Reducing Type II Errors for Suicide Ideation Detection

Nathan Clairmonte
McGill University
Montréal, Québec, Canada
nathan.clairmonte@mail.mcgill.ca

Abstract

Suicide and suicide ideation are health issues faced by many people globally. Social media provides an online space where individuals can express their thoughts, including suicidal tendencies. As a result, identification of this suicidal ideation in online texts is becoming increasingly necessary. However, as with many sensitive issues, some misclassifications are more detrimental than others. This project investigates the classification of online text based on the presence of suicide ideation and demonstrates using three models, that false negative misclassification errors can be minimized without a significant decrease to the overall accuracy.

1 Introduction

Suicide is largely considered one of the most prevalent health issues experienced worldwide. WHO (2021) estimates that 800,000 people globally have committed suicide in 2021, up from 788,000 in 2015. Suicide ideation refers an individual's thoughts of committing suicide and is often regarded as a potential indicator of the risk of a suicide attempt (Beck et al., 1979).

With the advent of the Internet and social media, there has been an increasing tendency for individuals to discuss their suicidal thoughts, feelings and intentions online. The ease of access, sense of community and anonymity of online interactions results in people more freely discussing topics which they generally might not in the real world, such as suicide and suicide ideation. This has led to a large corpus of online content, analysis of which could potentially provide an avenue into early suicide detection and ultimately suicide prevention. Therefore, classification of online text into suicide ideation and non-suicide ideation categories is becoming increasingly necessary.

While such classification is useful, it is not sufficient to simply attempt to maximize overall classification accuracy on this problem. This is because a false negative misclassification (i.e., predicting that an individual is not expressing suicidal ideation when they actually are) is much more detrimental than a false positive misclassification in this context. Therefore, in this project, we investigate the classification of online text into suicide ideation and non-suicide ideation categories, placing a specific emphasis on the reduction of misclassifications in the form of false negatives.

There are many different techniques that seek to minimize false negative misclassifications, some of which can be applied to various models. These different techniques can have varying effects on the model, including a negative effect on its overall accuracy. But with the right balance of hyperparamters, minimizing false negative errors should not significantly decrease the overall accuracy of a model.

The dataset analysed in this project consists of 232,074 reddit posts (taken from the r/SuicideWatch and r/teenagers subreddits) that are labeled as either suicide or non-suicide based on the subreddit they were taken from (Komati, 2021). The dataset is balanced, with 116,037 samples from each class.

2 Related Work

Detection of suicide ideation through online content has been the focus of much research in recent years. Fu et al. (2013) discovered that social media can be used to identify suicide ideation in its early stages, and Jashinsky et al. (2014) suggested that analysing suicidal risk factors in Twitter posts can in fact be used to prevent suicide attempts. While these works are similar to the work presented in this project, they differ in that they approached the problem from a psychological perspective rather than a machine learning (ML) or natural language processing (NLP). Despite this, many other researchers have taken the NLP route to detecting suicide ideation through online content.

For example, Vioules et al. (2018) used Twitter data to quantify suicide-based risk indicators for individuals and to detect suicide-related content in posts. Their work combined behavioural and textual features from the Twitter data, which were then applied to a martingale framework in an effort to recognize distinct changes in users' online behaviour; something they posit as being an indicator of suicide ideation risk. On the other hand, our work focuses solely on textual features and applies them to various ML models in an effort to reduce false negative classifications of suicide ideation.

In addition, Lv et al. (2015) attempted to detect suicide ideation in social media posts using a "suicide dictionary" of words from a Chinese social media platform which were likely to indicate risk of suicide ideation. This dictionary was then utilised in an attempt to detect suicide ideation in social media posts. Similarly, Huang et al. (2015) identified and collected data from users who had committed suicide in order to build a "psychological suicide lexicon" using psychological standards and word embedding techniques. They then utilised this lexicon along with ML algorithms such as topic models to identify suicide ideation. These works are, again, very similar to the work presented in this project, but differ in that they utilise a form of dictionary or lexicon related to suicide ideation while our work does not.

