DevFest Nairobi

2025

Optimising AI Agents with Retrieval Augmented Generation (RAG)

Cyril Michino
Co-founder, Zindua School

Google Developer Groups

{ DevFest }
Nairobi

About Me – Cyril Michino

Co-founder at Zindua School – Leading operations & AI Engineering for LMS 3.0. **Data Scientist & AI Engineer professionally**

Website: cyril.michino.co.ke
LinkedIn: /in/cyrilmichino
X: @cyrilmichino

Google Developer Groups

01

Background

Understanding how to choose the right LLM and the pillar of context engineering

Google Developer Groups

Pillars of Context Engineering

The art of AI Engineering today is all about quality context.

- Prompt Engineering
- Tool Calling
- Memory: Short-term and Long-term
- **Retrieval Augmented Generation (Our Focus)**
- Model Fine-Tuning

How to choose your model – Part A

Beyond Context Engineering, choosing the ideal model for your task is key. Here are some key things to consider:

- <u>Pricing:</u> Can you afford the model? Should you go for a mini version? Note price is anchored on input and output tokens
- <u>Performance benchmarks:</u> Intelligence, speed, niche indices
- <u>Context length:</u> Maximum input tokens

Continued in the next slide...

Google Developer Groups

How to choose your model – Part B

Continuation from the previous slide:

- Latency: Time to first answer token based target use
- Multimodality: Support for images, audio, videos
- Tool support: Code interpreter, browser, uploads, APIs
- Privacy/Open-source: Can you self-deploy the model

Always tie it down to your intended use case

02

What is RAG

In-depth
understanding of
retrieval augmented
generation

Google
Developer
Groups

How we typically work with LLMs – Prompt-based

When building AI Agents, your prompt will be a combine:

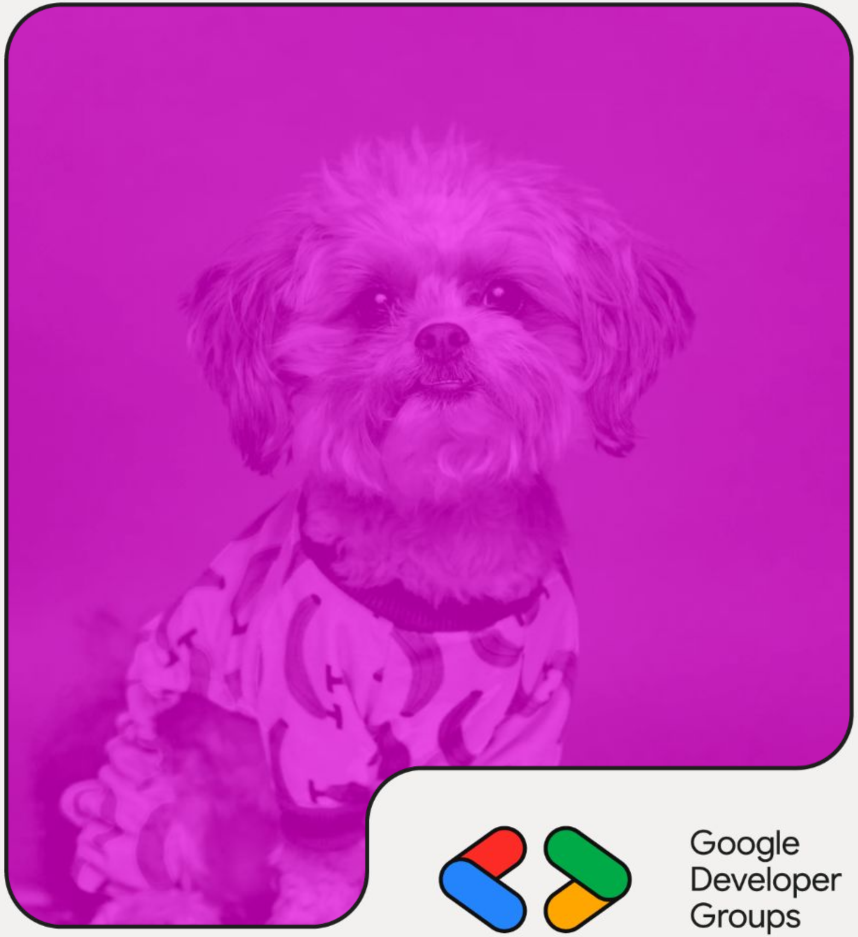
- System Prompt (what you encode into the agent)
- Human Prompt (from your end user)

Limitation of LLMs with Standard Prompts

- Lack of Up to Date Data
- Problems with Accuracy of Models
  - Hallucinations
  - Bias of sources
  - Domain-specific Context
- Limited Context Windows

