Agentic Retrieval Systems with Qdrant

Presented to you by Mohamed Arbi Nsibi

08/02/2026

Mohamed Arbi Nsibi

- ML engineer
- Qdrant Star ⭐
- Former GDSC Lead 23/24

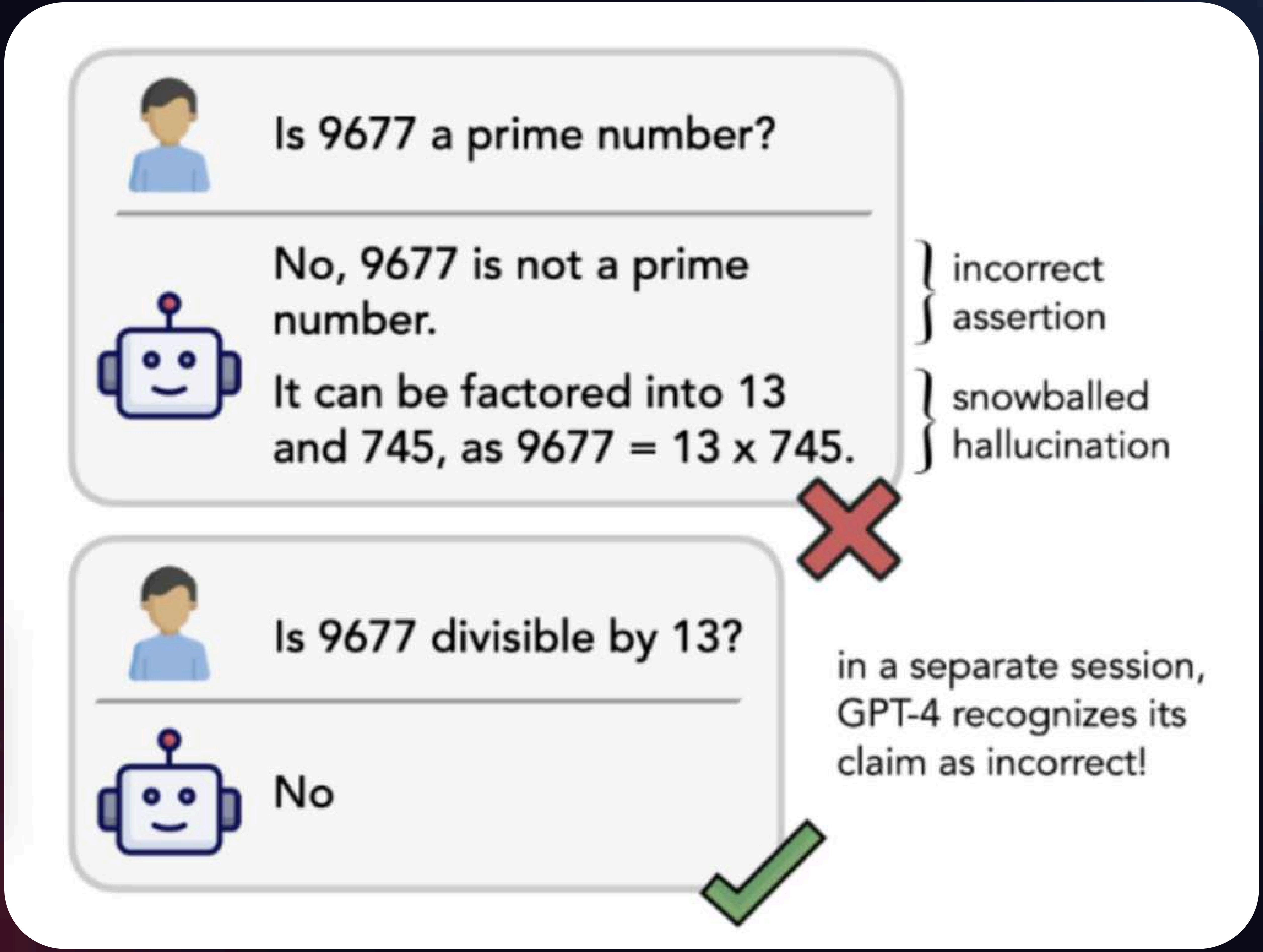
Content

- Motivation
- RAG components
- Vector stores deep dive
- Building basic RAG pipeline
- What makes an AI an Agent ?
- Demo
- Some Advanced RAG Techniques

1- The need for an external Knowledge !

2- Hallucinations

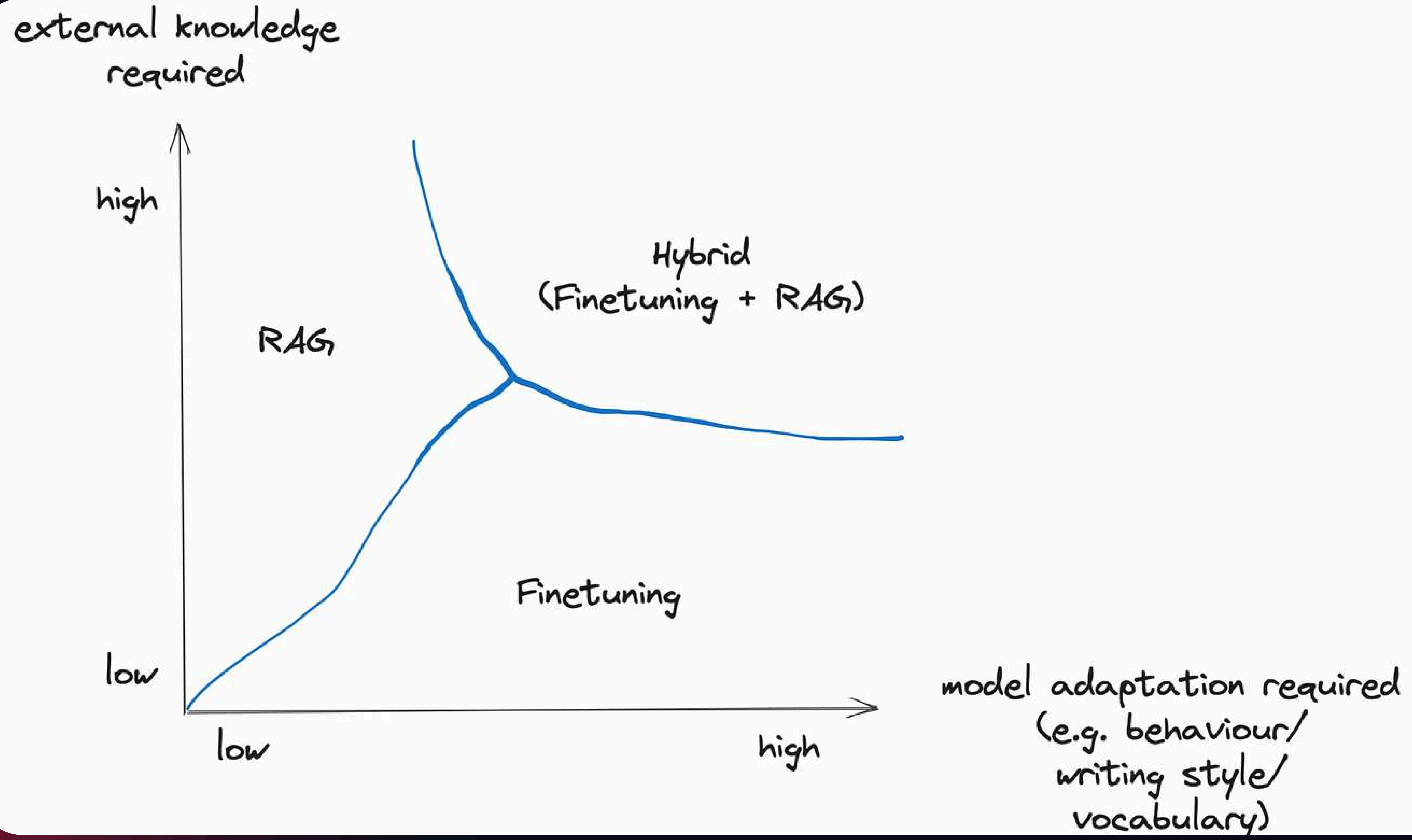
Hallucinations

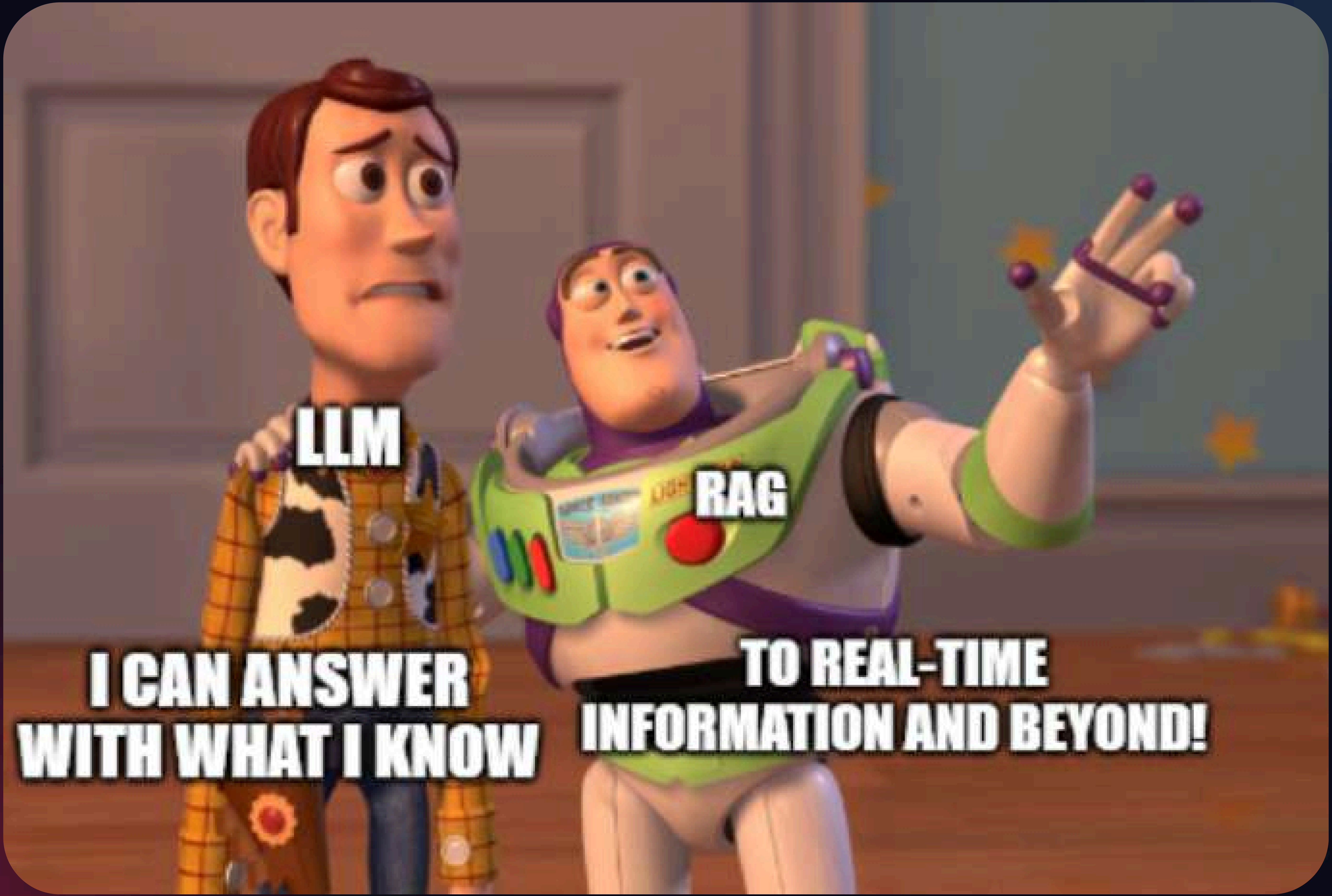


Hallucinations

- The model is not trained on enough data.
- The model is trained on noisy or dirty data.
- The model is not given enough context .
- The model is not given enough constraints (rules, guidelines, or limitations)

