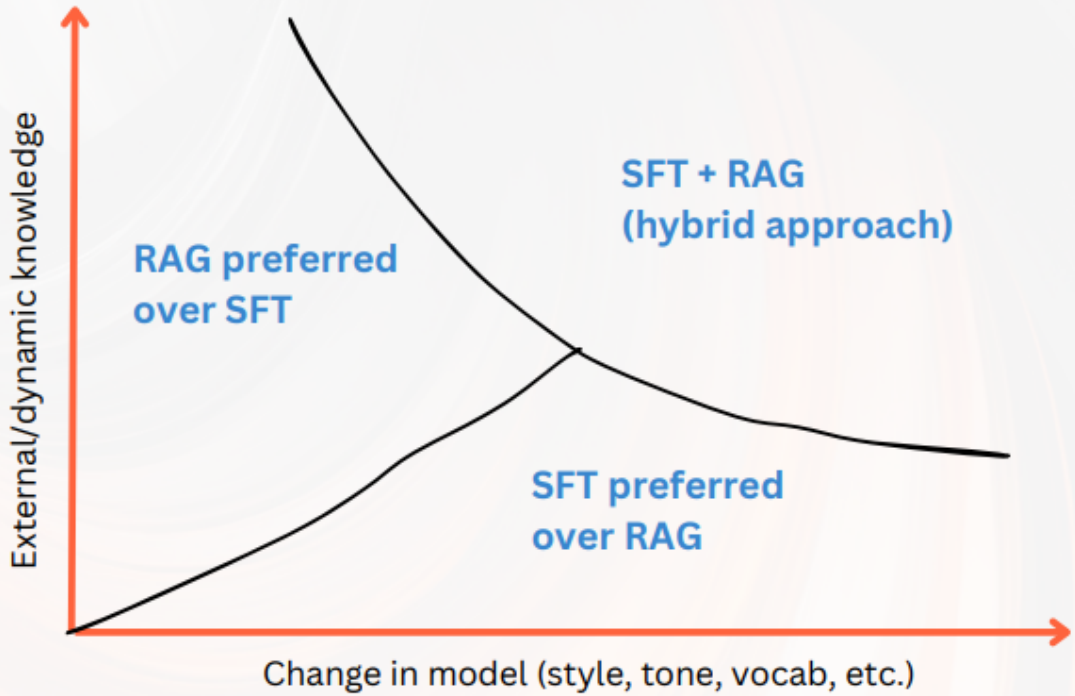
Important Use Case Considerations

Do you require usage of dynamic external data?

RAG preferred over SFT

Do you require changing the writing style, tonality, vocabulary of the model?

SFT preferred over RAG



(Ref: Generative AI with Large Language Model - Abhinav Kimothi)

YHK

Other Considerations

- **Latency:** RAG introduces inherent latency due to search and retrieval.
- **Scalability:** RAG pipelines are modular and scalable; SFT requires retraining.
- **Cost:** Both methods require upfront investment; costs vary.
- **Expertise:** RAG pipelines are moderately simple with frameworks; SFT needs deep understanding and training data creation.

YHK

Applications

Applications and Use Cases

- Code generation in software development
- Creative writing and storytelling
- Educational material generation
- Personalized product descriptions
- many many more . . .

