**Progress Report: CS 410 Fall 2022**
**Team: Pinto Beans (Physician Expert Search)**
**Date: November 13, 2022**

## Progress made thus far

Our team has completed our primary data collection and data cleansing tasks. We have completed development of several webscrapers to scrape data from five hospital and medical system websites in the Chicago metro area (UChicago Medical System, Northwestern Medical Group, etc.). We have used these scripts to collect several thousand doctor profile records from the metro region, which reflect common characteristics such as name, education, specialties, and location.

We have also begun scraping data from a Mayo Clinic website that will serve as a document collection for symptom to specialty mappings (that is, a web page might have a description of symptoms associated with a certain condition, and which medical specialty that condition falls under; since we ultimately want to map a user's query of medical symptoms to doctor specialties, we need to scrape additional data to create relationships between symptom descriptions and medical specialties).

We have determined the application architecture for remaining tasks: loading our scraped data into a NoSQL document database (possibly MongoDB), creating the back-end application to model medical symptoms to physician specialties, and creating a user interface (UI) to accept user queries and return results. We have begun dividing the remaining tasks and will be able to largely work in parallel for many of these items.

## Remaining Tasks

### Data cleaning and preparation

There are 2 datasets to clean before they can be used. The first one is the physician data scraped from the hospital websites. This step involves standardization by removing whitespaces and null values, as well as analysis including lemmatization or stemming. Finally we will store these records in a document data store for ease of retrieval.

The Mayo Clinic website data does not require cleaning but analysis techniques similar to the physician data might improve matching accuracy.

## Back-end Application

1. Input scraped texts into a clinical BERT model and obtain embeddings
    a. Use a BERT model pre-trained on clinical domain - *UmlsBERT - Clinical Domain Knowledge Augmentation of ContextualEmbeddings Using the Unified Medical Language System Metathesaurus*
    b. There will be two model inputs. As we are using a transformer model, it's recommended not to remove stopwords, punctuation, etc. as they help to capture more context compared to preprocessed text.
        i. Various symptom descriptions scraped from Mayo clinic. This will be treated as the first input.
        ii. The search application will prompt the user to enter a description of their conditions/symptoms experienced that led them to search for a healthcare provider. The user-entered description will be the second input.
    c. Extract pooled N-dimensional dense vector embeddings. Instead of developing our own pooling scheme on the output embeddings by averaging or summing the outputs of the last n hidden layers, we will rely on BERT to select the optimum pooling based on the necessary knowledge needed.
2. Evaluate user symptoms against the search space of scraped Mayo symptoms. Because there are no definitive labels associated with the Mayo descriptions, it is not possible to train a predictive model. However, we expect that the user-entered description contains words, sentences that are similar to one or more of the Mayo descriptions. We will find the closest match among the list of Mayo symptoms. This can be done by detecting how close together the embeddings of input one and two are in a vector space. We will adopt cosine similarity as the distance measure.
3. Use the matched symptom to determine the relevant physician specialties for the patient search. The matched symptom in step 2 essentially serves as a bridging label. It maps conditions experienced by the patient to physician specialties.
4. Use the list of relevant physician specialties in addition to other user-entered parameters/keywords on location, insurance, languages, as conditions to compile a list of physicians from the database as the final output.

## Front-end Application

Build a front-end web application that accepts user queries and displays the doctor profiles(s) returned by the back-end application. If time permits, we will consider hosting this on a cloud platform for free (given the expected low server usage and limited data storage required).

## Challenges/issues faced

A major challenge is coming up with a good dataset, which involves scraping multiple websites and retrieving fields. Not all websites have all the fields defined (or easily accessible) so we will have to work around that. Also, our scraper is heavily dependent on each website's current output format, so it will break if the website changes.

At the moment, we are unsure whether we will be able to use unstructured data as inputs successfully. If we can't, we will revert to using structured data as our input. Our ambitious idea is that users can input their symptoms and the system would suggest the suitable doctors for them. However, depending on the timeline and also the capability of our models, we might not be able to achieve such a feat.

The other challenge is that we are planning to use a deep learning framework called HuggingFace and we have limited experience with this framework, so learning it might take some time. Further, given the nature of the data itself, for example headache as a symptom can be associated with many illnesses, so our deep learning model would need to be relatively smart in capturing this.

Another issue that we might potentially encounter is that the doctor specialty might not be specific enough. For example, on the hospital website, if the doctor specializes in internal medicine, we don't know for sure if the doctor specializes in heart disease, lung disease or diabetes, so this is one of the big challenges of the modeling.

Moreover, as not all doctors have information about insurance that they are accepting, we will most likely omit this feature from our application.

## Citations

Zheng Yuan, Zhengyun Zhao, Haixia Sun, Jiao Li, Fei Wang, Sheng Yu, CODER: Knowledge-infused cross-lingual medical term embedding for term normalization, Journal of Biomedical Informatics, Volume 126, 2022, 103983, ISSN 1532-0464, https://doi.org/10.1016/j.jbi.2021.103983.
(https://www.sciencedirect.com/science/article/pii/S1532046421003129)