3 Method

To perform the investigation into reduction of false negative misclassifications, three approaches were analysed. First, a simple Logistic Regression model with Bag-Of-Words features (Section 3.1). Second, a linear Support Vector Machine (SVM) classifier with term frequency-inverse document frequency (TF-IDF) features (Section 3.2). Third, a convolutional neural network. (Section 3.3).

3.1 Logistic Regression with Bag-Of-Words

3.1.1 Bag-Of-Words Representation

Bag-Of-Words is a simple but effective way of classifying text by retrieving the word counts. It is to be expected that such a simple approach leads to lost meaning of the text, as the vector of counts does not take semantics into consideration. It simply looks at each word occurrence and values each instance equally, regardless of the order.

This representation was achieved by using Scikit-Learn's CountVectorizer from the text feature extraction library, which allowed us to view the resulting vocabulary representation of the dataset. We immediately noticed that from the word counts, there were many words relevant to the target study that were among the words with the highest number of occurrences. Table 1 shows some of these relevant words along with their rank in the complete list of words.

3.1.2 Logistic Regression Classification

The logistic regression classifier passes weighted input features to a sigmoid function which in turn outputs the resulting classification. For this reason, it works very well with a Bag-Of-Words representation. In our case, we are interested in two possible output values, which will allows us to know whether the model classifies the text as a suicide class or not.

The classifier was implemented with Scikit-Learn's LogisticRegression from the linear model library, which allowed us to train and get results on our dataset. In order to reduce the false negative rate obtained from the model as well as improve the resulting accuracy, we utilized Scikit-Learn's GridSearchCV method to perform a grid search on the dataset and check which regularization parameter leads to better results. We then adjusted the value of C to regularize the classification. The dataset used for this method was reduced to prevent runtime errors in the training phase due to limited computing resources.

3.2 Linear SVM with TF-IDF

3.2.1 Linear SVM Classification

An SVM attempts to find a hyperplane between two classes in a set of training samples that distinctly classifies the samples. It does this by defining a margin which maximizes the distance between the decision boundary and the closest samples from

Rank	Word
1	wonder
4	void
5	unstable
23	sad
40	mistake

Table 1: Most common relevant words from the dataset.

each of the two classes (Mavroforakis and Theodoridis, 2006).

Typically, data with non-linear class boundaries can be transformed with a kernel such that linear hyperplanes which separate classes can still be defined. However, in this project, only linear SVMs (i.e., no kernel transformations) were investigated. The reason for this was twofold. Firstly, Scikit-Learn's linear SVC estimator converges more quickly than their SVC function with the kernel set to linear, which reduced experimentation time significantly. Secondly, their linear SVC estimator allows for easier retrieval of confidence scores, which was necessary for the prediction threshold analysis described in Section 3.2.2.

Two analyses were performed as described in Sections 3.2.2 and 3.2.3, along with a baseline for comparison with no alterations made. All SVM models were trained on 80% and tested on 20% of the data, using TF-IDF on uni- and bi-grams as features.

3.2.2 Decreasing Prediction Threshold

As mentioned, Scikit-Learn's linear SVC estimator provides access to sample confidence scores, onto which a threshold can be applied to make class predictions. The positive class is predicted if a sample's confidence score is above the threshold, and the negative class is predicted otherwise. Therefore, decreasing the threshold will cause fewer negative predictions and more positive predictions to be made overall. This results in a desirable decrease in false negatives and increase in true positives, but has a trade-off because it also results in an undesirable increase in false positives and decrease in true negatives. However, for suicide ideation, this trade-off is still favourable because false negatives are largely more detrimental than false positives.

The default threshold used for the baseline estimator is 0, and experiments were performed for a decreased threshold of -0.1.

3.2.3 Altering Class Weights

Generally, the classes in a given set of training data are not completely separable. Therefore, the margin defined by an SVM is often softened to allow for some misclassifications. This softening of the margin is defined by a regularization hyperparameter, C (Batuwita and Palade, 2013). Usually C applies the same regularization weight to each class, but this is not a necessity. Instead, one class can be weighted less than the other, thereby increasing the penalization of misclassified samples for that class.

The baseline estimator weights both classes equally, and experiments were performed for altered class weights where the positive class was weighted more than the negative class, thereby penalizing false negative misclassifications more heavily. The weights were altered such that $C^{+} = 100C^{-}$.

