Project 1: Health Summary Generator

Objective

The project aims to reduce the time veterinarians spend reviewing a patient’s medical records before appointments.

Problem Statement

Veterinary patients often have extensive medical histories, including diagnostic details, lab reports, vaccination records, and more. These records are typically stored as unstructured PDF files, sometimes spanning hundreds of pages. Currently, veterinarians must spend 30-40 minutes manually reviewing these documents to find relevant information, which reduces overall clinic productivity.

**Plan**

**Approach:**

1. Read the complex pdf files and extract the text data as well as tabular data.
2. Chunk the raw data, for ease of LLM read and due to context/ token limit.
3. Convert chunks into embeddings
4. Store embeddings into Vector store, for similarity search.
5. Connect Vector store with LLM for query retrieval.
6. Prompt engineer the prompt template to get better and accurate response, and also mention the format of output.

**Technology Choices:**

1. Python (for a human like readability and extensive library support for data manipulation and extraction)
2. Llama parse (for extracting text from scanned pdf, by default it supports OCR)
3. Langchain Framework (ease of integration of LLm’s with other components like Prompt Template,vector stores etc)
4. FAISS vector store for storing embeddings.
5. Groq api, for faster response.

**Challenges and Solutions:**

1. Reading scanned pdf could be challenging as it needs Optical character recognition.
2. OCR could be time consuming.
3. Extracting Tabular data.

These hurdles can be cleared using llama parse, which by default offer OCR and speed, compared to traditional OCR’s.

**Solution Flow:**

****

**1. Input Layer**

* Starting Point: PDF Document containing veterinary records

**2. Document Processing Layer**

Document Parsing

* Tool: LlamaParse
* Input: PDF document
* Output: Markdown formatted content
* Configuration: result\_type="markdown"

Text Splitting

* Tool: MarkdownTextSplitter
* Parameters:
  + Chunk size: 1000 words
  + Chunk overlap: 200 words
* Purpose: Creates manageable text segments while maintaining context

**3. Embedding Generation Layer**

Embedding Creation

* Tool: JinaEmbeddings
* Model: jina-embeddings-v3
* Configuration: Requires Jina API key
* Purpose: Converts text chunks into vector representations

Vector Storage

* System: FAISS (Facebook AI Similarity Search)
* Function: Stores and indexes embeddings for efficient retrieval
* Purpose: Enables similarity-based document retrieval

**4. Query Processing Layer**

Similarity Search

* Input: User query
* Process: Converts query to embedding and finds similar documents
* Configuration: k=10 (retrieves 10 most relevant documents)
* Output: Retrieved relevant document chunks

**5. LLM Processing Layer**

Chain Configuration

* Model: ChatGroq
* Type: llama-3.2-90b-text-preview
* Parameters:
* temperature: 0 (for factual information)
* max\_retries: 2

QA Chain

* Type: "stuff" chain type
* Components:
* Custom prompt template
* Retrieved documents
* Question input
* Purpose: Orchestrates the question-answering process

Prompt Template

* Format: Structured template for health summary

**6. Output Layer**

Final Output

* Format: Structured health summary
* Storage: Written to output.txt
* Structure: Follows specified template format with:
* Tabular data where appropriate
* Chronological ordering
* Consistent date formatting
* Complete data fields (marked as "Not Available" if empty)

Data Flow Summary

1. PDF → Markdown Conversion
2. Markdown → Text Chunks
3. Chunks → Embeddings
4. Embeddings → Vector Database
5. Query → Retrieved Documents
6. Documents → LLM Processing
7. LLM → Structured Output

Error Handling

* Implementation: Try-catch block around main processing
* Output: JSON formatted error messages if processing fails
* Retry Logic: Maximum 2 retries for LLM calls

System Requirements

* Environment Variables:
* API key
* File System Access:
  + Read access to input directory
  + Write access to output directory

**Assumptions:** List down any assumptions you have made

* 1. All pdf files are scanned one and have same format