DroneX AI

AGENTIC RAG

---TECH STACK

Karn Singh

What is Agentic RAG?

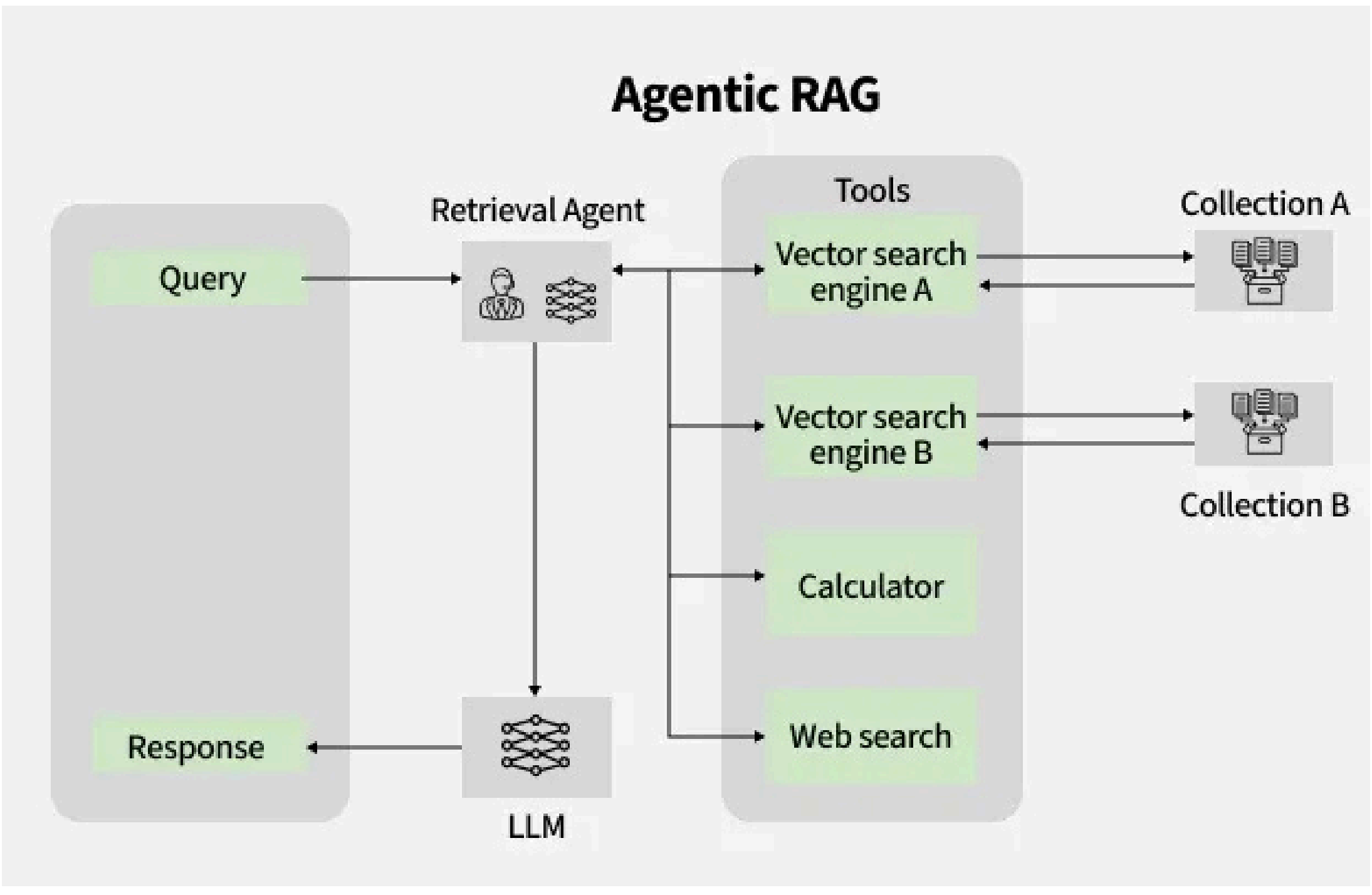
Agentic RAG is an advanced version of Retrieval-Augmented Generation (RAG) where an AI agent retrieves external information and autonomously decides how to use that data. In traditional RAG, system retrieves information and generates output in one continuous process but Agentic RAG introduces autonomous decision-making. This makes the AI smarter, more dynamic and adaptable to changing conditions. It can decide what to retrieve, how to interpret that data and what action to take next.

