RAG Fundamentals

Grounding Large Language Models in External Knowledge

A Technical Overview · 2026



Today's Journey

A complete mental model of RAG — from core concepts to production practice

01

Foundation

What RAG is, why it matters, and the core two-phase pipeline at a glance

PAGES 3–4

02

Deep Dive

Indexing, retrieval, generation — every component in detail with chunking strategies and embedding mechanics

PAGES 5–8

03

Frontiers

Evolution, advanced variants, real-world use cases, production challenges, and evaluation frameworks

PAGES 9–14

What is Retrieval-Augmented Generation?

Bridging the gap between static LLM knowledge and the dynamic world by retrieving external information at inference time.

Definition

RAG is an architectural pattern that enhances Large Language Models by integrating external, up-to-date knowledge from databases, documents, or the web at inference time — rather than relying solely on the model's internal parametric memory. Coined by Lewis et al. in their seminal 2020 paper at NeurIPS.

RAG =

Retrieval + Generation

Find the facts. Then write.

Four Problems RAG Solves

Knowledge Cutoff

Training data is static. RAG retrieves current information on demand.

Hallucinations

Models fabricate plausible answers. RAG grounds responses in evidence.

Fine-tuning Costs

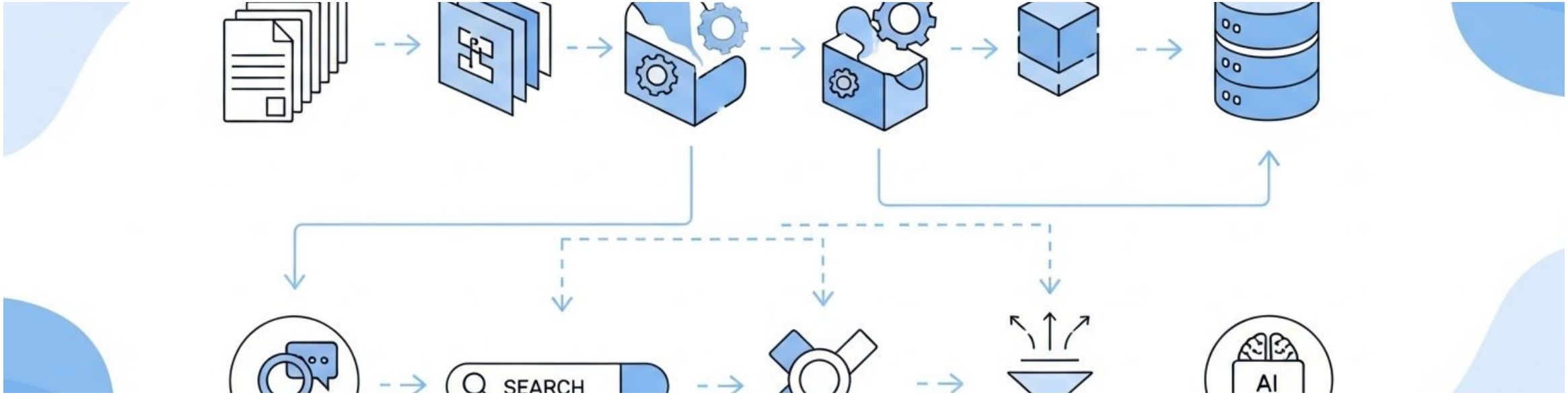
Adapting to new domains is expensive. RAG connects to data without retraining.

Private Data

Proprietary knowledge stays locked away. RAG indexes internal docs securely.

The RAG Pipeline: Two Phases

Offline indexing prepares the knowledge base; online query-time retrieves and generates grounded responses in milliseconds.



Indexing Phase (Offline)

Raw documents are ingested, split into chunks, converted to vector embeddings, and stored in a vector database. This preprocessing runs once and can be updated incrementally as new documents arrive.