YHK

Open Source Resources and Tools

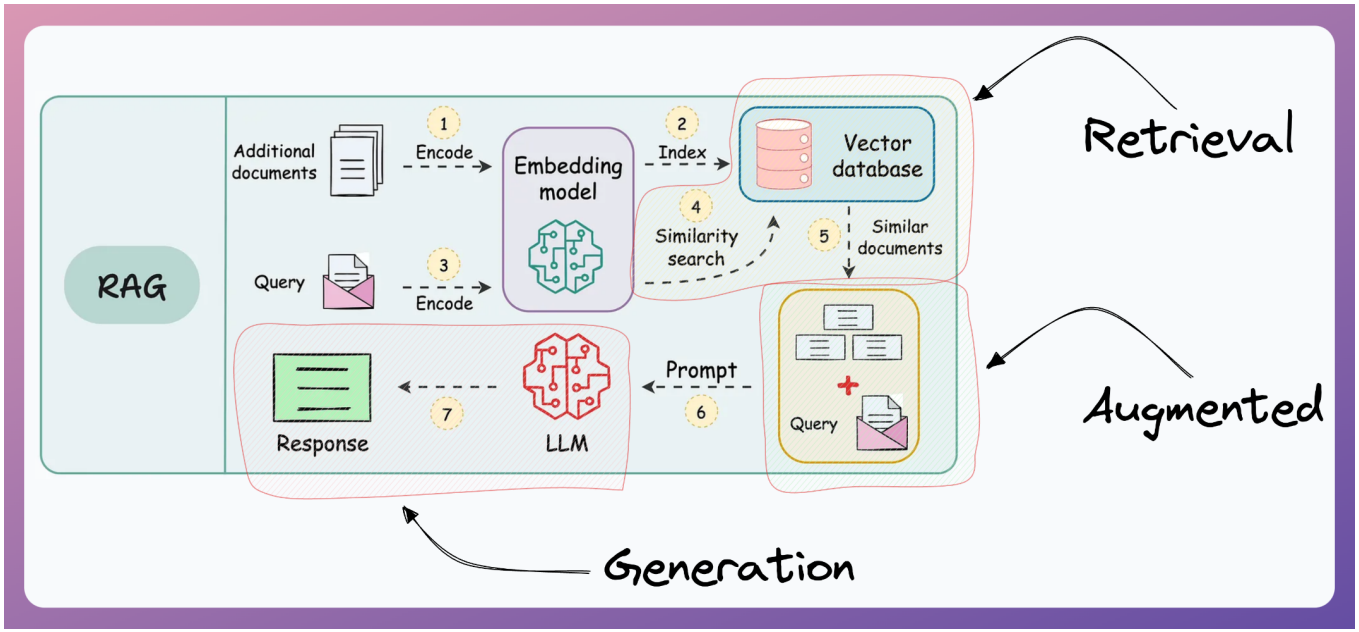
- LangChain, LlamaIndex like frameworks
- Transformers library, Hugging Face model hub
- Datasets and evaluation benchmarks

YHK

Conclusions

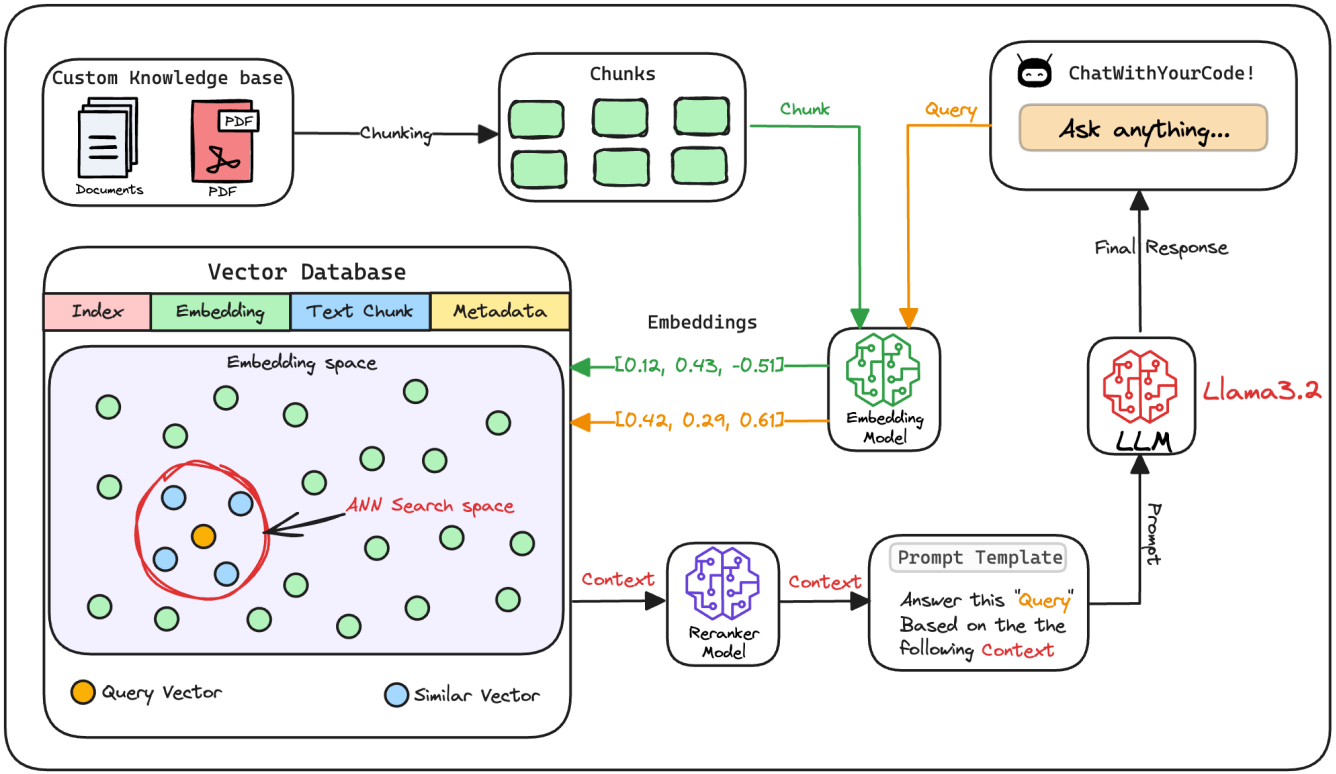
So, What is RAG?