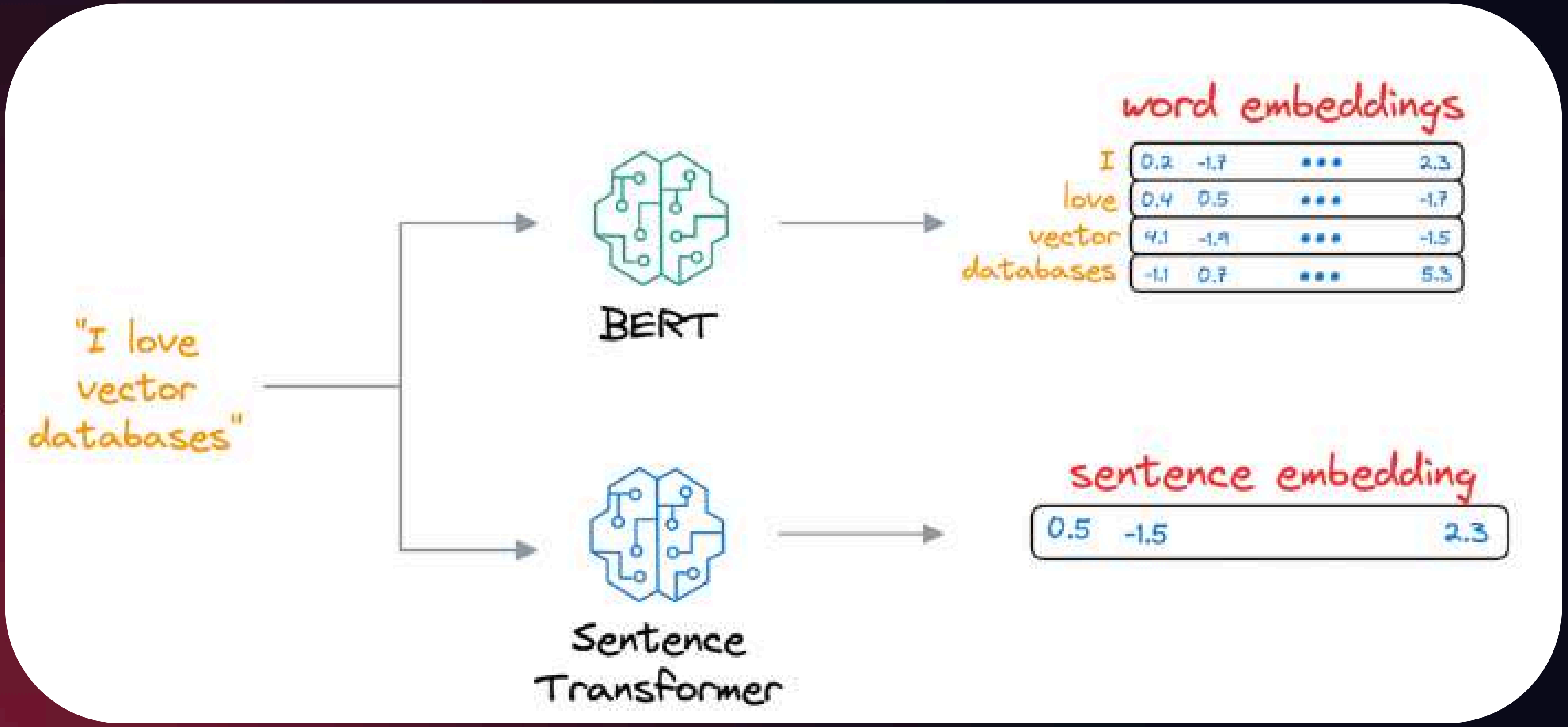
RAG / Fine-tuning



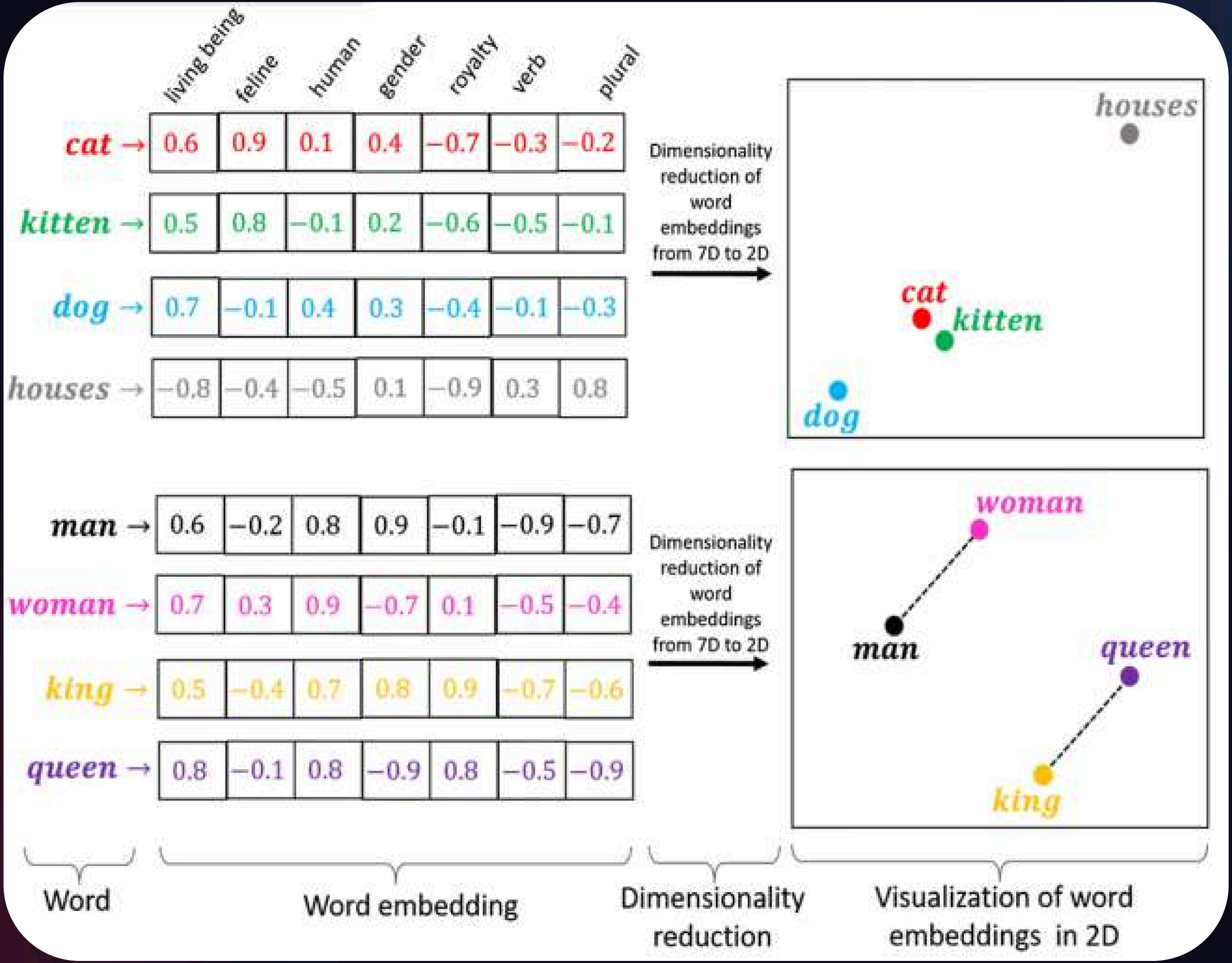

LLM
RAG
I CAN ANSWER
WITH WHAT I KNOW
TO REAL-TIME
INFORMATION AND BEYOND!

Embedding model

- These vectors live in a high-dimensional space where the proximity between vectors reflects the **relatedness** of the original items.

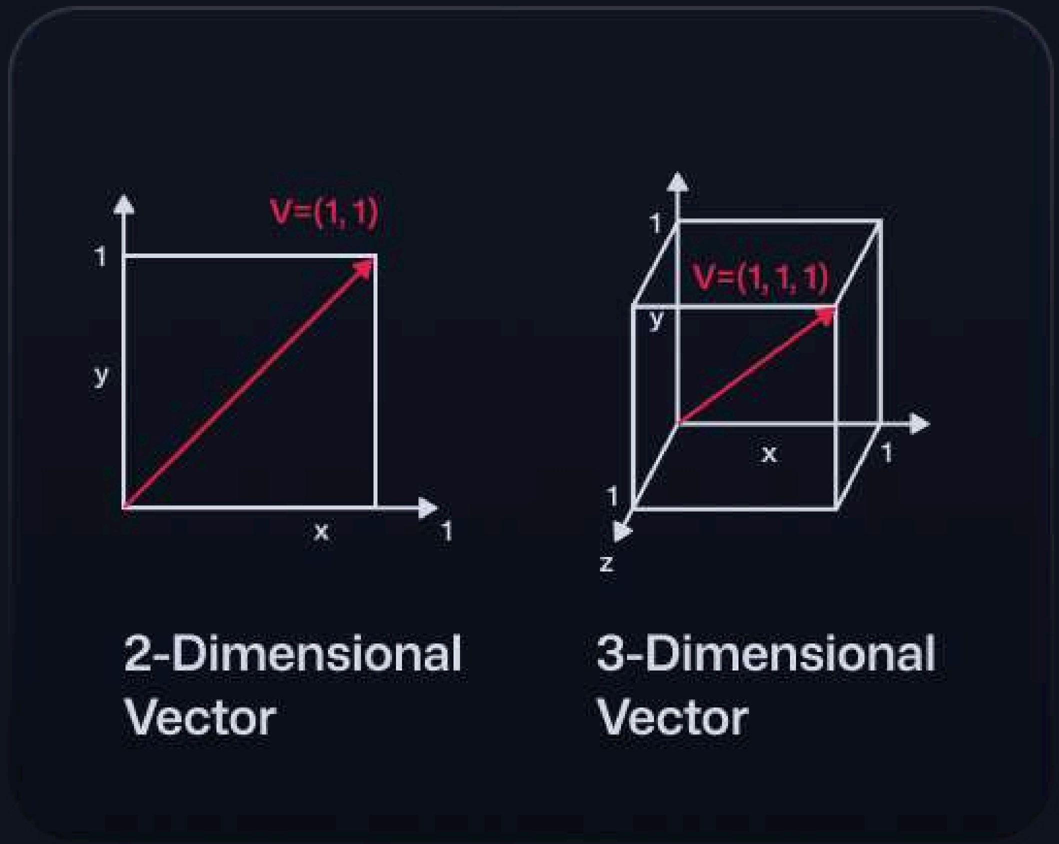


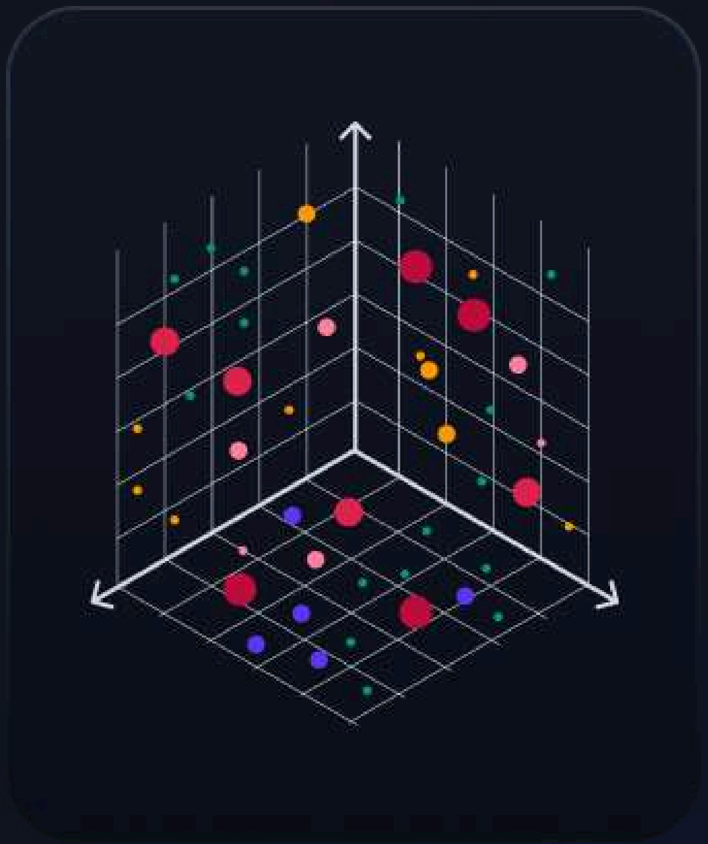
- Embedding model trained along LLM and learn to produce representation (vectors) based on context in word appear.

