Suicide Ideation Detection

Eric Chen, Matthew Wang, Jessica Zhao

17th April, 2024

1 Introduction

Mental health continues to be a pressing issue. In the US alone, an alarming 132 suicides happen every day ("Suicide Statistics"). Globally, depression affects over 280 million people, with these numbers rising annually (Koskie). Despite the severity and growing prevalence of these issues, there is not enough being done to prevent them. Our group wanted to explore whether natural language processing models could be used as a tool to detect suicide ideation and prevent it from happening. We aim to detect patterns and indicators of suicide ideation in text. This is not only innovative, but also very important in helping to prevent this critical issue.

2 Data

To begin building and training our models, we required a data set which was representative of the issue at hand. We chose a dataset consisting of approximately 50,000 Reddit posts, which were labeled under one of five categories: suicide watch, depression, anxiety, bipolar disorder, or off my chest. This diverse dataset enabled us to not only identify texts related to suicidal ideation but also to recognize expressions from individuals affected by other mental health issues. Consequently, this approach allowed us the flexibility to conduct both multiple-class and binary classifications. In the latter case, we could more effectively distinguish between negative texts that were suicidal and those that were not.

3 Models

3.1 Logistic Regression

One of the models we decided to train was scikit-learn's Logistic Regression. Logistic Regression is a robust statistical modeling technique that is widely used for classification tasks. Given its versatility and simple implementation, Logistic Regression acted as a great starting point for our project. To train the Logistic Regression model, we first must transform our text inputs into numerical form, which we did using scikit-learn's tf-idf vectorizer. The reason for choosing the tf-idf vectorizer is that it is effective at filtering out common words so that it focuses on the important words that distinguish between the labels.

3.2 Long Short-Term Memory Network

We opted to test the Long Short-Term Memory network, commonly known as LSTM, as our second model. LSTMs are particularly adept at analyzing textual data, where the order and context of the text play crucial roles. Given these capabilities, the LSTM model was a highly suitable choice for our comparative analysis. For our particular LSTM model, we use the pre-trained GloVe word vectors to provide the model with the ability to understand some semantic meanings. In order to begin training the LSTM model, each text data point is converted to a numerical vector using Keras's Tokenizer. Once the training data has been fitted to an instance of the tokenizer, we convert them to integer sequences using text_to_sequences(). Then we pad the sequences to the length of the biggest data point (reddit text) to ensure they are all the same dimension. For the multiple class classification, we convert the labels for each training input to one_hot encoding to help optimize the training process. As for the model itself, it consists of the three layers, an embedding layer, the LSTM layer, and an output layer. The LSTM layer consists of 210 units and a tanh activation function. The tanh activation was chosen solely due to compatibility with Google Colab's A100 GPU. Finally, the model was compiled with categorical_crossentropy and the adam optimizer. After many iterations, the final LSTM was fitted using 8 epochs and a batch size of 40.

3.3 Transformer

For the last model, we opted to test a pre-trained Transformer and fine tune it. The transformer's self-attention mechanism makes them particularly effective for tasks like suicide detection in text, where understanding the context and nuances of language is crucial for accurate classification, makes it a great last option to compare. To train our transformer, we used DistilBERT, a streamlined version of BERT, known for its adeptness in understanding contextual nuances in text. This made it a perfect fit for our task of identifying suicidal language. We then used the AdamW optimizer, which helped optimize training by effectively managing weight decay, thereby preventing overfitting and enhancing model generalization.

4 Results

4.1 Logistic Regression Results

Without modifying the dataset at all and allowing our logistic regression model to perform multiclass classification, we were able to obtain an accuracy of 66.64%. Because there are a total of 5 classes, our model will have a harder time distinguishing between the classes, especially when they are very similar. We decided to train our model on percentages of our training data to see how using more data allowed us to obtain more accurate results.

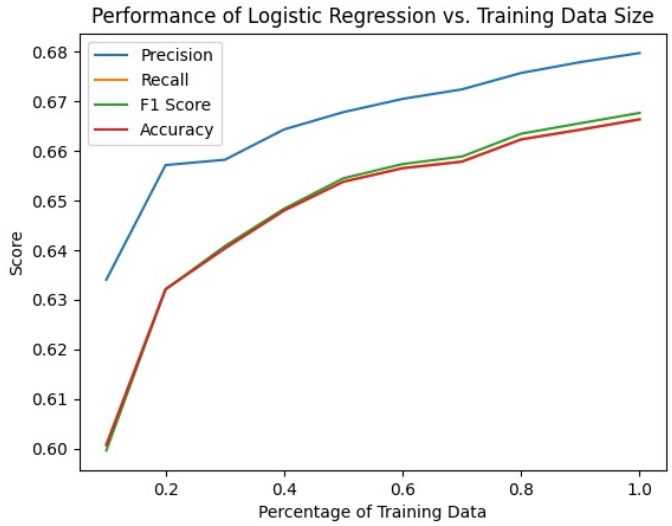

Figure 1: Performance of Logistic Regression vs. Training Data Size

As shown, there is a clear increase in our performance metrics every time we increase the percentage of training data used.

