426 Medical RAGs-to-riches Project Proposal

Matthew Fecco
Will Greenwood
Jack Skupien

March 2025

1 Introduction

Retrieval Augmented Generation (RAG) is a vectorized database that Large Language Models (LLM) utilize for greater context with specialized queries. Supplementing an LLM with RAG implementation can greatly increase accuracy of model responses and allow for up to date information without the need for expensive retraining or finetuning. RAGs operate on semantic data. In order to create a RAG with a dataset, extensive data cleaning and pre-processing must be done to transform data points into efficient text documents. These documents are then chunked, tokenized, and embedded into a multi dimensional vector database. Once a database has been build, queries for a LLM can be embedded and used in a similarity search of the database to locate relevant documents. These documents are then added to the original query and processed by the LLM for better results. For our group project, we are proposing to both build a RAG implementation and conduct feature exploration of specific parameters within the MIMIC-III critical care dataset using visualization and clustering techniques to both understand hidden trends and improve LLM inference.

2 Implementation

- Data cleaning and feature selection. MIMIC-III contains both structured and unstructured data. Spread across 26 unique tables, the dataset contains rich structured data including;
  - Static data such as hospital details, diagnosis codes, and patient demographics
  - Temporal data such as vital sign, lab results, and time of admittance.

  Beyond this structured, mostly numeric data, there is unstructured data in the form of in depth clinical notes. These notes represent valuable information about patient prognosis and care details beyond numerical data.

Our first step in implementation will be cleaning clinical notes by normalizing text, removing punctuation and unnecessary characters, and summarizing information for more efficient chunking and embedding. Due to the size of our dataset, we hope to use pySpark to spread these tasks across numerous nodes for increased scalability. At the same time, we will need to transform numerical data into semantic form for LLM retrial.

- LangChain pipeline implementation Our next step will be organizing and setting up our RAG pipeline. We are proposing to use LangChain to help streamline this process. LangChain offers a wealth of tools and streamlined processes to build RAG pipelines. It gives us access to a number of document loaders and vector databases to choose from. Some of the options of databases include:
  - FAISS
  - Chroma
  - Pinecone
  - InMemoryVectorStore

  We will need to select the best database for our purposes and may test the results of multiples. LangChain will also help us connect this to a LLM for simplified testing of model inference.

- Feature exploration and visualization of results along with database clustering Once our data has been cleaned and embedded within our LangChain vector db and the pipeline has been defined, we can then move on to feature exploration and visualization. We will explore visualization of vector databases using dimensionality reduction techniques such as t-SNE and PCA. If we are unable to implement these techniques, we can pivot to relational analysis of parameter pairs for example:
  - Is there a change in survivability based upon time of admittance?
  - How does care differ across genders?
  - How accurate are clinical notes in identifying proper diagnoses?