DocLens AI - Internship - 2025

Detail	Information
Company Name	DocLens.ai
Role/Position	Software Engineering Intern
Duration	12th May 2025 - 12th July 2025
Location	Bangalore, On-Site
Mentor/Supervisor	Amit Saha, Shivam Choudhary, Ramdoot Pydipaty
Certificate	https://drive.google.com/file/d/1PSE1GiS8FeUo1jcmlkYSVfmL_rRZBrDx/view?usp=sharing
Tech Stack	LangChain, LLM(Gemini, Claude, Mistral, Llama, Ollama), spaCy, NLTK, Vector Databases(Qdrant, Milvus, Chroma), Knowledge Graph(neo4j), DeepEval, RAGAs, Retrieval Augmented, Generation(RAG)
Company Website	https://doclens.ai/
Final Report	https://drive.google.com/file/d/1uwzkTLiZ2rvwJ4i2gU5LEGkJwp4-clWp/view?usp=sharing
Presentation	https://drive.google.com/file/d/1nbPWdKjyUf8w2-e3yGTW1CYTkYdn5pXX/view?usp=sharing

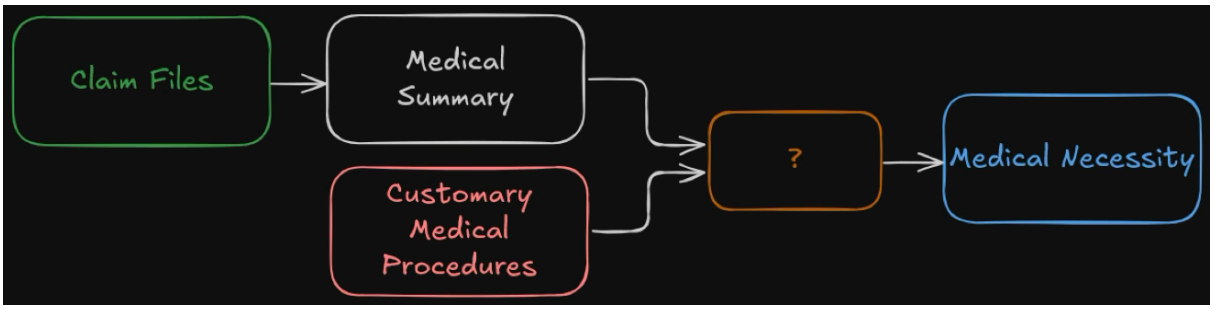
Projects

1. Creation of a '**Medical Necessity Evaluation**' report
    a. MayoClinic Web Scraper
    b. Embedded Hybrid Vector Search

c. RAG Pipeline

2. Context based **Question Recommendation System** for the chat interface
    a. Document based Knowledge Graph creation
    b. RAGAs(KG + LLM) QA Generation
    c. DeepEval(LLM) QA Generation
    d. Topic Modelling QA Generation

Project #1 - Medical Necessity Evaluation



Problem Statement

The current LLM-based approach to generating Medical Necessity reports lacks reliability due to hallucinations, outdated information, and hard to verify outputs.