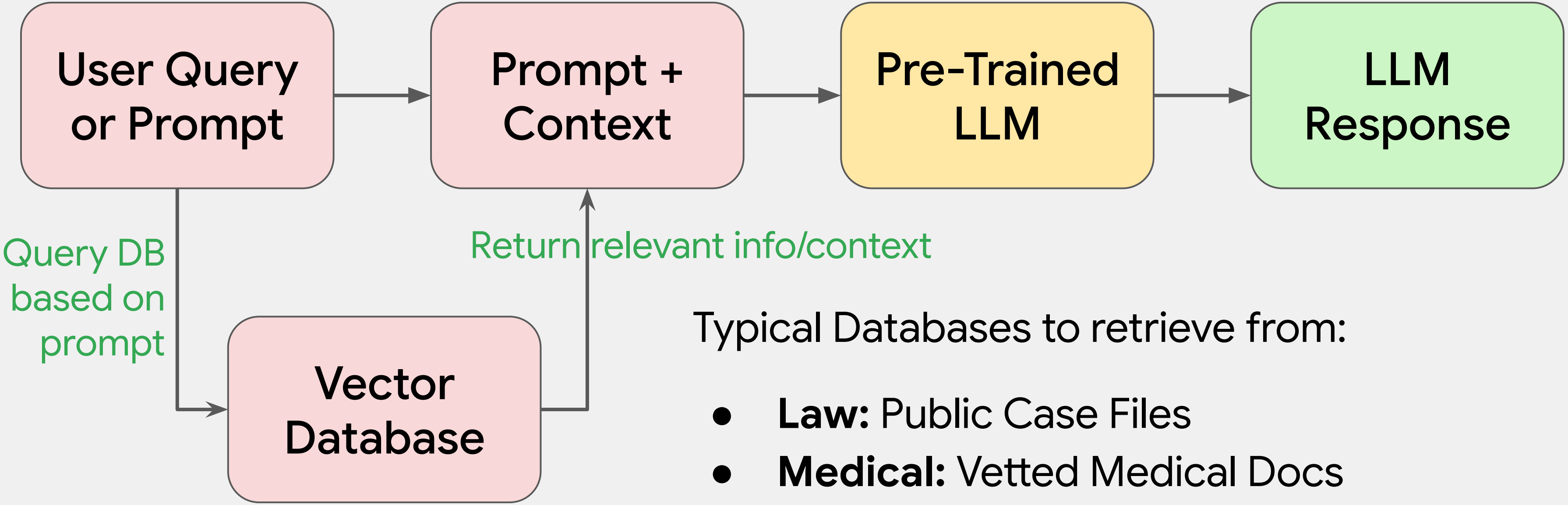
DEMO: Go to **Simple Agent Notebook** in this repo:

github.com/cyrilmichino/ai-agents-demo



Google Developer Groups

Now let's introduce a retriever – Vector RAG



Typical Databases to retrieve from:

- **Law:** Public Case Files
- **Medical:** Vetted Medical Docs
- **Zindua School:** Internal Course Content

Why care about RAG

Whether you are chatting with PDFs, Docs, or a Database:

1. Injects relevant external knowledge into the prompt
2. Keeps models lightweight (no need to fine-tune)
3. Enables domain-specific intelligence instantly

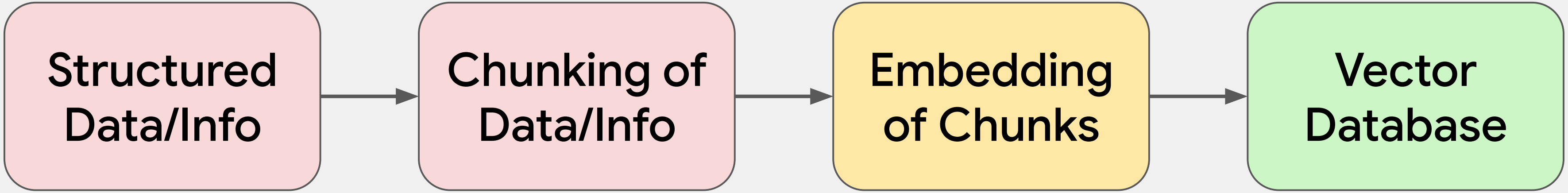
BONUS: External knowledge can be updated by adding new info or removing outdated info to the vector database

Intuition on Vector Databases

- **Problem 1:** Extremely large databases, limited context window
  - Retrieve only the most relevant data/info from the database
- **Problem 2:** Semantic Gap in traditional databases
  - Use vector embeddings to encode data into a vector store

