WeCloudData

Naive RAG, Advance RAG, Agents and How to start in GenAI?

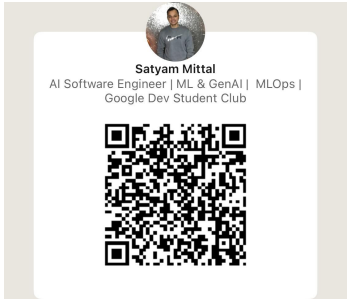
Instructor

Satyam Mittal

www.linkedin.com/in/satyam-sm

https://github.com/05satyam

Introduction



- Myself Satyam Mittal, currently working as AI Software Engineer in Retail organization(fortune 500).
- Did my research oriented M.S. in Computer Science with thesis in Machine learning.
- Designed an optimized version of Buggy Pinball Optimization algorithm named as Blindfolded Spiderman Optimization in higher dimensions

---

⚡Linkedin: Link(www.linkedin.com/in/satyam-sm)

⚡Github: Link(https://github.com/05satyam)

⚡AI-ML Repository/Resource: Link (https://github.com/05satyam/AI-ML/blob/main/README.md)

Table of Contents

Some key terminologies

- **LLM (Large Language Model):** AI trained on vast datasets to understand and generate human-like text.
- **RAG (Retrieval-Augmented Generation):** Combines LLMs with retrieval systems for context-aware responses.
- **Embeddings:** Numerical representation of data for semantic similarity in vector space.
- **Fine-Tuning:** Customizing a pre-trained model for specific tasks or domains.
- **Prompt Engineering:** Crafting inputs to elicit desired outputs from LLMs.
- **Guardrails:** Safety mechanisms to ensure reliable, ethical model outputs.
- **Vector Database:** Specialized database for storing and querying embeddings.
- **Agents:** Autonomous units in AI systems capable of decision-making and task execution.

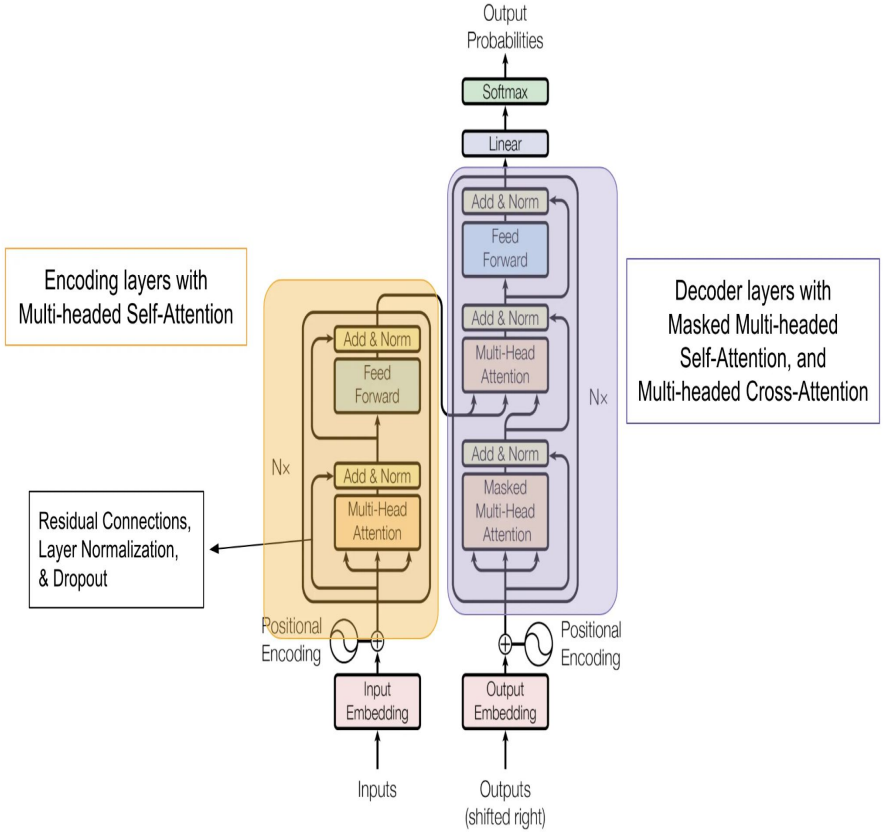
Transformer

Encoder-Decoder Structure:

