IST707 Project Deliverable

Real or Not? NLP with Disaster Tweets

Quinn Knudsen, Sean O'Neil & Chris Bryla

Syracuse University

Table of Contents

About the Challenge

Twitter has quickly grown to become one of the most popular social media channels in the world. With a wide range of users ranging from spoof accounts of peoples' pets to prominent public figures, organizations, and governmental bodies, this platform is used to express thoughts, feelings, and information at a clip of 500 million tweets per day (Sayce, 2019). With that level of volume, velocity, and variety it is of no surprise that data scientists have interest in exploring Twitter data. Combined with a projected Natural Language Processing (NLP) market size growth from $10.2 billion in 2019 to $26.4 billion by 2024, the timeliness of skill development in this area is apparent (Business Wire, 2019).

Specific to this challenge, disaster relief organizations and news agencies are interested in understanding whether tweets with seemingly similar language are referring to a disaster or some other life event that my linguistic appear in a similar manner to a disaster. Take, for example, the tweet in *Figure 1* (below) which uses the word 'ablaze' to describe the sky. The combination of the visual and the contextual clues given in the sentence such as 'sky' help a human extrapolate meaning and make the judgement that this tweet is not truly about a disaster. However, making that distinction apparent to a trained model will provide much more difficulty. This challenge aims to solve for that exact dilemma by classifying whether a tweet is about a disaster based solely upon the contextual indictors they provide.

Figure 1. Example tweet from the Kaggle challenge.

Exploratory Data Analysis

The columns in this dataset include a unique identifier (the tweet *id*), independent variables such as *keyword* (a particular keyword in the tweet), *location* (the location the tweet was sent from), *text* (the text of the tweet), and the dependent variable *target* (denotes whether the tweet is about a real disaster or not). ID is a nominal variable without any missing values, there are 221 unique key words (1% missing values), 33% of the location data is missing, and there is slight imbalance in the target variable with 43% of the trainset data classified as an actual disaster. See Table 1 (below) for a descriptive summary.

```
> sort(table(df$location),increasing=TRUE)[1:10]

                    'Merica              'SAN ANTONIOOOOO'                   'soooota
                          1                             1                           1
                  -?s?s?j??s-          -6.152261,106.775995          #????? Libya#
                          1                             1                           1
#1 Vacation Destination,HAWAII                #93???#734          #BlackLivesMatter
                          1                             1                           1
                 #BossNation!
                          1
```

Figure 2 (above). List of the least common locations highlighting the ineffectiveness with using locational data.

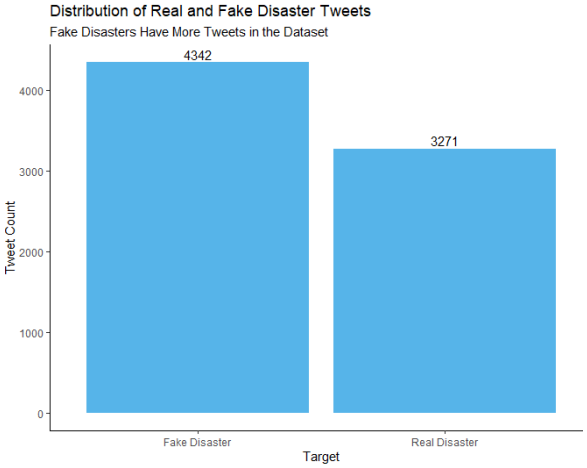


Figure 3. Distribution of the target variable showing more fake disaster than real.

