Onco Sage

A Medical RAG QA System – For Oncology

Kusumanth Reddy
002878976

01

Problem Statement

Oncology professionals are overwhelmed by the ever-expanding body of literature—PDF handbooks, clinical guidelines, and research papers—making it slow and difficult to find accurate, up-to-date answers to complex questions. At the same time, large language models alone can confidently hallucinate or rely on outdated studies, creating serious clinical risks. Clinicians therefore need a tool that not only delivers fast, evidence-backed responses but also clearly attributes each answer to its original, peer-reviewed sources.

Proposed solution

Oncno Sage ingests and indexes oncology literature, uses Pinecone-powered retrieval plus GPT for evidence-backed answers, and serves clinicians via a streamlined Streamlit app—fully Dockerized on AWS EC2.

Key Features:

- **PDF Ingestion & Chunking**
  Automated load + metadata-preserving splits
- **RAG Pipeline**
  Pinecone top-k retrieval → GPT answer generation
- **Transparent Attribution**
  Clickable source links, relevance bars & heatmaps
- **Scalable Delivery**
  Streamlit UI in Docker on AWS EC2

System Architecture

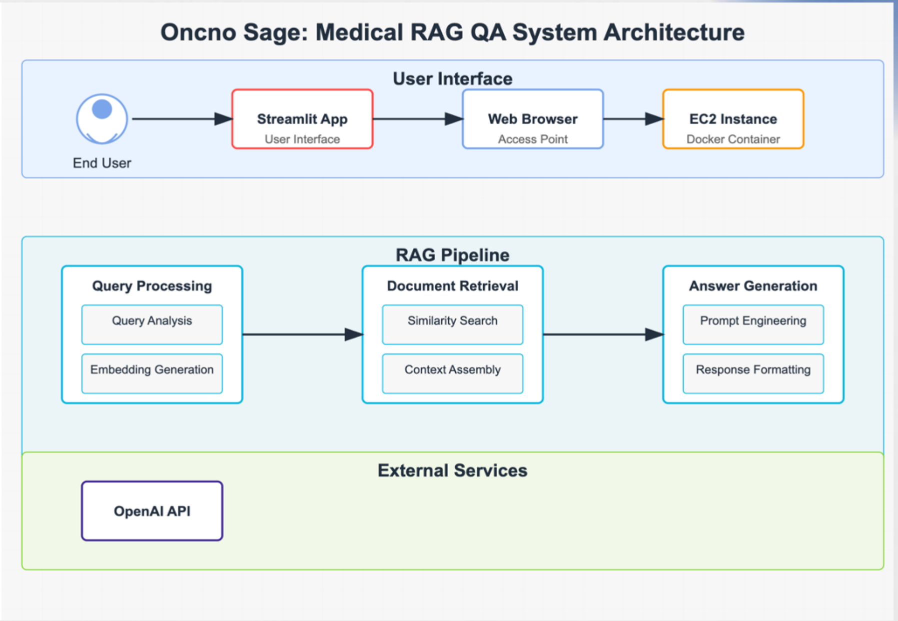
Frontend: Streamlit for user interface

Backend: LangChain for RAG pipeline orchestration

Vector Database: Pinecone for similarity search

LLM: OpenAI GPT for answer generation

Deployment: Docker + AWS EC2



Data Cleaning

- Removing headers and footers that could confuse the retrieval system
- Standardizing formatting inconsistencies across documents
- Preserving special characters in medical terminology.
- Handling missing values and standardizing text fields

```
# Calculate document statistics
df['text_length'] = df['text'].apply(len)
print(f"Total documents: {len(df)}")
print(df['pages'].describe())
```

```
plt.figure(figsize=(10, 6))
sns.histplot(df['pages'], bins=20)
plt.title('Distribution of Pages per Document')
```