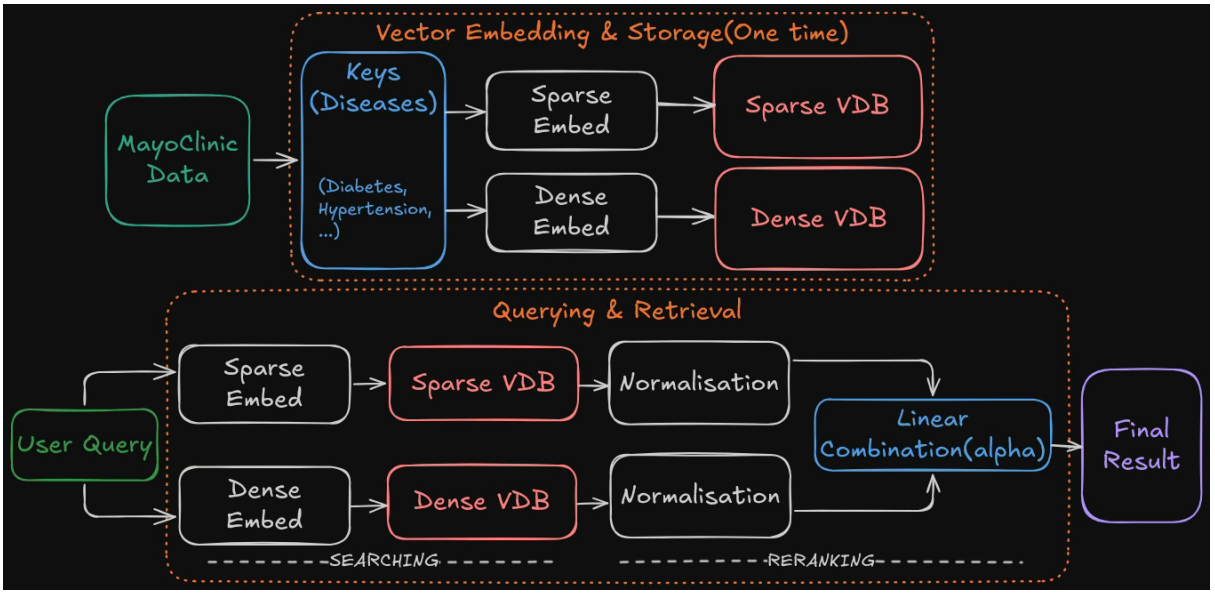
Hypothesis

Providing relevant scraped ground truth information (MayoClinic) as context to the LLM, can significantly improve response accuracy and reduce hallucinations.