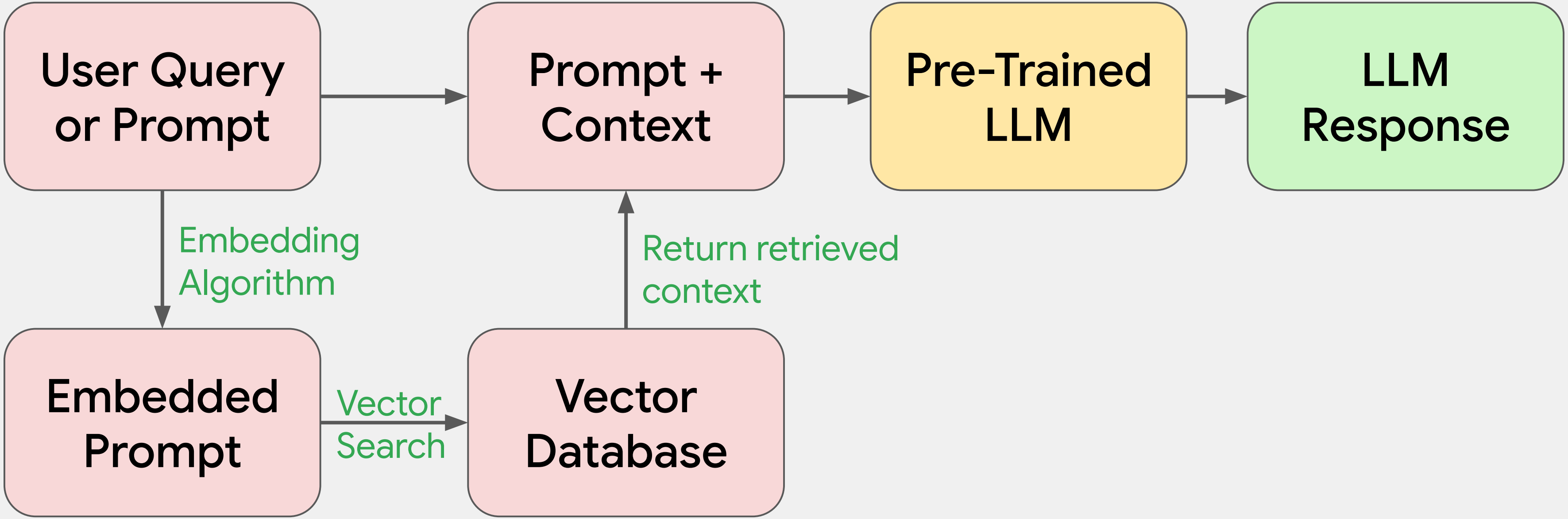
A vector database stores data as a series of vectors that encode the semantic meaning of data. Extremely similar data will be on the same area in your vector space

How to encode your data sources into a Vector DB



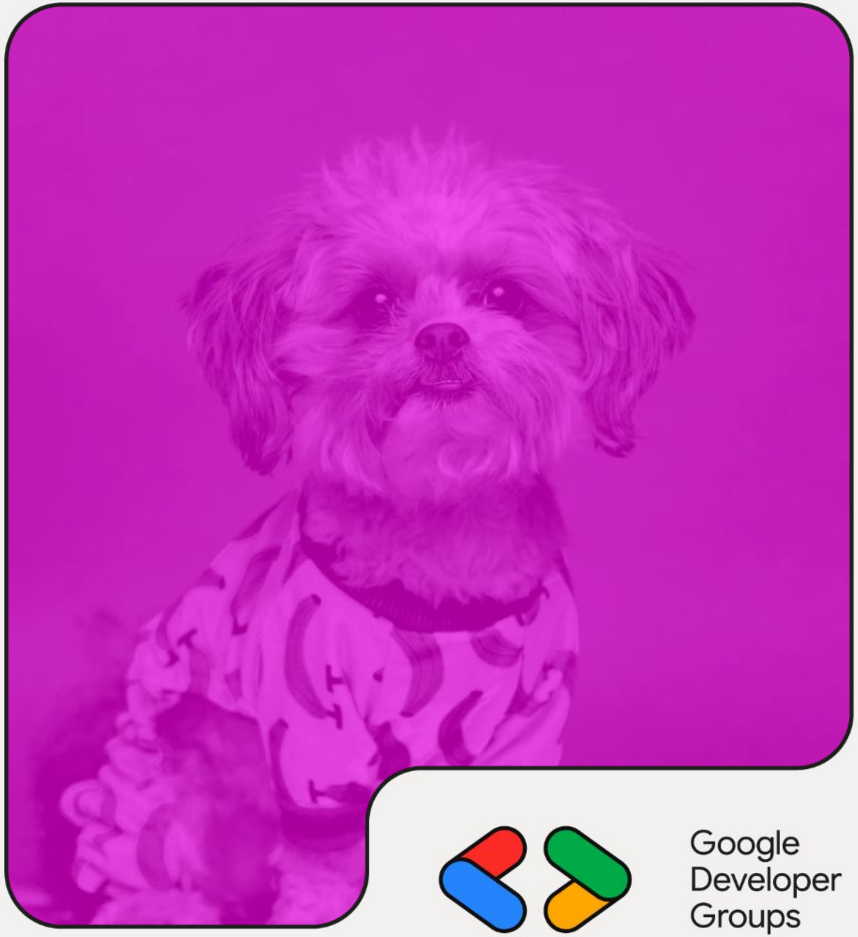
- **Chunking:** Breaking down high token documents into chunks of a standardised token size
- We use **embedding algorithms** relevant to the dataset to define the vector representation of each chunk (Semantic-based storage)

