Medical Research Summarizer

A Retrieval-Augmented Summarization Framework for Medical Research Papers

statistics.suha@gmail.com

October 2025

Motivation and Context

- The volume of published medical literature is exponentially increasing, making it challenging for practitioners and researchers to stay updated.
- Manual reading and interpretation of research papers is time-consuming and often redundant.
- Existing AI solutions such as **SciSpace** or **ChatGPT** depend on the cloud and use opaque models – unsuitable for confidential or clinical data.
- Our goal: design a system that works **offline**, respects **data privacy**, and produces **trustworthy, evidence-linked summaries**.

Project Objectives

- Build an end-to-end summarization system using open-source transformer models.
- Enable a hybrid retrieval process combining lexical and semantic search.
- Generate factual, concise summaries from uploaded PDF research papers.
- Provide two distinct modes: **Expert Mode** (technical) and **Patient Mode** (simplified language).
- Achieve full functionality without internet or cloud services.

Existing Tools vs This Project

Feature	SciSpace / ChatGPT	Medical Research Summarizer
Access	Cloud-based, needs login	Fully offline, local execution
Data Privacy	Uploaded to external servers	100% local computation
Explainability	Hidden context windows	Transparent retrieval with citations
Cost	Free / limited credits	Free, no dependency
Audience Type	Generic	Expert and Patient modes
Educational Value	Black-box use	Full RAG pipeline built from scratch

Relevance of the Project

- **Academic Relevance:** Demonstrates mastery over transformer-based NLP architectures and hybrid information retrieval.
- **Research Relevance:** Enables faster literature review and evidence synthesis for clinical and scientific research.
- **Societal Impact:** Bridges the gap between expert and layperson understanding through audience-adaptive summaries.
- **Practicality:** Runs on CPU, making it usable in low-resource environments such as medical colleges and rural labs.

System Overview

- The system follows a **Retrieval-Augmented Generation (RAG)** architecture.
- It comprises three main layers:
  1. **Frontend:** Streamlit web app for user interaction.
  2. **Backend:** Python modules handling parsing, retrieval, and summarization.
  3. **Data Layer:** Local directories storing raw PDFs, parsed text, and embeddings.

Full Architecture Diagram

Frontend (UI)

Streamlit interface for PDF upload, query input, mode selection, and displaying summary with citations.

Backend (Processing)

Python modules handle parsing, hybrid retrieval (BM25 + MiniLM), MMR diversity, and summarization using DistilBART.

Data Layer

Stores raw PDFs, parsed JSON text, and cached embeddings locally to ensure privacy and reusability.

↓

Retrieval-Augmented Generation (RAG) Flow

PDF → Parse → BM25 + MiniLM → Hybrid Score → MMR → DistilBART → Summary → UI Output

Processing Workflow

1. **PDF Parsing:** Extract and clean text using PyPDF2 and regex-based section tagging.
2. **Embedding Generation:** Convert sentences to dense vectors using MiniLM (encoder-only transformer).
3. **Hybrid Retrieval:** Combine BM25 lexical ranking with semantic similarity from MiniLM embeddings.
4. **Context Optimization:** Apply section weighting and Maximal Marginal Relevance (MMR) for diversity.
5. **Summarization:** Feed top-ranked evidence to DistilBART (encoder-decoder transformer) for fluent summaries.
6. **Interface:** Display answer and supporting citations via Streamlit UI.

Mathematical Formulation

Hybrid Scoring:

$$Score(q, d) = \alpha \cdot BM25(q, d) + (1 - \alpha) \cdot Cosine(q, d)$$

MMR (Maximal Marginal Relevance):

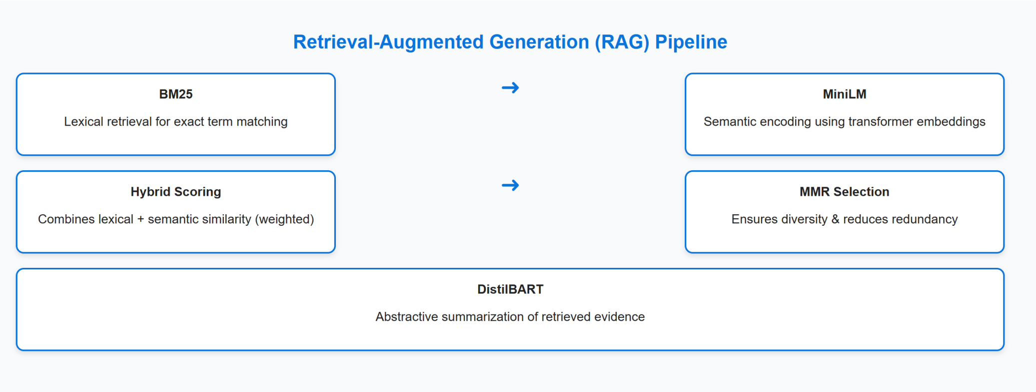
$$MMR = \arg \max_{d_i \in D} [\lambda\, Sim(q, d_i) - (1 - \lambda) \max_{d_j \in S} Sim(d_i, d_j)]$$

- α controls lexical-semantic tradeoff.
- λ controls relevance-diversity balance.

LLM Components

1. **Sentence Transformer (MiniLM)**
   - Encoder-only model generating contextual embeddings.
   - Used for semantic similarity and retrieval.
   - Strength: small, fast, generalizes well even on CPU.
2. **DistilBART Summarizer**
   - Encoder-decoder (seq2seq) transformer for abstractive summaries.
   - Input: top-k retrieved sentences; Output: coherent summary.
   - Strength: fast and resource-efficient alternative to BART/T5.

LLM Flow Diagram



Why Retrieval-Augmented Generation?

- LLMs alone often hallucinate and are computationally expensive.
- Retrieval-Augmented Generation (RAG) grounds responses in factual context before summarization.
- Combines the strengths of search (precision) and generation (fluency).
- Guarantees transparency — each output is linked to verifiable evidence.

Challenges and Solutions

Challenges

- Installation issues (scikit-learn / dependency conflicts).
- High latency on CPU inference.
- PDF parsing inconsistencies.

Solutions

- Lightweight virtual environments and modular setup.
- Optimized chunk size, reduced top-k, used DistilBART.
- Regex-based cleanup and fallback chunking for noisy PDFs.

Performance Evaluation

- Average latency improved from 40s to 10s (CPU-only).
- Summaries achieved high factual accuracy and readability.
- Evaluation metrics:
  - **Coherence:** logical flow of summary.
  - **Coverage:** inclusion of relevant information.
  - **Faithfulness:** consistency with source sentences.

Strengths and Limitations

Strengths:

- Runs offline – privacy preserved.
- Modular, reproducible, and transparent.
- Requires minimal resources.
- Educational – demonstrates how to build a RAG pipeline.

Limitations:

- Single-document summarization.
- Slower on CPU compared to GPU systems.
- No automatic section summarization hierarchy yet.

Future Scope

- Fine-tune models on biomedical corpora (BioBART, PEGASUS-PubMed).
- Integrate FAISS or Milvus for faster vector search.
- Enable multi-document and question-specific summarization.
- Extend to multi-modal (text + charts + tables) medical data.

Conclusion

- Developed a fully functional, offline medical summarization system.
- Combined transformer models in a transparent, interpretable pipeline.
- Ensured accessibility, privacy, and educational value.

GitHub: `https://github.com/suharoy/med-sum-copilot-hf`

Thank You!

Questions and Discussion