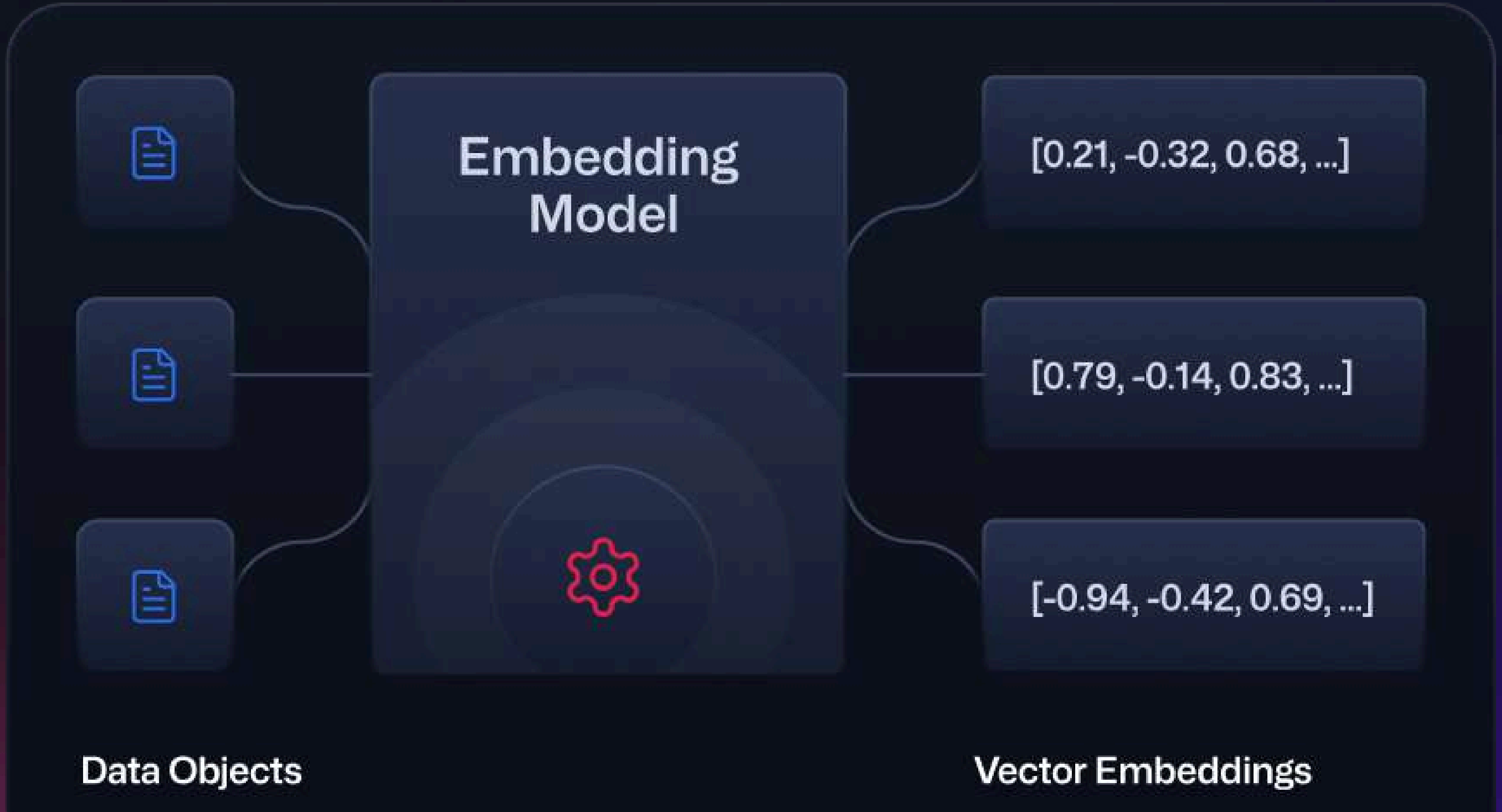

living being
feline
human
gender
royalty
verb
plural
cat → 0.6 0.9 0.1 0.4 −0.7 −0.3 −0.2
kitten → 0.5 0.8 −0.1 0.2 −0.6 −0.5 −0.1
dog → 0.7 −0.1 0.4 0.3 −0.4 −0.1 −0.3
houses → −0.8 −0.4 −0.5 0.1 −0.9 0.3 0.8
Dimensionality reduction of word embeddings from 7D to 2D
houses
cat
kitten
dog
man → 0.6 −0.2 0.8 0.9 −0.1 −0.9 −0.7
woman → 0.7 0.3 0.9 −0.7 0.1 −0.5 −0.4
king → 0.5 −0.4 0.7 0.8 0.9 −0.7 −0.6
queen → 0.8 −0.1 0.8 −0.9 0.8 −0.5 −0.9
Dimensionality reduction of word embeddings from 7D to 2D
woman
man
queen
king
Word
Word embedding
Dimensionality reduction
Visualization of word embeddings in 2D

Vector Search Basics

Two different vector embeddings should be close to each other if they represent a similar input object.
Embeddings are generated by neural networks and can represent thousands of dimensions.







Vector Search Basics

Although word counting produces embeddings, dense embeddings are needed to capture semantics

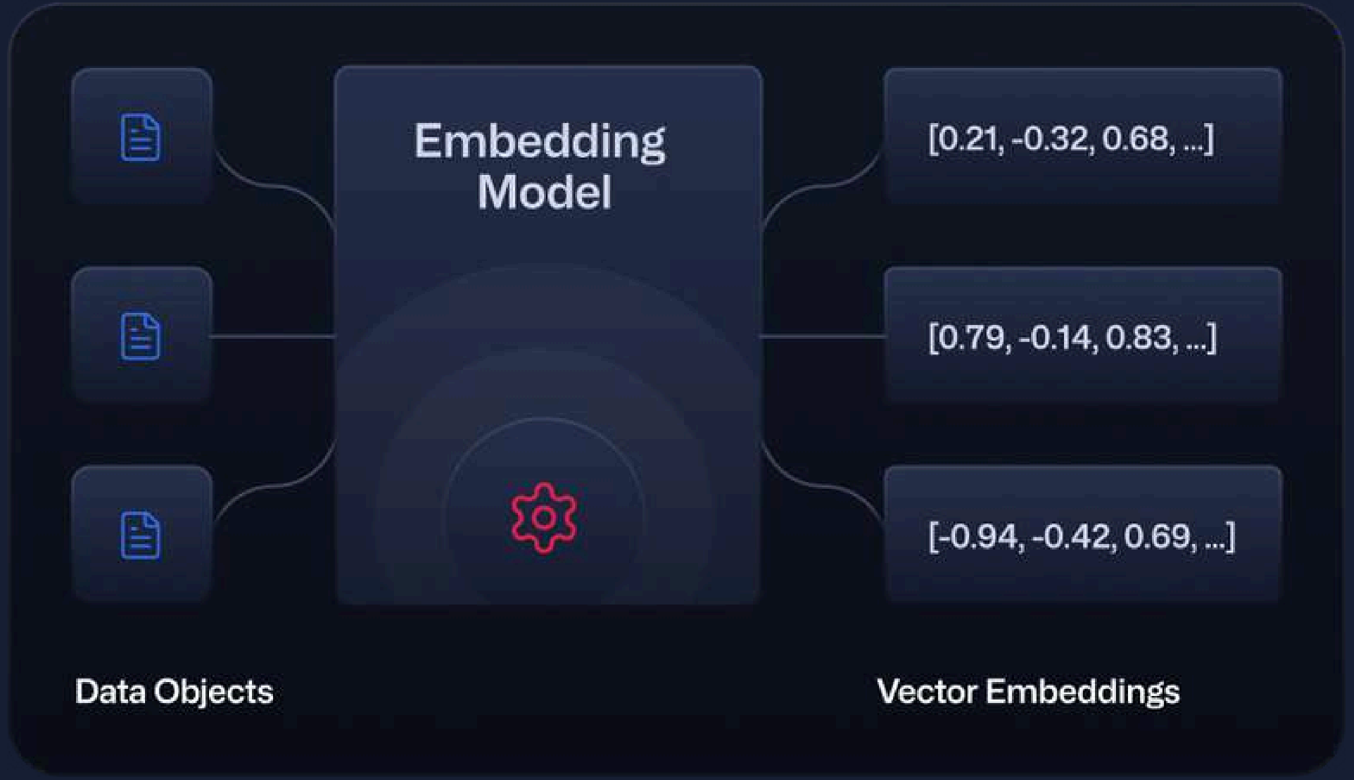
Sparse embedding:
e.g. One Hot Encoding

	an	another	embedding	is	this	Query Sim.
“this is an embedding”	[1,	0,	1,	1,	1]	3
“this is another embedding”	[0,	1,	1,	1,	1]	2

