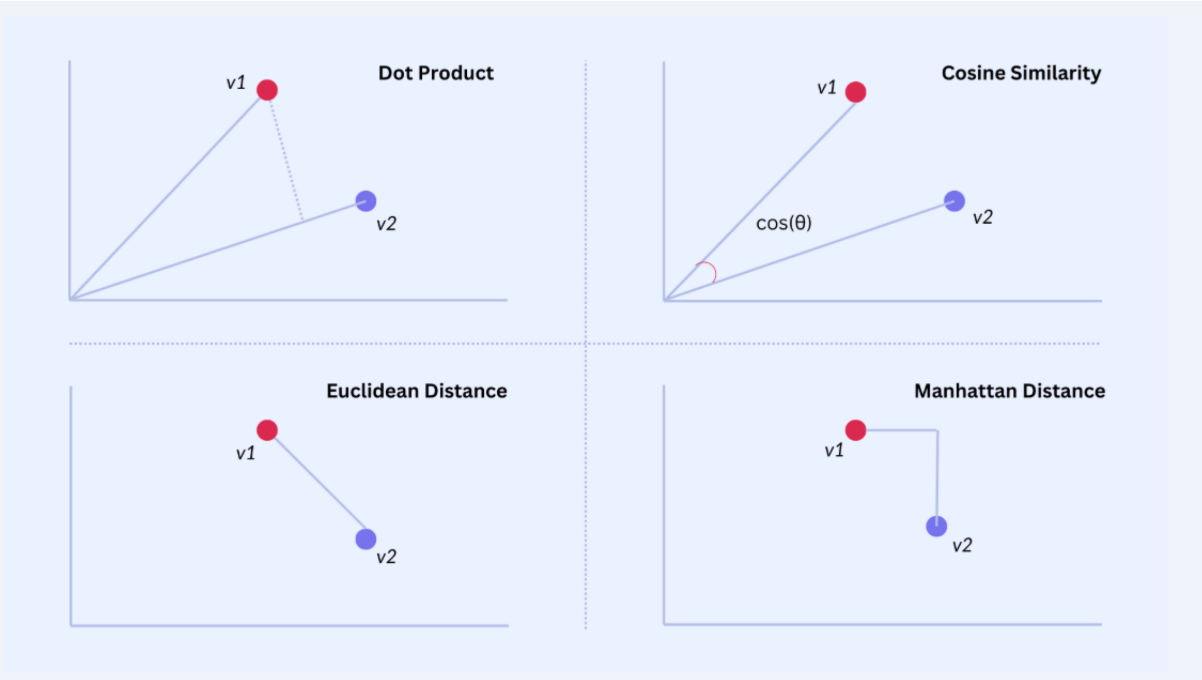
Introduction to the Vector Database

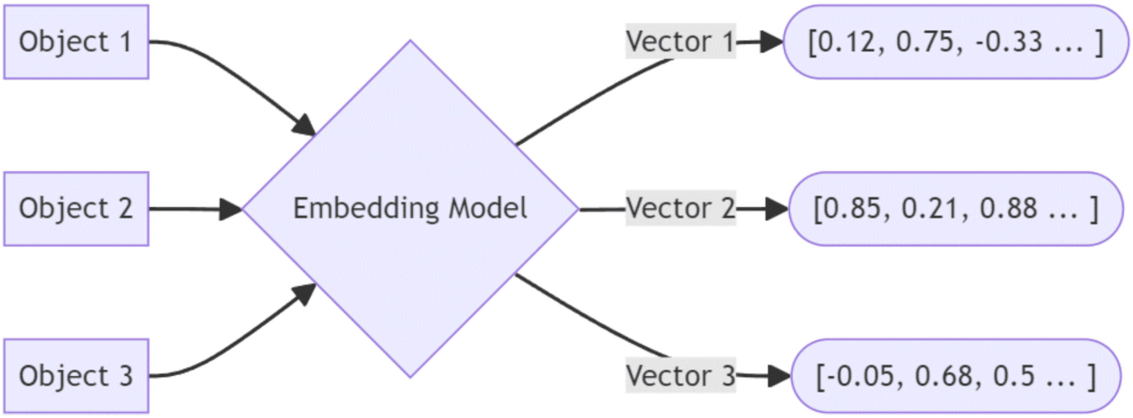
Vector Similarity



Distance Comparison:

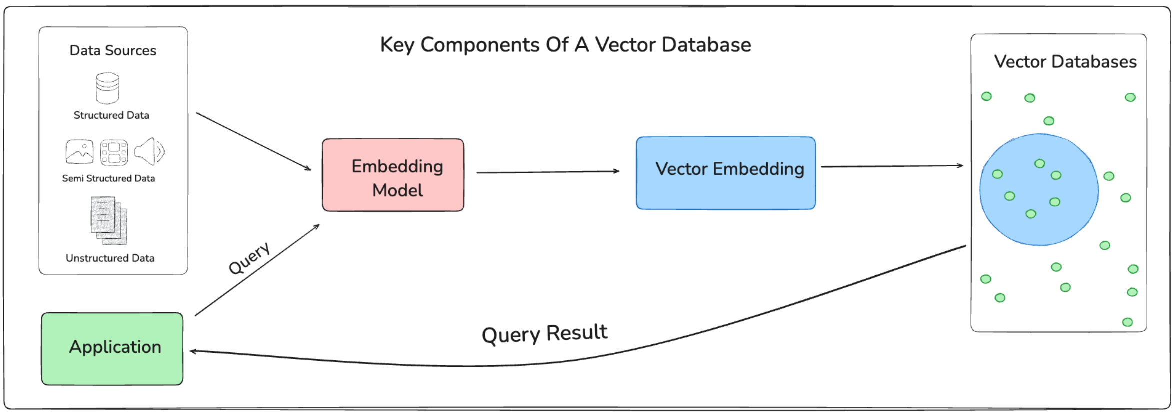
- **Dot Product**: Obtained by multiplying corresponding elements of the vectors and then summing those products. A larger dot product indicates a greater degree of similarity.
- **Cosine Similarity**: Calculated using the dot product of the two vectors divided by the product of their magnitudes (norms). Cosine similarity of 1 implies that the vectors are perfectly aligned, while a value of 0 indicates no similarity. A value of -1 means they are diametrically opposed (or dissimilar).
- **Euclidean Distance**: Assuming two vectors act like arrows in vector space, Euclidean distance calculates the length of the straight line connecting the heads of these two arrows. The smaller the Euclidean distance, the greater the similarity.
- **Manhattan Distance**: Also known as taxicab distance, it is calculated as the total distance between the two vectors in a vector space, if you follow a grid-like path. The smaller the Manhattan distance, the greater the similarity.

Vector Embedding



- Vector embedding is a numerical representation of text, images, or data.
- Each piece of content is converted into a high-dimensional vector of numbers.
- These vectors capture meaning based on patterns learned by the embedding model.
- Similar meanings produce vectors that lie close together in vector space.
- Dissimilar meanings produce vectors far apart from each other.
- Embeddings allow machines to understand semantic relationships mathematically.
- Instead of keywords, retrieval is based on semantic similarity.
- A typical embedding model outputs 256 to 1536-dimensional vectors.
- Each dimension captures latent features the model has learned.
- Embeddings support search, clustering, classification, and ranking tasks.
- In RAG, both documents and queries are converted into embeddings.
- Vector databases store and index these embeddings efficiently.
- During retrieval, similarity metrics like cosine similarity are applied.
- The closest vectors represent the most relevant information.
- Embeddings make modern semantic search and intelligent retrieval possible.

Vector Database



When to use Vector Database

Feature	OLTP Database	OLAP Database	Vector Database
Data Structure	Rows and columns	Rows and columns	Vectors
Type of Data	Structured	Structured/Partially Unstructured	Unstructured
Query Method	SQL-based (Transactional Queries)	SQL-based (Aggregations, Analytical Queries)	Vector Search (Similarity-Based)
Storage Focus	Schema-based, optimized for updates	Schema-based, optimized for reads	Context and Semantics
Performance	Optimized for high-volume transactions	Optimized for complex analytical queries	Optimized for unstructured data retrieval
Use Cases	Inventory, order processing, CRM	Business intelligence, data warehousing	Similarity search, recommendations, RAG, anomaly detection, etc.

Retrieval-Augmented Generation

1. Introduction

Retrieval-Augmented Generation (RAG) is a technique that improves Large Language Model (LLM) responses by grounding the model in real, trusted information sources. Instead of allowing the LLM to answer solely from its internal parameters, RAG retrieves relevant context from external knowledge bases such as PDFs, databases, websites, or enterprise documents, and supplies them to the model.

This reduces hallucination, improves accuracy, and enables domain-specific responses without retraining the model.

---

2. Why RAG Is Needed

Pure LLMs rely on:

- Pretrained patterns
- Probabilistic reasoning
- Knowledge up to a cutoff

They cannot access:

- Private company data
- Updated information
- Regulatory documents
- Customer-specific instructions

RAG solves this by injecting external knowledge into the prompt at runtime.