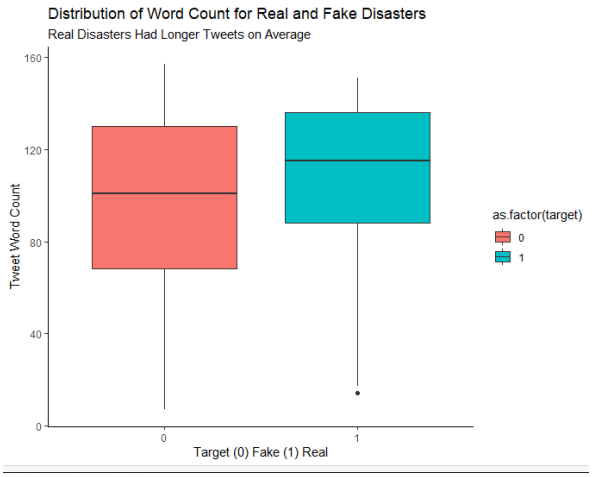


Figure 4. Distribution of word count per tweet of the target variable. Real disasters had more words per tweet on average.

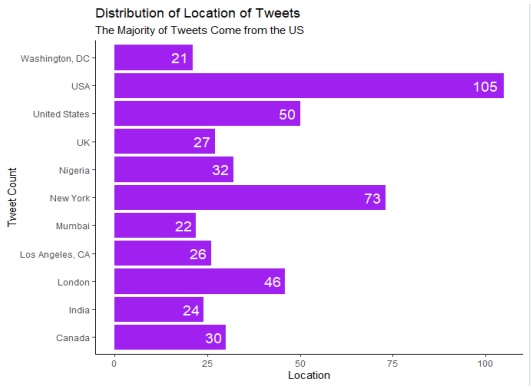


Figure 5. Distribution of tweet location shows the majority come from the USA.

Summary of Trainset				
id	**keyword**	**location**	**text**	**target**
Min: 1	**Length:** 7613	**Length:** 7613	**Length:** 7613	**Min.:** 0.0000
1st Qu: 2734	**Class:** character	**Class:** character	**Class:** character	**1st Qu:** 0.0000
Median: 5408	**Mode:** character	**Mode:** character	**Mode:** character	**Median**: 0.0000
Mean: 5442				**Mean:** 0.4297
3rd Qu: 8146				**3rd Qu:** 1.0000
Max:10873				**Max:** 1.0000

Table 1. Summary descriptive analysis of the training dataset provided by Kaggle.

Data Preparation

Based upon the exploratory analysis, we decided to not use the location in our analysis because most of the locations were either missing, misspelled, or not a real location. The keyword column was also not included, because the word was extracted from the text along with all the other words.

In order to analyze textual data, a document-term matrix must first be created. This process was initialized by first converting all the text to lowercase, then creating a text corpus by tokenizing the tweets, breaking them into lists of words, using a custom regex tokenization developed at CMU for twitter data. Several different preprocessing methods were tested, including removing punctuation, URLs, mentions, numbers, emojis, and stopwords. Once the initialization of the words has completed, a document-term matrix was built and the words that only occurred in one document were removed. The same processing was applied to the test dataset, and the words that didn't occur in both training and test sets were removed, since this would allow our models to still work on the test data. The training set was then split into train (70%) and validation (30%) parts for initial model evaluation.

Before Preprocessing:

After Preprocessing:

Figure 6 (above). Word cloud of the most popular words before and after removing stopwords.

Sentiment Analysis

In order to test our hypothesis that the wording of true disaster tweets would have a more negative sentiment than non-disaster tweets, the text column of this dataset was evaluated against a sentiment function that gave individual scores to each sentence of the tweet. This was stored as a data frame before sentences were aggregated to provide one final sentiment score for each line. This sentiment score was then combined with the original dataset as a newly engineered feature. Grouping on target, a t-test was used to determine significant differences. The results indicated that true disasters (-0.15), and non-disasters (-0.06) had significant differences in means $p<.001$. As it would be expected, both scored negatively in their overall sentiment, but tweets written about true disasters were significantly more negative. However, since the distribution of sentiment overlaps by so much between the two classes (see figure 7), it did not improve the classification accuracy when included in models.