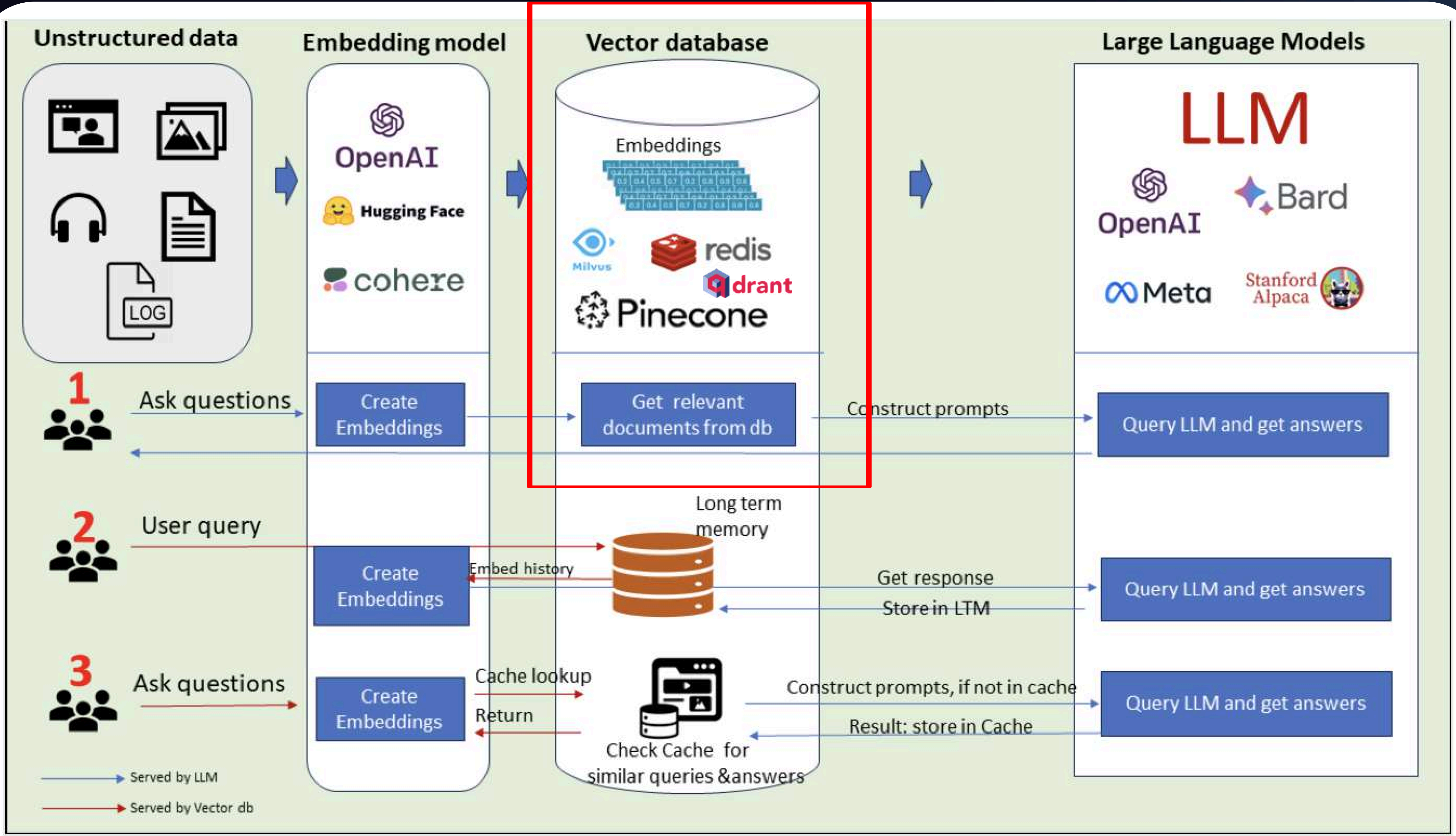
Query:

“What is an embedding?”

Dense embedding:
e.g. from BERT



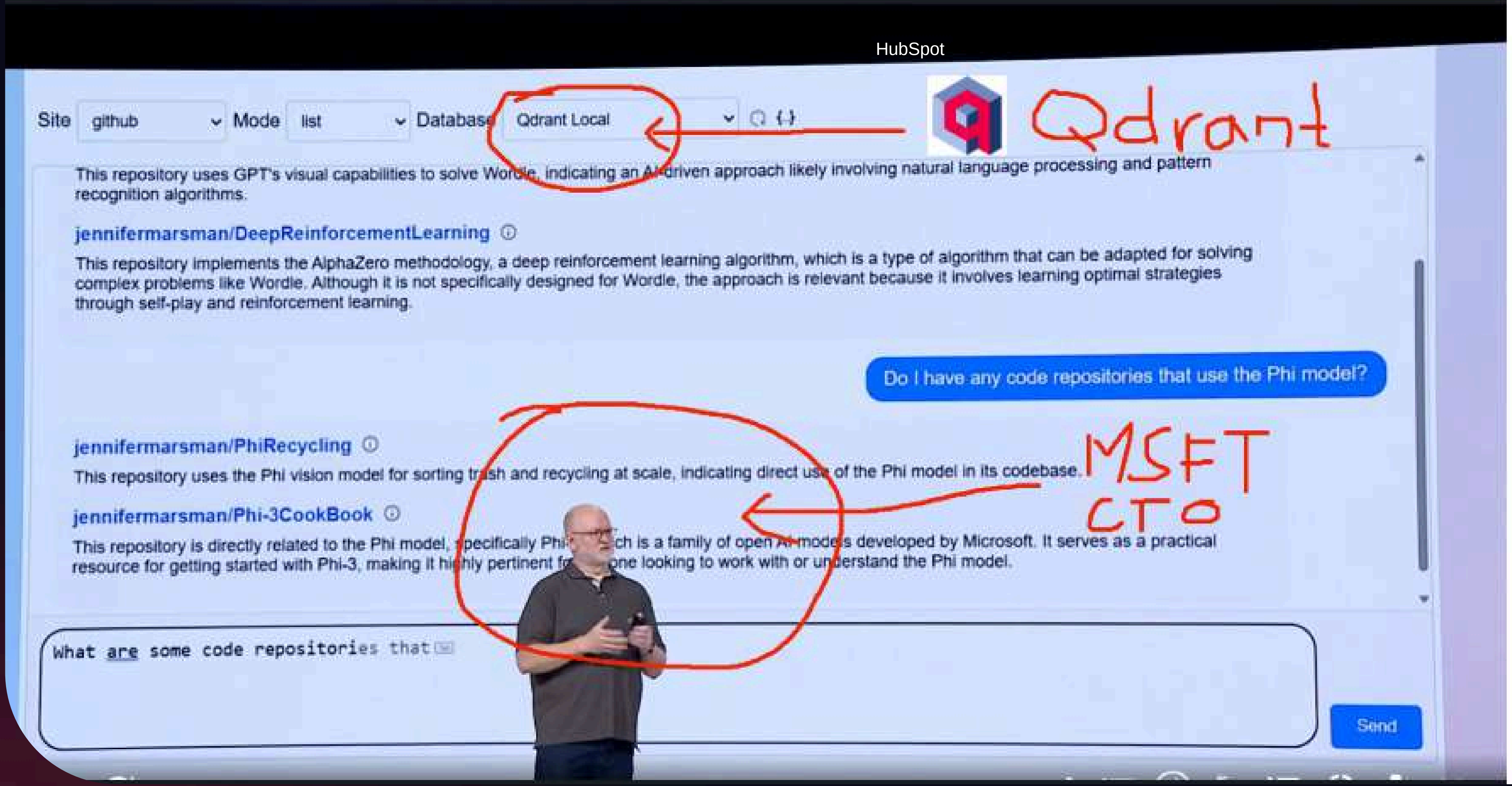
Vector Database

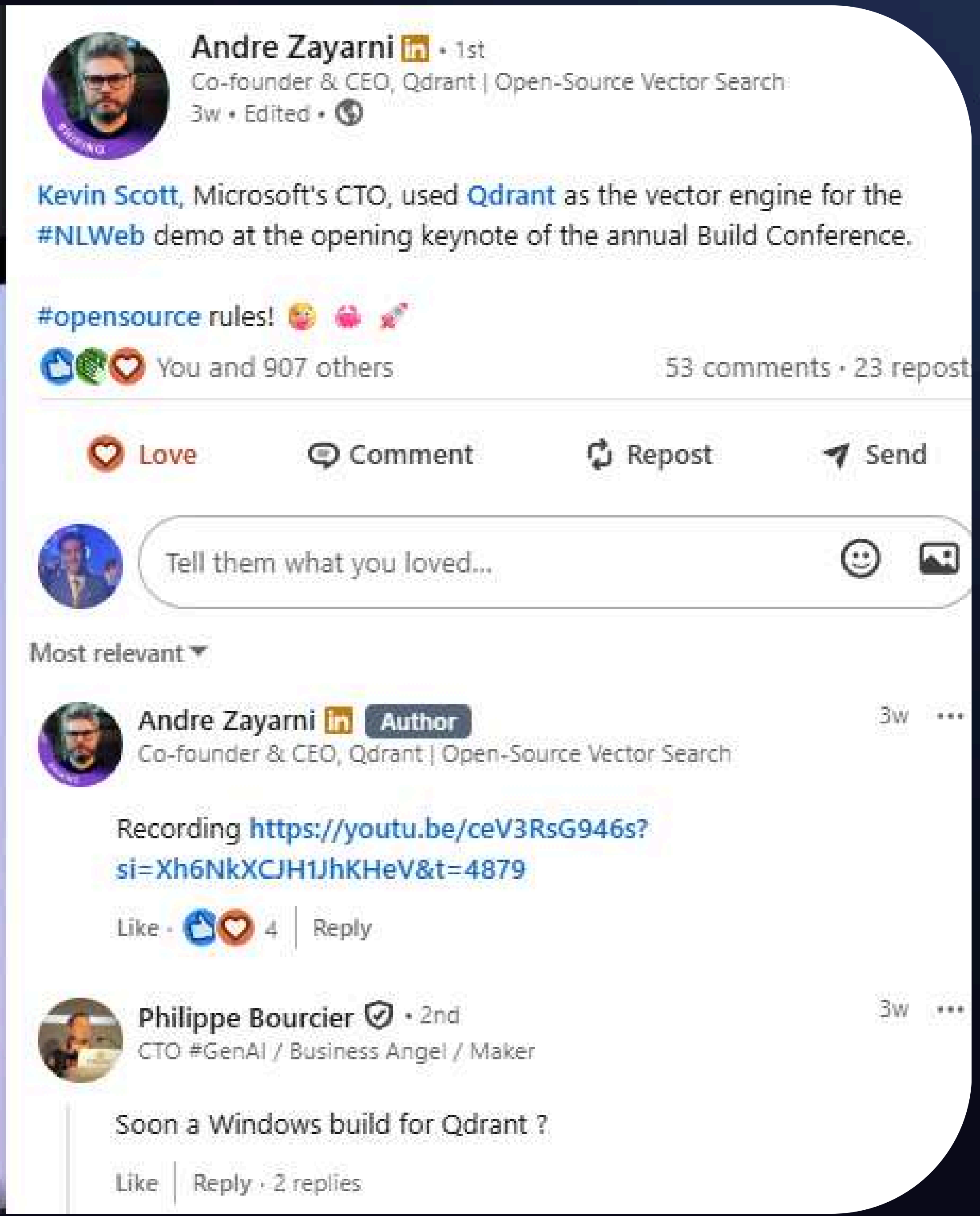


Fully Open-source
Self-hosting : run on ur own infra
Super fast: Latency ~0.024s (~24ms)
Hybrid Search
UI support
Free Tier ~1M(vectors) 768-dim vectors

