Retrieval-Augmented Generation (RAG) Template 🚀

This repo provides a **plug-and-play RAG pipeline** in Python. Use it to index your documents and query them with an LLM (OpenAI, Gemini, HuggingFace, etc.).

Files Included

- `rag_template.py` → The main RAG script.
- `sample_docs.txt` → Example text file for testing.

Setup Instructions

1. Clone this repo

```
git clone https://github.com/yourusername/rag-template.git
cd rag-template
```

2. Create a virtual environment (recommended)

```
python -m venv venv
source venv/bin/activate   # On Mac/Linux
venv\Scripts\activate      # On Windows
```

3. Install dependencies

```
pip install -r requirements.txt
```

Minimal requirements (`requirements.txt`):

```
langchain
langchain-community
langchain-openai
faiss-cpu
```

4. Set your API Key

Export your API key for security:

```
export OPENAI_API_KEY="your_api_key_here"   # Mac/Linux
setx OPENAI_API_KEY "your_api_key_here"     # Windows
```

Or edit the script to place your key directly.

5. Run the RAG pipeline

```
python rag_template.py
```

How it Works

1. **Load Documents** → from `.txt` files (can swap for PDFs, CSVs, or DBs).
2. **Split Text** → into manageable chunks.
3. **Embed Chunks** → using OpenAI embeddings.
4. **Store Vectors** → inside FAISS (local vector database).
5. **Ask Questions** → queries are matched against your docs.
6. **LLM Response** → retrieved chunks are passed to the LLM for context.

Customization

- Change `TextLoader` to `PyPDFLoader`, `CSVLoader`, etc.
- Swap embeddings: `OpenAIEmbeddings` → `HuggingFaceEmbeddings`.
- Replace FAISS with other vector DBs (Pinecone, Chroma, Weaviate).
- Use another LLM: `ChatOpenAI` → Gemini / HuggingFace / LLaMA.

Example Query

```
YOU: What is this document about?
RAG: This document explains how to set up a Retrieval-Augmented Generation
pipeline.
```

Contributing

Feel free to fork, modify, and use this template for your own projects.

🚄License

MIT License. Free to use and modify.