---

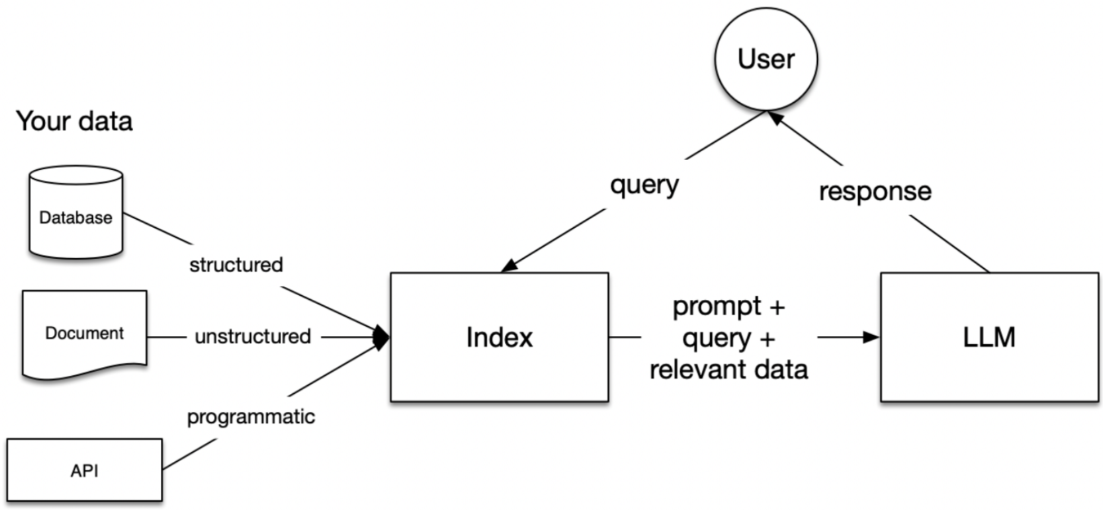
3. High-Level Workflow

A RAG pipeline follows this sequence:

1. The user asks a question.
2. The system converts the question into an embedding (a vector representation).
3. The vector is used to search a vector store (FAISS, Chroma, Pinecone, etc.) for semantically similar documents.
4. Relevant documents are retrieved.

5. The LLM receives both the query and the retrieved context.
6. The LLM generates an answer grounded in real information.

This ensures responses are accurate and verifiable.



---

4. Detailed Mechanics of Each Stage

4.1 Input Question

The user provides a natural-language question. Example:
"What are the steps to configure system access for new employees?"

This becomes the starting point for the retrieval process.

4.2 Embedding Generation

The system transforms the query into an embedding using an embedding model.
An embedding is a numerical vector representing semantic meaning.

Important characteristics:

- Similar meanings are close in vector space.
- The embedding model must match the one used for indexing documents.

4.3 Vector Search in the Knowledge Base

All documents stored in your knowledge base are pre-converted into embeddings.

During retrieval:

- The query vector is compared with stored document vectors.
- Techniques like cosine similarity or inner product determine relevance.
- The system selects the top-k most similar chunks.

This step is the core of RAG retrieval.

4.4 Chunking and Metadata

Documents are chunked into smaller units (200–1,000 tokens each).
Each chunk is stored with metadata such as:

- Source file path
- Page number
- Section title

Chunking ensures better recall and precision during search.

4.5 Context Assembly

The retrieved chunks are combined into a prompt block known as the "context window".
The system must:

- Limit context length to avoid token overflow
- Remove duplicates
- Maintain chronological or structural order if needed

4.6 Prompt Construction

The final prompt given to the LLM includes:

- System instruction
- User query
- Retrieved context

Example structure:

System: "Use only the context below to answer the question."
Context: [All retrieved chunks]
User: [Original question]

This ensures the LLM answers based on real data.

4.7 Generation

The LLM generates an answer grounded in the provided context.

Characteristics of RAG answers:

- Factually anchored
- Citeable (if required)
- Less hallucination
- Context-specific

4.8 Optional Enhancements

RAG can be extended with advanced components:

1. Query rewriting
   Improves retrieval quality by expanding or reformulating the question.
2. Multi-hop retrieval
   Retrieves related context iteratively.
3. Re-ranking
   Uses cross-encoders to choose the most relevant chunks.
4. Validation
   Checks if the generated answer is grounded in context.

These turn RAG into an enterprise-grade system.

5. RAG vs Model Training

Dimension	RAG	Model Training
Cost	Low	High
Speed to Update	Instant	Slow
Freshness of Knowledge	Real-time	Requires retraining
Hallucination Risk	Low (grounded)	Higher
Auditability	High	Low
Data Privacy	Strong isolation	Risk of leakage
Scalability	Unlimited	Model size limited
Control	Explicit	Implicit
Maintenance	Easy	Complex