Used by

NLWeb link

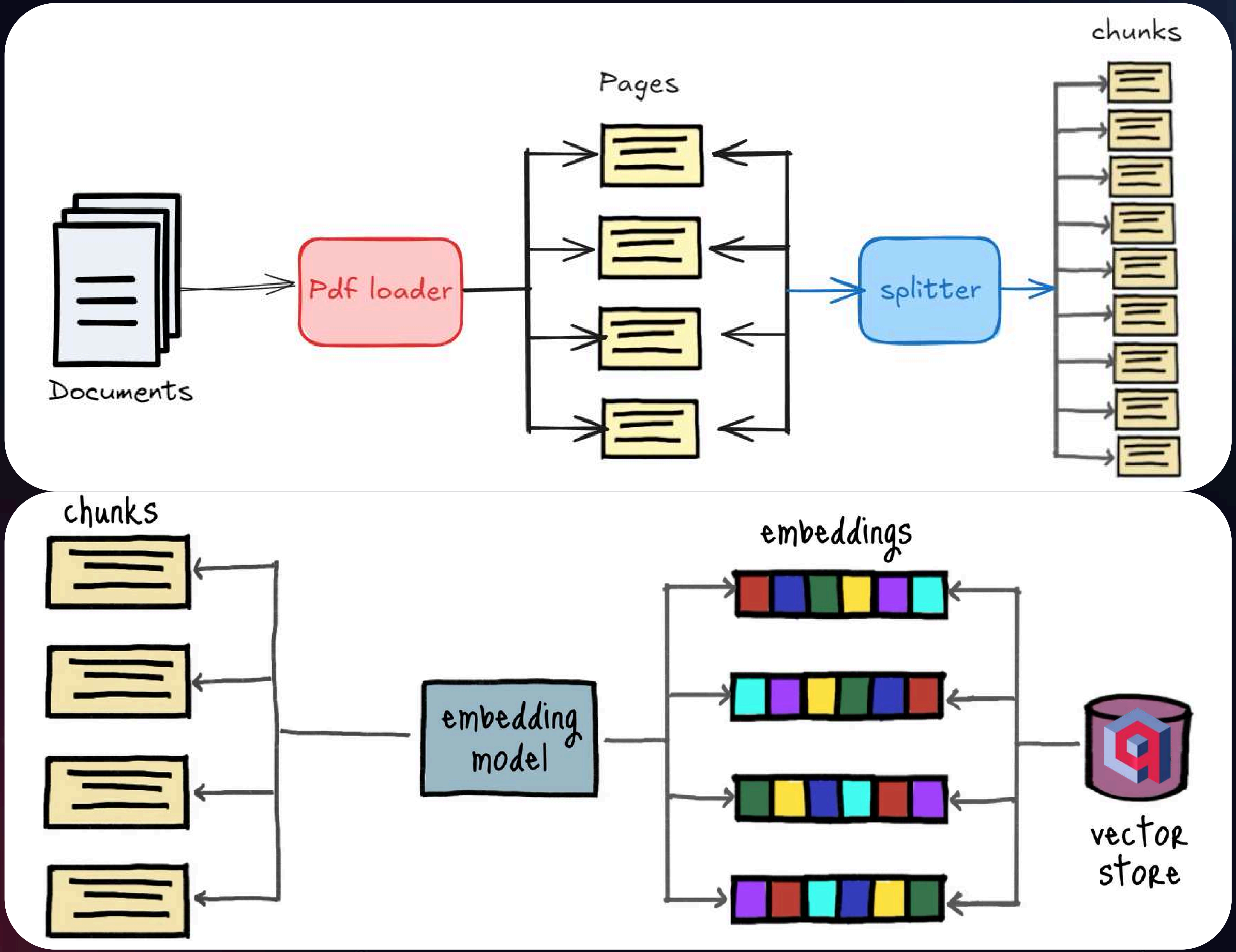


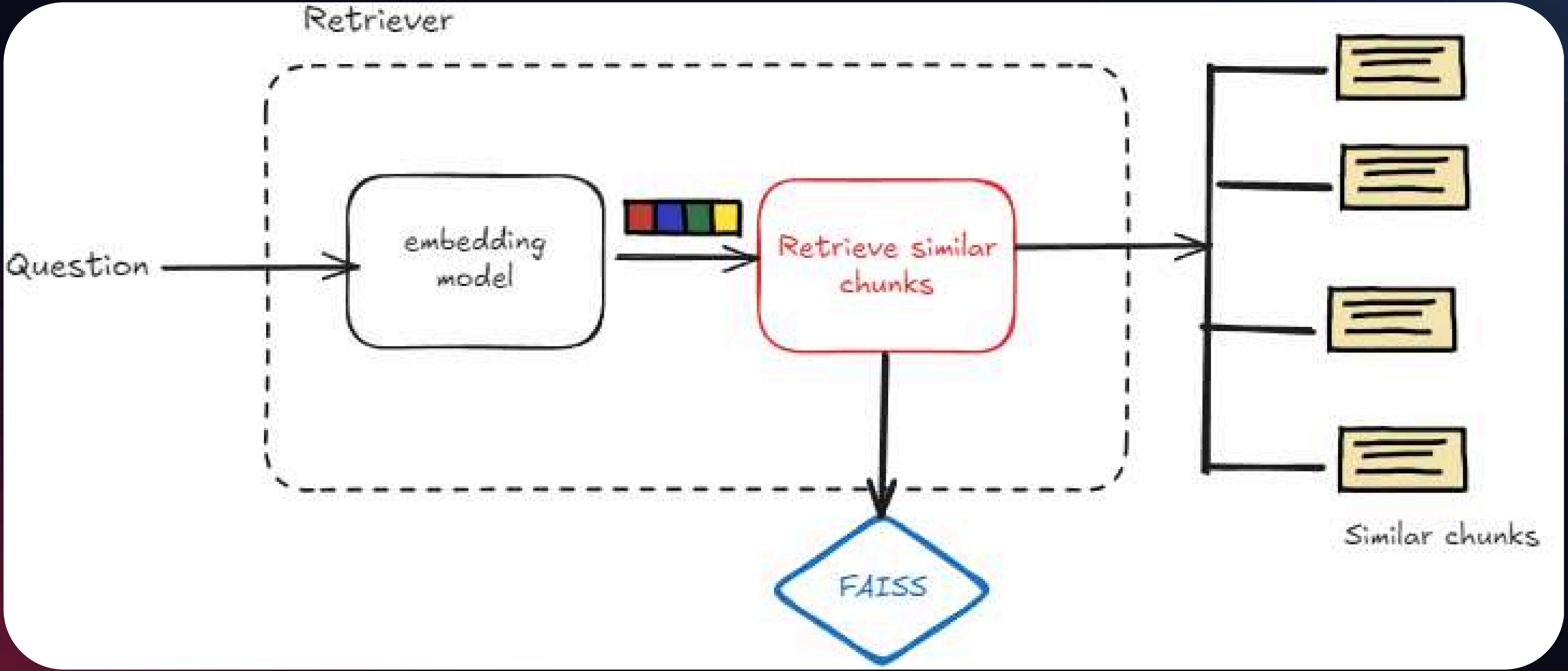


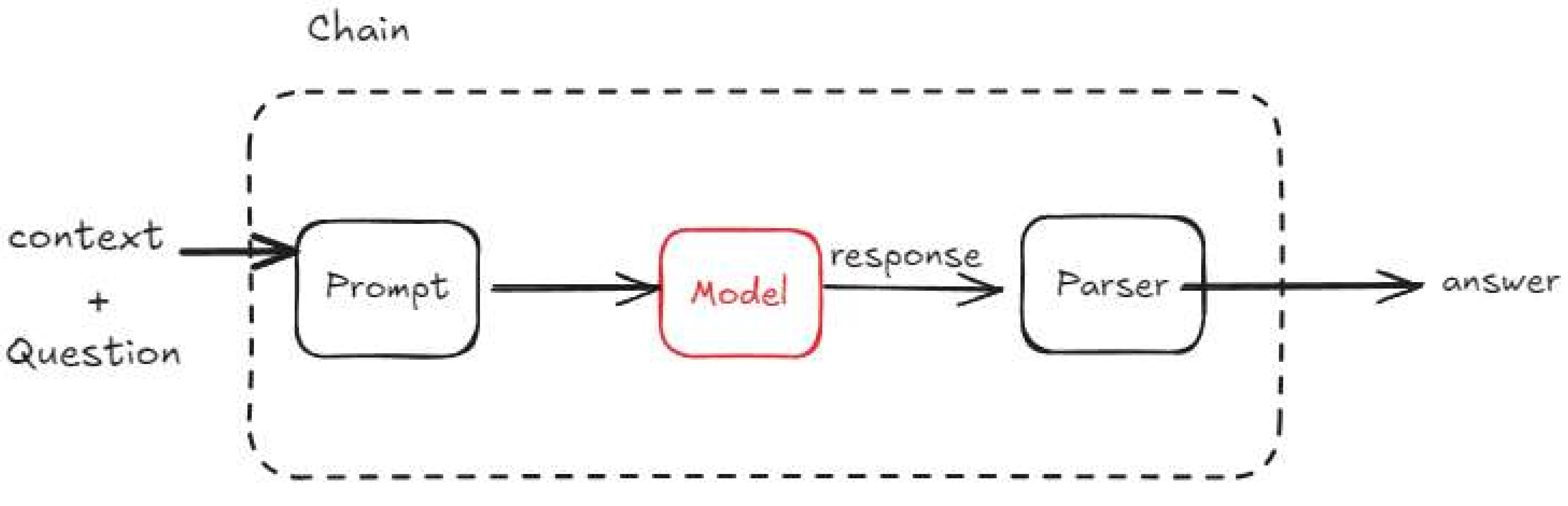
Why Qdrant is super fast ?

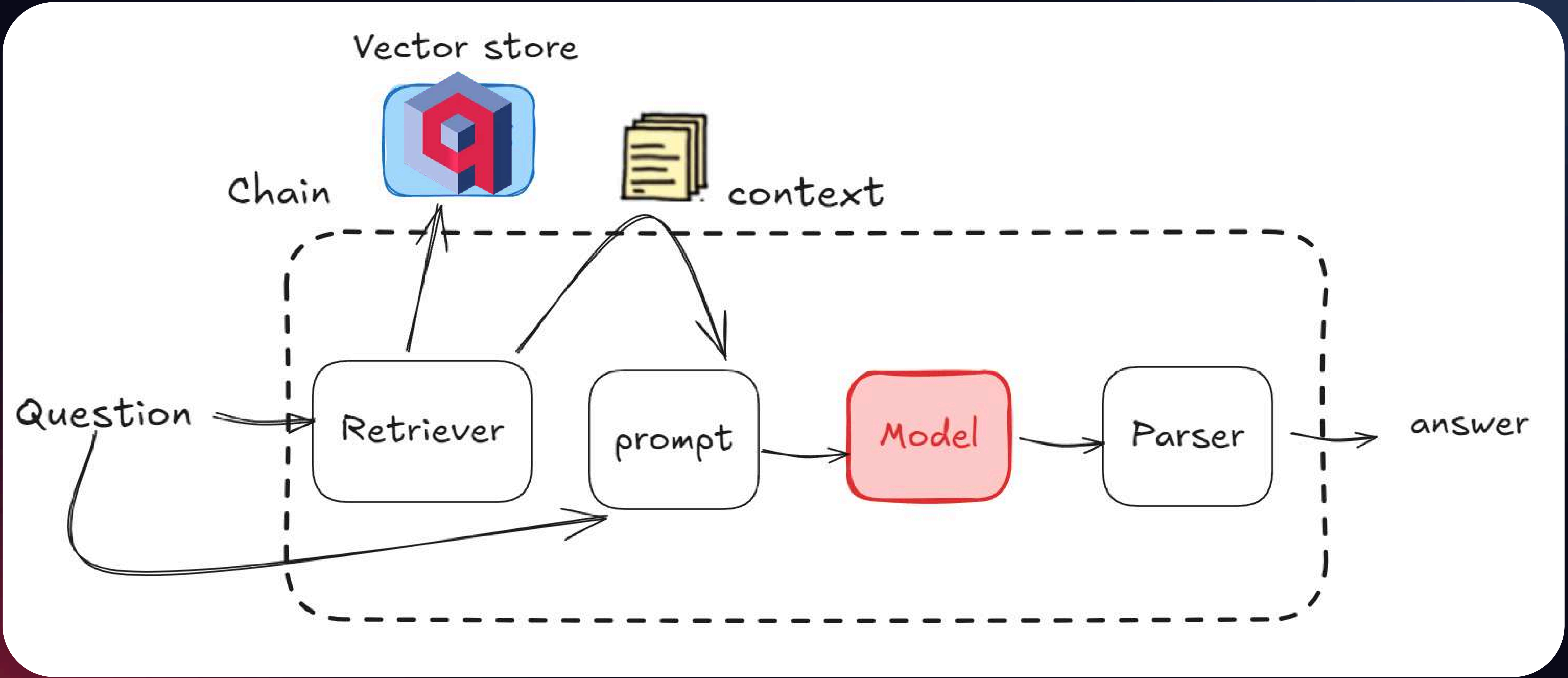
- Rust-Based Engine
- HNSW (Hierarchical Navigable Small World) Indexing : fast ANN search (on the best matches without scanning everything.)
- Vector Quantization (useful for large-scale datasets) : Saves RAM (up to 16x)
- Batch & Parallel Processing