How it actually works in practice



DEMO: Go to **RAG Agent Notebook** in this repo:

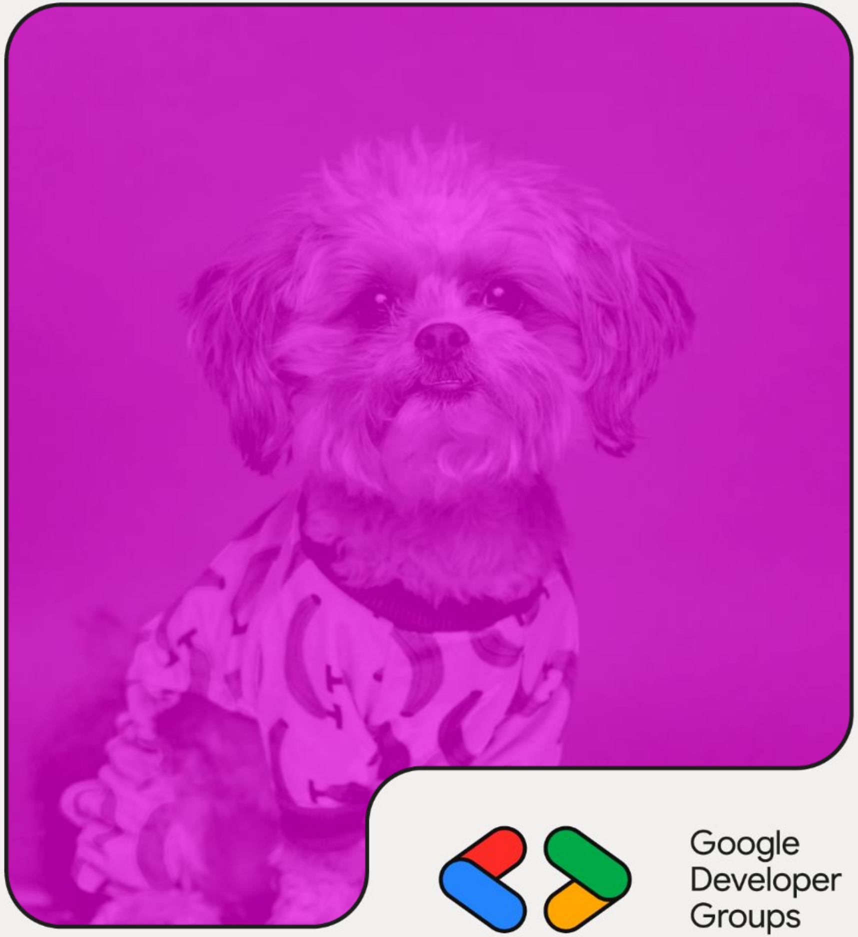
github.com/cyrilmichino/ai-agents-demo



Google Developer Groups

From the demo, we'll need:

- AI Model (Of course)
- Embedding Algorithm
- Vector Database
- Vector Indexing Algorithm



Google Developer Groups

Example of Embedding Algorithms

- Contrastive Language-Image Pretraining (CLIP)
- Global Vectors for Word Representation (GloVe)
- Word2Vec (Words)
- Wav2Vec (Audio)
- Bidirectional Encoder Representations from Transformers (BERT)

Google Developer Groups

Popular Vector Databases

- Chroma
- Pinecone
- Weaviate
- QDrant
- Milvus
- PGVector

Vector Indexing Algorithms

Vector indexing is critical if working with a large vector DB.

Reduce the vector space to be searched for better latency/cost:

- Flat: Brute-Force
- IVF (Inverted File Index) – Clusters
- HNSW (Hierarchical Navigable Small World) – Graph
- PQ (Product Quantization) – Dimensionality Reduction
- Annoy (Approximate Nearest Neighbors Oh Yeah) – Trees