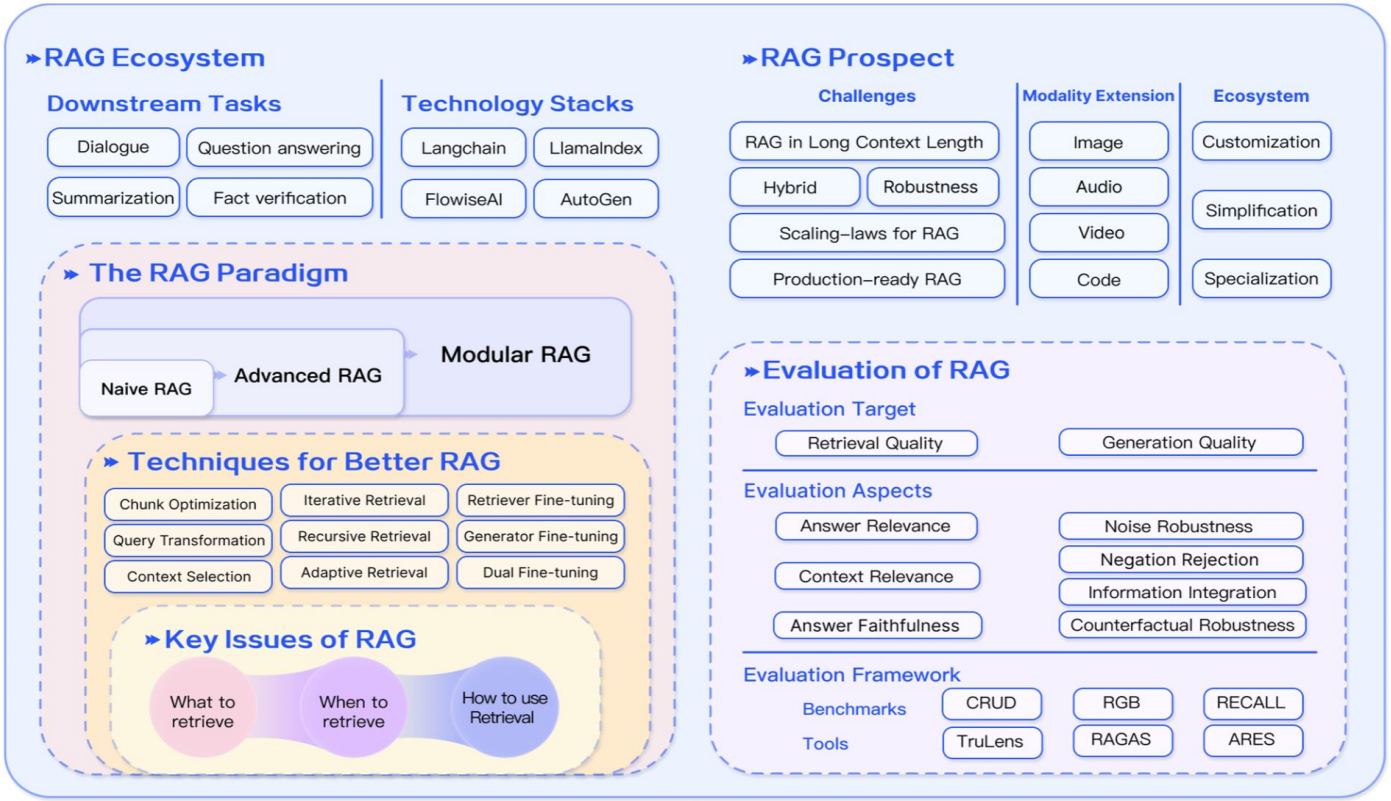
- **Encoder**: Processes input sequence with multi-headed self-attention and feed-forward layers.
- **Decoder**: Generates output sequence using masked multi-headed self-attention, cross-attention, and feed-forward layers.

Key Components:

- **Multi-Headed Attention**:
  - **Self-Attention** in Encoder: Focuses on relationships within the input sequence.
  -
- **Feed-Forward Networks**: Add non-linearity and feature transformation.
- **Positional Encoding**: Adds order information to sequences.



Summary of RAG Ecosystem



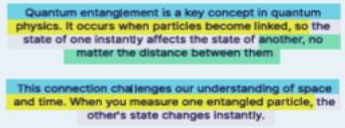
https://arxiv.org/pdf/2312.10997

Data Preprocessing

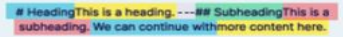
Based on use-case we will have data-preprocessing.

