CS325 Team23 Midterm Progress Report

1st Haoran Chen
University of Tennessee
Electrical Engineering and Computer Science dept
Knoxville, USA
hchen73@vols.utk.edu

2nd Fort Hunter
University of Tennessee
Electrical Engineering and Computer Science dept
Knoxville, USA
lhunte21@vols.utk.edu

3rd Ngan Huynh
University of Tennessee
Eletrical Engineering and Computer Science dept.
Knoxville, USA
nhuynh2@vols.utk.edu

***Abstract*—There is a growing prevalence of mental challenges that individuals face nowaday. We have created a Natural Language Processing model named OM aim to analyze a patient's status from their given statement. We implemented the current model with a baseline logistic model and trained it with a sentinel analysis for mental health dataset from Kaggle. We have developed several plans to refine and improve the current model performance including a more balanced dataset, hyperparameter turing and combining multiple models.**

***Index Terms*—Machine Learning, Natural Language Processing, Sentiment Analysis**

I. Introduction

The increasing prevalence of mental challenges has created an urgent need for tools that can assist therapists in assessing patients' mental states effectively. Traditional methods of mental health evaluation often rely on subjective interpretations of patient statements, which can lead to inconsistencies. The goal of this project is to predict a patient's status based on their written or spoken statements. Our project, Om, aims to address this issue with a Natural Language Processing (NLP) model that predicts patients' mental states based on their statements. OM will provide a data-driven approach to assist therapists in real-time analysis and evaluation using a dataset that includes patient statements and their corresponding mental health status. Our dataset contains statements and the corresponding mental health status.

II. Data Exploration

The data set we worked with contained approximately 52,000 phrases, each accompanied by one of seven mental health statuses of the person who said the phrase. These statuses likely included categories such as anxiety, depression, normal, suicidal, stress, bipolar, and personality disorder mental states. The phrases are sourced from several different web-based sources, such as Facebook, Reddit, and Twitter. While the data we were working with was generally clean, there were more cleaning steps we had to do, as well as preparing it for our NLP tasks, because eliminating unnecessary data reduces the computational resources required for processing and analysis. The first step of cleaning our data was removing missing values. The Exploratory Data Analysis shows that 362 rows contain NULL statements that must be eliminated from our dataset. This data cleaning step was necessary because NULL values could cause the model to break or produce unreliable results and introduce noise into the analysis. Removing these NULL values ensures that our model is trained on clean, informative data, likely improving its accuracy and performance. The next step in the data preprocessing was removing unnecessary substrings. These included hyperlinks, punctuation, and hashtags. It was necessary to remove these because they would add confusion to our model. These elements would likely detract from sentiment prediction and not contribute positively to the model's performance. Next, we made everything lowercase so that everything was in the same format. This will help us ensure words with different capitalization are seen as the same.

The following steps of data manipulation are done solely to prepare the data for NLP. The first step in this process is to tokenize the phrases in the dataset. Tokenization is the process of breaking down our phrases into smaller, more manageable chunks. For the data we were working with, that meant breaking them down word by word. We tokenize because it allows us to see the frequency and relationships between certain words and the emotions they are associated with. It also gives our model a uniform way to analyze each piece of data we feed into it. Most importantly, during the tokenization process, emoticons such as ":)," ":(," and "..." were retained due to their prevalence in chat language and their ability to convey emotional polarity [1]. These emoticons can significantly contribute to understanding mental states, with ":)" indicating positive sentiment and ":(" suggesting negative sentiment]. To effectively capture these chat-specific features, the TweetTokenizer() was employed. After we tokenize, the next step is to remove stop words. Stop words in the English language are words that do not contribute to the sentiment of a text.

Words such as "and," "but," and "or" help with the flow of

Identify applicable funding agency here. If none, delete this.

language but do not contribute to the overall understanding of the statement. Removing these words will help with confusion within our model. Since these words will be in almost every text, our model could draw false conclusions when they see these words. However, essential negation words such as "not," "no," "none," or "never" are intentionally retained in our analysis. In mental health contexts, statements like "I don't want to feel this way" or "I never give up" carry different semantic weight about an individual's emotional state (ref:NLTK :: nltk.sentiment.util module). These negative determiners were excluded from a custom list of stop words due to their crucial roles in distinguishing between affirmative and negative expressions. This led to a more reliable classification of mental health states. Another advantage of removing stop words is that it reduces the number of tokens we work with. This should improve the efficiency of training our model and save us time in later steps of the data processing pipeline.