03

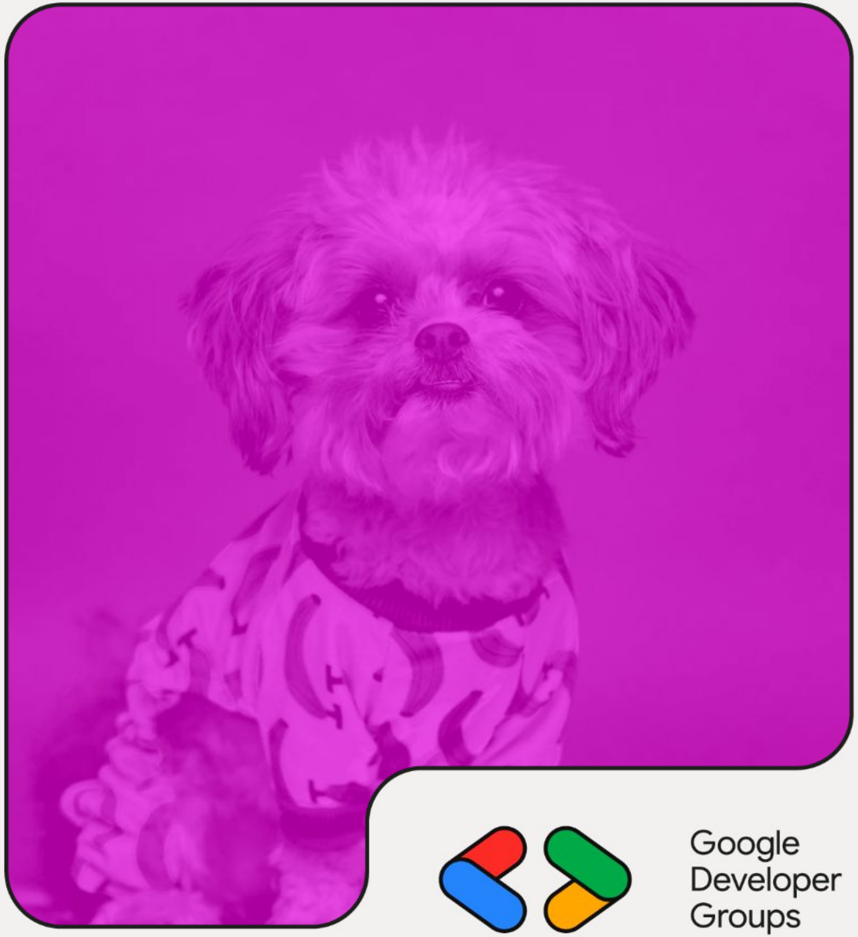
Optimising RAG

How to improve the performance of your RAG workflows (Vector RAG)

Google Developer Groups

Improving results of Vector RAG:

- Chunking Strategies
- Reranking
- Agentic RAG
- Cache Augmented Gen.
- NEW RAG Paradigms



Google Developer Groups

3.1. Chunking Techniques

When breaking down large documents into chunks, we might end up with chunks that do not have a complete semantic thought encoded. Here is how we can optimise our chunks:

- Sliding Windows (Simplest Approach)
- Context-aware Chunks
- Semantic Chunking
- Recursive Chunking

Google Developer Groups

3.2. Reranking

Cosine similarity results from a vector database will not always be relevant to the prompt. With reranking:

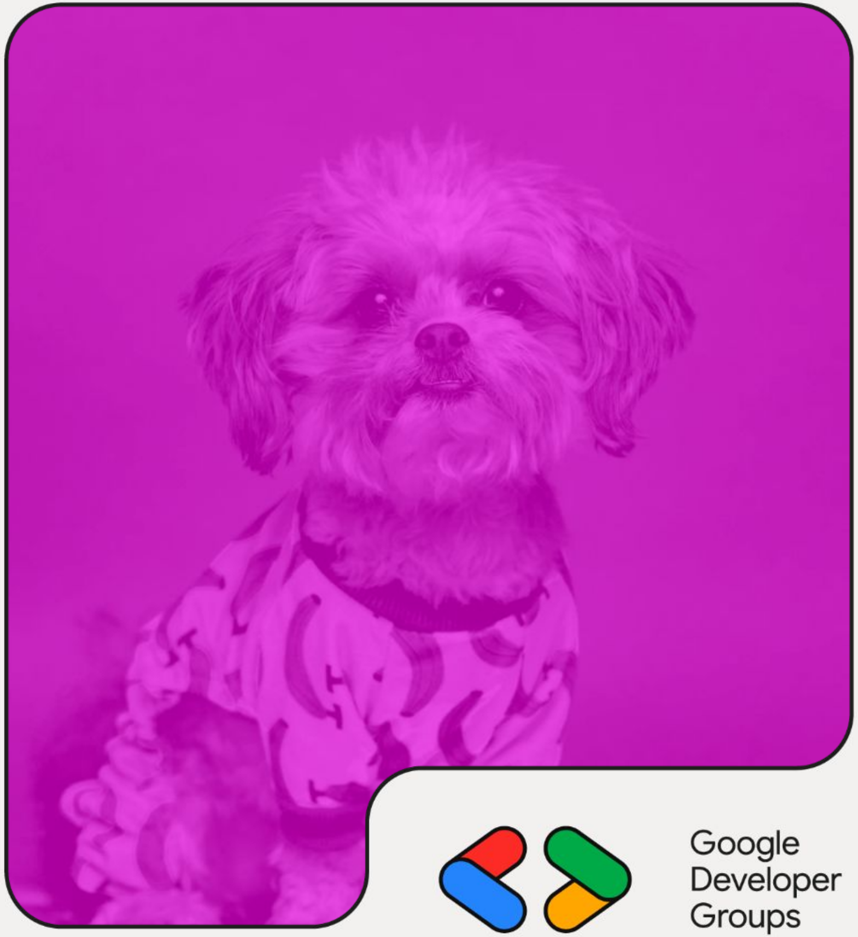
- We use an LLM or a cross-encoder to score the relevance of each retrieved chunk
- Based on the score, we filter out irrelevant chunks before feeding retrieved context back to the LLM