Actual Tweet	Sentiment Score
[1] "This real [explicit] will damage a [explicit]"	-1.70
[2] "wreck? wreck wreck wreck wreck wreck wreck wreck wreck wreck wreck	-1.67
[3] "Bomb Crash Loot Riot Emergency Pipe Bomb Nuclear Chemical Spill Gas Ricin Leak Violence Drugs Cartel Cocaine Marijuana Heroine Kidnap Bust"	-1.40
[4] "anxiety attack ??"	-1.24
[5] "Suicide bombing is just the fear of dying alone"	-1.20

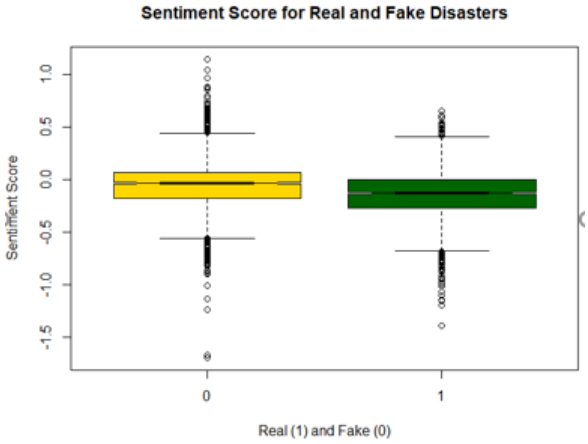


Table 2 (above). The lowest scored training tweets based upon their sentiment. Although on average the disaster tweets had a more negative sentiment, only tweets 3 and 5 here were coded as real disasters.

Figure 7 (left). Real disaster tweets were more negative on average than non-disasters.

Modeling

Preliminary models were trained on 70% of the data and tested on 30% of the data. Modeling was conducted using R and Python, and external datasets were not included beyond what was provided from the Kaggle challenge.

Preliminary tests showed that Support Vector Machine, Naïve Bayes, Random Forest, Logistic Regression, and Neural Network models may perform well on this dataset. 5-fold cross validation was used to find more precise values for the expected score to compare to the final

submissions to Kaggle to see if the model generalizes well. Since the Kaggle competition metric was the F1 score, we used this to compare the models, however, we also looked at the accuracy, precision, and recall of the models.

SVM

Strengths: SVM's can model non-linear decision boundaries, and there are many kernels to choose from. They are also robust against overfitting, especially in high-dimensional space.

Weaknesses: SVM's are memory intensive, trickier to tune due to the importance of picking the right kernel. This is also a "black box" approach which offers no probabilistic explanation for classification and is more difficult to use for feature selection/importance. SVMs are also susceptible to missing data. This can limit some of its applied use.

Results: The best SVM results were found with a linear kernel and a C of around 0.03. The best model used minimal preprocessing and on average had an F score of 83.73%, an accuracy of 80.36%, a precision of 79.33%, and a recall of 88.67%.

Naïve Bayes

Strengths: Even though the conditional independence assumption rarely holds true, NB models perform well, especially for how simple they are and offer probabilistic outputs. NB models scale well with larger, high dimensional data which make it effective for NLP problems.

Weaknesses: Due to their sheer simplicity, NB models are often beaten by models listed here when they are properly trained and tuned.

Results: Multinomial Naïve Bayes was used, with the best smoothing value found to be 0.85. The best Naïve Bayes model used bigrams and TF-IDF transformed data, and on average had an F score of 83.27%, an accuracy of 78.59%, a precision of 75.09%, and a recall of 93.46%.

Random Forest

Strengths: They are robust to outliers, scalable, and able to naturally model non-linear decision boundaries thanks to their hierarchical structure. Random forests are also uniquely able to handle missing and unbalanced data.

Weaknesses: Interpreting a random forest model can be difficult, because the decision relies on the aggregate predictions of many decision trees. A caveat with random forest is that they can overfit the training data, particularly when the data is noisy.

Results: The best random forest models were trained with 500 trees. The best performing model used minimal preprocessing an on average had an F score of 82.98%, an accuracy of 78.63%, a precision of 76.01%, and a recall of 91.37%.