- A new paradigm for generation tasks, combining retrieval and generation models.
- Motivation: Overcoming limitations of pure generative models in factual consistency, efficiency, and diversity.
- Impact: Improved performance in text summarization, question answering, and other NLP tasks.



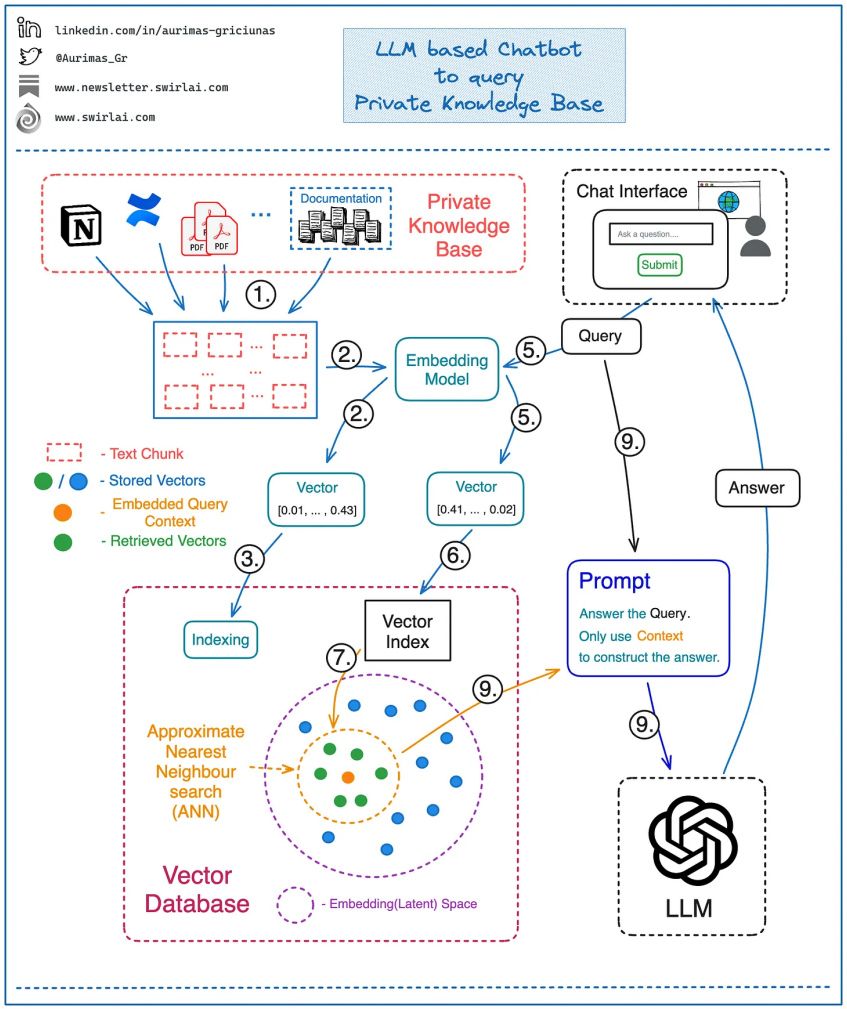
(Ref: A Crash Course on Building RAG Systems -Daily Dose)

