Practicum Sprint #8

Mental/Physical Illness Chatbot

Tusheet Goli, Tejas Pradeep, Akshay Sathiya, Pranav Khorana, Sanket Manjesh, Rahul Chawla
tgoli3@gatech.edu, tpradeep8@gatech.edu, asathiya6@gatech.edu, pkhorana3@gatech.edu, smanjesh3@gatech.edu, rchawla36@gatech.edu

1 ACCOMPLISHMENTS THIS WEEK

1.1 Akshay Sathiya's Progress

Akshay Sathiya began writing test cases for the physical illness prediction (PIP) part of the project. The test cases consist of ten messages that are sent to the chatbot that uses the PIP system and the response builder to communicate to the user the physical illnesses that they are most likely at risk of.

The PIP system works by first extracting symptoms from a message and then passing that symptom information to the PIP models to predict the physical illnesses those symptoms correspond to. Akshay inspected the results of the PIP system on the test cases and found that the symptom extraction step could be more effective.

The former technique for symptom extraction Akshay developed involved text cleaning, keyword extraction using RAKE (Saxena, 2020), text vectorization using a TF-IDF vectorizer (scikit-learn developers, 2022, *sklearn.feature_extraction.text.TfidfVectorizer*), and distance comparisons between vectors for keywords and symptoms. Akshay developed a new machine learning technique to replace the former technique.

Akshay wrote code to assemble a symptom extraction dataset that consists of several sentences with fixed templates (e.g. "I am not feeling well. I am experiencing …", "This is weird, I am feeling …") and lists of one, two, and three symptoms substituted in the templates. This dataset is meant to simulate the types of messages users would submit to the chatbot.

Akshay then rewrote the text cleaning logic as a Texthero preprocessing pipeline (Texthero, n.d.) for ease of customization and modification. Akshay then trained

a multi-output classifier (built from several neural networks) (scikit-learn developers, 2022, *sklearn.multioutput.MultiOutputClassifier*) on this dataset to predict the symptoms that a given message pertains to. Each output corresponds to a unique symptom in the physical diseases dataset (Patil, 2020), and represents whether or not the user's message includes that symptom. This symptom extraction (SE) model is saved as a pipeline (scikit-learn developers, 2022, *sklearn.pipeline.Pipeline*) that includes a TF-IDF vectorizer as well as the multi-output classifier itself.

Similar to the PIP models, the SE model is also evaluated on the train/test set (65-35 split) and evaluated using k-fold cross-validation (scikit-learn developers, 2022, *sklearn.model_selection.Kfold*), where k = 5. The SE model had very high accuracies and F1 scores under these evaluations. The F1 score used to evaluate the SE models is a weighted (by the number of positive/true samples where a symptom is present) average of the F1 scores for each symptom (scikit-learn developers, 2022, *sklearn.metrics.f1_score*).

The SE model's very high accuracies and F1 scores are a sign that it may be overfitting on the generated symptom extraction dataset, so Akshay decided to test the updated PIP system with the new SE model on the test cases (which are not a part of the generated symptom extraction dataset). After inspecting the results of the PIP system (with the SE models) on the test cases, Akshay noticed that a significant majority of extracted symptoms were correct and the new symptom extraction technique is much more accurate than the former symptom extraction technique. However, minor improvements can still be made.

Next week, Akshay will continue improving the symptom extraction code and inspecting the results of the PIP system (with the SE models) on the test cases. Akshay will also help integrate the PIP system with the backend and resolve any issues that come up there.

Akshay's work has been merged with the *main* branch of the project's GitHub repository.

1.2 Pranav Khorana's Progress

This week, Pranav worked on connecting the flask server to the front end allowing the messages to be stored in the database. The NLP model can formulate responses back to the user based on the messages sent and stored in the database. He also started working

on the authentication process for user login. This will probably involve a simple database query for username and password.

1.3 Rahul Chawla's Progress

This week, Rahul worked on developing the mental illness prediction models, via a sentiment analysis approach, as outlined by the tasks. He created a subdirectory in the repository for mental illness prediction and began exploring datasets to train the model. This week he will continue to pull messages from the database and use this information to pass into a time series sentiment analysis model which will be used to determine if the user is at risk for anxiety, depression, or bipolar disorder.