Logistic Regression

Strengths: Outputs have a nice probabilistic interpretation. Logistic models can be updated easily with new data using stochastic gradient descent.

Weaknesses: Logistic regression tends to underperform when there are multiple or non-linear decision boundaries. They are not flexible enough to naturally capture more complex relationships.

Results: Logistic regression models were trained as simple neural networks using Tensorflow Keras. Since models converged at different rates, early stopping based on validation accuracy was used to avoid overfitting. The best model was the one using minimal preprocessing, and on average had an F score of 83.88%, an accuracy of 80.66%, a precision of 79.95%, and a recall of 88.24%. This was the best model out of all the models that were trained and tested.

Neural Network

Strengths: The depth of network allows for more complex non-linear decision boundaries to be learned. They lend to complex relationships and can be easily updated with gradient descent.

Weaknesses: Neural networks become difficult to interpret when additional layers are added and having too many layers can lead to overfitting. Given all the possible neural network architectures, it can be difficult to find the optimal architecture.

Results: Several different neural networks architectures were trained with TensorFlow Keras with varying hidden layers, numbers of hidden layer neurons, and activation functions. However, none of the models performed as well as simple logistic regression, and adding additional layers only leads to overfitting. The best model used minimal preprocessing, had only one hidden layer, and used dropout to reduce overfitting. On average, the F score was 83.87% (just barely less than for logistic), the accuracy was 80.61%, the precision was 79.77%, and the recall was 88.44%.

Conclusion: Results and Submission

Some of the best models were trained on the full training dataset and then predictions were made on the test dataset. These results were submitted to Kaggle to get the final F scores for the models. The models which were tested were the best Random Forest, Logistic, SVM, and Naïve Bayes models, along with a few other models to compare the models with and without preprocessing. All models except Naïve Bayes performed best with minimal preprocessing, which is interesting, because it indicates that there is important information in the words which are commonly removed, like stopwords. All of the models scored a few percentage points less than the average score from cross-validation, indicating slight overfitting. However, overfitting is to be expected with text data like this, because the training data and test data are very unlikely to

have the same distribution of words used. The best model was logistic regression with minimal preprocessing, with a score of 0.8098.

Table 3. Kaggle submission scores.

Type of model	**Preprocessing**	**Final score**
Random Forest	Minimal	0.7955
Random Forest	Stopwords, TF-IDF	0.7832
Logistic Regression	Minimal	0.8098
Logistic Regression	Stopwords, TF-IDF	0.7822
Logistic Regression	Bigrams	0.7955
Support Vector Machine	Minimal	0.7935
Naïve Bayes	Bigrams, TF-IDF	0.7965