Once we have tokenized and eliminated unnecessary words, we will break down the words into simpler forms. This will allow us to normalize words. Words such as "hate," "hated," and "hating" are all the same word but with different tenses. Breaking them down to their simplest form, in this case, "hate," would allow our model to see them all as the same. Two main ways of breaking these words down are stemming and lemmatization. Stemming is the more accessible and much more efficient method. Stemming simply cuts off suffixes and prefixes based on predefined rules. While it is efficient, it does have some drawbacks. It will often produce words that do not exist in the English language. For example, a word like "scaling" would likely be reduced to "scal" when the correct way to reduce it would be "scale". Lemmatizing solves this problem. Lemmatization takes grammatical knowledge and a dictionary of sorts to reduce words to their actual base, making the results more interpretable for both humans and machines [4]. Lemmatization maintains the core meaning of words, ensuring that the processed text retains its original semantic intent [2]. Unlike stemming, lemmatization considers the context and part of speech, resulting in more accurate word transformations [3], but it does come at the price of much more computation. To illustrate the impact of lemmatization, consider the following examples:

- Statement 1: "Sensitive feelings make heart restless" lemmatized into "Sensitive feeling make heart restless"
- Statement 2: "Gr gr dreaming ex crush game god" lemmatized into "Gr gr dream ex crush game god"
- Statement 3: "Anyone know of a way that has no chance of failure, is decently quick and is not a gun? It would be much appreciated Painless way except gun?" lemmatized into "Anyone know of a way that have no chance of failure, be decently quick and be not a gun? It would be much appreciate Painless way except gun?"

As demonstrated, lemmatization effectively reduces words to their base forms while preserving grammatical and semantic integrity. This process enhances the subsequent TF-IDF vectorization by ensuring that related words are treated as a single feature, thereby improving the accuracy of sentiment analysis and other NLP tasks [5].

The last step in the data preprocessing step is to vectorize our words. Vectorizing is the process of transforming our tokens into numerical representations. Since most algorithms deal with numerical data as opposed to text, this is an essential step in making the data usable in our model. The most straightforward approach to vectorizing is to use what is known as "Bag of Words." In Bag of Words, each word is assigned a unique index in a vector known as the vocabulary. Once we have read every word, each phrase will have a unique vector based on the words that are in it.

III. Baseline Model

Logistic regression is a statistical model used for binary classification tasks. It estimates the probability that a given input belongs to a particular class. There are several key benefits of choosing a logistic regression model. It's a simple model suitable for future improvement and also effective with text features.

Logistic regression is a simple and baseline model for future improvements. Each feature weight indicates the direction and magnitude of its contribution to the prediction result. This feature of interpretability helps us visualize and understand which feature contributes most strongly to the corresponding mental health status. In addition, logistic regression is straightforward to implement and provides a good starting point for further improvement or determining whether a more complex model is necessary.

Logistic regression is effective when using Bags of Words or TF-IDF to represent each text feature. Using TF-IDF helps identify which words are more informative for classification, providing valuable insights into what drives sentiment in text. In addition, Logistic Regression has a lower risk of overfitting since it has fewer parameters to tune in this case.

We also began work on a Convolutional Neural Network model that will serve as an advance extension to what we currently have. We chose to start with the logistic regression model because of it's simplicity and overall decent accuracy. However, moving forward we will implement the CNN model to more accurately make predictions.

For our implementation, we implemented a logistic regression model and converted the text data into numerical vectors using TF-IDF. We then convert the sparse TF-IDF matrix into a dense array so they can be fitted in the scaler. We also implemented Logistic Regression with 'C' as a hyperparameter. The value of C is set to 100 for the moment, which implies low regularization strength. This might lead to overfitting, especially if the model complexity is high. We plan to use techniques such as cross-validation to find the optimal C value. In conclusion, the current baseline model is able to achieve an accuracy score around sixty-three percent. The current model is able to achieve a higher accuracy on "normal" cases, but it also requires improvement on other minority cases.

	precision	recall	f1-score	support
Anxiety	0.62	0.65	0.64	755
Bipolar	0.68	0.64	0.66	527
Depression	0.55	0.57	0.56	3016
Normal	0.85	0.78	0.81	3308
Personality disorder	0.65	0.47	0.55	237
Stress	0.41	0.41	0.41	536
Suicidal	0.47	0.51	0.49	2158
accuracy			0.62	10537
macro avg	0.60	0.58	0.59	10537
weighted avg	0.63	0.62	0.63	10537