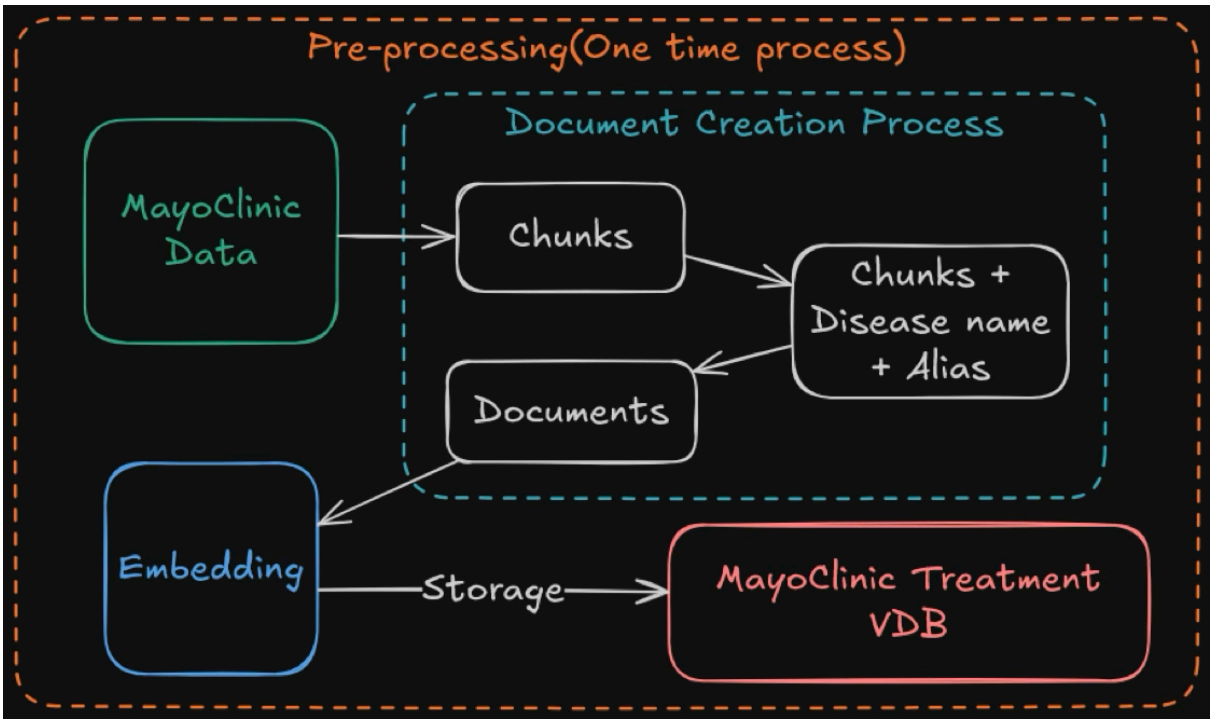
Ideas

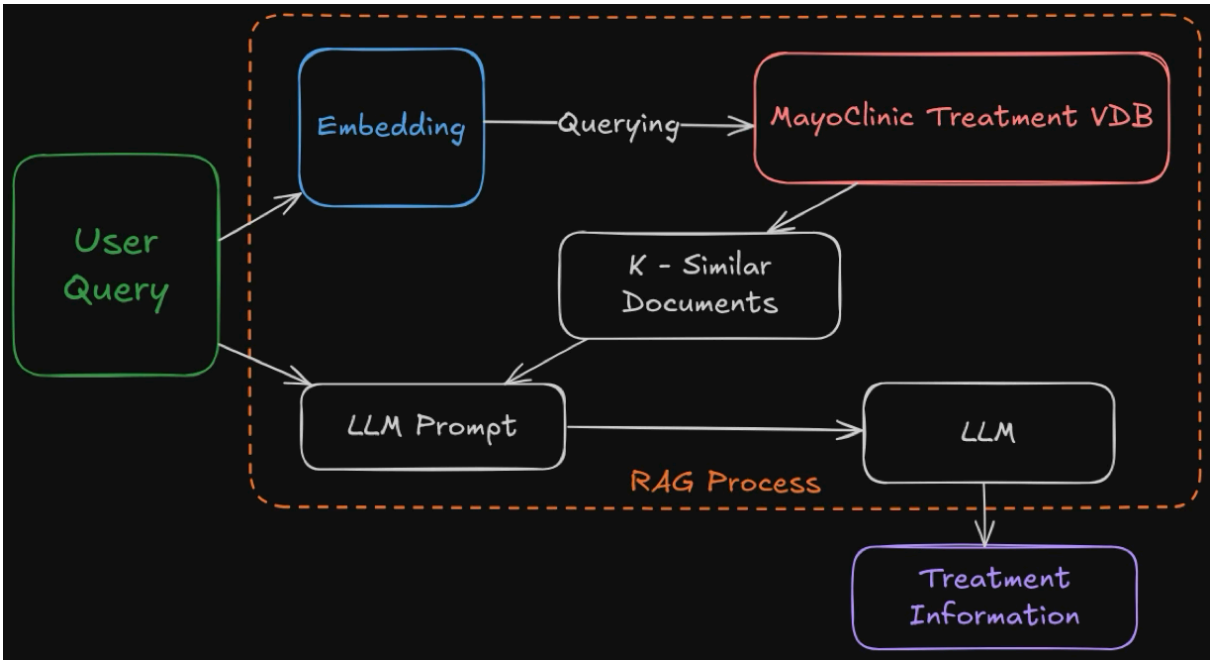
1. LLM - Knowledge Base(Current)
2. LLM + Tools(Internet Access)
3. LLM + Local database of MayoClinic's publicly available data
4. LLM + database passed as System Prompt

Approach #1 - Embedded Vector Search - Semantic + Lexical Similarity



Approach #2 - RAG Pipeline





Results

Approach #1 - Embedded Vector Search

- All obvious diseases and the ones present in the database get matched with little to no problems
- Too **generic**(Slip and Fall) or when they're too **niche**(non-infectious) and the dataset is lacking, it picks the closest, which we don't have a lot of control over
- **Keywords** like Syndrome, Abscess and **Acronyms** are a problem for the model to understand and match
- Highly dependent on the dataset

Approach #2 - RAG Pipeline

- Lack of exact treatment, generic solution to all problems, raises more flags than necessary.
- High inclusivity, low specificity
- Fetches all possible injuries and treatments with alternatives and implications
- Makes for a lacking database of medical information

Next Steps(Recommendations)

1. Trying different embedding models, better embeddings fine tuned for diseases
2. Scraping different websites, e.g. Cleaveland Clinic, better data ⇒ improved output
3. Running these on more diseases, to get a better idea of deficiencies
4. Creation of a Knowledge Graph of scraped medical data
5. Classification of queries, at runtime to choose approach of searching(Vector search, RAG)

Learnings & Implementation

- Vector Embeddings and Searching: Sparse + Dense Embeddings, Similarity Search methods, Hybrid Search method(self-implemented)
- Vector Databases: Qdrant, Milvus, Chroma
- Hugging Face Fine-tuned Models
- RAG & Re-Ranking of chunks
- AWS specific technology(via Bedrock): AWS Nova models, AWS Embeddings models

---

Project #2 - Question Recommendation System

Problem Statement

Process documents to generate and suggest questions in chat, relevant to the contexts of the documents

