Medical Q&A Project Documentation

Project Overview:  
This project builds a medical question-answering system using PubMed abstracts. It combines retrieval-based search with generative summarization, providing readable, accurate answers to user queries.

Key idea:

* Retrieve relevant abstracts using BioBERT embeddings + FAISS.
* Generate concise answers using Flan-T5 (Generative QA).

Project Goals:

1. Collect PubMed abstracts for medical search terms.
2. Build semantic embeddings using BioBERT.
3. Create FAISS index for fast retrieval.
4. Implement Hybrid RAG QA: Retrieve top-k abstracts → Generate answer with Flan-T5.
5. Deploy interactive Gradio web interface.

Technology Stack:

* Python 3.10+
* Libraries: transformers, sentence-transformers, faiss, numpy, pickle, os, gradio
* Models: dmis-lab/biobert-base-cased-v1.1 (embeddings), google/flan-t5-base (QA)
* Environment: Laptop CPU/GPU or Google Colab GPU
* Deployment: Gradio web app

Project Structure:

* 01\_fetch\_pubmed\_abstracts.ipynb: Fetch abstracts and save as all\_pubmed\_abstracts.pkl
* 02\_embeddings.ipynb: Preprocess abstracts and compute embeddings
* 03\_build\_embeddings\_faiss.ipynb: Build FAISS index and save
* 04\_query\_faiss.ipynb: Test FAISS retrieval
* 05\_model\_integration.ipynb: Hybrid QA pipeline and Gradio deployment

Methodology:  
Step 1: Fetch PubMed Abstracts

* Define search terms (diseases, symptoms, drugs, lab tests, procedures)
* Fetch abstracts via PubMed API
* Save as all\_pubmed\_abstracts.pkl

Step 2: Preprocess and Compute Embeddings

* Clean abstracts
* Flatten nested dictionary to list of texts
* Load BioBERT model
* Compute embeddings in batches
* Save embeddings (all\_pubmed\_embeddings.pkl, embeddings.npy, texts.npy)

Step 3: Build FAISS Index

* Initialize FAISS IndexFlatL2 (dim=768)
* Add embeddings
* Save index pubmed\_faiss.index

Step 4: Test Retrieval

* Embed query using same BioBERT
* Retrieve top-k abstracts
* Inspect for quality control

Step 5: Hybrid QA Integration

* Input query → Embed → Retrieve top-k → Generate answer via Flan-T5 → Output readable answer
* Deploy using Gradio

Example Gradio snippet:

import gradio as gr

def hybrid\_medical\_qa(query):

retrieved\_texts = retrieve(query)

answer = flan\_t5\_generate(query, retrieved\_texts)

return answer

gr.Interface(fn=hybrid\_medical\_qa, inputs="text", outputs="text").launch(share=True)

Evaluation:

* Test questions: type 2 diabetes treatment, vulvar pruritus, unanswerable, adverse effects, etc.
* Observed: Some questions perfectly answered, some partially.
* Adding more abstracts improves recall for rare topics.

Potential Pitfalls:

1. Large corpus (>20k abstracts) → Use Colab GPU
2. Memory usage → FAISS + embeddings require GBs of RAM
3. Generative hallucination → Ensure grounding in retrieved abstracts
4. Consistency → Embed queries with same model as abstracts
5. Corpus update → Adding more terms requires rerunning Notebooks 01–03

Future Improvements:

* Expand search terms and abstracts
* Use MedAlpaca or ChatGPT API
* Fine-tune Flan-T5 on PubMed abstracts
* Implement answer ranking & confidence scoring
* Cloud-based Gradio deployment with persistent embeddings

Project Sharing:

* GitHub: Include notebooks, requirements.txt, README.md; large files stored externally
* Gradio Deployment: Colab demo (share=True) or Hugging Face Spaces

References:

* BioBERT: <https://huggingface.co/dmis-lab/biobert-base-cased-v1.1>
* Flan-T5: <https://huggingface.co/google/flan-t5-base>
* FAISS: <https://github.com/facebookresearch/faiss>
* PubMed API: <https://www.ncbi.nlm.nih.gov/home/develop/api/>