RAG architecture




Retriever
Question
embedding model
Retrieve similar chunks
FAISS
Similar chunks


Chain
context
+
Question
Prompt
Model
response
Parser
answer



https://arxiv.org/abs/2005.11401

What makes an AI an Agent

```
{%- if tools %}
    {{- '<|im_start|>system\n' }}
    {%- if messages[0].role == 'system' %}
        {{- messages[0].content + '\n\n' }}
    {%- endif %}
    {{- "# Tools\n\nYou may call one or more functions to assist with the user query.\n\nYou are provided with function signatures within <tools></tools> XML tags:\n<tools>" }}
    {%- for tool in tools %}
        {{- "\n" }}
        {{- tool | tojson }}
    {%- endfor %}
    {{- "\n</tools>\n\nFor each function call, return a json object with function name and arguments within <tool_call></tool_call> XML tags:\n<tool_call>\n{\"name\": <function-name>, \"arguments\": <args-json-object>}\n</tool_call><|im_end|>\n" }}
{%- else %}
    {%- if messages[0].role == 'system' %}
        {{- '<|im_start|>system\n' + messages[0].content + '<|im_end|>\n' }}
    {%- endif %}
{%- endif %}
{%- for message in messages %}
    {%- if message.content is string %}
        {%- set content = message.content %}
    {%- else %}
        {%- set content = '' %}
    {%- endif %}
    {%- if (message.role == "user") or (message.role == "system" and not loop.first) %}
        {{- '<|im_start|>' + message.role + '\n' + content + '<|im_end|>' + '\n' }}
    {%- elif message.role == "assistant" %}
        {{- '<|im_start|>' + message.role + '\n' + content }}
        {%- if message.tool_calls %}
            {%- for tool_call in message.tool_calls %}
                {%- if (loop.first and content) or (not loop.first) %}
                    {{- '\n' }}
                {%- endif %}
                {%- if tool_call.function %}
                    {%- set tool_call = tool_call.function %}
                {%- endif %}
                {{- '<tool_call>\n{"name": "' }}
                {{- tool_call.name }}
                {{- '", "arguments": ' }}
                {%- if tool_call.arguments is string %}
                    {{- tool_call.arguments }}
                {%- else %}
```

- If a model can use tools or functions, you can classify it as an agentic AI model. A simple way to confirm this is to look at its chat_template.

What makes an AI an Agent

```python
class AdditionTool(Tool):
    """
    A class-based tool for adding two numbers.
    """
    name = "add_numbers"
    description = "Adds two numbers (integers or floats) together
and returns the result."
    inputs = {
        'a': {
            'type': "integer",
            'description': "The first number to add."
        },
        'b': {
            'type': "integer",
            'description': "The second number to add."
        },
    }
    output_type = "integer"

    def forward(self, a: int, b: int) -> int:
        """
        The core logic of the tool. This method is executed when the
tool is called.
        """
        return a + b

addition_tool = AdditionTool()
```

creating a subclass of Tool.

- Explicit control over schema.
- Good when you want strict typing, validation, or more complex tools.

```python
@tool
def add_numbers(a: int, b: int) -> int:
    """
    Adds two numbers (integers or floats) together and returns the
result.
    Args:
        a: The first number to add.
        b: The second number to add.

    Returns:
        The sum of the two input numbers."""
    return a + b
```

decorator based

- Short.
- Simple.
- Feels natural when the tool is just a single operation.
- Ideal for quick tools

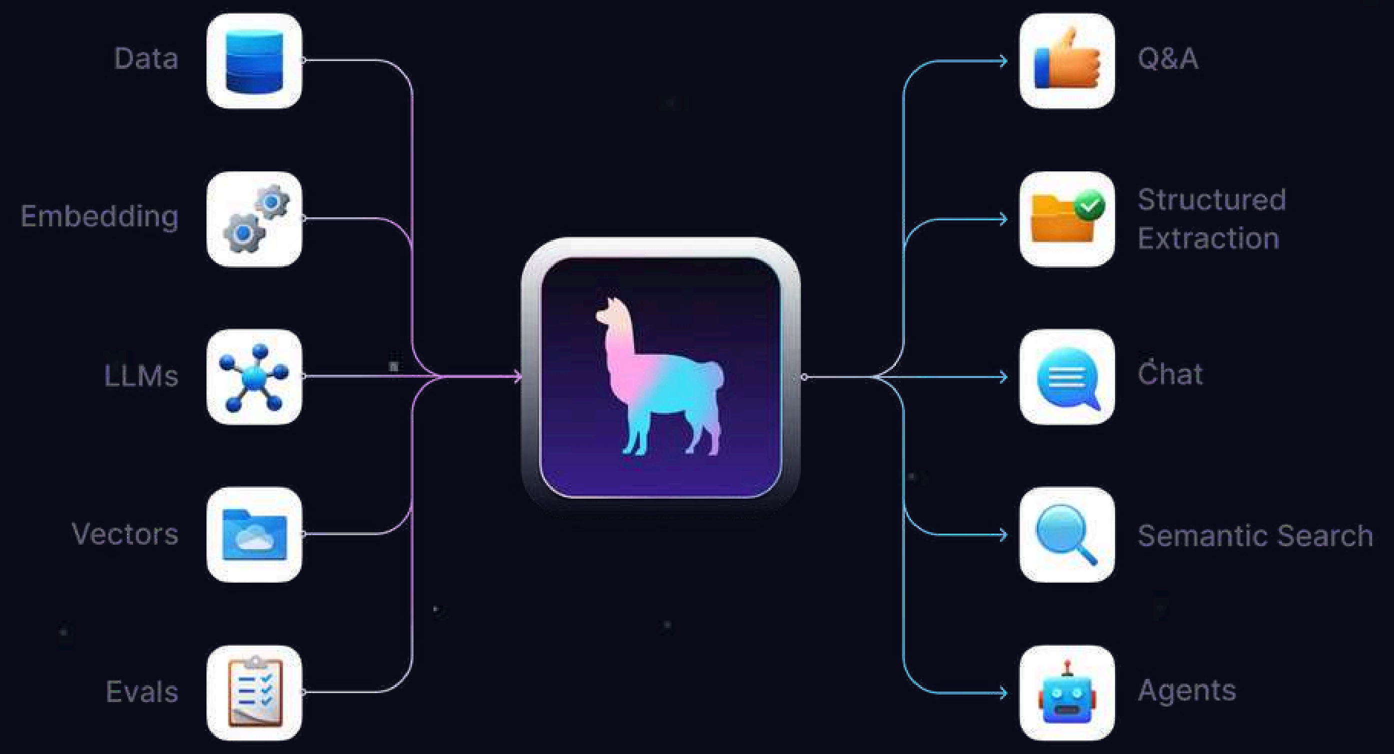
How to get started ?

LangChain

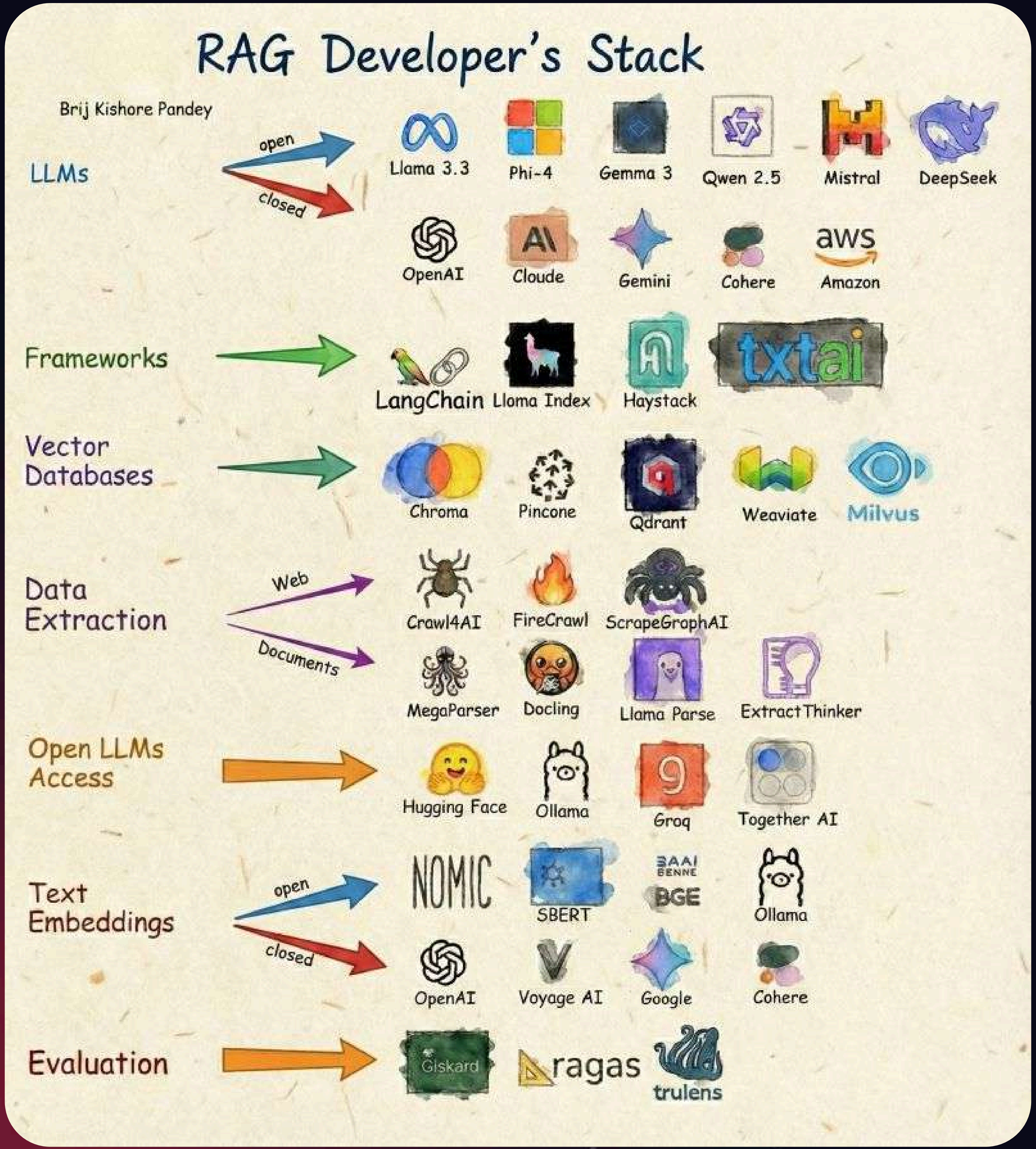
- LangChain is a framework designed to simplify the creation of applications using large language models.

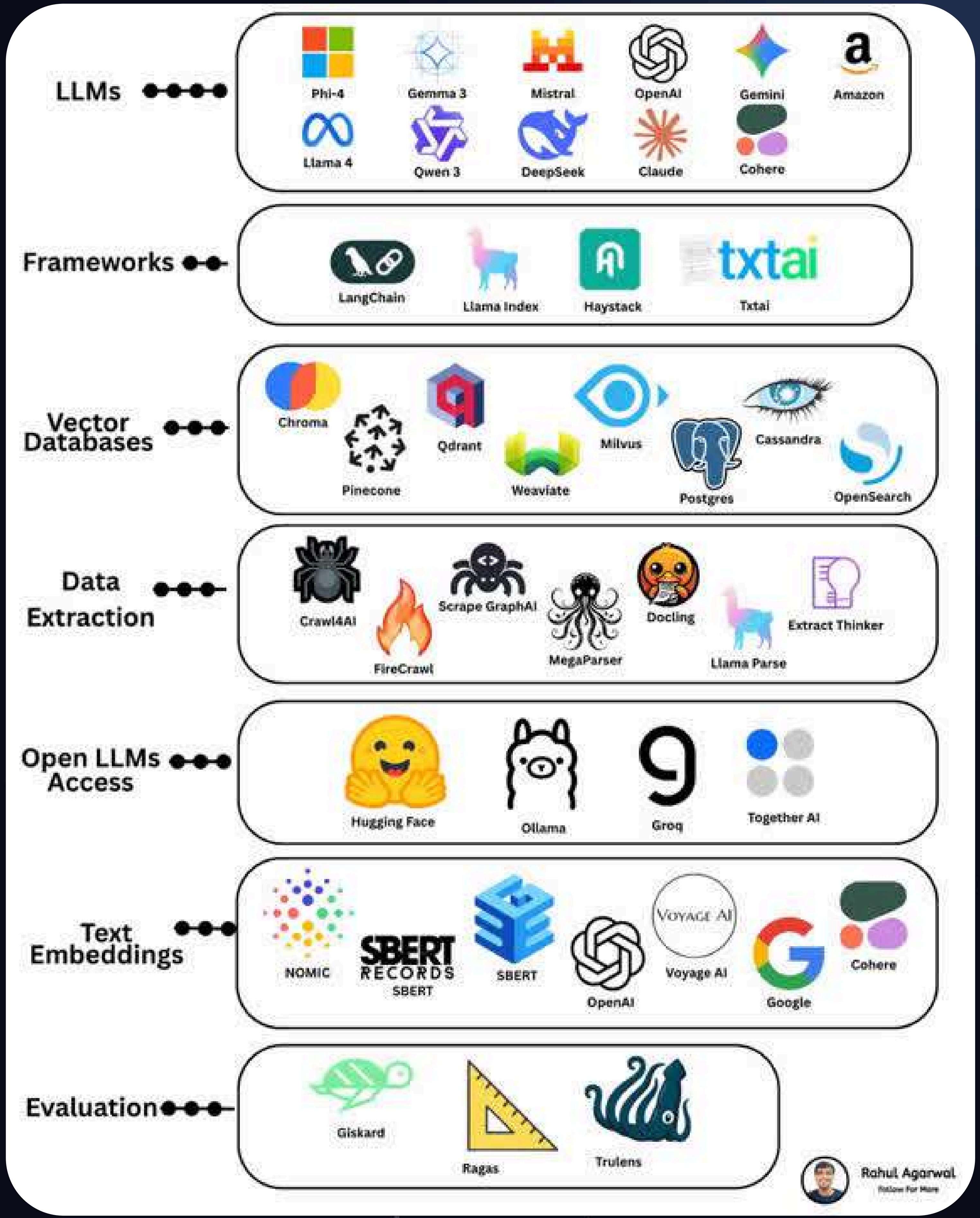
LlamaIndex

- LlamaIndex is a handy tool that acts as a bridge between your custom data and large language models (LLMs) which are powerful models capable of understanding human-like text.



How to get started ?





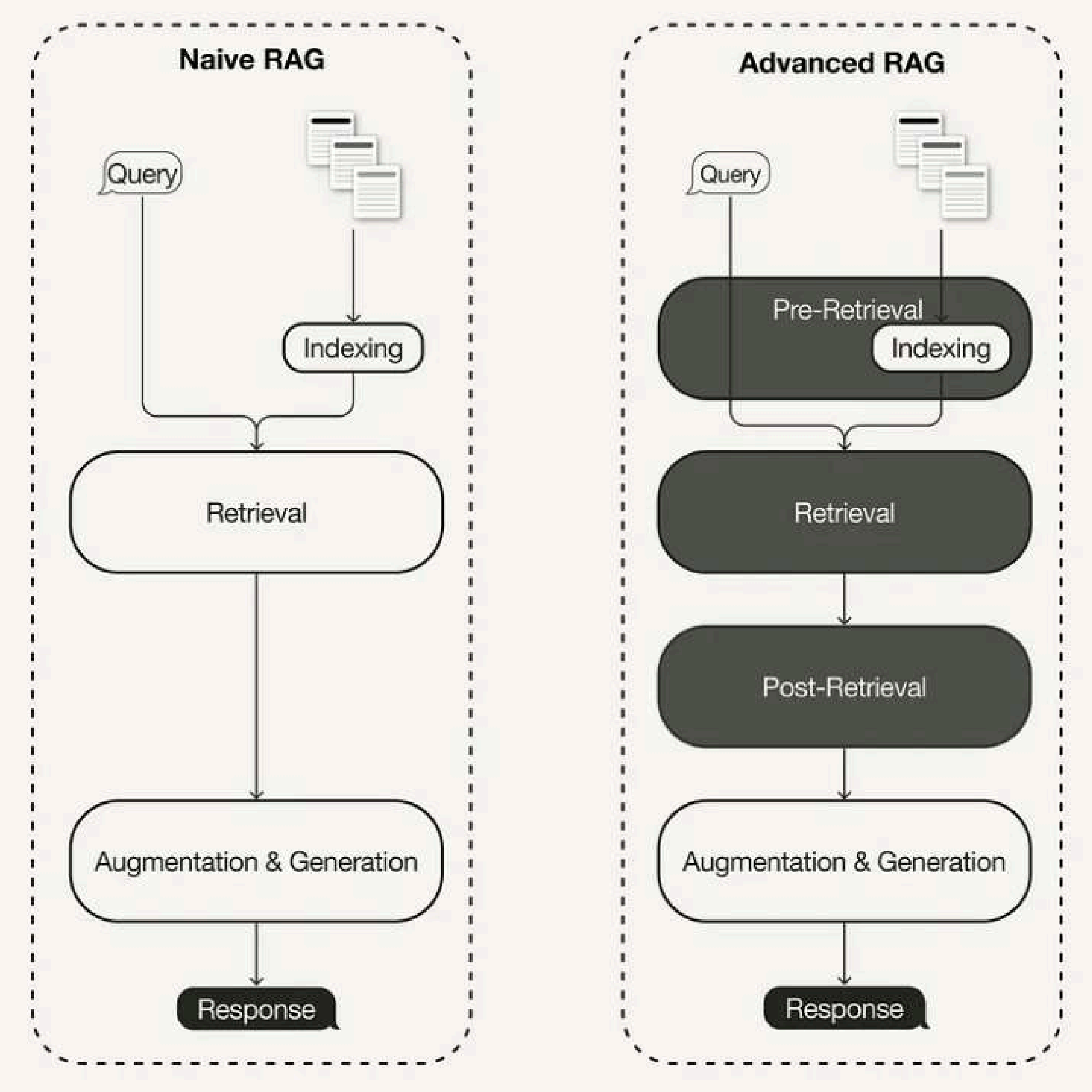
smolagents

DEMO

[smollagents]

Naive RAG vs Advanced RAG

- There are many implementation to further improve performance of Naive RAG.
- Advanced RAG has evolved as a new paradigm with targeted enhancements to address some of the limitations of the naive RAG paradigm.
  - Advanced RAG techniques can be categorized into
    - pre-retrieval optimization,
    - retrieval optimization, and
    - post-retrieval optimization
- some examples :
  - Feedback loops (re-ranking, similarity score thresholds)
  - Hybrid Search (dense + keyword)
  - Contextual compression (summarize before feeding to LLM)
  - Multi-vector per chunk (dense embeddings per aspect)

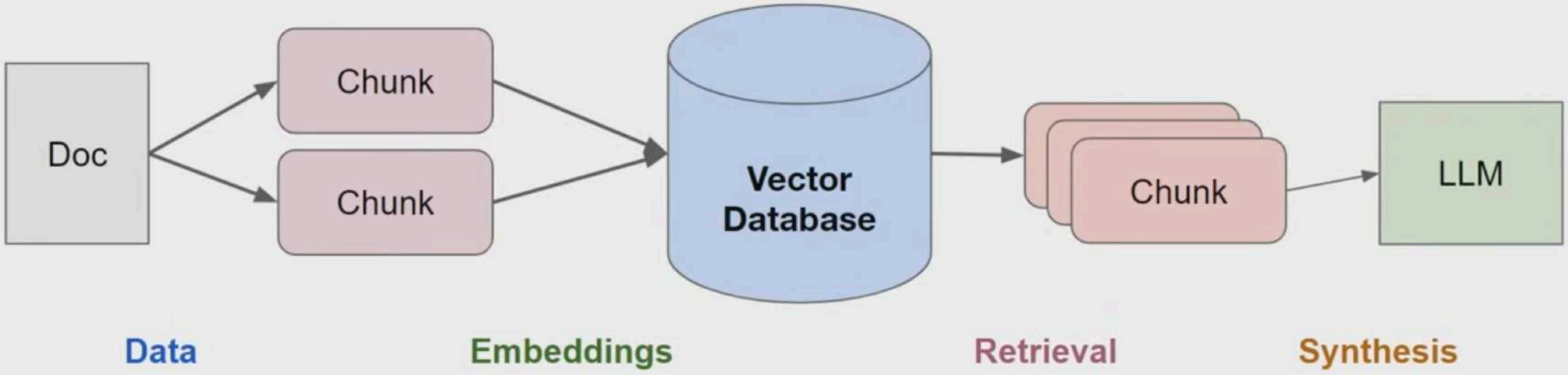


Difference between Naive and Advanced RAG (Image by the author, inspired by [1])

Naive RAG vs Advanced RAG

What do we do?

- **Data:** Can we store additional information beyond raw text chunks?
- **Embeddings:** Can we optimize our embedding representations?
- **Retrieval:** Can we do better than top-k embedding lookup?
- **Synthesis:** Can we use LLMs for more than generation? v



The Shift to AI Native Search

Qdrant

Unstructured Data Is Exploding

(Data isn't in a spreadsheet)

AI Agents Are the New Users

Legacy Search Falls Short

Vector Search Is the Missing Layer

Wave 1

RAG 1.0 - Static Assistants

(2023 - 2024)

Wave 2

Agentic AI - Multi-Step Reasoning

(2024 - Now)

Wave 3

Embedded AI – Physical & On-Edge Agents

(2025 +)

Qdrant-at-a-Glance

Vector Search Engine. Not Database. optimized for scalability and high availability

Built-Out for Search-First Workflows

Qdrant is built from the ground up with **search as the core functionality**. Conventional databases focus on ACID transactions and strong consistency.
In contrast, search engines are optimized for scalability, low-latency search, and high availability.