1.4 Tusheet Goli's Progress

This week, Tusheet continued working on linking the backend service with the database and the machine learning model. He worked along with Tejas to integrate the backend API with the front-end UI buttons to properly relay information between the PostgresSQL database and the machine learning model of the physical illnesses prediction. Using last week's progress where he was successfully able to link it with a Heroku-hosted PostgresSQL database, he was able to test the functionality of the services and their endpoints to match the expectations. Tusheet set up a similar pipeline for the mental illnesses prediction machine learning model as well. Unfortunately, he was unable to test it since the mental illnesses prediction machine learning model is still being worked on. But the template pipeline for all of this is set up. Tusheet also helped with logging the chatbot conversations to the backend database to keep a record of conversations and the resultant illness classification from the machine learning model.

Tusheet has thus set up the backend pipeline to relay information between the front end screen and the machine learning model to accurately transmit information to the appropriate services at each step and display the resultant illnesses classification. Next week, Tusheet is going to test the pipeline for the mental illness prediction and try to integrate all aspects of the app to get a fully functioning MVP out. This is Tusheet's main goal for next week. After that, he will work on finetuning this pipeline and ensuring all edge cases have been tested and accounted for to finally release a fully working application by Sprint#10.

1.5 Tejas Pradeep's Progress

This Tejas worked on various odd fixes in the backend. This week Tejas also worked on beginning to link the front end with the backend. During this upcoming week, Teja s shall work with Tusheet to set up the overall pipeline and put together the various components of the project to wrap up the project. Currently, I am blocked by the other aspects of the project finishing up.

1.6 Sanket Manjesh's Progress

This week, Sanket pushed his progress with the Flask server so far and will let Tusheet fix the issues dealing with linking the Flask server with the PostgreSQL database as he has more experience in the area. Now, Sanket has shifted his focus to the ML side of the project and will be helping with developing the NLP models to discern sentiment from chats. These tasks will involve setting up models to detect bipolar disorders through random switches in positive/negative sentiment and also keyword detection for diseases mentioned within texts. Sanket will look into using the TextBlob library for sentiment analysis.

2 CHALLENGES ENCOUNTERED

2.1 Tejas's Challenges

This week Tejas had a bunch of other projects due and hence was not able to donate much time to this project. This next week Tejas can spend more on this project.

2.2 Pranav's Challenges

This week, Pranav had a lot of projects to work on and was not able to make too much progress on the project. Next week he will spend more time on the project by polishing the UI and ensuring the backend is well integrated with the frontend.

3 FUTURE PLANS

The team members made significant progress on their tasks. In the future, each team member will continue working on their respective tasks and prepare for their respective parts of the project to be integrated with one another.

4 REFERENCES

1. Patil, P. (2020, May 24). *Disease Symptom Prediction*. Kaggle. Retrieved March 6, 2022, from https://www.kaggle.com/itachi9604/disease-symptom-description-dataset
2. Saxena, N. (2020, September 6). *Extracting Keyphrases from Text: RAKE and Gensim in Python*. Extracting Keyphrases from Text: RAKE and Gensim in Python | by Nikita Saxena | Towards Data Science. Retrieved April 3, 2022, from https://towardsdatascience.com/extracting-keyphrases-from-text-rake-and-gensim-in-python-eefd0fad582f
3. scikit-learn developers. (2022). *sklearn.feature_extraction.text.TfidfVectorizer*. sklearn.feature_extraction.text.TfidfVectorizer — scikit-learn 1.0.2 documentation. Retrieved April 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html
4. scikit-learn developers. (2022). *sklearn.metrics.f1_score*. sklearn.metrics.f1_score — scikit-learn 1.0.2 documentation. Retrieved April 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html
5. scikit-learn developers. (2022). *sklearn.model_selection.Kfold*. sklearn.model_selection.KFold — scikit-learn 1.0.2 documentation. Retrieved April 3, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.KFold.html
6. scikit-learn developers. (2022). *sklearn.multioutput.MultiOutputClassifier.* sklearn.multioutput.MultiOutputClassifier — scikit-learn 1.0.2 documentation. Retrieved April 16, 2022, from

https://scikit-learn.org/stable/modules/generated/sklearn.multioutput.MultiOutputClassifier.html#sklearn.multioutput.MultiOutputClassifier
7. scikit-learn developers. (2022). *sklearn.pipeline.Pipeline*. sklearn.pipeline.Pipeline — scikit-learn 1.0.2 documentation. Retrieved April 16, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html
8. Texthero. (n.d.). *Getting started*. Getting started · Texthero. Retrieved April 16, 2022, from https://texthero.org/docs/getting-started#custom-pipeline