For ingesting into vector-database, we also add **chunking-strategies**

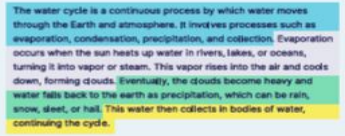
Fixed-size chunking is a simple technique that splits text into chunks of a predetermined size, regardless of content structure. While it's cost-effective, it lacks contextual awareness. This can be improved by using overlapping chunks, allowing adjacent chunks to share some content.



Recursive chunking offers more flexibility by initially splitting text using a primary separator (like paragraphs) and then applying secondary separators (like sentences) if chunks are still too large. This technique respects the document's structure and adapts well to various use cases.



Document-based chunking creates chunks based on the natural divisions within a document, such as headings or sections. It's particularly effective for structured data like HTML, Markdown, or code files but less useful when the data lacks clear structural elements.



Semantic chunking divides text into meaningful units, which are then vectorized. These units are then combined into chunks based on the cosine distance between their embeddings, with a new chunk formed whenever a significant context shift is detected. This method balances semantic coherence with chunk size.

LLM-based chunking is an advanced technique that uses an LLM to generate chunks by processing text and creating semantically isolated sentences or propositions. While highly accurate, it's also the most computationally demanding approach.

Image credit: Weaviate

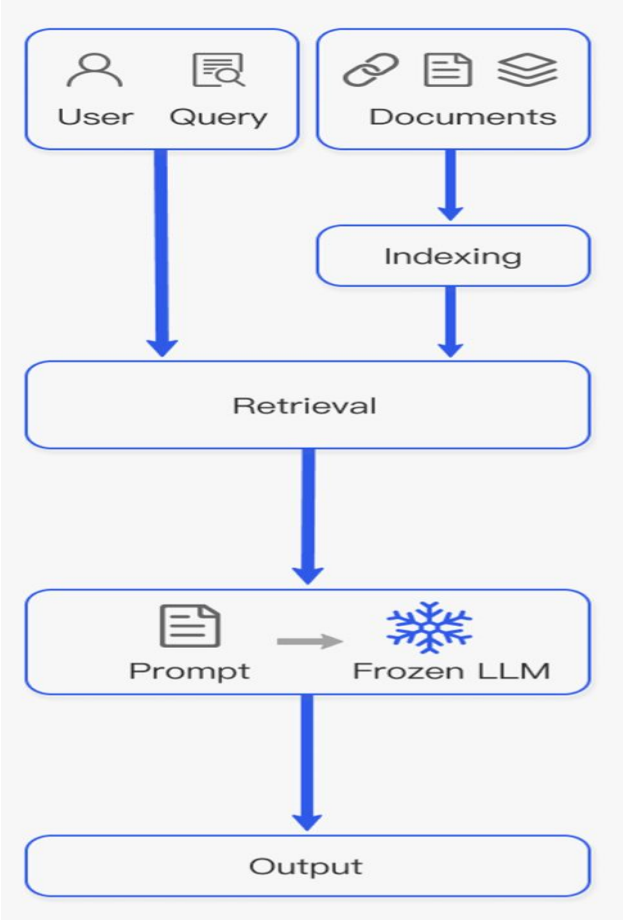
Naive RAG

The workflow starts with a **user query**, paired with **indexed documents** for efficient retrieval.

Relevant documents are retrieved and combined with the query to create a **contextual prompt**.

A **frozen LLM** processes the prompt, leveraging retrieved context to generate accurate outputs.

This approach ensures **domain-specific responses** without costly model fine-tuning.



