MAIS 202 - PROJECT DELIVERABLE 1

Our group's idea for this project is to create a program capable of helping healthcare professionals gain time by delegating the repetitive task of pre-consultation questions to the computer in favor of nurses completing more complex tasks that require their expertise, and thus to disencumber emergency rooms in hospitals. This would also allow for a more in-depth knowledge of the patients' symptoms and antecedents, since human and time limits would not allow to spend too much time on each patient individually.

1. Choice of dataset:
https://www.kaggle.com/datasets/itachi9604/disease-symptom-description-dataset?resource=download. We plan on using this dataset because it includes, not only diseases and their corresponding symptoms, but also the associated recommended course of action. This will be helpful to assess the seriousness of the situation, and maybe create a priority among patients' order for doctor's consultation based on that. The usability is fairly high, but we might want to switch to another dataset with a higher usability if necessary (ex: https://www.kaggle.com/datasets/dhivyeshrk/diseases-and-symptoms-dataset). We might also change and write our own dataset loading script if we want to add extra inputs, such as medical antecedents (rather than simply symptoms) or add a feature recognizing pictures. Finally, we will also need to find a conversational dataset for the program to be able to ask questions in a conversational mode and make the symptoms' summary.

2. Methodology:

a. Data Preprocessing:
The dataset that we chose is feasible since it directly maps symptoms to diseases which is the heart of our diagnosis component.However, for any real-word application, this dataset would be potentially unsafe since, in our dataset, the symptom diseases mapping and the recommended course of action would be the most useful piece of information for our model since they enable both diagnosis and triage prioritization. The first step of our data preprocessing would be data cleaning which consists of first identifying any missing symptom or disease labels by removing entries with critical missing data, as imputing a symptom could introduce inaccuracies. We will also standardize all symptom names (e.g., "Fever" and "fever" become the same) to create a clean list.The second step will be the data transformation and in this case we will use binarization to convert the symptom lists into a numerical format. We will create a list of all unique symptoms and for each disease create a vector where 1 indicate the presence of the symptom and 0 indicating its absence.For example with a list [fever, cough, headache, rash], a disease with [fever, cough, rash] becomes the vector [1,1,0,1].

b. Machine learning model:
We want to help doctors predict what disease someone has based on the symptoms that they shared with our chatbot. After some research, I found that a library that would help turn sentences into vector values would help us turn text input explaining symptoms into numerical values that reflect these symptoms. Based on those, we can classify and find the disease with a random forest algorithm. A random forest algorithm can work well for diagnostic type problems because they're based on decision trees. During training, they

generate a multitude of decision trees and that makes sense when evaluating whether someone has a symptom or not, but random trees can also interpret “ordinal” or “numerical” data, like temperature or heart rate, which are important for medical diagnosis as well. The “bagging” and “feature randomness”, which are features of the random forest that help reduce overfitting, seem to fit with the diagnostic-type problem, since many diseases share similar symptoms. The weakness of random forest is that on its own, it won’t work with text based inputs, which is why we also need to use an “NLP pipeline”, so text pre-processing, embedding and classifier. I am not very familiar with the inner workings of them, but the “fine-tuned transformer” would be the best classifier. Alternatively, there is the logistic regression/linear models, because they’re simple to use, but they’re not very complex and so they might not give good results.

c. Evaluation Metric:

Our project is a diagnostic chatbot that analyzes textual input, and provides a possible text diagnosis through data classification. Indeed, our textual response relies on correctly assigning likely labels (potential diseases) from the data inputted (symptoms) out of a large set of possibilities (our dataset). Therefore, we shall evaluate its performance off of standard evaluation metrics for multi-label classification and text generation models. These include:

- Accuracy (multi-label classification): This will measure the fraction of correctly predicted labels over the total number of actual labels for that instance. It will then take the average over each instance and produce our accuracy score. We would hope to have a baseline accuracy percentage of at least 80%.
- Confusion matrix and F1-score (multi-label classification): Takes into account precision (being correct when predicting something) and recall (catching all the right cases) to ensure a balance. For instance, if either precision or recall is low, even though the other may be very high, our F1-score will be low. We aim to have a baseline F1-score of 70% since it is harder to inflate.
- BLEU score (text generation): This will measure how similar our generated text is to reference/template text, ensuring quality in sentence structuring. For this, we would like to have a score higher than 40% as this is a hard task to accomplish.
- ROUGE score (text generation): This will establish whether the most important information was included in the generated text. In our case, we should make sure to actually include the symptoms and possible diseases in our text. We will aim once again for a score higher than 40%.

3. Application:

In the application, the user enters through a text input their information and planned time of arrival at the hospital (this is not AI but would be essential if the solution was to be implemented), then they feed the program with the input we are interested in: the symptoms. Then, there would be two types of outputs delivered to two different receivers. First, the patient would receive follow-up questions to which they would answer again, generating additional input. Then, the program would generate a summary of the symptoms / antecedents in order of priority for the doctor to look at easily. This could include the top 5 potential diagnoses with their likelihood, but it should not be too straightforward, as this could bias the doctor’s diagnosis, and that for ethical purposes the doctor should remain the main decision-maker. The program has an idea of what the diagnosis might be, but this can be used for highlighting the most important symptoms, instead of clearly stating a diagnosis.