Agentic AI

Agentic AI refers to artificial intelligence systems that can perform autonomous decision-making. Unlike traditional AI systems that process data reactively, agentic AI systems interact with their environment to make decisions and take actions. These agents don't just follow a fixed set of instructions but they learn from environment and adapt their actions based on real-time data. This makes them capable of dynamically adjusting to new situations, making them more flexible and intelligent in complex real-world applications.

Architecture of Agentic RAG

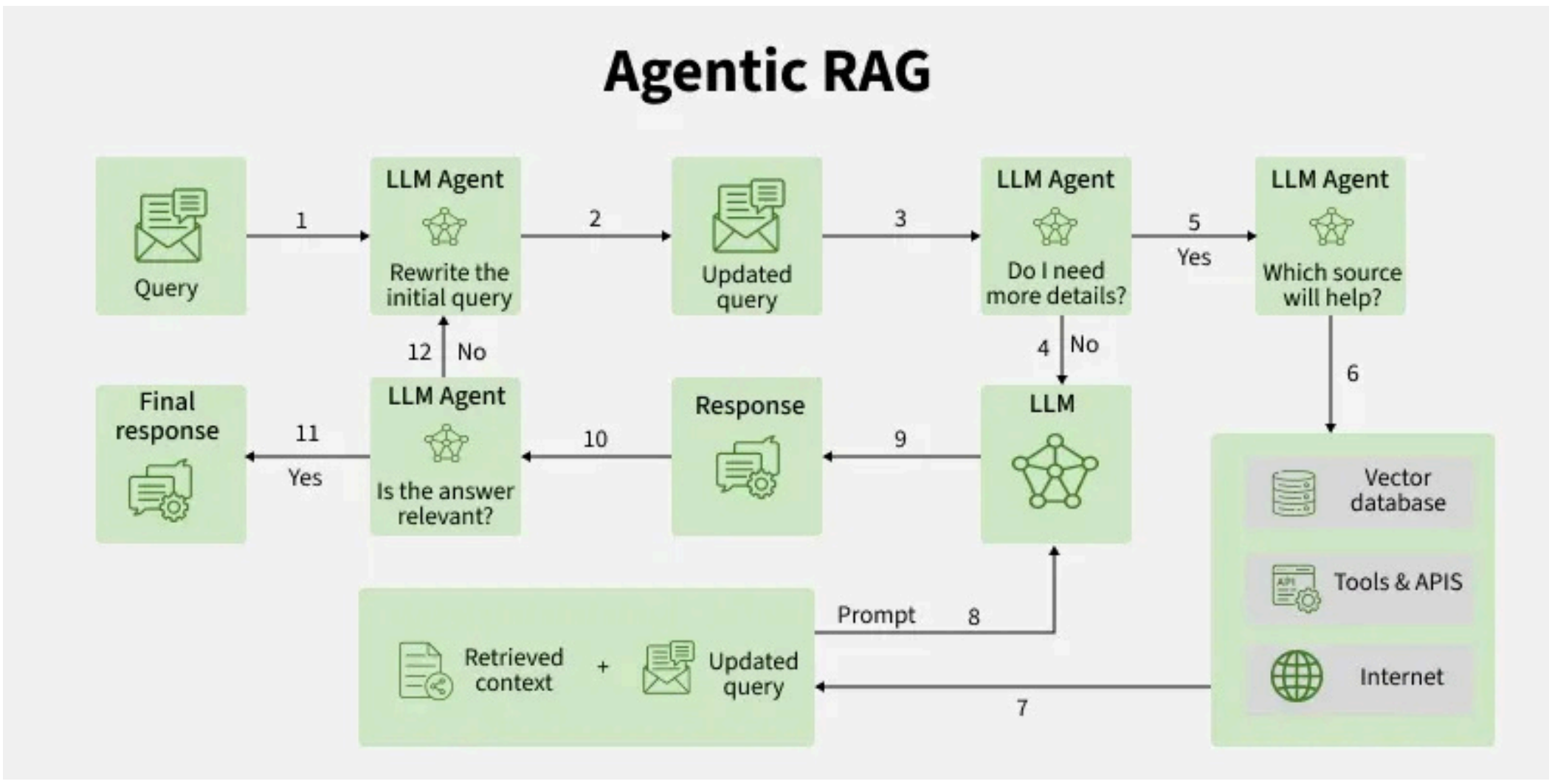
Agentic RAG (Retrieval-Augmented Generation) system consists of several key components that work together to help autonomous operation and dynamic decision-making. These components allow system to retrieve relevant data, process it and generate contextually correct responses or actions based on user queries.



