RAG (Retrieval-Augmented Generation) — Short Primer

What is RAG? Retrieval-Augmented Generation (RAG) combines a generator model with an external knowledge index. At query time, the system retrieves the most relevant chunks from a vector store and feeds them (as context) to the model to produce grounded, more accurate answers.

Why use RAG? • Keeps answers up-to-date without retraining the model. • Reduces hallucinations by citing retrieved passages. • Lets you control and audit sources.

Key building blocks 1) Ingestion pipeline - Load documents (PDF, TXT, MD). - Split into chunks (e.g., ~500 chars, 100 overlap). - Embed chunks into vectors. - Store vectors in a similarity search index (e.g., FAISS). 2) Retrieval - Convert user question into an embedding. - Similarity search finds top-k chunks. 3) Generation - Provide retrieved chunks + question to the LLM to craft an answer. - Optionally return sources and scores.

What “good” chunks look like • Self-contained: each chunk carries enough local context to be useful alone. • Consistent style: similar size, limited noise, minimal boilerplate. • Traceable: metadata includes filename, page, and a stable ID. RAG_TEST_TOKEN_002

Ingestion tips for this PDF • This file has short headings and compact paragraphs—ideal for your RecursiveCharacterTextSplitter. • Your script will add: chunk_id and filename to each chunk’s metadata. • Expect multiple chunks due to chunk_size≈500 and overlap≈100. • Use this to verify end-to-end storage and retrieval.

Suggested retrieval prompt skeleton “Use the context chunks to answer. If the answer isn’t in the context, say you don’t know. Cite the chunk filenames and chunk_id values that support the answer.”

Common pitfalls (and fixes) • Pitfall: Overlapping or duplicated documents inflate the index. Fix: Deduplicate by a content hash before embedding. • Pitfall: Huge chunks reduce recall; tiny chunks reduce coherence. Fix: Tune chunk_size and overlap via small experiments. • Pitfall: Embedding model mismatch. Fix: Keep the same embedding model for both indexing and querying. • Pitfall: Metadata loss. Fix: Always preserve source (path), page numbers (for PDFs), and custom IDs.

Minimal test you can run 1) Build the index using this file as the only document. 2) Query: “What is RAG?” — The system should retrieve one or more chunks containing the ‘What is RAG?’ paragraph. 3) Query: “Name two benefits of RAG.” — Expect bullets from ‘Why use RAG?’ 4) Query: “What fields does the ingestion code add to metadata?” — Expect: chunk_id and filename. RAG_TEST_TOKEN_003

Glossary (fast) • Chunk: a short slice of text used for retrieval. • Embedding: a numeric vector representing text meaning. • Vector store: a database optimized for similarity search over embeddings. • Top-k: number of nearest chunks returned by the retriever.

That’s it—happy indexing! RAG_TEST_TOKEN_004