**📌 Summary: What We Did So Far in Our RAG-Based Research Paper Analyzer**

We built a **Research Paper Analyzer** using **RAG (Retrieval-Augmented Generation)** to extract, store, and query academic content efficiently. Here’s a **detailed breakdown** of what we did, the tools we used, and how RAG was implemented.

**🚀 What We Built**

✅ **A system to extract and retrieve relevant information from research papers** using **FAISS + Gemini API (and later considering GPT-4/Claude)** for answering queries.  
✅ **Improved text extraction, chunking, vector storage, and retrieval** to ensure better results.  
✅ **Added Google Search & Custom Database as options for hybrid retrieval.**

**🔧 What Tools & Technologies We Used**

| **Component** | **Technology Used** | **Why We Used It?** |
| --- | --- | --- |
| **PDF Text Extraction** | pdfplumber, PyMuPDF (fitz), OCR (Tesseract) | Extracted text from PDFs reliably |
| **Text Chunking** | NLTK, RecursiveCharacterTextSplitter | Broke text into **meaningful, retrievable chunks** |
| **Vector Embeddings** | SentenceTransformers (MiniLM, E5-Large) | Converted text into **vector embeddings** |
| **Vector Database (Storage & Search)** | FAISS (Facebook AI Similarity Search) | Efficiently stored & retrieved **top-matching research chunks** |
| **LLM for Answering Queries** | Gemini API (Google AI) *(now considering GPT-4/Claude)* | Used an LLM to generate answers **based on retrieved chunks** |
| **Hybrid Retrieval** | Google Search API + Custom DB (planned) | Enabled external search alongside research papers |

**🔍 How We Implemented RAG (Retrieval-Augmented Generation)**

**1️⃣ Data Preparation & Extraction**

* Extracted text from **research papers (PDFs)** using **multiple extraction methods**:
  + **pdfplumber** → Best for structured PDFs.
  + **PyMuPDF (fitz)** → Extracted text while preserving layout.
  + **OCR (Tesseract)** → Used for scanned images inside PDFs.
* Cleaned extracted text **by removing headers, footers, and duplicate lines**.

✅ **Why?**  
We needed **clean, structured text** for better retrieval accuracy.

**2️⃣ Text Chunking & Preprocessing**

* Split extracted text into **manageable, contextually relevant chunks**:
  + **NLTK Sentence Tokenizer** → Split into sentences.
  + **RecursiveCharacterTextSplitter** → Created **overlapping chunks** (to preserve context).
  + **Used section-aware chunking** (Abstract, Methods, Results, etc.).

✅ **Why?**  
Chunks ensure **efficient retrieval** and **prevent missing critical context.**

**3️⃣ Converting Text into Vector Embeddings**

* Used **Sentence Transformers** to convert text chunks into vector embeddings:
  + **Initially used all-MiniLM-L6-v2 (Fast but small model)**
  + **Later switched to intfloat/multilingual-e5-large** for better embedding quality.

✅ **Why?**  
LLMs like Gemini, GPT-4, or Claude **cannot search PDFs directly** → Instead, we used **semantic search** via embeddings.

**4️⃣ Storing Embeddings in FAISS (Vector Database)**

* **Stored research paper chunks in FAISS (Facebook AI Similarity Search)**.
* Indexed **111 chunks** from **multiple research papers**.
* Used **L2 similarity search** to find the **most relevant** chunks for any user query.

✅ **Why?**  
FAISS allows **fast, scalable similarity search** over large datasets.

**5️⃣ Querying FAISS to Retrieve Relevant Chunks**

* When a user asks a question:
  + Converted **the question into an embedding** (same model as chunking).
  + **Searched FAISS** to get the **top 10 most relevant chunks**.
  + **Filtered results** to remove noise and irrelevant content.

✅ **Why?**  
This ensures that **only relevant research paper data** is passed to the LLM.

**6️⃣ Passing Retrieved Chunks to LLM (Gemini/GPT-4)**

* **Initially used Gemini API** to generate answers from retrieved chunks.
* **Problem:** Gemini **ignored retrieved content** and said "No relevant information found."
* **Fix Attempted:**
  + **Summarized retrieved chunks first** before passing to Gemini.
  + **Cleaned formatting (removed citations, extra spaces, etc.).**
  + **Forced Gemini to use only retrieved data.**
* **Still an issue?** → Considering **switching to GPT-4-Turbo or Claude 3**.

✅ **Why?**  
LLMs **refine and contextualize search results** to provide **human-like answers.**

**7️⃣ Adding Hybrid Retrieval (Google Search + Custom DB)**

* **Planned feature**:
  + **Google Search API** → Fetch latest research articles for better context.
  + **Custom DB** → Store **past user queries** + **additional datasets**.

✅ **Why?**  
Combining **RAG + external web search** improves answers beyond our stored PDFs.

**📌 Current Status:**

✔ **Text extraction, chunking, and FAISS retrieval are working well.**  
❌ **LLM response generation (Gemini) is inconsistent.**  
✔ **Considering switching to GPT-4 or Claude for better results.**

**📌 Next Steps**

1️⃣ **Switch to GPT-4-Turbo or Claude 3** for better handling of research-based queries.  
2️⃣ **Optimize FAISS retrieval further** (adjust embedding model, fine-tune chunking).  
3️⃣ **Implement Google Search & Custom DB hybrid retrieval.**

🚀 **Do you want me to switch your query\_faiss.py to use GPT-4-Turbo or Claude 3 now?** 🔥