Google Developer Groups

3.3. Agentic RAG

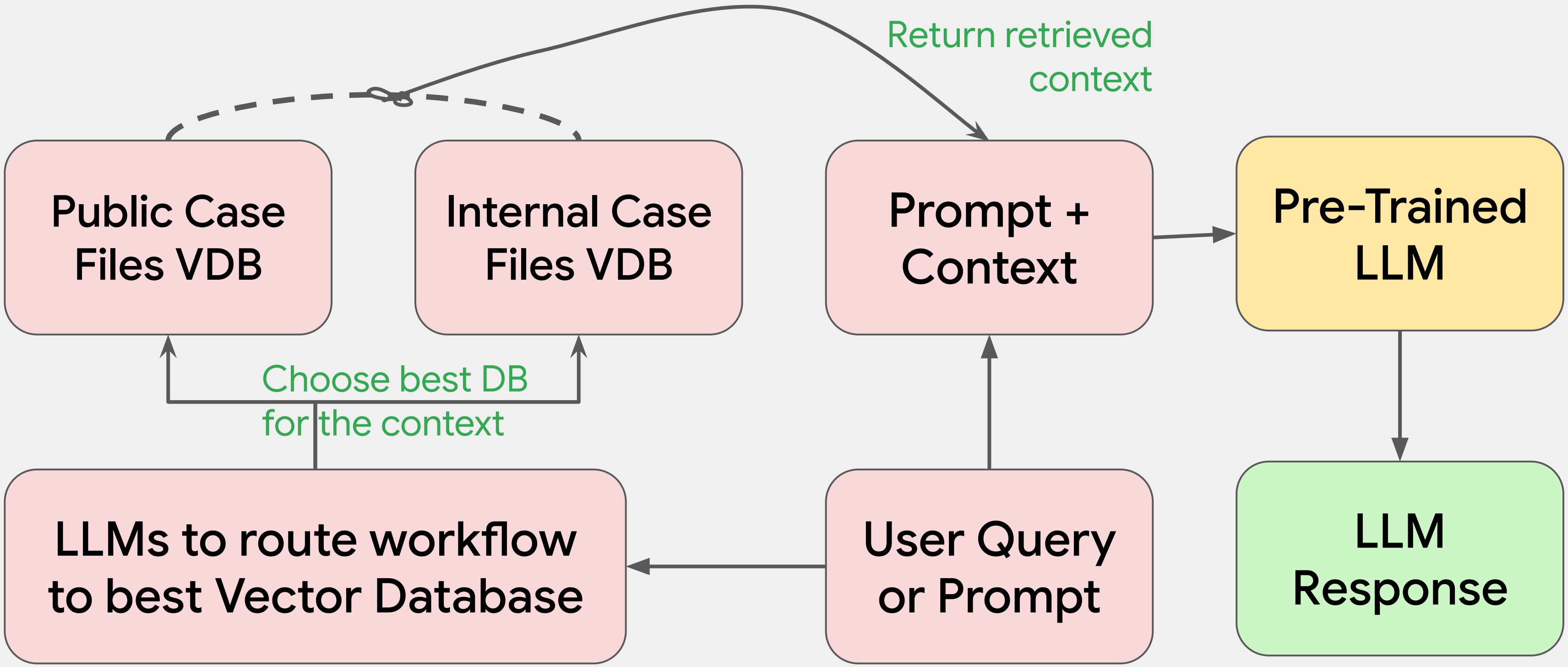
In certain scenarios, we have multiple databases and/or tools to be called.

Agentic RAG allows you to redirect your RAG workflow to the right Vector DB or tool