Documents → Chunks → Embeddings → Vector DB

Query-Time Phase (Online)

The user query is embedded, top-K similar chunks are retrieved, combined with the original query, and fed to the LLM to produce a source-grounded response — all in milliseconds.

Query → Embed → Retrieve → Augment → Generate

Inside the Indexing Phase

Four sequential steps that transform raw documents into a searchable knowledge base.

1

Document Ingestion

Load and parse raw documents from diverse sources: PDFs, web pages, text files, databases. Extract structured text and metadata for citations.

2

Chunking

Split parsed text into semantically coherent, self-contained passages (500-2000 chars) with overlap to preserve cross-boundary context.

3

Embedding

Convert each chunk into a dense vector using an embedding model (BGE, E5, OpenAI). Same model must be used for both indexing and retrieval.

4

Vector DB

Store embeddings in a vector database: Pinecone, Weaviate, FAISS, Milvus. Fast ANN search at scale.

Key insight:
Each stage is critical to retrieval quality — a mistake in chunking or embedding propagates through the entire pipeline.

Inside the Query-Time Phase

Four steps that retrieve context and generate a grounded answer — in real time.

1

Query Embedding

Encode the user's question into the same vector space used during indexing. Identical embedding model is required.

2

Retrieval

Similarity search (cosine/dot-product) against vector DB to find top-K relevant chunks. Hybrid = dense + BM25 sparse.

3

Augmentation

Construct a prompt template: system instructions + retrieved context + original query.

4

Generation

LLM produces a grounded, source-aware response with inline citations.

Performance note:
Production systems combine these steps with streaming, caching, and hybrid search for sub-second end-to-end latency.

Chunking: The Retrieval Unit

Chunking strategy is the most impactful RAG design decision — too large dilutes relevance, too small loses context.

Fixed-size Chunking

CHARACTER / TOKEN COUNT

Split by character or token count with configurable overlap. Start with chunk_size=500, overlap=100.

Best for:

Uniform text with consistent structure

Recursive Chunking

HIERARCHICAL BOUNDARIES

Split hierarchically on natural boundaries: paragraphs, then sentences, then words. Preserves coherence.

Best for:

Narrative documents, articles, transcripts

Format-specific Chunking

DOCUMENT STRUCTURE AWARE

Use document structure as the splitting guide: Markdown headers, HTML sections, PDF chapters.

Best for:

Structured docs: Markdown, HTML, codebases

Semantic Chunking

EMBEDDING SIMILARITY

Use embedding similarity to find natural conceptual boundaries. Most coherent chunks for complex content.

Best for:

Complex, heterogeneous document collections

How Embeddings Power Retrieval

The mathematical foundation that makes RAG retrieval work — semantic similarity equals geometric proximity.

From Text to Vector Space

An embedding model maps each text chunk to a dense vector — typically 768 or 1536 dimensions. In this space, "Paris is the capital of France" sits close to "What is France's capital city?" but far from "How to bake a croissant."

Similarity Search Mechanics

Cosine similarity or dot product measures the angle between the query vector and each stored chunk vector. Approximate Nearest Neighbor (ANN) algorithms make this fast at millions of documents.

The Critical Rule

The same embedding model must be used for both indexing and retrieval. Mismatched models produce vectors in incompatible spaces.

Embedding Models (2025)

BGE-base-en-v1.5
E5-base-v2
GTE-large
NV-Embed-v2
text-embedding-3-small

Vector Dimensions

Typical: 768 (base) or 1536 (large)

Vector DB Options

Pinecone · Weaviate · Qdrant · FAISS · Milvus

The Evolution of RAG Architectures

Four generations — from simple retrieve-then-read to self-improving agentic systems.

01

Naïve RAG

2020–2021

Single retrieval, no reranking.
Retrieve → Read only.

02

Advanced RAG

2022–2023

Hybrid search (dense+sparse), cross-encoder reranking, query expansion, RRF fusion.

Production-ready quality.

03

Modular RAG

2024

Pluggable components for domain customization. Hierarchical (RAPTOR), graph-enhanced pipelines, composable stages.