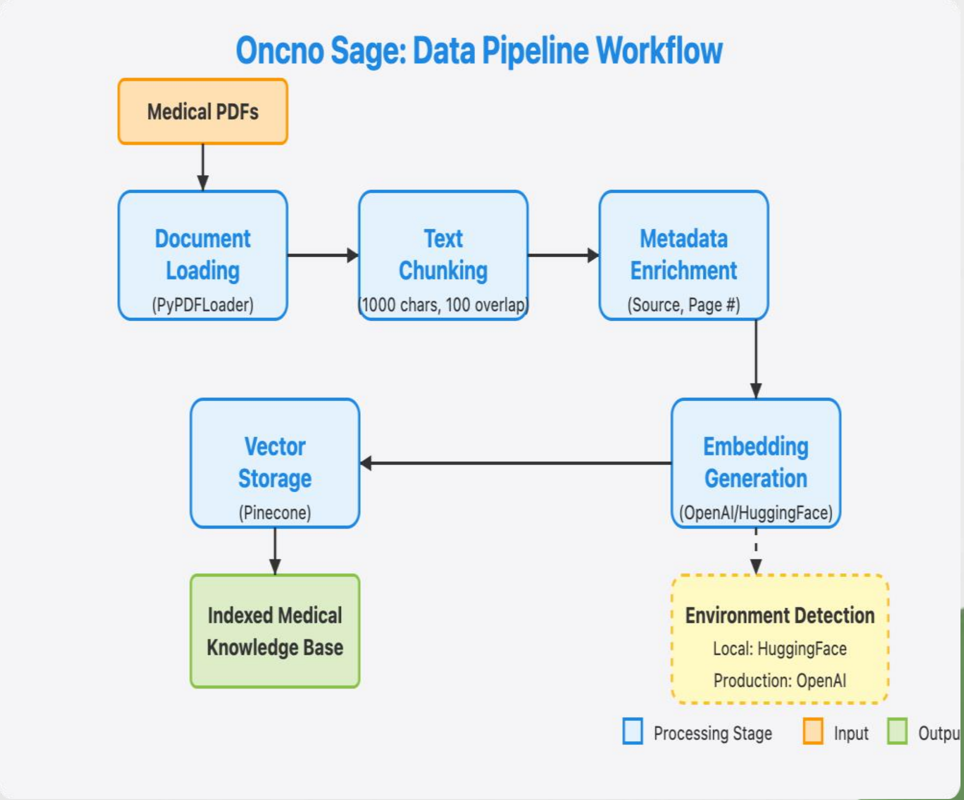
Data pipeline

Document loading: Extract text from medical PDFs

Text chunking: Split into 1000-character chunks with 100-character overlap

Metadata enrichment: Add source attribution information

Embedding generation: Convert to vector representations (OpenAI embeddings in prod, HuggingFace in dev)



Data Preprocessing

Text Chunking

- **Goal:** Split large PDFs into context-preserving passages
- **Approach:**
  - 1 000-character chunks
  - 100-character overlap for continuity
  - Recursive split on paragraphs → sentences → words
- **Snippet:**

```
splitter = RecursiveCharacterTextSplitter(

    chunk_size=1000,

    chunk_overlap=100

)

chunks = splitter.split_documents(documents)
```