RAG Workflow



(Ref: A Crash Course on Building RAG Systems -Daily Dose)

RAG Architecture



(Ref: Overview of Large Language Models - Aman AI)

RAG Architecture

- **Powerful Enhancement:**
  - RAG provides additional memory and context.
  - Increases confidence in LLM responses.
- **Architecture:**
  - Two essential pipelines for an effective RAG system.
- **Indexing Pipeline:**
  - Retrieves knowledge from diverse sources.
  - Enables loading, splitting, and embedding creation for offline use.
  - Can be on-the-fly for lower volume and single-use scenarios.
- **Generation Pipeline:**
  - Retriever fetches information from the knowledge base.
  - Retrieved data augments user prompt and is sent to the LLM.
  - LLM generates text and sends the response back to the user.
- **Evaluation Pipeline:**
  - Optional pipeline for assessing groundedness and relevance of responses.

(Ref: RAG Architecture -Abhinav Kimothi)

YHK

The Core RAG Bottleneck

- 90% of RAG systems fail due to poor retrieval.
- Better LLMs do not fix bad retrieval.
- Naive RAG: Chunk, embed, retrieve top_k — too simplistic.
- Poor retrieval causes hallucinations, omissions, and vague outputs.
- Quality of generation depends on quality of retrieved context.

YHK

Step 1: Fix the Basics

- Use dynamic chunking that respects document structure.
- Tune chunk size to avoid loss or fragmentation.
- Apply metadata filtering to narrow semantic scope.
- Combine vector search with keyword filters (hybrid).

YHK

Step 2: Advanced Retrieval Techniques

- Apply re-ranking (learned or rule-based).
- Use small-to-big retrieval: start with sentences.
- Employ recursive retrieval methods like LlamaIndex.
- Use multi-hop and agentic retrieval for complex queries.

YHK

Step 3: Evaluate or Die Trying

- Don't iterate blindly — evaluate systematically.
- Use end-to-end evaluation with ground truths or feedback.
- Run component-level evals using MRR, NDCG, success@k.

YHK

Step 4: Fine-Tuning — A Last Resort

- Fine-tune only if domain is highly specific or LLM is weak.
- Ensure retrieval and prompting are fully optimized first.
- Be aware of added cost, latency, and system complexity.

YHK

Pros and Cons of RAG

- Pros: Improved factual accuracy, diversity, and efficiency compared to pure generation.
- Cons: Increased complexity, potential bias introduced by the retrieval component, and dependency on external corpus quality.

YHK

Challenges and Future Directions

- Bias Mitigation: Addressing potential biases introduced by the retrieval component.
- Domain Adaptation: Adapting RAG models to specific domains with limited training data.
- Interpretability Enhancement: Understanding the reasoning behind generated outputs for better model debugging.

YHK

References

Many publicly available resources have been refereed for making this presentation. Some of the notable ones are:

- Retrieval Augmented Generation - Medium blogs by Abhinav Kimothi
- Mastering RAG: How To Architect An Enterprise RAG System - Pratik Bhavsar
- Prompt Engineering Guide https://www.promptingguide.ai/techniques/rag
- 3 Day RAG Roadmap: Understanding, Building and Evaluating RAG Systems 2024 - Aishwarya Naresh Reganti
- Explanation of RAG by DeepLearning AI
- What is RAG by DataStax
- Advanced RAG series (6 videos) by Sam Witteveen
- LangChain101: Question A 300 Page Book (w/ OpenAI + Pinecone) by Greg Kamradt
- Blog on advanced RAG techniques by Akash
- RAG hands-on tutorials on GitHub gkamradt/langchain-tutorials/

YHK

Thanks . . .

- Office Hours: Saturdays, 3 to 5 pm (IST); Free-Open to all; email for appointment to yogeshkulkarni at yahoo dot com
- Call + 9 1 9 8 9 0 2 5 1 4 0 6

(https://www.linkedin.com/in/yogeshkulkarni/)

(https://medium.com/@yogeshharibhaukulkarni)

(https://www.github.com/yogeshhk/)

YHK