3.3 Neural Network

With the advancements in deep learning, neural networks have become an increasingly popular tool for classification. Specifically, convolutional neural networks which scan text encoded as sequences using 'windows' of different sizes, can be extremely effective for text classification (Kim, 2014). Different concepts are captured by each window and the most important feature is extracted using a max pooling layer. Then these outputs are concatenated and a dropout layer is applied which randomly nullifies inputs to avoid overfitting. Finally, a layer using an activation function computes the final output. Our neural network uses window sizes 3, 4, and 5 with a dropout rate of 0.5 and a sigmoid activation function to determine the final output. The model was compiles with a binary cross entropy loss function and a RMSProp optimizer,. This model was built using the Keras library which provides an API for the Tensorflow library. All neural networks were trained on 80% of the dataset and tested on the remaining 20%, Due to time and resource constraints, training was limited to 3 epochs.

3.3.1 Altering Class Weights

Altering class weights is another common method for reducing specific types of errors and it works well for various models. By default, both classes are treated the same and have equal impact on the loss function. By specifying class weights, we are able to adjust the calculations such that classes with higher weights have a greater impact on the training. For our experiment, we false negative misclassifications are much more significant and by assigning a greater weight to the positive class, their effect on the loss function will be greater and thus the number of them will be minimized. For our neural network, we set the weight of the positive class to 0.6 and the weight of the negative class to 0.4.

3.3.2 Custom Loss Function

The loss function used in the baseline neural network is binary cross entropy. This function is a measures the amount by which the predicted probabilities diverge from the actual labels. For the hypothesis of our experiment, we were looking for a way to specifically reduce false negative results. From our class notes, recall and precision are defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{1}$$

$$Precision = \frac{TP}{TP + TN} \tag{2}$$

$$F1 = (1 + \beta^2)\frac{Recall * Precision}{Recall + \beta^2 * Precision} \tag{3}$$

To attempt to lower false negatives without affecting accuracy, we explored a custom loss function that used the f1 score as a loss function, with a custom $\beta = 2$ to capture that recall is twice as important as precision in our case. This will weigh recall as more important and thus minimize false errors.

4 Results

To evaluate the performances of the logistic regression classification, the three SVM variants analysed (baseline, decreasing prediction threshold and altering class weights) and the neural network variants, similar metrics on the dataset were computed and compared in order to obtain comparable results for the models.

The metrics measured were the false negative rate (FNR), true positive rate (TPR) and overall model accuracy. These metrics are defined in Equations (4), (5) and (6) respectively, using true positives (TP), true negatives (TN), false positives (FP) and false negatives (FN). Since the goal was to reduce false negative predictions, decreases in FNR and increases in TPR were desired. In addition, it was desired that these changes did not result in more than a 1% decrease in the overall model accuracy.

$$FNR = \frac{FN}{TP + FN} \tag{4}$$

$$TPR = \frac{TP}{TP + FN} \tag{5}$$

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

4.1 Logistic Regression with Bag-Of-Words

The metrics for the Logistic Regression with Bag-Of-Words representation are shown in Table 2.

4.2 Linear SVM with TF-IDF

The aforementioned metrics are shown in Table 3 for each of the three SVM variants analysed. Decreasing the prediction threshold resulted in an FNR reduction of 19.9% and an overall accuracy reduction of only 0.07%, while altering the class weights resulted in an FNR reduction of 33.6% and an overall accuracy reduction of only 0.47%.

4.3 Neural Network

Table 4 displays metrics for the baseline neural network as described in the Methods section of the paper, as well as the metrics for both variations which attempted to minimize false negative misclassifications.

5 Discussion and Conclusion

Comparing the results between the Logistic Regression model and the SVM model, it is clear that the baseline variant of the SVM model was a drastic improvement over the fine-tuned LR model. This was to be expected as the LR model with BoW representation compromises accuracy due to its simplistic approach. In turn, the false negative rate was also much higher.

The decreased threshold variant of the SVM model produced less false negatives than the baseline variant and hardly compromised the overall

FNR (%)	TPR (%)	Acc (%)	TP	TN	FP	FN
11.189	88.811	90.548	3,667	3,872	325	462

Table 2: FNR, TPR, overall model accuracy, TP, TN, FP and FN for the Logistic Regression classifier with Bag-Of-Words representation.

