BUAN 6342

Group Project

Fall 2024

Sri Sai Sashank Pyneni (sxp220217)

Zach Landry (zxl230003)

Phuc Minh Le (pml160230)

Ryan Adnan Saleh (ras141430)

Harsha Kolachina(sxk180148)

## Use Case:

Mental health is a critical aspect of overall well-being, and early detection and intervention are crucial for effective treatment. Social media platforms provide a rich source of data for understanding mental health trends and identifying individuals in need of support. This project aims to develop a sentiment analysis model that can accurately classify mental health statuses from text-based statements.

## Methodology:

### Corpus Used:

The dataset utilized in this project is a meticulously curated collection of mental health statuses tagged from various statements. The dataset has been cleaned and compiled to create a robust resource for developing chatbots and performing sentiment analysis.

### Data Preparation/Data Cleansing:

The following pre-processing steps were performed on the dataset:

* Stop Word Removal
* Punctuation Removal
* Change to Lower Case
* Tokenization
* Label Encoding

### Python Code:

#### Splitting the Data

The data was split into an 80-20 train-test split. The target variable became the "status\_encoded" column, which was numerically encoded using LabelEncoder.

#### Tokenizing the Text Data:

The text data was tokenized using the first 5000 frequently-occurring words. All other words were mapped to a default value of 0.

#### Padding Sequences

The sequences were padded to ensure uniform input length for efficient batch processing in deep learning models.

#### Converting Target Variable to Categorical Format

The target variable was converted to a categorical format for the softmax activation function in the output layer.

#### Model Attempts:

##### Deep Learning LSTM Attempt 1

* Used an LSTM model with a dropout of 20% and a recurrent dropout of 20%.
* Training Accuracy: ~84%
* Validation/Test Accuracy: ~74%

##### XGBoost Model

* Used TfIdfVectorizer() to convert the data to a numeric matrix.
* Trained for 100 rounds using mlogloss as the loss metric.

##### Naive Bayes, Decision Tree, & Logistic Regression Model

* Used TF-IDF encoded matrices for training and testing.
* Output a classification report and a confusion matrix for each model.

##### Deep Learning Model (LSTM) - Attempt 2

* Dropped empty cell values by converting to NaN and dropping null values.
* Training Accuracy: ~83%
* Validation/Test Accuracy: ~74%

##### Deep Learning Model (LSTM) - Attempt 3

* Dropped empty cell values and removed the "personality disorder" and "bipolar" categories from the target variable.
* Oversampled the data to balance the dataset across the remaining 5 statuses.
* Training Accuracy: ~95%
* Validation/Test Accuracy: ~90%

##### Deep Learning Model with VADER Sentiment Analysis (LSTM) - Attempt 4

* Added a feature to determine the negativity of the text using VADER sentiment analysis.
* Training Accuracy: ~95%
* Validation/Test Accuracy: ~90%

### Testing the Model:

The model was evaluated using accuracy as the main metric to ensure the statements are classified correctly. We also recorded the accuracy on a validation to ensure our models did not suffer from overfitting. The accuracy graphs for the models have been provided in the appendix below:

### Conclusion or Summary:

We chose LSTM as our primary model since these RNNs have a "hidden state" that helps them remember long-term dependencies through the use of gates to remember/forget information. Text data tends to be sequential since the context & meaning of words depends on the surrounding words, leading us to believe the LSTM model as the best fit. The LSTM model generally performs better than the Naive Bayes, Decision Tree, Logistic Regression, and Gradient Boosting models. This is in part due to the different regularization techniques used such as dropout & recurrent dropout for the hidden state. It is also dependent upon the batch size used since smaller batch sizes can lead to better convergence for deep learning models. However, dropping the "bipolar" and "personality disorder" categories from the target variable in the dataset led to a significant jump in model performance, both in training & validation/testing accuracy. This is most likely due to the text between both categories being similar, leading to confusion for the model during predictions regarding which is the correct classification. This resulted in a training accuracy jump from ~84% to ~95% and a validation/test accuracy jump from ~74% to ~90%. After adding VADER sentiment scores as a feature to the dataset in Attempt 4, the accuracy stayed around the same.

The main limitation of this model is that if it encounters out-of-sample text that corresponds to the either of the status labels that were dropped, it will misclassify the statements. Future improvements can be done via feature engineering to increase accuracy without dropping any labels to avoid compromising prediction accuracy in production environments.

## References

https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health

https://realpython.com/python-keras-text-classification/

https://www.kaggle.com/code/annastasy/mental-health-sentiment-analysis-nlp-ml#-5.-Model-Traning-and-Evaluation-

https://medium.com/@tomaslruffo/sentiment-analysis-whit-vertex-generative-ai-7a5e6c10eda4

## Appendix:

### A close-up of words Description automatically generatedWord Clouds:

### Deep Learning Model (LSTM) - Attempt 1

A graph with a line

Description automatically generated

### XGBoost

A screenshot of a graph

Description automatically generated

A graph of a number of boosting rounds

Description automatically generated

### Deep Learning Model (LSTM) - Attempt 2

A graph with a line

Description automatically generated

### Deep Learning Model (LSTM) - Attempt 3

A graph with a line

Description automatically generated

### Deep Learning Model (LSTM) - Attempt 4

A graph with a line

Description automatically generated

### Final Best Confusion Matrix:

A graph of confusion matrix

Description automatically generated