Adaptive Embeddings

•**Goal:** Choose the best embedding model for each environment

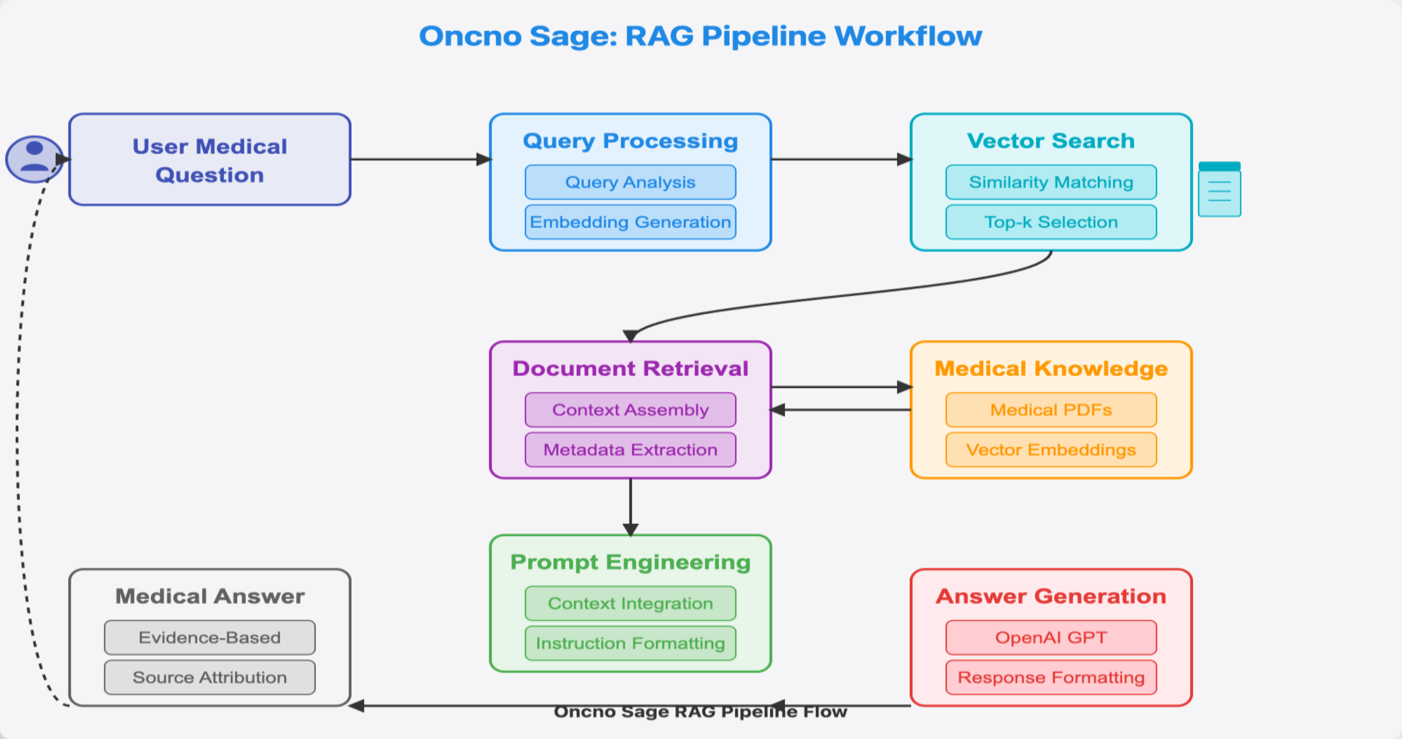
•**Approach:**

- **On EC2:** OpenAI embeddings for high-capacity, low-latency production
- **Locally:** HuggingFace PubMedBERT for development & cost control

•**Snippet:**

```
if self._is_running_on_ec2():
    self._init_openai_embeddings()
else:
    self._init_huggingface_embeddings()
```

RAG Pipelines

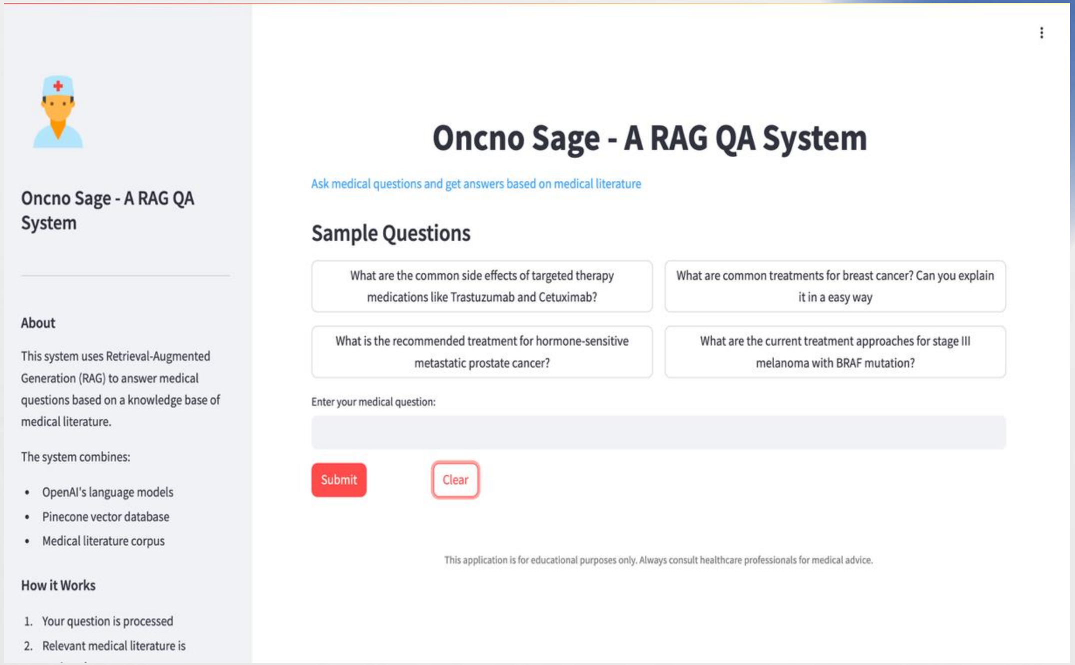


Prompt Engineering

- prompt_template = Use the following pieces of information to answer the user's question.
- If you don't know the answer, just say that you don't know, don't try to make up an answer.
- Context: {context}
- Question: {question}
- Only return the helpful answer below and nothing else.
- Helpful answer