Google Developer Groups

Agentic RAG for a Legal Firm – Two Vector Databases

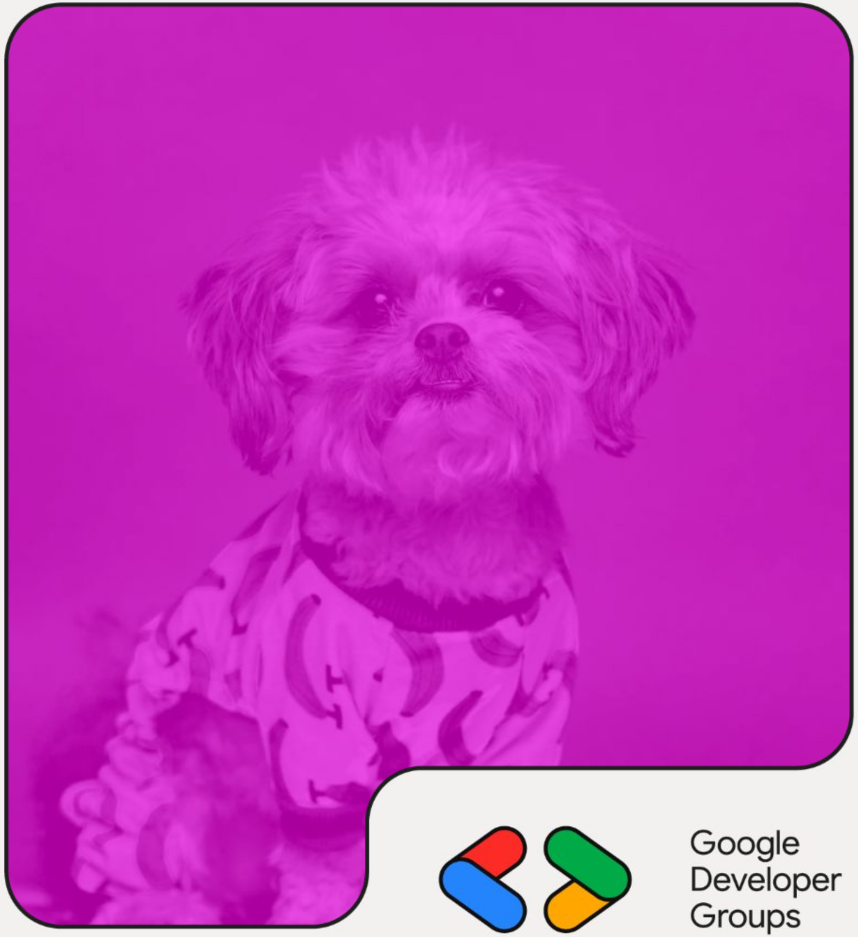


3.4. Cache Augmented Generation

- RAG is powerful, but recomputes answers every time:
  - Every query – vector search – generate answer
  - Very costly (repetitive computation)
  - High Latency (retrieve fast before responding)
- CAG: Save the question and its LLM response. If a new query is similar, serve the cached answer instead of calling the LLM
  - Precompute common answers
  - Have a key-value store as your cache
  - Can default to RAG if answer is not in cache

