RAG-Based Document Summarizer

This project is a simple but powerful pipeline for document summarization using **Retrieval-Augmented Generation (RAG)**. It accepts long PDF documents, breaks them into meaningful chunks, retrieves the most relevant parts, and generates a coherent summary using a pre-trained large language model.

Setup Instructions

1. Clone the Repo or Download Files

Place your desired PDF in the same folder as `main.py`. Example filename: `sample.pdf`

2. Install Requirements

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

for faster HuggingFace model downloads:

```
pip install huggingface_hub[hf_xet]
```

How It Works

1. **Document Ingestion**
    - Loads a PDF using PyMuPDF
    - Chunks the document using a sliding window
2. **Embedding & Storage**
    - Converts text chunks into vector embeddings using `SentenceTransformers`
    - Stores them in a FAISS index for fast semantic search
3. **Retrieval & Summarization**
    - Retrieves top-k chunks relevant to the query: “Summarize this document”
    - Passes these chunks to `facebook/bart-large-cnn` to generate a summary
4. **Output**
    - Prints a clean summary in the terminal

Example Usage

```
python main.py
```

Output:

```
 SUMMARy:
Artificial Intelligence (AI) is transforming industries with its capabilities
in NLP, vision, and decision-making...
```

Files

- `main.py` – core logic
- `sample.pdf` – test document
- `requirements.txt` – dependencies
- `README.md` – setup & overview
- `sample_output.txt` – sample summary
- `report.pdf` – brief explanation of the pipeline

Credits

- Hugging Face `transformers`
- `sentence-transformers`
- `faiss-cpu`
- PyMuPDF

REPORT.PDF

Document Summarization using RAG – Report

Objective

To build a summarization system that combines retrieval-based context selection with large language model generation. The system ingests long documents and returns short, accurate summaries.

Components

1. Document Ingestion

- Used PyMuPDF (`fitz`) to extract text
- Split into chunks of 500 words with 100-word overlap

2. Embedding + Vector DB

- Used `all-MiniLM-L6-v2` from `sentence-transformers`
- Stored chunk embeddings in FAISS index

3. Semantic Retrieval

- Used cosine distance to retrieve top-5 relevant chunks based on the query: *"Summarize this document"*

4. LLM Summarization

- Used `facebook/bart-large-cnn` via `transformers.pipeline`
- Limited input text to ~3000 characters for efficiency

5. Output

- Clean summary printed in terminal

Sample Result

Summary Output:

> Artificial Intelligence (AI) is transforming industries with its capabilities in natural language processing, computer vision, and decision-making. From healthcare diagnostics to financial forecasting and personalized education, AI applications are expanding rapidly. Ethical concerns remain, including data privacy, algorithmic bias, and the potential displacement of jobs.

Deliverables

- Python Code (`main.py`)
- PDF Test File (`sample.pdf`)
- `README.md`
- `requirements.txt`
- Sample Output File
- This Report

Conclusion

This project demonstrates a complete, functional pipeline for retrieval-augmented summarization of long documents using modern NLP tools.