■ RAG + Vector DBs: Conceptual Guide

■ Project Overview

This notebook walks through how to use a vector database (FAISS) and a retrieval-augmented generation (RAG) pipeline to enable a language model (LLM) to "remember" facts and generate more accurate outputs.

■ Key Components

• SentenceTransformer: Transforms text into dense embeddings (vectors). • FAISS: Facebook AI Similarity Search library for fast vector retrieval. • Transformers Pipeline: Uses distilgpt2 to generate text responses. • Prompt Engineering: Controls LLM behavior using query strings.

■ Step-by-Step Concepts

1. Install & Import Dependencies
Install FAISS, SentenceTransformers, and Transformers for vectorization, retrieval, and generation.

2. Baseline Prompt
A prompt like 'The first president of the United States was...' is sent to the LLM without context to measure baseline accuracy.

3. Knowledge Base
A list of factual historical sentences is defined. These are what the LLM will use to retrieve supporting context.

4. Generate Embeddings
Each fact is converted into a 768-dim vector using a pretrained SentenceTransformer. Normalization is applied to ensure cosine similarity behaves correctly.

5. FAISS Index
The embeddings are stored in FAISS, allowing efficient similarity search between a query and stored facts.

6. Query → Embedding → Retrieve
A new user question is embedded and compared against the vector database to retrieve relevant documents.

7. Retrieval-Augmented Prompt
The retrieved context is prepended to the original prompt and sent to the LLM for a more accurate, context-informed answer.

■ Learning Takeaways

• LLMs are not databases — they benefit from external context. • Vector embeddings allow semantic retrieval of relevant documents. • RAG pipelines turn LLMs into open-book systems. • You can scale

this for PDFs, websites, or business documents.

End of Guide – Happy Building!