Google colab: Link

https://arxiv.org/pdf/2312.10997

Advance RAG

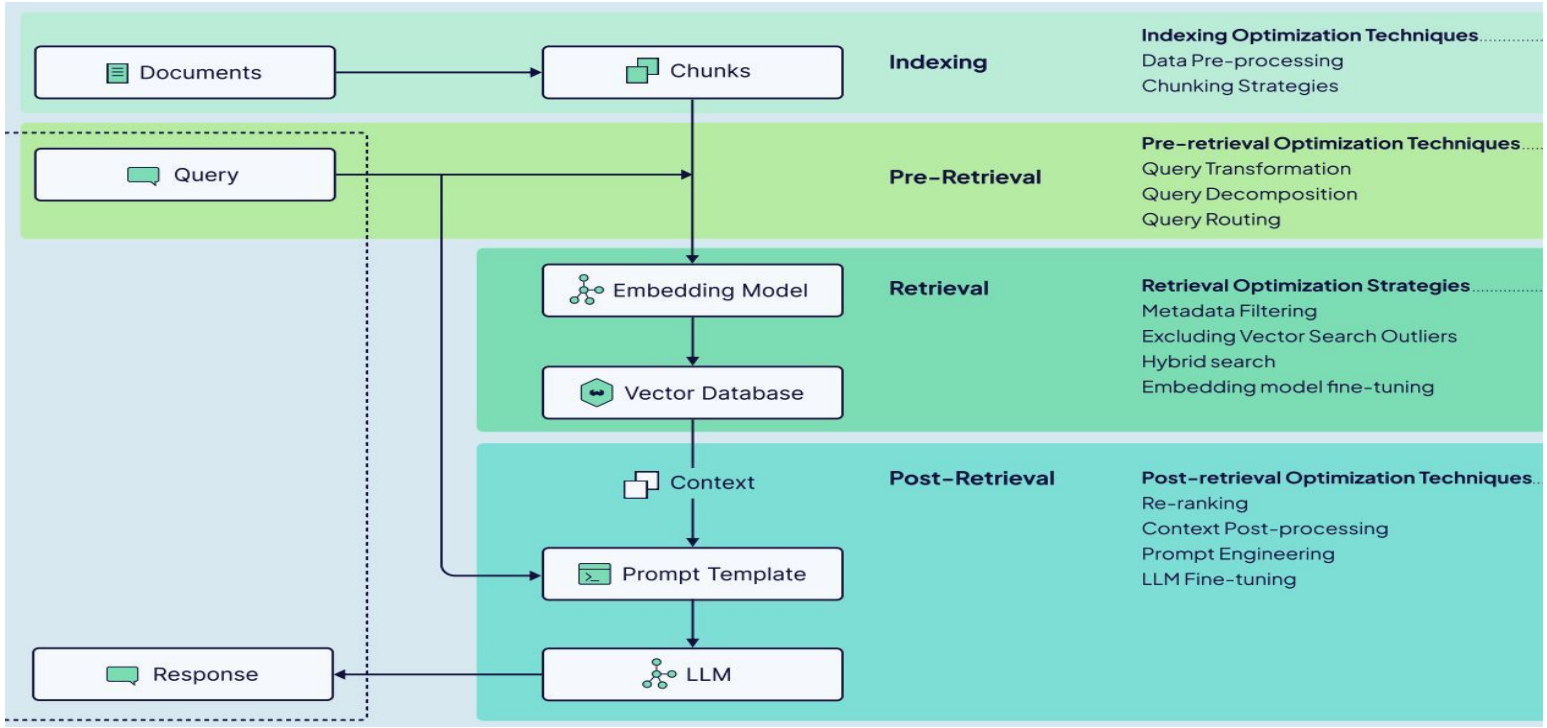


Image credit: Weaviate

Google colab: Link

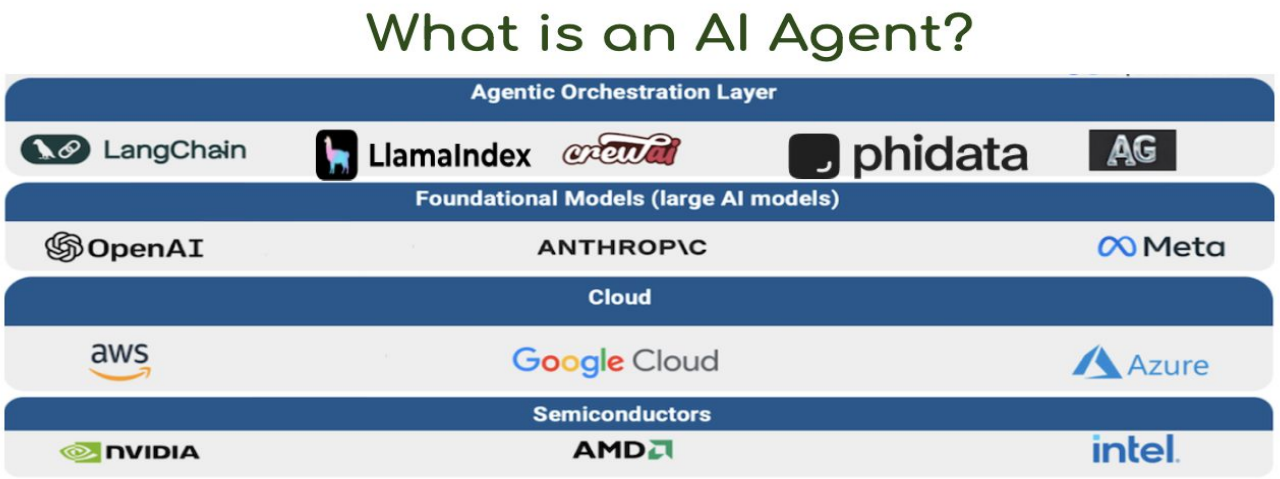
Advance RAG Continued…

Pre-Retrieval Optimization: Enhances query quality through **query rewriting**, expansion, and routing for better retrieval accuracy.

Post-Retrieval Processing: Refines retrieved results using **reranking**, summarization, and fusion to ensure high relevance and contextual alignment.