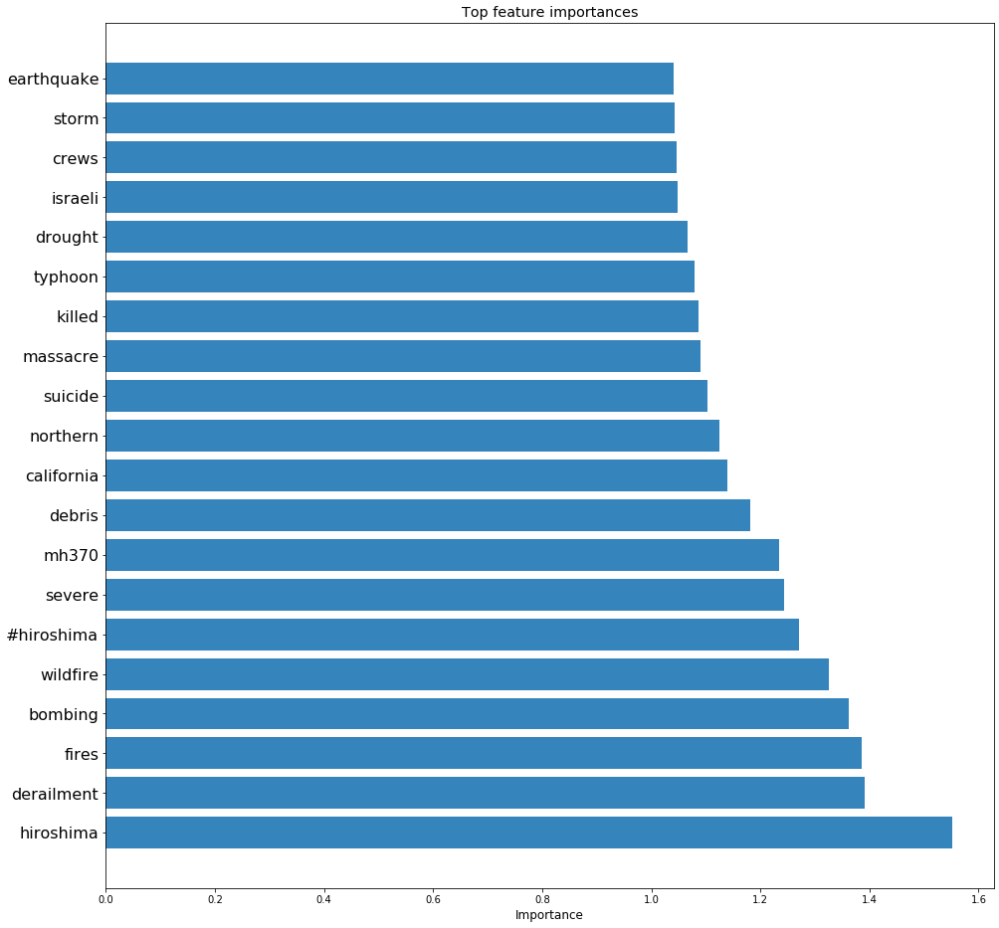


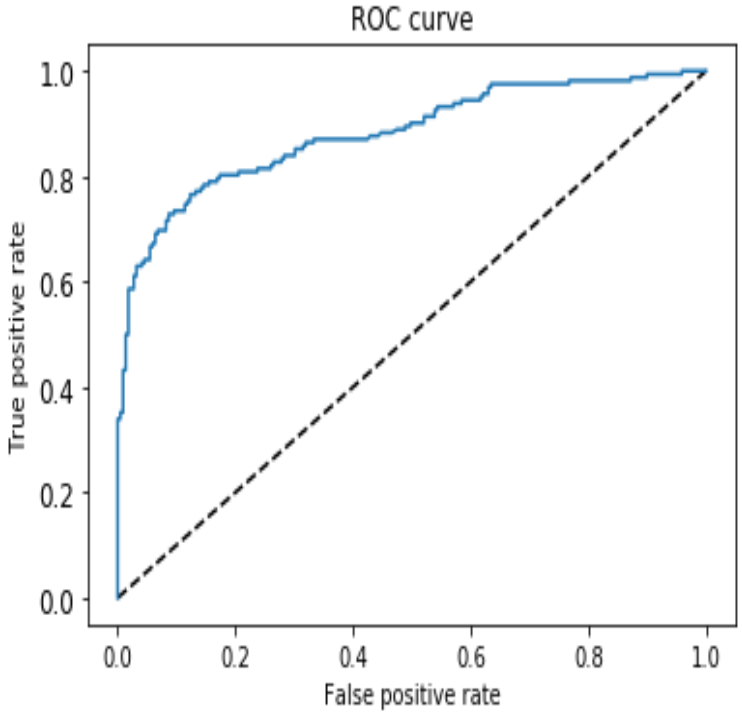
Figure 8. Top feature importances, in terms of weights, for the best logistic regression model.

Looking at the above plot of feature importances for the top model, we can see that words that are commonly associated with disasters, like "Hiroshima", "bombing", "fire", and "killed", were given large weights. However, "California" was also commonly associated with disasters.

This is an example of where the model may be overfitting, because while California does have many disasters, more context would be needed to distinguish a non-disaster tweet about California.