04

Agentic RAG

2025–now

Adaptive retrieval, multi-hop reasoning, self-evaluation with confidence re-query, auto query decomposition.

Self-improving systems.

Beyond Basic RAG: Advanced Variants

Three extensions that push RAG into structured reasoning, self-critique, and cross-modal retrieval.

GraphRAG

Leverages knowledge graphs for structured, entity-aware retrieval instead of flat vector search.

Impact:

Up to 35% accuracy improvement on complex multi-entity reasoning.

Legal · Science · Enterprise KG

Self-RAG

The model learns to decide when retrieval is necessary and critiques its own outputs for faithfulness.

Impact:

Reduces hallucinations by training the LLM to recognize uncertainty.

High-stakes · Verifiable · Audit

Multi-modal RAG

Extends retrieval beyond text to images, audio, and video alongside traditional text document retrieval.

Impact:

Retrieve X-ray images + clinical notes in one unified retrieval system.

Healthcare · Design · Multimedia

RAG in Production: Key Use Cases

Not a research curiosity — RAG powers enterprise systems across industries today.

Enterprise Q&A

Internal knowledge bases, policy docs, FAQs with citations. Instant, verifiable answers from company data.

Healthcare

Clinical decision support, medical literature synthesis. EHR summarization with traceable evidence trails.

Legal

Case law retrieval, contract analysis, regulatory compliance. Every assertion backed by a specific precedent or clause.

Customer Support

Grounded chatbots citing product docs instead of hallucinating policies. Consistent, auditable interactions.

Code Assistants

Course- or codebase-specific programming help. Retrieve relevant docs and examples for context-aware code.

Research

Automated systematic literature reviews. Query thousands of papers with grounded, cited summaries in seconds.

Production Challenges & Emerging Solutions

Each challenge has a maturing solution pattern — turning blockers into engineering decisions.

Retrieval Quality

Noisy or irrelevant context degrades generation quality downstream.

Solution

Hybrid search (dense + BM25), cross-encoder reranking, query rewriting, metadata filtering.

Latency

Multi-stage retrieval + LLM generation can exceed user patience budgets.

Solution

Streaming architectures, embedding caching, smaller distilled models, edge deployment.

Hallucinations

LLMs may ignore evidence or fabricate despite having retrieved context.

Solution

Citation grounding, self-evaluation loops, abstention when confidence is low.

Privacy & Security

Sensitive documents must not leak through the LLM pipeline.

Solution

On-premise vector DBs, federated retrieval, access-control-aware chunk filtering.

Production Stack

A production-grade system addresses all four challenges simultaneously. No single technique solves everything.

Hybrid Search

Dense vector + BM25 sparse

Reranking

Cross-encoder precision

Guardrails

Citation + abstention

Isolation

On-premise + access control

Measuring RAG Quality: The RAGAS Framework

Standardized evaluation measuring both retrieval and generation quality — because one without the other is broken.

Faithfulness

Are claims in the answer fully supported by the retrieved context? The most critical metric — measures hallucination.

Retrieval + Generation

Answer Relevancy

Does the answer address the question, or did context distract the model into a tangent?

Generation Quality

Context Precision

What fraction of retrieved chunks is actually relevant? High precision means efficient use of the context window.

Retrieval Quality

Context Recall

Did retrieval find all relevant information? High recall means nothing important was missed.

Retrieval Quality

How RAGAS Works

RAGAS computes these metrics by prompting an evaluator LLM to judge each dimension against the retrieved context, producing scores that correlate with human judgment. Combined with traditional IR metrics (NDCG, MRR), it gives a complete picture of RAG quality.

RAG is the bridge from language models to knowledge systems.

The journey from naive retrieve-and-read to agentic, self-improving RAG mirrors the broader arc of AI: from pattern matching to reasoning.

Understanding the fundamentals is the foundation for building AI systems that are not just fluent, but truthful.