

RAG-Based Summarization Project

This project is a simple RAG-based (Retrieval-Augmented Generation) application focused on document summarization. The main idea is to let users upload files in different formats like PDF, TXT, or Markdown, and get a clean summary of the document. I built the system in a way that it works completely offline, without needing any paid APIs or tokens.

How the App Works

The user interface is built using Streamlit, which allows anyone to use the app through a web browser without needing to touch any code. Once a file is uploaded, it's processed depending on the file type.

- For **PDFs**, I used the PyPDF2 library to extract text from each page.
- For **TXT** and **.md** files, the content is simply read as plain text.

After this, the extracted text is cleaned up and prepared for the next steps.

Splitting the Text

Since large documents can't be summarized in one go (because of model token limits), I split the text into smaller parts. Instead of doing random or fixed-size splits, I used a smarter method called RecursiveCharacterTextSplitter from LangChain. It breaks the text based on structure like paragraphs or sentences, and also keeps some overlap between chunks. This makes sure the chunks are still meaningful and don't lose context.

Each chunk is around 2048 characters long with some overlap to help keep the flow between segments. The chunk size is larger than average 512 because for summarization longer context works better.

Embedding and Search

Once the text is chunked, each chunk is turned into a vector using a SentenceTransformer model called all-MiniLM-L6-v2. This model converts each chunk into a 384-dimensional vector. These vectors can be compared with each other to find similarities, which is helpful for tasks like semantic search or clustering.

I'm using **FAISS (Facebook AI Similarity Search)** to store these vectors and search through them. It runs on CPU and is fast enough for small to medium-sized documents.

Retrieving Relevant Chunks

When the user gives a query (for example, "Summarize the general policy of internet usage from the document"), it is also converted into a vector using the same embedding model. Then FAISS is used to find the top 5 chunks that are most similar to the query. These chunks are what we call the "retrieved context."

Summarization with a Local LLM

For the actual summarization, I needed a language model that can generate text. Instead of using cloud APIs (which need API keys and credits), I went for an open-source option that I could run on my machine.

I chose **LLaMA 3.2 Instruct (1B version)**. The 3-billion parameter version is small enough to run on CPU, but still good at understanding and summarizing text. I pass the retrieved chunks to this model along with a prompt like:

```
prompt = (
    f"User query: {query}\n"
    f"Relevant text:\n{combined_text}\n\n"
    "Summarize this for the user." )
```

```
messages = [
    {"role": "system", "content": "You are a helpful assistant who summarizes the information for the user."},
    {"role": "user", "content": prompt},
]
```


Why Local Setup

I decided to keep everything offline because calling APIs costs money and also needs a stable internet. Most API-based models require a key and have token limits that become expensive for long documents. By choosing open-source models for both embeddings and generation, the app becomes easier to test, run, and experiment with, especially on local machines with limited resources. The requirements file is also in the folder to easily download all the required packages. This setup is useful for students, researchers, or anyone who wants to try out RAG without relying on commercial services.


Deploy

Document Summarizer

Upload a PDF, TXT or Markdown file and get a summary using RAG-style retrieval with LLAMA.

 Drag and drop file here
Limit 200MB per file • PDF, TXT, MD

Browse files

 ptcl_general_privacy_policy.txt 15.9KB ×

Document loaded and split into 12 chunks.

Enter your query (or general summary request):

Summarize this document

Generate Summary

Final Summary

This document outlines the terms and conditions for using services and products offered by PTCL Group, a telecommunications company. Here's a summary:

What's Included:

- Information about transactions, usage, and subscription details
- Response data related to your requests for service or assistance
- Any other information provided to PTCL Group, authorized partners, and third parties

How PTCL Group Collects and Stores Your Information: