


RAG Assistant – Ibrahim Knowledge Base

 **Repo/Script Entry:** `__main__` triggers indexing via `create_vector_store()`

Aim of the Project

This project builds a lightweight Retrieval-Augmented Generation (RAG) assistant fine-tuned to answer questions about **Ibrahim's work and project documents**. It indexes PDFs and CSVs, stores vector embeddings in a local FAISS index, and serves grounded answers with linked sources. When questions are too vague or off-topic, it gently steers users to ask about Ibrahim's projects.

Objectives - Ingest and clean documents (PDFs, CSVs) about Ibrahim's work. - Create a persistent FAISS vector store for fast semantic search. - Retrieve the most relevant chunks (k=4) and generate faithful answers with citations. - Provide a robust default/fallback message for vague or unsupported queries.

Tech Stack

- **Core:** Python, LangChain
- **Vector Store:** FAISS (local persisted index)
- **Models (OpenAI-compatible via GitHub Models):**
 - Chat LLM: `gpt-4o-mini` (override via `GHM_MODEL`)
 - Embeddings: `text-embedding-3-small` (override via `GHM_EMBED`)
 - Endpoint: `https://models.inference.ai.azure.com`
- **Interface:** Script with Streamlit import (future UI), CLI entry in `__main__`

Environment Variables - `GITHUB_TOKEN` (required) - `GHM_MODEL` (optional) - `GHM_EMBED` (optional)

Data Collection & Cleaning

Data Sources - Folder: `documents/` - Supported types: **PDF, CSV**

Loaders - **PDFs:** `PyPDFLoader` splits by pages, emits `Document` objects per page. - **CSVs:** Loaded into pandas; UTF-8 first, with ISO-8859-1 fallback; serialized back to CSV-text for embedding.

Text Cleaning (`clean_text`) - Concatenates hyphenated line breaks. - Joins single newlines into spaces, preserves paragraph breaks. - Compacts 3+ newlines to 2. - Drops standalone numeric lines (page artifacts, TOC counters).

Result: Cleaner, denser passages that embed better and reduce noise from PDF layout artifacts.

Document Chunking

- **Splitter:** `RecursiveCharacterTextSplitter`
 - **Parameters:** `chunk_size=1000`, `chunk_overlap=200`
 - **Rationale:** Keeps topical coherence while limiting token bloat; overlap retains cross-chunk context for retrieval.
-

Embeddings & Vector Store

- **Embeddings:** `OpenAIEmbeddings` via GitHub Models endpoint, keyed by `GITHUB_TOKEN`.
- **Store:** `FAISS.from_documents(chunks, embeddings)`
- **Persistence:** Saved to `faiss_index/` (created if absent). Enables warm restarts without re-embedding.

Index Build Pathways 1. **Explicit:** Run the script (`__main__`) → builds and saves FAISS. 2. **Lazy:** On first query, if `faiss_index/` is missing, auto-index the `documents/` folder.

Retrieval & Question Answering

Retriever - Built from the FAISS store with `search_kwargs={"k": 4}` to surface top-4 chunks.

LLM - `ChatOpenAI` with low temperature (`0.2`) to favor factual, grounded outputs.

Prompt Strategy - System-style prefix sets scope: "helpful assistant trained on Ibrahim's work." - **Vagueness Guardrail:** If user asks general or unrelated questions, assistant nudges toward Ibrahim's data projects. - **Chat History Conditioning:** Optional `chat_history` (list of `{role, content}`) is concatenated into the prompt to preserve context across turns.

QA Chain - `RetrievalQA.from_chain_type(...)` returns both the **answer** and **source_documents**. - Sources are normalized to `filename[, page N]` for user-readable citations.

Fallback Behavior - If no sources or the LLM signals insufficient question quality, returns a friendly, scoped prompt suggesting topics like "football clustering," "prediction systems," etc.

Error Handling & Robustness

- **Missing Token:** Raises at startup if `GITHUB_TOKEN` is not set.
 - **CSV Encoding:** Graceful fallback from UTF-8 to ISO-8859-1.
 - **Dangerous Deserialization:** `allow_dangerous_deserialization=True` for FAISS load (required by some LangChain/FAISS versions); keep index directory trusted.
 - **Cold Start:** Auto-builds index if FAISS directory not found.
-

Security & Privacy

- Secrets isolated via environment variables; tokens never hard-coded.
 - All content remains local to the machine; FAISS index is on-disk under project control.
 - If deploying Streamlit or an API, restrict file uploads and validate content paths.
-

Usage

Build / Rebuild the Index

```
python rag_assistant.py  
# or run the module that contains __main__
```

This scans `documents/`, cleans and chunks, embeds, and saves FAISS to `faiss_index/`.

Ask a Question (Programmatic)

```
answer, sources = get_rag_response("What is Ibrahim's clustering workflow?")  
print(answer)  
print(sources)
```

Inputs - `query: str` - user question. - `chat_history: list[dict] | None` - optional prior messages: `{"role": "user|assistant", "content": "..."}.`

Outputs - `answer: str` - grounded response. - `sources: list[str]` - list of deduplicated `filename[, page]` indicators.

Design Decisions & Rationale

- **k=4** passages: balances breadth (mitigates single-chunk bias) and token cost.
 - **Low temperature (0.2)**: increases determinism and reduces hallucinations.
 - **Simple custom prompt**: scoped expertise avoids off-topic drift for a personal knowledge base.
 - **Local FAISS**: lightweight, dependency-free at runtime; no managed vector DB required.
-

Example Interaction

Q: "How does Ibrahim approach player clustering?"

A (summarized): Explains scraping from FBRef, cleaning & feature engineering (per-90 metrics across defensive/attacking/possession), scaling + K-means, elbow method to pick K, PCA visualization (2D/3D), and insights (e.g., unique creativity profiles).

Sources: e.g., `Clustering Project Report.pdf, page 3`; `Clustering Project Report.pdf, page 5` (actual pages depend on indexed files).

Limitations

- **No re-ranking:** Pure similarity search may surface redundant chunks; consider hybrid retrieval or MMR for diversity.
 - **No structured citations in text:** Current chain returns raw sources; a custom prompt template could format inline citations.
 - **Streamlit import unused:** Present code doesn't expose a UI; add an app layer to chat and upload docs.
-

Appendix – Key Functions

- `clean_text(text)`: removes layout noise and artifacts.
 - `load_and_clean_documents(folder_path)`: loads PDFs/CSVs and applies cleaning.
 - `split_documents(docs, chunk_size=1000, chunk_overlap=200)`: chunking strategy.
 - `create_vector_store()`: builds and persists FAISS.
 - `get_rag_response(query, chat_history=None)`: complete RAG pipeline with fallback.
-

Summary

This RAG assistant creates a maintainable, token-efficient, and privacy-respecting knowledge base for Ibrahim's portfolio and technical work. It lays the groundwork for a Streamlit UI and more advanced retrieval while already providing high-signal, source-grounded answers for common queries about his projects.