Fig. 1. Performance Report

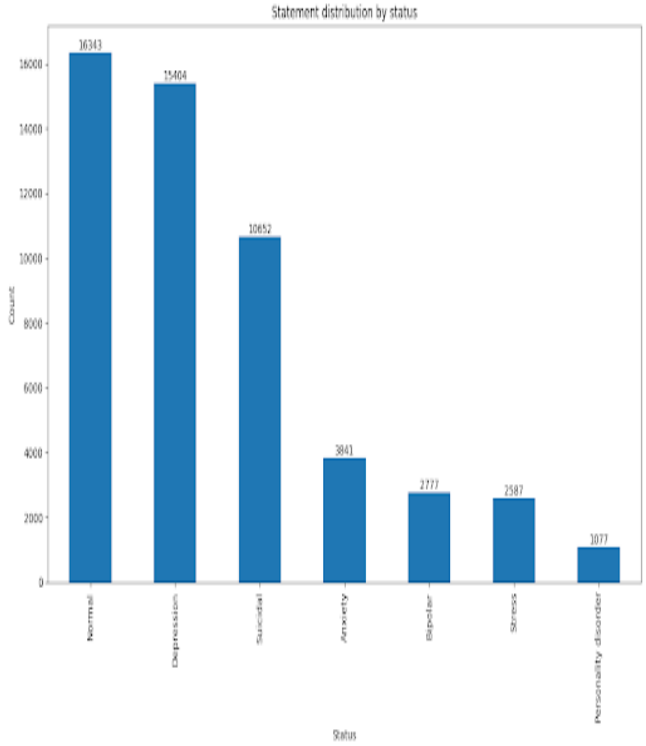

Fig. 2. Sentiment Analysis Data Set

IV. Proposed Extension

While the current version of model OM merely serves as a basic model, we have discussed several methods and possibilities for improving and enhancing the current model performance. Inherent limitations of logistic regression prevent the model from advancing further. The current logistic regression model may not reach its optimal performance due to the complexity of the given statement and its incapability to capture dependencies across words and phrases.

We plan to enhance the current model performance with several different methods including a more improved and refined data set to replace the current data set. Implementing hyperparameter tuning within our model. Furthermore we can choose a more suited model, such as a convolutional neural network, aside from logistic regression model and combine multiple models to optimize performance.

A. Data Balancing

The data set comprises over 52,681 statements and their corresponding mental health status. Only around thirty-one percent of the statements are Normal, with the remaining sixty-nine percent of the statements feature negative mental status. In this case, the provided data set has imbalanced classes with more negative sentiments than positive statements. As a result, this can affect the model's ability to generalize, producing suboptimal results and accuracy. The model may achieve a high accuracy by predicting the majority classes for most instances, while failing to identify the minor class.

B. Limits of Logistic Regression

Advanced models would better capture ambiguous wording (lack of context-sensitive meanings, use of slang, syntax, and grammar errors) by dealing with completed sentences instead of independent tokens. Models such as CNN, Random Forest, Principal Component Analysis, LSTM or BERT are designed to enhance text prediction by resampling biased datasets where some features dominate others. Because this project aims to build an interactive chatbot or contribute the model to similar projects, applying more complex or balanced models for more advanced text classification would become demanding.

Logistic regression models are relatively ineffective when processing and handling complex sentences. Complex sentences consist of sarcasm, ambiguous nuances, and slang. In this case, logistic regression's linear decision bounty might not be sufficient enough to provide the most optimal score.

C. Choosing a More Advanced Model

Models designed to understand a sequence of words, in other words, identify complex relationships between words. Their algorithm helps grasp the context and nuances in the text by learning whole sentences, especially in cases when users provide only a few or unclear sentences. The advanced models can balance skewed target distributions, which enhances prediction accuracy by reducing possible high variance. Underrepresented labels would be weighted similarly.

D. Hyperparameter Tuning

Tuning parameters like C, penalty type, solver, max_iter, and class_weight directly affects how well the model fits the data, handles imbalanced classes, generalizes to unseen data, and efficiently converges. In this case the data set is not balanced with "normal" status statements as a minority class. In comparison, all remaining status statements feature negative mental status. In imbalanced datasets, the model may be biased toward the majority class. Using class weights helps the model pay more attention to the minority class, improving recall and reducing bias.

E. Combining Multiple Models(Ensemble)

Sentiment analysis often deals with ambiguous language, sarcasm, and mixed sentiments. By combining different models, we can reduce the risk of missing these complexities. For example, while the baseline logistic regression model can focus on the word frequencies and patterns on mental status, we can implement a decision tree that focuses on certain keywords rules.

V. Conclusion

The current model is successful at predicting patients' mental status based on their given statements. The current implementation of model OM is a logistic regression model trained from the Sentiment Analysis for Mental Health data set from kaggle. The imbalanced data set as well as the inherent limit of logistic regression impose challenges on us to further improve our model. However, we have developed several plans to enhance and improve the overall performance of our model. We plan to combine multiple models, implement hyperparameter turnings as well as a more refined and balanced dataset will further improve the model performance.

VI. Distribution of Work

We all worked on our individual models and processed the data ourselves. Chen and Huynh used two Logistic Regression models, while Hunter employed the Convolutional Neural Network (CNN) model. We were all involved in every step of building the models for our understanding. The area where we had limited knowledge was primarily the text preprocessing part. We chose different Python libraries and tools such as SpaCy, Scikit-Learn, TensorFlow, and NLTK to explore the best cleaning process before training our models. The experimentation process was extensive and time-consuming. However, as a result, we added more advanced text cleaning and contextual considerations into the preprocessing stage as we aimed to achieve the best accuracy in text prediction. Our goal was not only to base predictions on static keywords but also to take into account contextual identifiers. This comprehensive approach to preprocessing allowed us to create more robust models that could better understand the nuances and context of the mental health-related text data we were analyzing.

References

[1] D. Kalita, "A Comprehensive Overview of Sentiment Analysis," Analytics Vidhya, Apr. 01, 2022. https://www.analyticsvidhya.com/blog/2022/04/a-comprehensive-overview-of-sentiment-analysis/