Engineered for Vector Search at Scale

Qdrant is purposed to handle extremely high-dimensional embeddings. It's designed with a **vector index as a central component of the system**, allowing a custom, finely tuned approach to data and index management that secures high performance even as data grows and changes dynamically

Specialized for Advanced Vector Operations

Qdrant is designed from the ground up to handle high-dimensional vector math and (dis-)similarity-based retrieval. This allows for leveraging the full potential of vector search **beyond simple similarity ranking** from multi-stage filtering to dynamic exploration of high-dimensional spaces**.**

Quick and Easy to Start

Performance Centric

Fully Open Source Project

All Embeddings Types Supported

Scalability Oriented

Resource Optimized

How Qdrant Achieves Search

Core Capabilities

Vector Search
Scalable similarity and discovery search (billions of vectors)

Hybrid Search
Combine dense + sparse embeddings, filters, and metadata

Filtering
Numeric, categorical, geo, temporal filters out-of-the-box

Distributed & Resilient
Replication, sharding, multi-tenancy

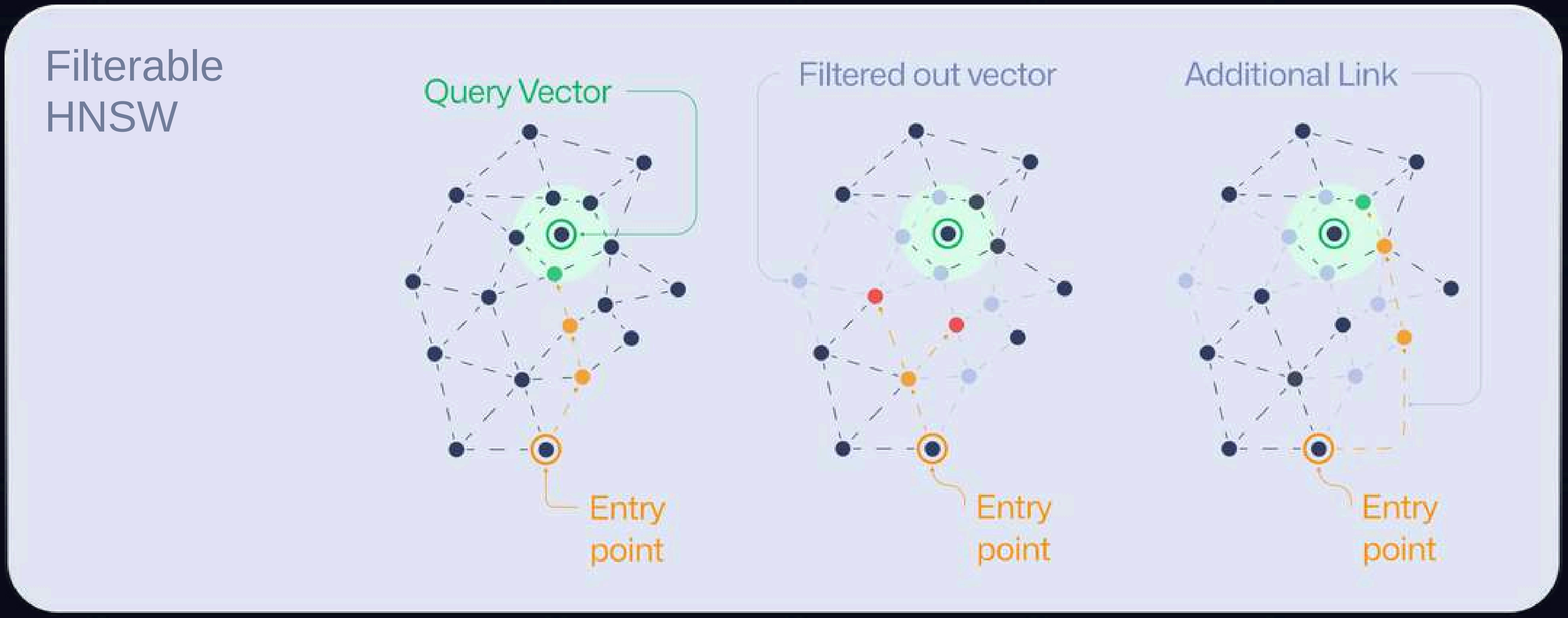
Advanced Features

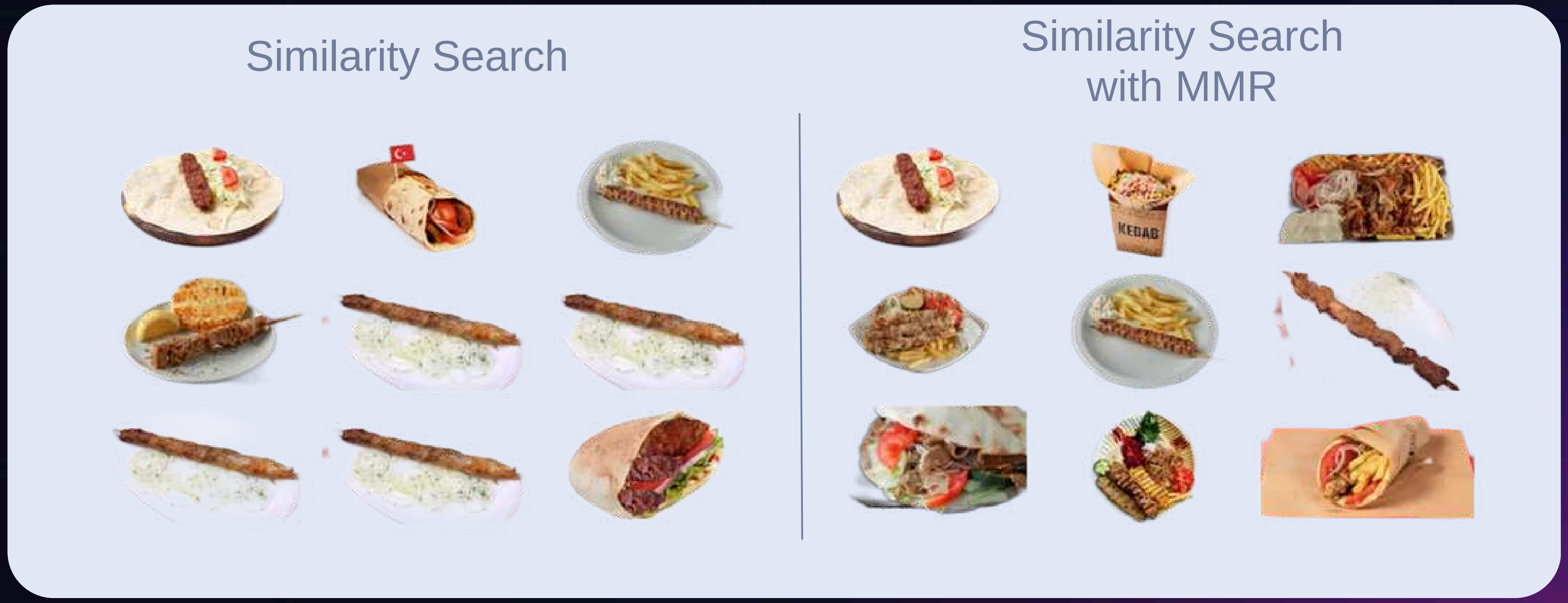
Re-ranking
Maximum Marginal Relevance (MMR), score boosting

Quantization
Binary, scalar & product; lower cost without major recall loss

Multi-vectors: Late interaction for retrieval models (e.g. ColBERT)

Performance Optimizations
HNSW tuning, payload indexing, prefetching





60K

Community Members

250M+

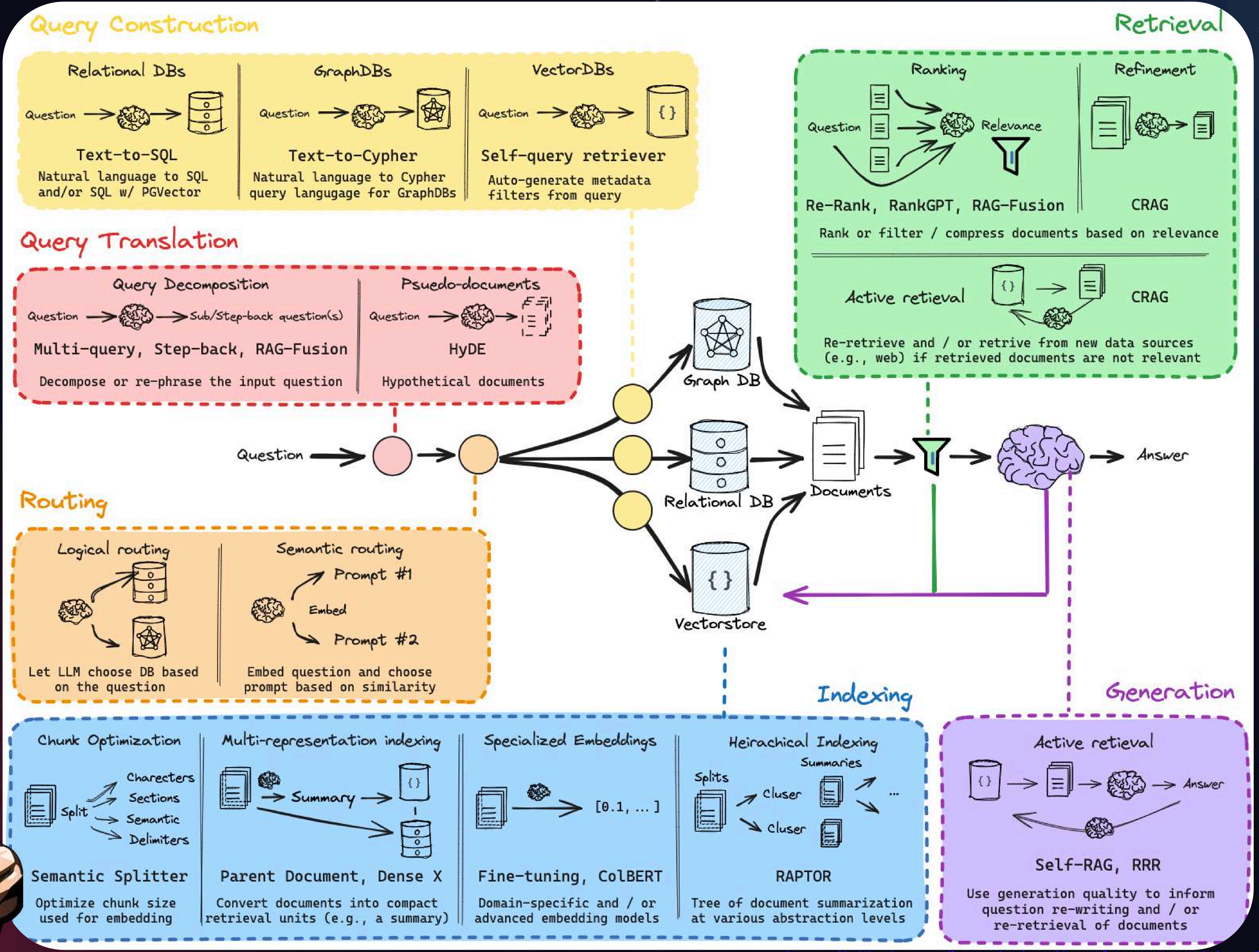
OSS Downloads

26K+

Github Stars

>140

Contributors



https://github.com/langchain-ai/rag-from-scratch/tree/main?tab=readme-ov-file

Resources

- Qdrant + DataTalks.club free course
- Just-RAG Github Repo
- How to get started with Qdrant
- Similarity search HNSW
- Building nerual search service with ST and Qdrant
- Your RAG powered by Google Search Technology
- Embedding models leaderboard
- Let’s talk about LlamaIndex and LangChain
- Retrieval-Augmented Generation (RAG) framework in Generative AI
- (RAG): From Theory to LangChain Implementation
- Free Perplexity Pro for 3 months

THANK YOU FOR YOUR ATTENTION!!

Where to find me?

Goodnight

Linkedin Profile

MedArbiNsibi

THANK YOU FOR YOUR ATTENTION!!

Khushal Kumar • Following
Software Engineer – GenAI @Wingify | Masters in AI/ML | Follow me to Learn AI Engineering in Quick Si...
1w •

I once took an interview where a candidate really stood out,
all because of how he handled RAG.

Most candidates presented the same basic Retrieval-Augmented Generation setup.
You know, plug in a vector DB, chunk text, retrieve, generate... nothing unusual.

But this candidate went deeper.

He didn't just talk about how RAG works.
He showed how tables and other unstructured data could be extracted and indexed from documents.
He thought about real-world use cases, not just the standard pipeline.

The tools weren't what impressed me, it was the thinking.
The attention to detail.
The willingness to go beyond what everyone else was doing.

It's been months, and I still remember that interview.

In a world where everyone knows the basics, it's the "depth" that makes you unforgettable.

Curious, what's the most memorable interview experience and why?

And by the way, if you're trying to level up your GenAI interview or assignment game, I've created something new to help with exactly that.

🔗 Link's in the comments.

#coding #codingInterview #RAG #python

Ahmed SIDI AHMED and 102 others · 4 comments