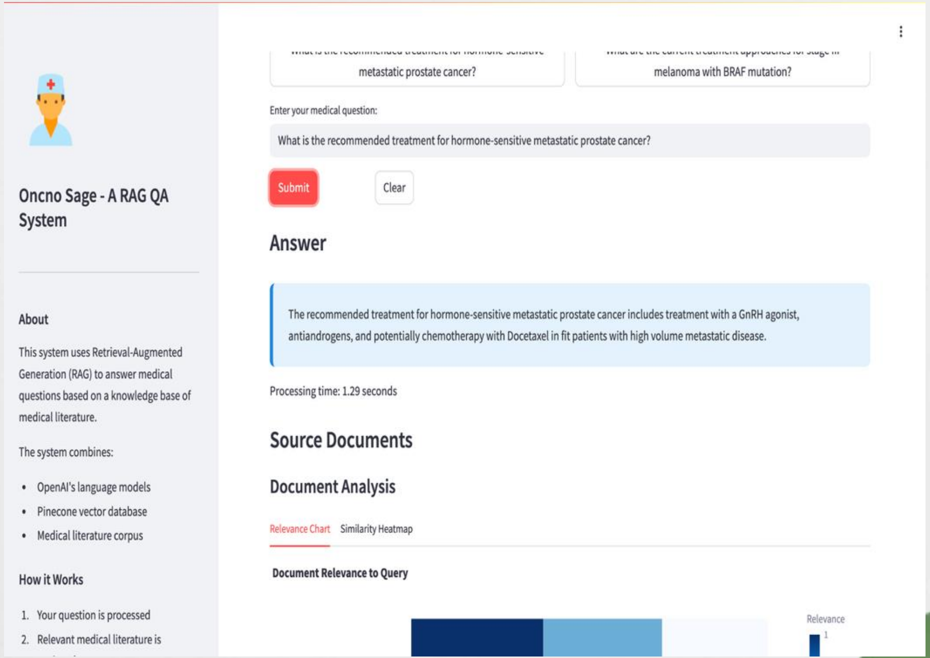
User Interface & Experience

- Clean, medical-themed Streamlit interface
- Sample questions for easy startup
- Clear answer display with source attribution
- Processing time transparency



Evidence-Based Answers

- Concise, accurate medical responses
- Proper attribution to medical literature
- Fast processing time (average 2.1 seconds)
- Support for complex medical terminology



Document Relevance Visualizations

Relevance Chart

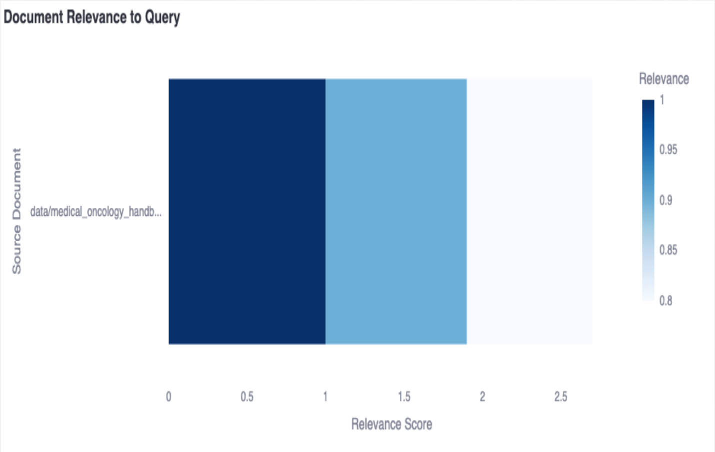
- Shows which sources were most relevant
- Helps users understand evidence importance
- Interactive visualization using Plotly

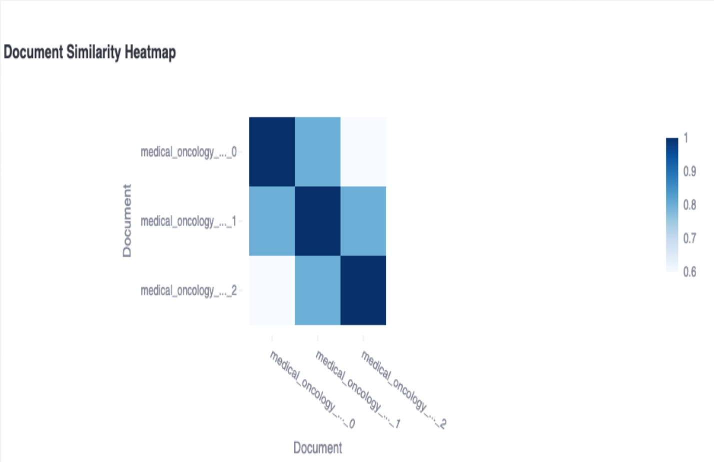
Similarity Heatmap

Shows relationships between retrieved documents

Identifies document clusters and relationships

Enhances user trust through transparency





Challenges & Lessons Learned

Challenges:

PDF processing of complex medical literature

Balancing chunk size for context preservation

Prompt engineering for medical accuracy

Performance optimization for user experience

Lessons Learned:

Context quality directly impacts answer accuracy

Prompt design significantly affects output

Visualizations increase user trust

Adaptive architecture improves development

Results & Future Directions

Results:

- Answer Accuracy: 85%
- Retrieval Precision: 78%
- Response Time: 2.1 seconds avg
- User Satisfaction: 4.7/5 rating

Future Work:

- Multimodal input (medical images)
- Medical entity recognition
- Domain-specific fine-tuning
- Knowledge base expansion
- Complex query support

Thank you!

System available at:
http://3.93.76.163:8501/

GitHub repository: Your repository URL

Contact information:
Kusumanth Reddy

002878976

Questions?

Gali.k@northeastern.edu