Adaptive Prompting: Combines processed retrieval data into a dynamic, structured prompt for the **frozen LLM**, enhancing response precision.

AI AGENTS - Frameworks for Development

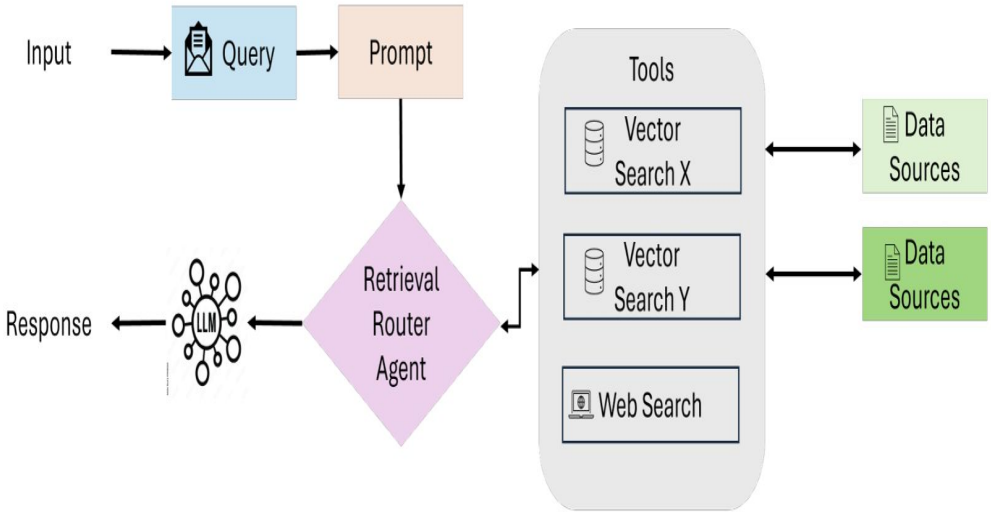


Detailed Article: Link

Agentic RAG

Single-Agent Agentic RAG:

A centralized decision-making system where a single agent manages the retrieval, routing, and integration of information