- **Query:** Process starts with user's query which is the input provided by the user helps in triggering system's operation.
- **Data Sources:** System retrieves data from external systems such as APIs, databases, documents or knowledge graphs. These data sources provide the raw information needed to address user's query.
- **Retrieval System:** It queries relevant data from the sources based on user's query. It uses techniques like semantic search or vector search to ensure retrieved data matches with the context of the query.
- **Tools:** System also includes tools such as a Calculator for performing numerical operations and a Web Search tool for retrieving real-time, external information when needed. These tools expand system's ability to gather and process relevant data.
- **Agent Decision Module:** After the data is retrieved, it processes and evaluates its relevance and decides how to proceed. Agent autonomously identifies next steps such as whether to combine data, request additional information or refine the current query.
- **Generation Module:** Once the agent has made decisions, Generation Module takes over to produce the final response. Using advanced language models like GPT-3, the module transforms the processed data into a contextually relevant response such as answering a question, executing a command or providing a report.

How Does Agentic RAG Work?

Agentic RAG (Retrieval-Augmented Generation) system consists of several key components that work together to help autonomous operation and dynamic decision-making. These components allow system to retrieve relevant data, process it and generate contextually correct responses or actions based on user queries



- **Query Input:** Process begins when system receives user's query. This is the starting point of the entire process where user's request or question is input into the system.
- **Rewrite the Initial Query:** LLM Agent checks the query for clarity and rewrites it if necessary for better understanding.
- **Updated Query:** After the query has been rewritten, system moves forward with the updated query. This ensures the input is optimized for accurate data retrieval and processing.
- **Checking if more info needed:** It evaluates whether the updated query still requires more details. If the query is sufficiently clear and specific, it moves to the next step or additional details are needed.
- **If More Details Needed:** If LLM Agent finds that additional information is needed, system will request or search for more details to ensure the response is as accurate as possible.
- **Deciding Source:** Once system has enough context, LLM Agent decides which data source will best to solve the query. It identifies various options such as vector databases, tools and APIs or even internet for real-time data.
- **Data Retrieval:** It proceeds to query selected data source. These can include specialized vector databases for structured data, tools and APIs for specific queries.

- **Updated Query + Retrieved Context:** After retrieving relevant data, system combines this retrieved context with the updated query. This ensures that query is now updated with external data or context.
- **Processing the Data:** LLM processes updated query and retrieved context. It uses this information to understand query in-depth and helps in giving contextually appropriate response.
- **Response:** LLM generates response based on the data and context provided. This is the important step of the process where system changes raw data into a meaningful output which could be a direct answer or a recommendation.
- **Checking Answer Relevance:** Once response is generated, LLM Agent will checks whether the response is relevant to the original query or not. It ensures that the response accurately answers user's question and aligns with the context of the query.
- **Final Response:** Once the LLM Agent confirms that the generated response is relevant and accurate and then final response is delivered to the user.