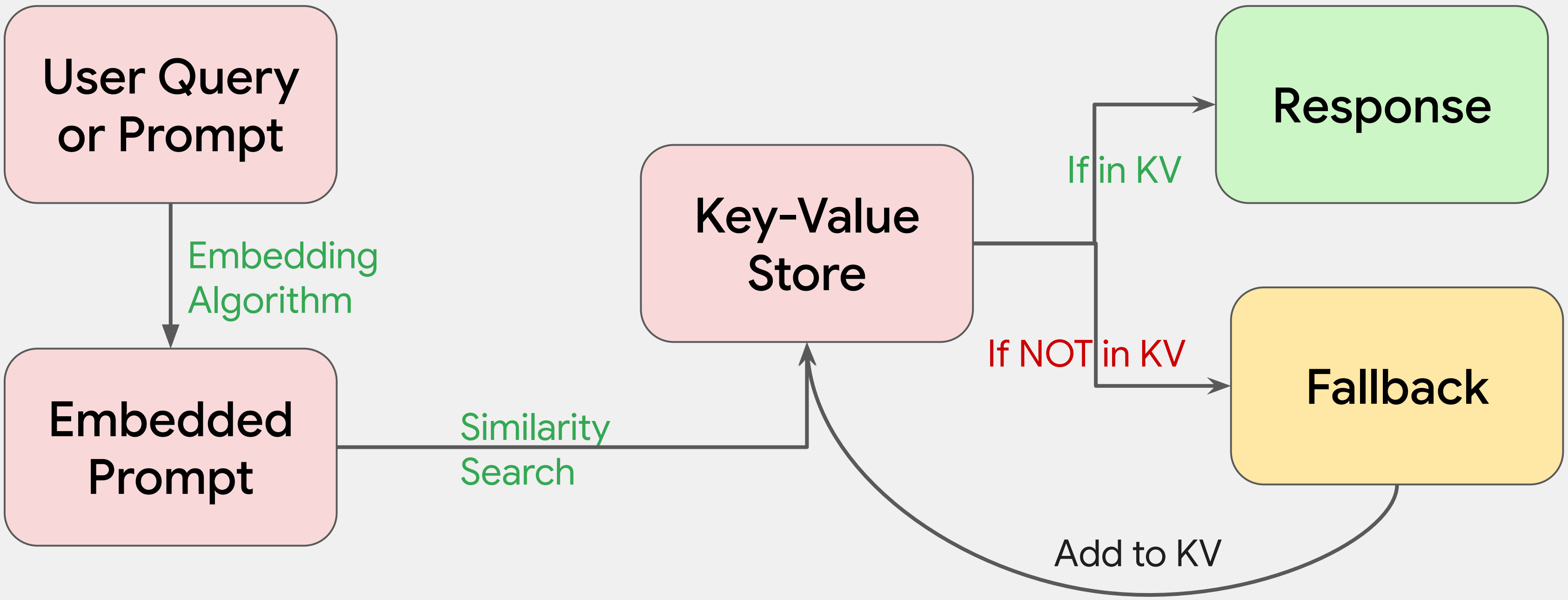
Key Layers for CAG

1. Key Value Store
2. Embedding Model
3. Similarity Engine
4. Fallback (RAG, or LLM response)



Google Developer Groups

How it actually works in practice



Cache Augmented Generation vs standard RAG

	RAG	**CAG**
Data Source	Vector Database	Key-Value Store
Latency	High Latency	Low Latency (Fast)
Data Consistency	Changes frequently	Almost always constant
Good for	Large external knowledge	Not so large external KV
Not good for	High traffic similar queries	Always changing database

3.5. Other Paradigms of RAG

Vector RAG (using vector databases) is just one way to implement retrieval augmented generation. There are other paradigms that improve upon its limitations:

- **Graph RAG:** Use Knowledge Graphs instead of Vector DBs
- **RL RAG:** Fine-tune an LLM to know when best to use RAG vs Tool Calls vs its own Embedded Knowledge

The biggest problem in using AI and building AI agents is passing **NOT ENOUGH context.**

The second biggest is passing **TOO MUCH context.**

DevFest

Nairobi

What we do at Zindua School

Coding School in Kenya offering programs in **Software Engineering** and **Data Science**. We've also launched specialisation programs in DevOps, Data Engineering, and **AI Engineering**.

Learn more: zinduaschool.com

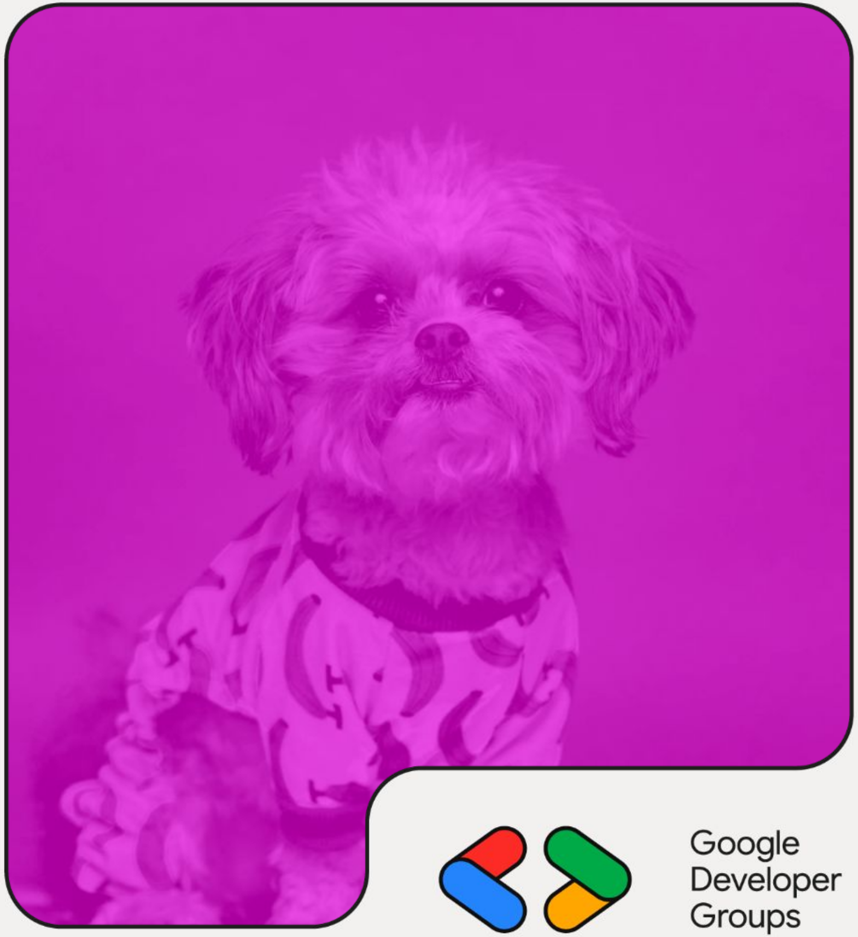
Beyond Zindua – I write, talk, and teach

Check out either of the following publications

- Data HQ by Michino – #DataScience #ML #AI
- Startup Ops by Michino – #Founders #Operators

DM me on X or LinkedIn (Responses within a week)

SCAN to get the Slides & drop your feedback



Google
Developer
Groups