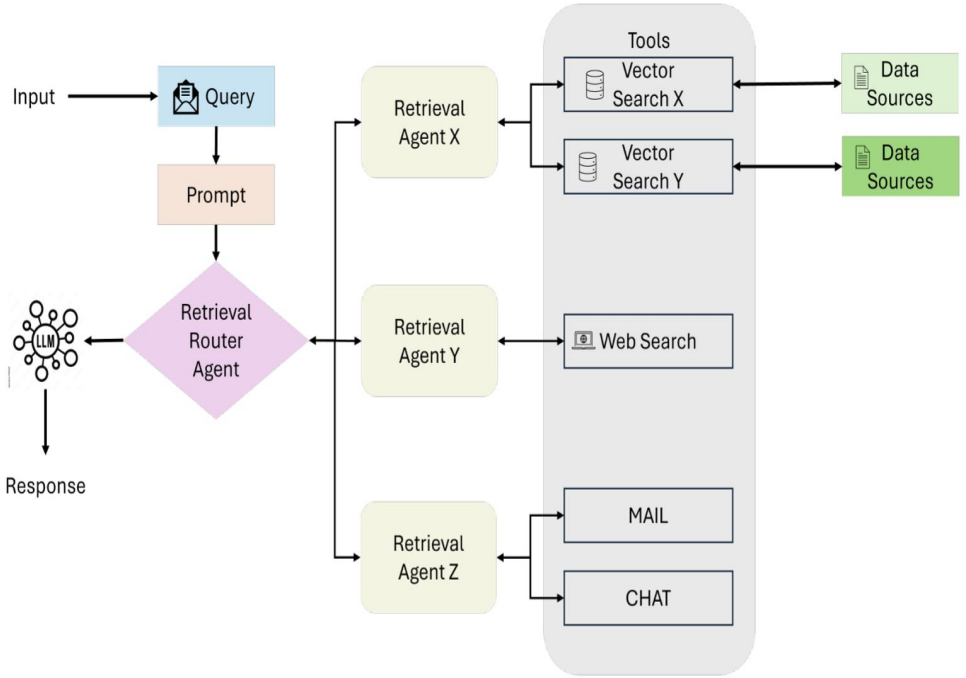
Google colab: Link

https://arxiv.org/pdf/2501.09136

Multi-Agent RAG

Instead of relying on a single agent to manage all tasks—reasoning, retrieval, and response generation

—This system distributes responsibilities across multiple agents, each optimized for a specific role or data source.



Google colab: Link

https://arxiv.org/pdf/2501.09136

Hands-on Google colab: Link

Or

Copy below given URL:

https://colab.research.google.com/drive/1_Uhu5jVfW3MqlaqCWrUscqU0lS5Wb33t#scrollTo=ERQf6jqBWkc5

Poc2Production continue: Evaluations

Holistic Evaluation: Evaluate each stage separately while measuring end-to-end impact. Some top-of-mind metrics are:

⛳ Indexing Optimization:
- Chunk Overlap Ratio: Redundancy in chunking.
- Token Utilization Efficiency: % of useful tokens.
- Indexing Latency: Time taken for processing.

⛳ Retrieval Optimization
- Recall@K, MRR, NDCG: Ranking effectiveness.
- Embedding Cosine Similarity: Relevance score.
- Retrieval Latency: Speed of document retrieval.

⛳ LLM Generation Quality
- Faithfulness & Relevance Score: Accuracy of responses.
- Exact Match / F1-score: Correctness vs. ground truth.
- Hallucination Rate: % of unsupported/generated content.

⛳ End-to-End & UX Metrics
- Latency Analysis: Speed from query to response.
- User Satisfaction: A/B testing, surveys.
- Diversity & Readability: Response variability & clarity.

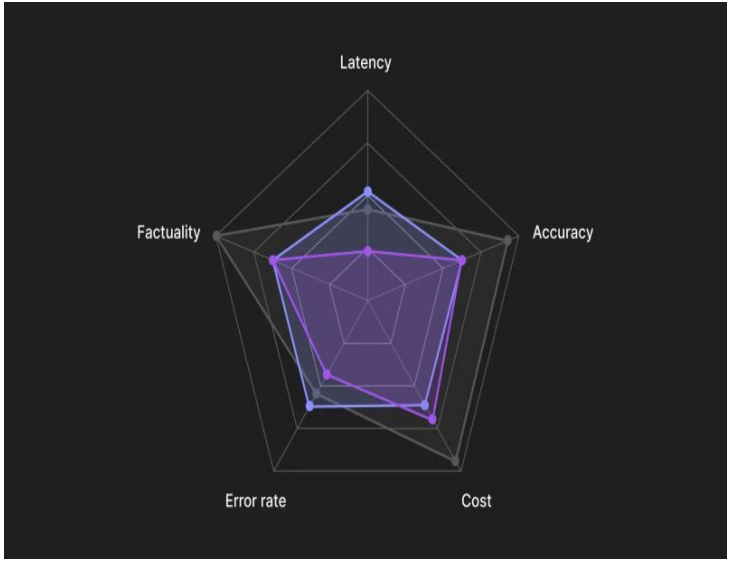


Image credit: https://lnkd.in/dCSJi-pB

Some available frameworks for evaluation:
Ragas(https://lnkd.in/d5yk686K); LangChain evaluation tools; LlamaIndex evaluation library; OpenAI Evals;

Poc2Production continued:

2. LLM Observability 🔭 RAG based application

Observability lets us:

⚡ Spot slowdowns or errors early, pinpointing root causes quickly.