An ROC curve was created for the logistic regression model, and the AUC was calculated to be about 0.8811. This indicates that the model separates the classes well, but there is some overlap between the real and fake tweets.

Figure 9. ROC curve for the best logistic regression model.



Limitations and Future Directions

One major limitation of recognizing disasters from text is that there is a much higher rate of false positives than false negatives. This makes sense, because people can use words that sound like a disaster without there actually being any disaster (e.g. "The party was a blast"). Being able to interpret these nuances of natural language correctly would require more sophisticated models than ones using a bag of words approach. In particular, using more advanced NLP techniques like syntax tree parsing and word sense disambiguation could help to more correctly identify when particular words are being used in a disaster related context, making their meanings easier for a model to interpret. Higher scoring results on Kaggle use advanced models like BERT, which have been getting state-of-the-art results in NLP tasks. These models, however, are extremely computationally expensive to train.

For future analysis, it might be worthwhile taking a deeper examination of how the one-word description and location could be better utilized in the model. Greater work could be done weighting words, exploring ideas such as tweet volume per minute using similar disaster

language. It might also help to build anomaly detection for the volume of tweets using disaster language in a specific geographic location that might clear the noise caused by infrequent but muddling instances like the 'ablaze sky' example articulated earlier. Additional accuracy may result from incorporating additional data sources that were not explored in this Kaggle submission.

One additional step we could have taken would have been to go through the dataset and preprocess the data by hand by either: fixing any obvious spelling errors or normalizing the locational data in order to make use of it. The problem with doing this is it would be very time consuming going through all of the tweets and it very simply could have a negative or zero-sum effect on the models. There is a chance that people tend to make spelling errors when a real disaster is occurring, which would be lost in our extensive preprocessing. This project really showed us the strengths and limitations of the models and techniques that we had to use.

Challenges

One of the challenges we experienced was around the training time required to build a model. Another main challenge was that a lot of important information seemed to be contained in the words which were usually removed, like stopwords. We found that the models which did not do extensive preprocessing performed better in almost all cases. Also, another challenge that we faced was not all of us were using the same programming language. Two of us used R to code while the other person used Python. This left our code fragmented and a lot of redundancies occurred.

References

Koo, Ching & Liew, Mei & Mohamad, Mohd & Salleh, Abdul. (2013). A Review for Detecting Gene-Gene Interactions Using Machine Learning Methods in Genetic Epidemiology. BioMed research international. 2013. 432375. 10.1155/2013/432375.

Pang-Ning Tan, Michael Steinbach, and Vipin Kumar (2005) Introduction to Data Mining. Addison Wesley, 2005.

Schmunk, Sergej & Höpken, Wolfram & Fuchs, Matthias & Lexhagen, Maria. (2014). Sentiment Analysis: Extracting Decision-Relevant Knowledge from UGC. Information and Communication Technologies in Tourism 2014. 10.1007/978-3-319-03973-2_19.

https://www.businesswire.com/news/home/20191230005197/en/Global-Natural-Language-Processing-NLP-Market-Size

https://www.dsayce.com/social-media/tweets-day/

https://elitedatascience.com/machine-learning-algorithms

https://www.cs.cmu.edu/~ark/TweetNLP/#pos

https://towardsdatascience.com/understanding-auc-roc-curve-68b2303cc9c5

Appendix

Table 4. Cross-validated metrics for the best model of each type. Also includes the Kaggle score for comparison to the F score.

Model	F1 Score	Kaggle Score	Accuracy	Precision	Recall
Logistic Regression	0.8388	0.8098	0.8066	0.7995	0.8824
Random Forest	0.8298	0.7955	0.7863	0.7601	0.9137
Neural Network	0.8387	0.8027	0.8061	0.7977	0.8844
SVM	0.8373	0.7935	0.8036	0.7933	0.8867
Naïve Bayes	0.8327	0.7965	0.7859	0.7509	0.9807
Average Model Score	0.8355	0.7996	0.7977	0.7803	0.9096