Next, we wanted to see how well a simple logistic regression model performed on binary classification, either suicide or not suicide. We replaced every label that was suicide with a 1, and every other label a 0. We then ran into a different problem: there were significantly more 0's than 1's, around 37k to 8k. We decided to run two models using binary classification, one with upsampled 1's, and one with downsampled 0's.

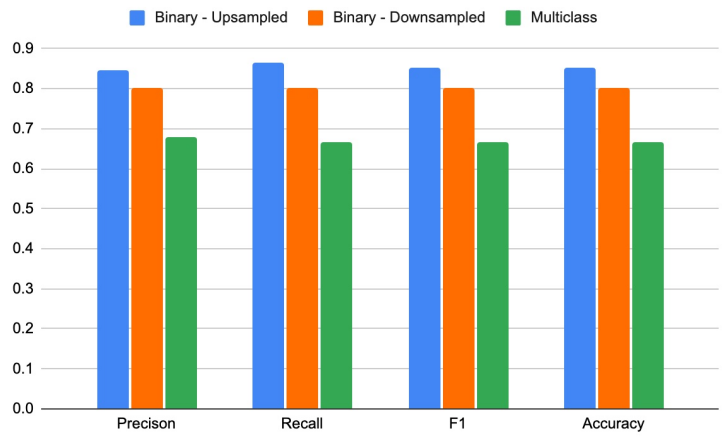

Figure 2: Comparison of Logistic Regression Metrics

Using a binary classification logistic regression model with upsampled data obtained the highest accuracy of around 85.24%, almost a 20% increase from our original model. The model using downsampled data also saw a significant increase from the original, with all of its metrics sitting around 80%.

4.2 LSTM Results

The initial LSTM multi-class classification model was trained using 2 epochs, a batch size of 32, and the LSTM layer had 128 units. The metrics of this initial model are illustrated below.

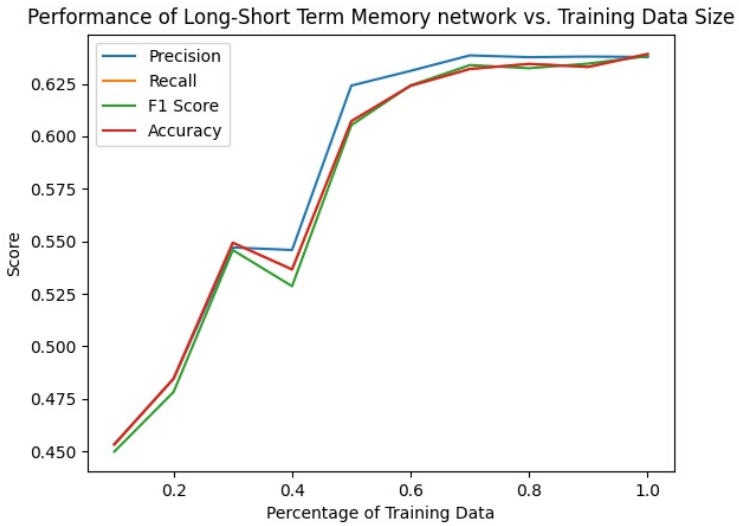

Figure 3: Comparison of LSTM Metrics

The figure showed the overall performance of the LSTM improved as more data was given, however it only had an accuracy of 63.7%. After further optimization and incorporating GloVe word embeddings, the final LSTM model yielded an accuracy of 65.6%, an increase of roughly 1.9% when tested on the same test split. The same LSTM was then trained on the binarized data, which was both upscaled and downscaled. Most of the hyperparameters remained consistent with the previous multi-class version except for the output activation function which was changed to a sigmoid function. The metrics of all three models were compared and graphed below.

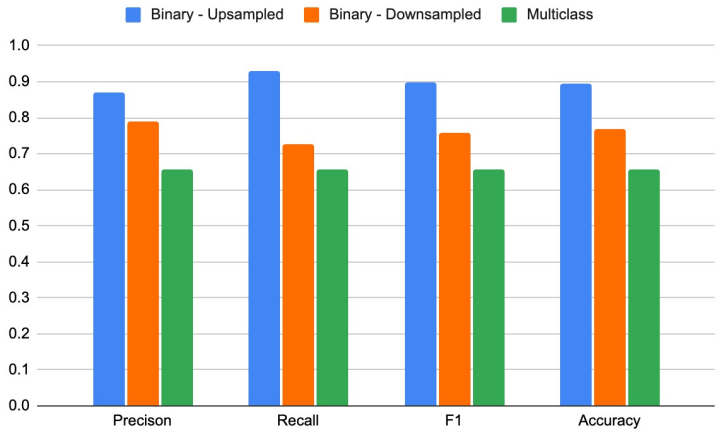

Figure 4: Comparison of LSTM Metrics

When the binarized data was upscaled, the model yielded an accuracy of 0.8946 on the 20% test split and the downscaled data yielded an accuracy of 0.7689. This meant that the LSTM generally performed better than the Logistic regression model when the data was binarized rather than non-binarized.