AGENTIC RAG

Level-8	**Alignment (Safety & Control)** Keeping AI in check	Guardrails AI, Arize, LangFuse, Helicon
Level-7	**Memory (Context Persistence)** Retain and recall	Zep, Meo, Cognee, Letta
Level-6	**Data Extraction (Ingestion Layer)** From raw data to structured knowledge	Firecrawl, Scrapy, Docling, LlamaParse
Level-5	**Embedding (Semantic Understanding)** Turning data into meaning	BAAI, Nomic, Ollama, Voyage AI, OpenAI
Level-4	**VectorDB (Retrieval Layer)** Where context lives	Pinecone, Chroma, Milvus, Weaviate
Level-3	**Framework (Orchestration)** Glue your pipeline together	LangChain, LlamaIndex, Haystack
Level-2	**LLM (Reasoning Engine)** The brain behind the operation	Llama-4, Gemini 2.5 Pro, Claude 4, GPT-5
Level-1	**Evaluation** Measure to improve	LangSmith, Phoenix, DeepEval, Ragas
Level-0	**Deployment** Where AI meets infrastructure	Groq, AWS, GCP

Level 0: Deployment

- This is the infrastructure foundation. Without deployment platforms, none of the upper layers can actually run in production.
- **Role:** Provides compute resources, scaling, networking, and runtime environments.
- **Providers:** Groq (AI accelerator hardware for ultra-low latency inference), AWS (scalable cloud hosting with GPU/TPU instances), Google Cloud (Vertex AI and TPU pods), Together.ai (optimized inference hosting for open-source LLMs).
- **Importance:** Different AI workloads demand different compute (e.g., training vs inference, small-batch vs large-batch). Specialized providers like Groq allow millisecond inference for real-time chatbots, while hyperscalers like AWS/Google Cloud manage enterprise workloads.
- **Connection:** All upper layers (LLMs, Frameworks, Databases, etc.) are hosted here. The performance of the entire stack depends on deployment reliability.

Level 1: Evaluation

- Before models are trusted, they must be measured, tested, and validated.
- **Role:** Ensure AI outputs are accurate, consistent, fair, and safe.
- Tools:
- **LangSmith** – evaluates chain-of-thought reasoning and structured workflows.
- **Phoenix** – focuses on observability and debugging of AI pipelines.
- **DeepEval** – regression testing for AI applications.
- **Ragas** – evaluates Retrieval-Augmented Generation (RAG) pipelines.

Key Methods:

- Automatic metrics (BLEU, ROUGE, perplexity, cosine similarity).
- Human-in-the-loop feedback (safety checks, annotation).
- Scenario testing (adversarial prompts, edge-case evaluation).
- **Connection:** Feedback loops from Evaluation guide Alignment (Level 8) and improve Frameworks (Level 3).

Level 2: LLMs

- This is the intelligence core — the reasoning brain of the system.
- **Role:** Large Language Models process prompts, generate natural text/code, and perform reasoning tasks.

Examples:

- **Llama 4** (open-source, Meta's most advanced LLM).
- **Gemini 2.5 Pro** (Google DeepMind's multimodal model).
- **Claude 4** (Anthropic, safety-optimized, strong long-context handling).
- **GPT-4o** (OpenAI, optimized for multimodal reasoning with speech/image input).
- **Capabilities:** Natural language understanding, summarization, reasoning, translation, coding, multimodal processing.
- **Limitations:** Prone to hallucination, lack of long-term memory, context window limitations.
- **Connection:** LLMs consume embeddings (Level 5), integrate via frameworks (Level 3), and are evaluated (Level 1).

Level 3: Framework

- Frameworks are orchestrators that bridge LLMs with tools, APIs, and custom workflows.
- **Role:** Provide pipelines for building applications like chatbots, RAG systems, or autonomous agents.