🚀 Identify resource bottlenecks, optimizing our infrastructure.

📊 Gain insights into user behavior and system health.

Google colab: Link

Some of the good Observability Tools for LLM Applications 🛠️:

⚡ LlamaTrace (Hosted Arize Phoenix)

⚡ Langsmith

⚡ OpenLLMetry : Open-source project based on OpenTelemetry

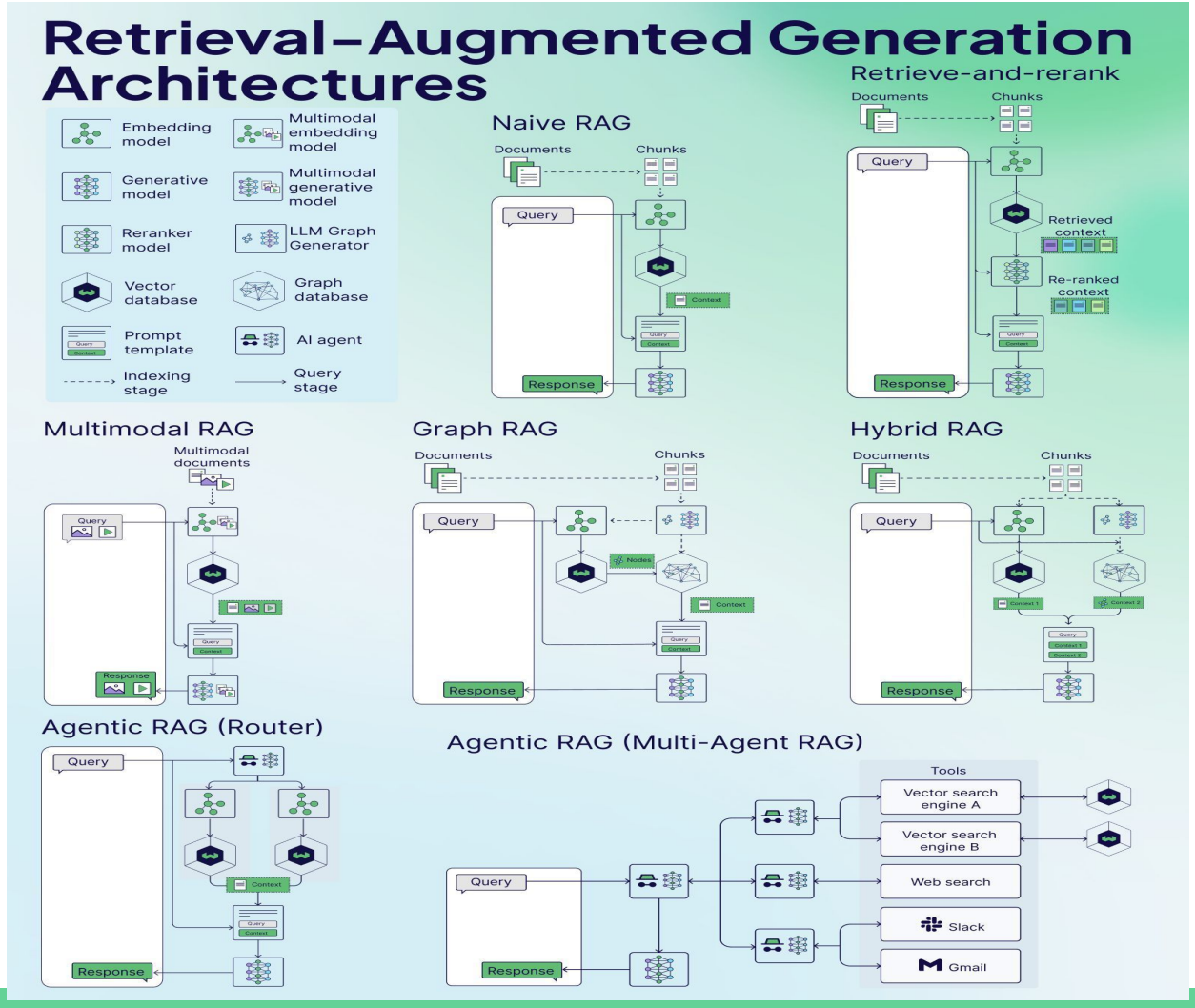
⚡ Arize AI- Phoenix (Local)

