Build a Simple Offline RAG Demo (Step-by-Step)

*Learn RAG by building a local, offline-first retriever + answerer with optional LLM grounding.*

Author: Waseem Ibn Yousef CM

# Build a Simple Offline RAG Demo (Step-by-Step)

This tutorial walks you through building a Retrieval-Augmented Generation (RAG) demo that runs offline using TF‑IDF. You’ll learn the core ideas and have a working CLI app you can extend with LLMs, embeddings, or a web UI.

## Why RAG?

RAG reduces hallucinations by grounding answers in retrieved documents. Instead of forcing the model to “know everything,” you fetch relevant passages and ask the model (or a simple heuristic) to answer using those passages.

## What we’ll build

* A local corpus of `.txt` files
* A chunker that splits documents
* A TF‑IDF vector index for retrieval
* Two answerers:

- OfflineAnswerer: extractive summary with citations

- OpenAIAnswerer: optional LLM answer grounded in retrieved context

* A CLI to query the system

## Prereqs

* Python 3.10+
* Windows PowerShell or your shell of choice

## Setup

python -m venv .venv  
. .venv/Scripts/Activate.ps1  
pip install -r requirements.txt

## Project layout

src/  
 app/  
 rag\_pipeline.py  
 cli.py  
data/  
 intro\_rag.txt  
 tfidf\_baseline.txt  
tests/  
 test\_rag.py  
README.md

## Core code explained

### Loading and chunking

We load `.txt` files and split into overlapping chunks so retrieval can score localized passages.

Key parameters:

* `chunk\_size`: characters per chunk (default 600)
* `chunk\_overlap`: overlap between chunks (default 80)

### TF‑IDF retrieval

We use `sklearn` to vectorize chunks and compute cosine similarity. It’s fast, transparent, and offline.

### Answering

* Offline: pick sentences that share terms with the query; append citations `[doc#chunk]`.
* OpenAI: if `OPENAI\_API\_KEY` is set and `--provider openai`, the system sends context + query to a chat model.

## Try it

python -m src.app.cli "What is RAG and why use it?"

Optional with OpenAI:

$env:OPENAI\_API\_KEY = "sk-..."  
python -m src.app.cli "Explain TF-IDF in RAG" --provider openai --model gpt-4o-mini

## Testing

pytest -q

## Extend it

* Swap TF‑IDF for sentence embeddings (e.g., `sentence-transformers`) and a vector DB (FAISS, Chroma).
* Add loaders for PDFs/HTML/Markdown.
* Build a Streamlit UI with source highlighting.
* Add multi-turn memory and tool use (web search, calculators) for agentic workflows.

## FAQ

* No API key needed to run offline mode.
* Add your own `.txt` files to `data/` and ask domain-specific questions.

## License

MIT

## About the author

Waseem Ibn Yousef CM is an AI engineer focused on practical, developer-friendly AI systems. Interests include RAG, agents, and shipping clear tutorials that help others learn fast.