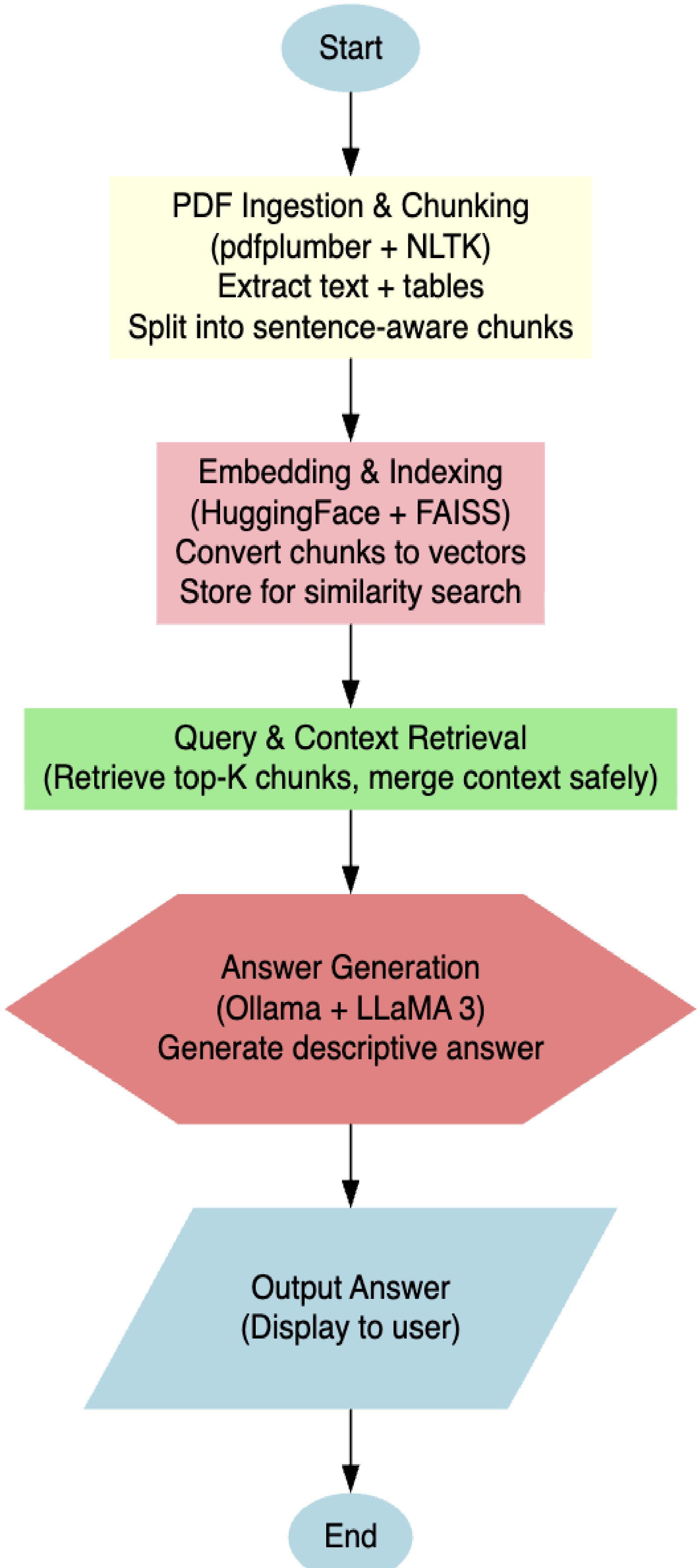
Retrieval-Augmented Generation (RAG) Pipeline with LLaMA 3 + Ollama

This repository implements a local RAG pipeline for question answering over PDF documents.
It combines document ingestion, embedding, retrieval, and LLM generation into one flow.

I implemented a multi-document RAG pipeline using four Apple financial reports. Text and tables were extracted, chunked, embedded, and indexed in a single FAISS store. The retriever fetches relevant chunks from all reports, merges context, and sends it to LLaMA via Ollama. This allows generating descriptive answers informed by multiple sources.

Pipeline Overview

1. PDF documents are ingested and text + tables are extracted
2. The extracted text is divided into manageable chunks
3. Each chunk is converted into embeddings using HuggingFace models
4. Embeddings are stored in a FAISS index for similarity search
5. Given a query, the most relevant chunks are retrieved
6. Retrieved chunks are merged into a single context while staying within model token limits
7. Ollama with LLaMA 3 generates descriptive answers from this context


Start
PDF Ingestion & Chunking
(pdfplumber + NLTK)
Extract text + tables
Split into sentence-aware chunks
Embedding & Indexing
(HuggingFace + FAISS)
Convert chunks to vectors
Store for similarity search
Query & Context Retrieval
(Retrieve top-K chunks, merge context safely)
Answer Generation
(Ollama + LLaMA 3)
Generate descriptive answer
Output Answer
(Display to user)
End

Pipeline Steps

1. PDF Ingestion

- **Library:pdfplumber**
- Read PDFs page by page.
- Extracts both plain text and tables (e.g., financial statements, charts).
- Tables are flattened into text using separators so that language models can process them (since most LLMs don’t understand table formats directly).