⚡ Literal AI : Designed for team collaboration on LLM evaluation and observability

⚡ Comet Opik

⚡ Langfuse (YC W23)

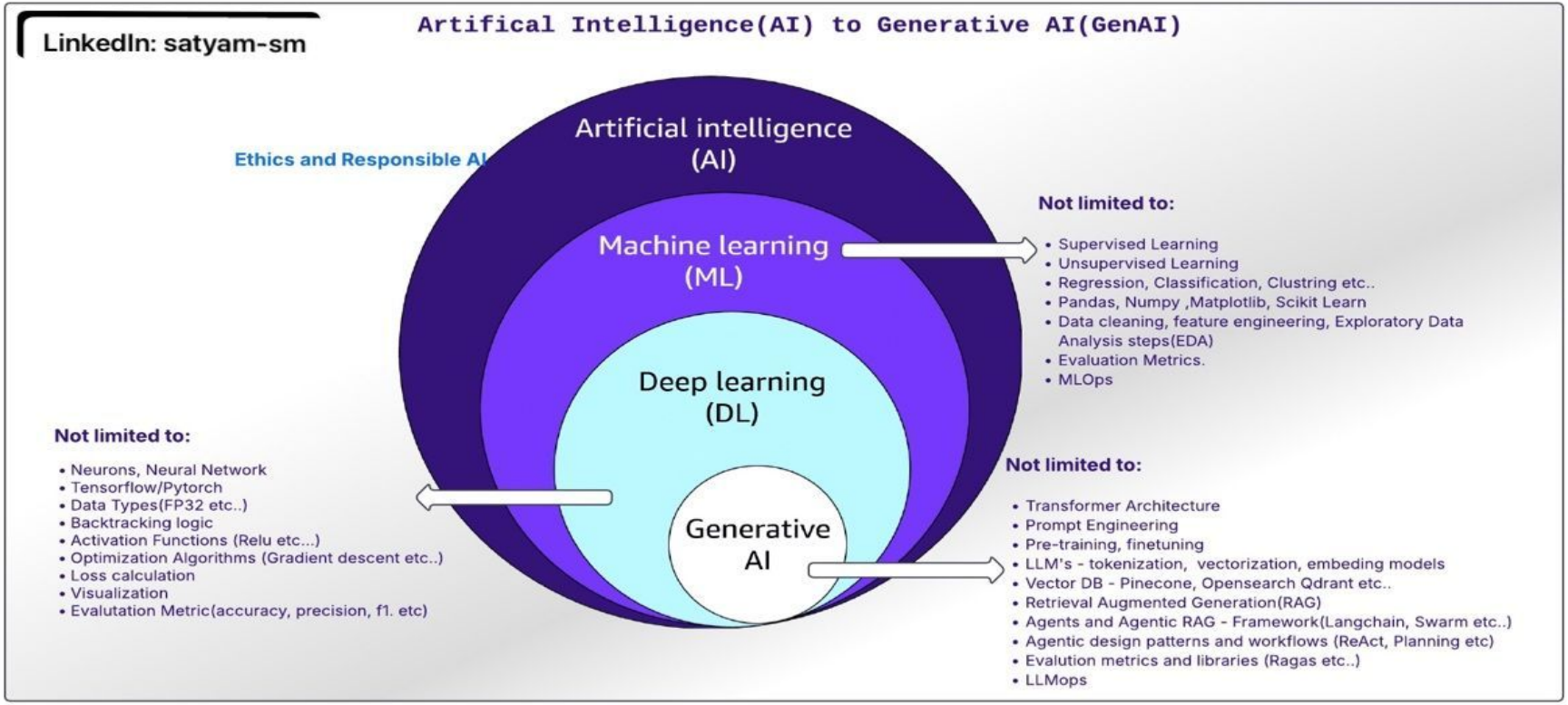
⚡ Weights & Biases and Biases Prompts



Different types of RAG Architectures

Image credit: RAG architectures by Weaviate

How to start in AI and especially in GenAI?



How to start in AI and especially in GenAI continue?

Basic Courses:

1. Basics of Python in AI: Link
2. Basics of AI: Link
3. Basics of GenAi: Link
4. Courses from WeCloudData

Advance Course: Post Link

Post Link

Question and Answer

Thank You!

Special thanks to all the organization/people for publishing such great images and content and open source material