Examples:

- **LangChain** – modular pipelines for chaining LLM prompts.
- **LlamaIndex** – focused on data indexing + retrieval integration.
- **Haystack** – enterprise-ready NLP framework for QA and search.
- **DSPy** – declarative framework for prompting and optimization.
- **Why Needed:** LLMs alone can't interact with external data, APIs, or memory. Frameworks enable dynamic tool use, structured reasoning, and knowledge integration.
- **Connection:** Frameworks connect to VectorDBs (Level 4), Memory (Level 7), and Evaluation tools (Level 1).

Level 4: VectorDb

- A vector database is the knowledge retrieval engine.
- Role: Stores embeddings (numerical representations of meaning) and allows fast similarity searches.

Examples:

- **Pinecone** – cloud-native, scalable vector DB.
- **Chroma** – open-source lightweight DB.
- **Milvus** – industrial-scale, billion-vector indexing.
- **Weaviate** – hybrid search (semantic + keyword).

Use Cases:

- RAG (retrieval-augmented generation).
- Semantic search (find meaning, not keywords).
- Recommendation systems.
- **Tech:** Uses Approximate Nearest Neighbor (ANN) algorithms (HNSW, IVF, PQ) for high-speed similarity search.
- **Connection:** Embeddings (Level 5) feed into VectorDB; frameworks (Level 3) query VectorDB to provide relevant context for LLMs.

Level 5: Embedding

- Embeddings are the mathematical foundation of meaning representation.
- Role: Convert words, sentences, or images into high-dimensional numerical vectors.

Providers:

- **Nomic** – visual embedding exploration.
- **Ollama** – lightweight LLM and embedding tools.
- **Voyage AI** – specialized high-quality embeddings.
- **OpenAI** – leading general-purpose embeddings API.

Importance:

- Enables semantic similarity (e.g., "car" and "automobile" close in space).
- Used in clustering, search, classification, and reasoning.
- Connection: Embeddings power VectorDB (Level 4), inform LLMs (Level 2), and help in downstream tasks like recommendation engines.

Level 6: Data Extraction

- This is the raw knowledge intake layer.
- Role: Gather, clean, and structure raw information for AI systems.

Examples:

- **Firecrawl** – automated web scraping.
- **Scrapy** – open-source crawling framework.
- **Docling** – document parsing & conversion.
- **Llamaparse** – parsing unstructured data into structured formats for LLMs.
- **Importance**: AI models are only as good as their data. Clean, structured, and updated knowledge ensures accuracy.
- **Connection:** Extracted data → transformed into embeddings (Level 5) → stored in VectorDB (Level 4).

Level 7: Memory

- Memory makes LLMs context-aware across time.
- Role: Store and retrieve user/session history, knowledge, and state.

Examples:

- **Zep** – scalable memory management.
- **Mem0** – lightweight memory plugin.
- **Cogngee** – enterprise-grade contextual memory.
- **Letta** – conversational memory manager.

Why Important:

- LLMs are stateless; they forget once the session ends.
- Memory allows personalization, long-term assistants, and continuity.
- Connection: Frameworks (Level 3) use memory to feed back relevant user history into LLM prompts.

Level 8: Alignment

- This is the safety + ethics layer, ensuring AI behaves responsibly.
- Role: Monitor, constrain, and guide model outputs according to rules, policies, and human values.

Examples:

- **Guardrails AI** – ensures outputs follow defined safety rules.
- **Arize** – ML observability, fairness checks.
- **Langfuse** – tracing and debugging.
- **Helicone** – LLM observability and monitoring.

Key Goals:

- Prevent harmful outputs (bias, toxicity, disinformation).
- Ensure compliance with regulations.
- Maintain transparency and auditability.
- Connection: Uses signals from Evaluation (Level 1), applies rules at the LLM (Level 2) and Framework (Level 3) layers, and feeds into deployment governance.

DroneX AI

WAS THIS POST USEFUL?

FOLLOW FOR MORE!

REPOST