Why?
LLMs need structured text. Annual/quarterly reports often contain key figures in tables, so ignoring them would lose important information.

2. Text Chunking

- Library: nltk (Natural Language Toolkit)
- Splits the extracted content into chunks of sentences rather than arbitrary cuts.
- Chunks are limited by length (e.g., 2000 characters).

Why?

- Keeps sentences intact (avoids splitting "Apple reported $82B revenue..." mid-sentence).
- Balances granularity (too small = lots of noise, too large = hard retrieval).
- This makes downstream similarity search much more accurate.

3. Embeddings

- Libraries: langchain-huggingface, HuggingFace sentence-transformers
- Each chunk is converted into a vector embedding (dense representation).

- We use **all-MiniLM-L6-v2** → a small, efficient model that captures semantic meaning.

Why?

- Phrases like *"net sales"* and *"revenue"* may not be identical textually but are semantically close → embeddings capture that.
- LLMs can't search raw text efficiently → embeddings make similarity search possible.

4. Vector Indexing

- Library: FAISS (Facebook AI Similarity Search)
- Stores embeddings in a highly optimized index for **fast nearest-neighbor search.**
- Works even when documents are large (millions of chunks).

Why?
Instead of searching every chunk manually, FAISS quickly finds the k most similar chunks to a query in milliseconds.

5. Retrieval

- Library: langchain retrievers
- For a given question, the retriever queries FAISS and returns the **t**op-k chunks most relevant.
- Parameter k controls recall vs. precision:
  - Higher k = more coverage (risk of irrelevant context).
  - Lower k = shorter, sharper context.

Why?
Keeps only the most relevant slices of the document. Without retrieval, the model might hallucinate or drown in irrelevant details.

6. Context Merging

- **Logic:** Custom Python
- Retrieved chunks are concatenated into one context string.
- A cap (e.g., 4000 characters) ensures we don't exceed LLaMA's token limit.

Why?
LLMs have finite context windows. If we overfeed, inference fails or slows drastically.
By capping and merging, we feed just enough information for the question at hand.

Step 7: Answer Generation with Ollama

This is where all the preparation pays off. After retrieving and merging the most relevant chunks, the final question + context is passed to Ollama, which serves as the local runtime for LLaMA 3 models.

How it works

- We build a structured prompt that explicitly separates the context, question, and expected answer format.
- The prompt is then sent to Ollama through a subprocess call.
- Ollama runs a local instance of LLaMA 3 (your choice: 8B or 70B parameters depending on hardware).
- The model processes the merged context and generates a descriptive, grounded answer.

Why Ollama?

- **Local-first** → No need for cloud APIs; all inference runs on your own machine.
- Supports multiple models → You can easily switch from llama3 to llama3:70b by changing a single command.
- **Optimized performance** → Ollama is designed to handle large models efficiently on consumer hardware, supporting CPU and GPU acceleration.

- **Privacy** → Your documents never leave your machine. This is crucial for sensitive or proprietary PDFs.

- Simple interface → Just one CLI command (ollama run llama3 "your prompt") makes experimentation easy.

The answer generated will be context-aware (grounded in the retrieved chunks) and avoids hallucinating unrelated information.

Benefits of using Ollama here

- Offloads all the heavy lifting to a **local LLM runtime**.

- Keeps the pipeline modular: retrieval and preprocessing are handled by Python, while Ollama handles generation.

- You can swap models (e.g., LLaMA 3, Mistral, Gemma) without changing the pipeline logic.

- Supports streaming output, which means answers can appear token-by-token instead of waiting for the full response.

OUTPUT

Question:

How has Apple's total net sales changed over time?

```
--- Generated Answer ---
Based on the context, Apple's total net sales have increased over time. According to the data provided, the
 company's total net sales were:

* $81,434 in the third quarter of 2021
* $82,959 in the third quarter of 2022 (an increase of 2% year-over-year)

For the first nine months of the year, Apple's total net sales also increased:

* $282,457 in the first nine months of 2021
* $304,182 in the first nine months of 2022 (an increase of 8% year-over-year)

Overall, Apple's total net sales have shown a steady growth trend over time.
```

T5 Transformer Integration Approach

I also implemented the same RAG pipeline using the T5 transformer. T5 is capable of generating short, concise answers to questions, making it very effective for quick retrieval and summarization tasks.

Demerit:

- T5 struggles with producing long, descriptive answers. While it excels at short responses, it may miss finer details and nuanced explanations compared to LLaMA + Ollama.