Experiment	FNR (%)	TPR (%)	Acc (%)	TP	TN	FP	FN
baseline SVM	5.839	94.161	94.618	21,933	21,984	1,138	1,360
decreased threshold	4.675	95.325	94.553	22,204	21,683	1,439	1,089
altered class weights	3.877	96.123	94.172	22,390	21,320	1,802	903

Table 3: FNR, TPR, overall model accuracy, TP, TN, FP and FN for each of the linear SVM variants analysed.

Experiment	FNR (%)	TPR (%)	Acc (%)	TP	TN	FP	FN
baseline CNN	7.148	92.852	94.112	21,549	22,133	1,074	1,659
altered class weights	6.407	93.593	94.377	21,721	22,084	1,123	1,487
custom loss function	7.687	92.313	90.438	21,424	20,553	2,654	1,784

Table 4: FNR, TPR, overall model accuracy, TP, TN, FP and FN for each of the neural network variants analysed.

accuracy at all, resulting in a reduction of only 0.07%. However, altering the class weights of the SVM model proved to be the technique that resulted in the lowest FNR. This technique did result in a larger accuracy reduction of 0.47%, but such a reduction was negligible in comparison to the 33.6% reduction in FNR. Overall, both techniques were largely successful because they managed to significantly reduce FNR without decreasing the overall accuracy by more than 1%.

The baseline CNN model used in the experiment had an accuracy comparable to the SVM model, though an initially higher FNR. Like for the SVM counterpart, altering class weights proved to be effective for the neural network as well and resulting in an FNR reduction of 10.4%. However, the custom loss function decreased the FNR and the overall accuracy of the model. This is likely due to the unsophisticated nature of the loss function which took twice as long to run per epoch and would most likely require more epochs to achieve comparable results.

Our results confirmed our original hypothesis that simple techniques can be used to reduce false positive misclassifications for a variety of models without having an adverse effect on overall accuracy. When the situation demands, as in the case of suicidal ideation, there are a number of effective strategies to minimized type II errors.

In future, our experiment could be extended to include more models. It would also be interesting to analyze similar techniques across all models. As we have seen with altering class weights for SVM and neural networks, the same technique can have very different application methods and results depending on the model. It would also be beneficial to apply these techniques to other datasets with similar sensitivity to false negative misclassifications, like convictions or medical diagnostics.

References

Rukshan Batuwita and Vasile Palade. 2013. Class imbalance learning methods for support vector machines. *Imbalanced learning: Foundations, algorithms, and applications*, pages 83–99.

Aaron T Beck, Maria Kovacs, and Arlene Weissman. 1979. Assessment of suicidal intention: the scale for suicide ideation. *Journal of consulting and clinical psychology*, 47(2):343.

King-wa Fu, Qijin Cheng, Paul WC Wong, and Paul SF Yip. 2013. Responses to a self-presented suicide attempt in social media. *Crisis*.

Xiaolei Huang, Xin Li, Tianli Liu, David Chiu, Tingshao Zhu, and Lei Zhang. 2015. Topic model for identifying suicidal ideation in chinese microblog. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation*, pages 553–562.

Jared Jashinsky, Scott H Burton, Carl L Hanson, Josh West, Christophe Giraud-Carrier, Michael D Barnes, and Trenton Argyle. 2014. Tracking suicide risk factors through twitter in the us. *Crisis*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the*

2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Nikhileswar Komati. 2021. Suicide and depression detection.

Meizhen Lv, Ang Li, Tianli Liu, and Tingshao Zhu. 2015. Creating a chinese suicide dictionary for identifying suicide risk on social media. *PeerJ*, 3:e1455.

Michael E Mavroforakis and Sergios Theodoridis. 2006. A geometric approach to support vector machine (svm) classification. *IEEE transactions on neural networks*, 17(3):671–682.

M Johnson Vioules, Bilel Moulahi, Jérôme Azé, and Sandra Bringay. 2018. Detection of suicide-related posts in twitter data streams. *IBM Journal of Research and Development*, 62(1):7–1.

WHO. 2021. Global health observatory - mental health.