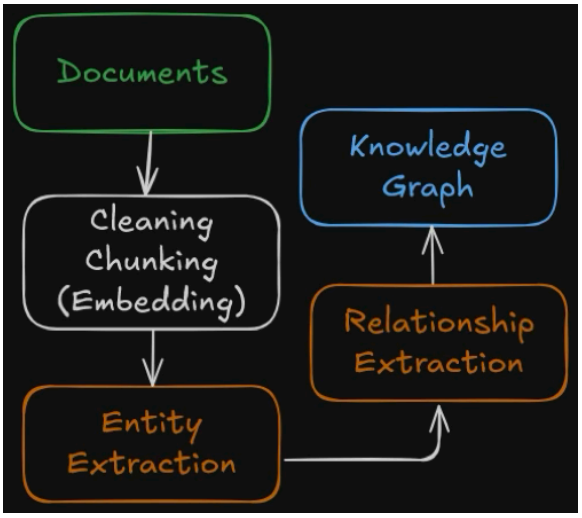
Hypothesis

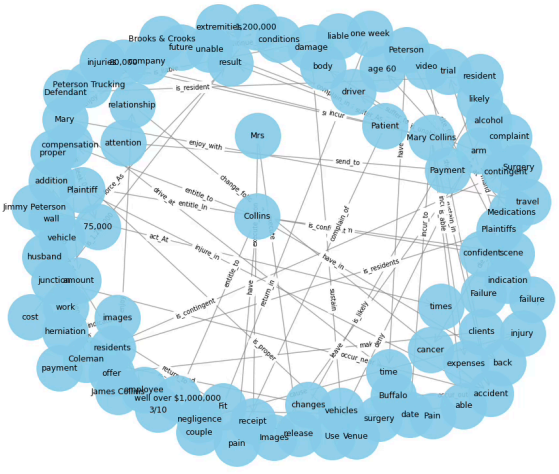
Using a knowledge graph to model relationships in the document would generate accurate and relevant questions

Ideas

1. Knowledge Graph based modelling + QA Generation
2. LLM based QA Generation
3. Topic Modelling + QA Generation

Approach #1 - Knowledge Graph





- Attempted to create a KG and generate questions
- Time taking, hard to implement, lack of resources
- Moved onto using RAGAs, ran into similar issues of entity and relationship recognition and extraction

Entity Extraction:

- Getting the nodes of the graph
- Named Entity Recognition(NER)
- LLM

Relationship Extraction:

- Getting the edge of the graph
- Rule based(syntax)
- LLM

Approach #2 - DeepEval, LLM

- **Evolutions**: Customise the type of QA sets:
  - `REASONING`: cause-effect
  - `MULTICONTEXT`: QA from >=1 contexts
  - `CONCRETIZING`: Only from context
  - `CONSTRAINED`: QA with limits to count/contexts
  - `COMPARATIVE`: Comparative QA
  - `HYPOTHETICAL`: Creative QA, what-if
  - `IN_BREADTH`: QA on the general theme, rogue
- **Filtering**: select a threshold - filtering out the responses & contexts below that threshold along every step of the process
- **Styling**: Influence the framing & wording of QA

Approach #3 - Topic Modelling

- Latent Dirichlet Allocation (LDA) for getting the topics from the documents
  - Each document is a mixture of topics.
  - Each topic is a mixture of words.
- Templates can be used to automatically generate questions (e.g., "What is *X*?" or "How does *Y* work?") around those key terms
- LLM is prompted for the wording and question logic, replicating the types from DeepEval

Results

RAGAs:

- Hallucinated, produced questions with slang, lost context

DeepEval:

✅ How do I calculate Mary Collins's total economic damages, including future care costs?

✅ What are the specifics of the April 3, 2024, Peterson Transport collision?

✅ Where was Mrs. Collins airlifted to after the accident, and who examined her in the emergency room?

Topic Modelling:

✅ If Mrs. Peterson had been driving slower, could the accident have been avoided?

✅ Is a person entitled to compensation for medical pain with an expense of $1,000,000?

❌ Is Dr. Collins related to Mrs. Coleman?

Next Steps(Recommendations)

1. Including ongoing chat context during the question generation process

Learnings

- Knowledge Graph implementation from a document context
  - Named Entity Recognition methods
  - Relationship Extraction methods
  - neo4j implementation of knowledge graphs
- RAGAs and DeepEval for QA generation
- Topic Modelling and QA generation from it