4.3 Transformer Results

The initial transformer multi-class classification model was trained over 3 epochs, a batch size of 16. The metrics are plotted below over the 3 epochs.

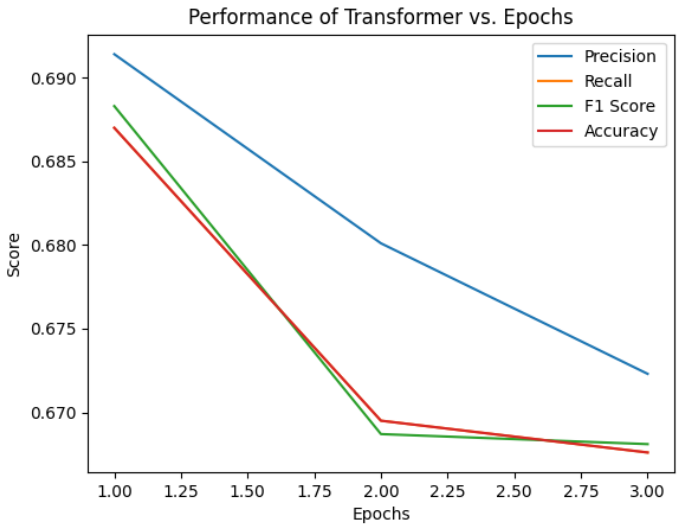

Figure 5: Comparison of Transformer Metrics

The performance of the transformer is the best after one epoch where it achieved an accuracy of 0.6870. After the first epoch, the model very clearly begins to overfit on the training data. Across the three epochs, the training loss decreases from 0.7230 to 0.3608 while the validation loss increase from 0.3608 to 1.0739.

We then ran the transformer model with binary classes.

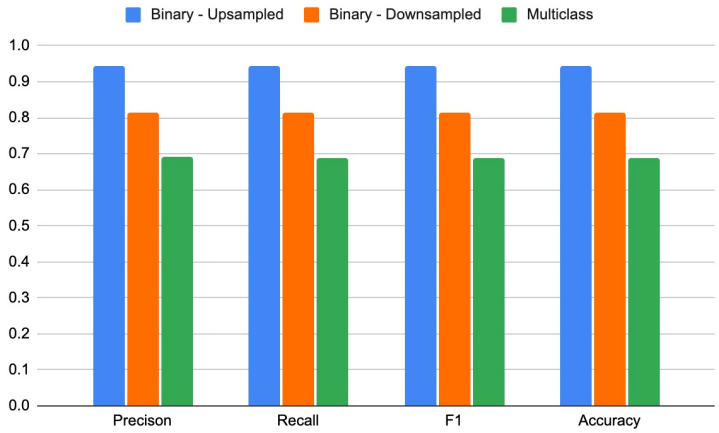

Figure 6: Comparison of Transformer Metrics

The binary classification transformer model with upsampled data obtained the highest accuracy of 0.9432 after three epochs, almost a 26% increase from the multiclass model. The metrics of the upsampled model all increased throughout the three epochs and only began to show signs of overfitting when the validation loss increased slightly at the third epoch. The model using downsampled data also saw a significant increase from the multiclass model, with an accuracy of 0.8145. Unlike the upsampled model, the downsampled metrics were the best over at the first epoch and all decreased over the three epochs.

4.4 Overall

Out of the the three models, the Transformer performed the best on both binary and non-binary classification when it came to suicide ideation. The transformer was shown to be the most consistent as well as being the most accurate. From this, we could conclude that the transformer would generally perform the best when it comes to detecting suicide ideation in texts.

5 Future Work

This evaluation was done with a culmination of texts related to people with mental health, however it was solely from Reddit. In order to be more confident in our overall decision it is crucial to use a diverse data set, one that includes other social media platforms such as Twitter. It would also be beneficial to be able to have a data set of personal texts and direct messages though it is difficult to attain such a data set due to privacy issues.

6 Link to Project Code

https://github.com/22ericchen/Suicide-Ideation-Detection

7 References

[1] "Suicide Statistics - SAVE: Suicide Prevention, Information, and Awareness." www.save.org, 25 Jan. 2024. Accessed 17 Apr. 2024.

[2] Koskie, Brandi. "Depression: Facts, Statistics, and You." Healthline, 2018, www.healthline.com/health/depression/facts-statistics-infographic.

[3] Ji, Shaoxiong. "Supervised Learning for Suicidal Ideation Detection in Online User Content", Hindawi, 2018, https://www.hindawi.com/journals/complexity/2018/6157249/