Identifying Real Disaster Tweets

IDS 703 Final Project | Team Green

Caleb O’Neel, Nikhil Bhargava, Junbo Guan, Rhayoung Park, Xiaohan Yang

Abstract

This research paper discusses different NLP methods and classification models used to predict whether tweets were talking about real natural disasters from a dataset from Kaggle. We processed data by tokenizing, lemmatizing and extracting features. Different NLP methods such as TF-IDF and Word Embeddings were implemented to build the model. After vectorizing the corpus, several classification models such as Logistic Regression, Naive Bayes, Decision Tree Classifiers, Random Forest Classifiers, and Multi-layer Perceptrons were used to predict the target variable. As a result, the Logistic Regression produced the best output with AUC of 0.86.

I. Introduction

Over the past decade, Twitter has become a powerful source of news for individuals and organizations. In times of emergency, Twitter can provide real-time information empowering first responders and community members to make informed decisions. With the sheer volume of tweets published to Twitter every second, sifting through them to find relevant posts can be nearly impossible. To cut through the noise, our team built a classifier that, based on a tweet, designates whether or not it is talking about a natural disaster. To construct the best possible classifier, our team explored various natural language processing and classification methods.

II. Background

The data used in this project comes from Kaggle's "Real or Not? NLP with Disaster Tweets" to enable media to monitor Twitter for information programmatically. A large portion of the dataset consists of tweets that use vocabulary that sounds like a natural disaster, but it is not. For example, the sentence "I felt a tidal wave of emotion hit me as I walked down the street" uses similar syntax to "There was an outpouring of emotion from community members after a tidal wave descended on the town last week covering the streets in 10 feet of water". Obviously, only the second example is talking about a natural disaster, but it is not as simple as looking for words related to disasters for a computer to make a classification. Working with Twitter data presents a unique set of challenges that are not present in other NLP disciplines. The tweets had misspelled words, punctuation was inconsistent, and the introduction of links, pictures, emojis, and hashtags. Applying NLP in a social media context is not a novel idea, however.

One of the primary resources we leveraged dealt with fake news classification, which looks at quotes from politicians and news articles and gives them a truthfulness classification on six scales (Wang, 2017). Like our dataset, the paper examined data from social media platforms and attempted to verify the validity of the contents. This study introduced us to the idea of "meta-data" (Wang, 2017), which we implemented in our analysis through hashtag counts and keywords.

We also implemented ideas from the study conducted by Go et al., which directly applied to our project because it analyzed sentiments on Twitter (Go et al., 2009). This has conceptual overlap with our project, as we, too, are attempting to derive the sentiment or meaning from tweets that use similar syntax to one another. We expanded upon the idea of using links in our data so the model would be able to use them as a signal indicator. We also applied this idea to hashtags in our model creation to emphasize hashtags in tweets for our model. This overlaps with the type of text we dealt with in our analysis.

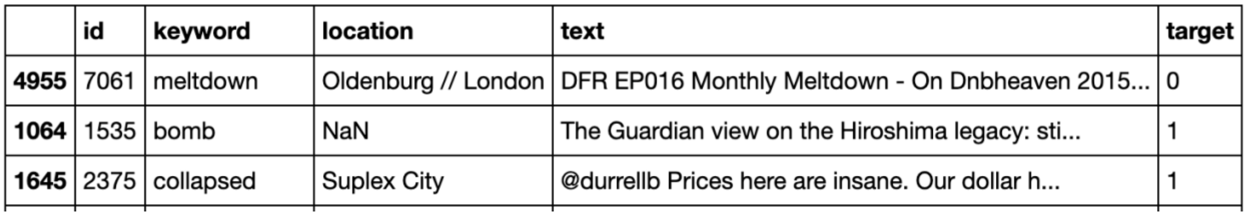
Using some of the ideas from these papers on dealing with social media text and performing feature engineering offered a solid jumping-off point for our project.

Combining these techniques for handing Twitter data with our pre-existing knowledge of language processing allowed us to work towards a better tweet classifier.

III. EDA & Feature Engineering

III-I. Data Exploration

The original training dataset consisted of five columns: unique id of each tweet, keyword, location, text, and target. The keyword column was words that sounded like they might be about natural disasters such as “earthquake” and “burning.” In addition to these data pieces, we created features for links and hashtags that appeared in our tweets.



	id	keyword	location	text	target
4955	7061	meltdown	Oldenburg // London	DFR EP016 Monthly Meltdown - On Dnbheaven 2015...	0
1064	1535	bomb	NaN	The Guardian view on the Hiroshima legacy: sti...	1
1645	2375	collapsed	Suplex City	@durrellb Prices here are insane. Our dollar h...	1

Figure 1: Snapshot of the dataset

(i) “Keyword” column

The top 15 keywords used in disaster tweets and non-disaster tweets did not differ significantly. The top 15 keywords from disaster tweets included ‘outbreak,’ ‘typhoon,’ and ‘evacuated.’ On the other hand, keywords from non-disaster tweets (‘harm,’ ‘ruin,’ ‘siren,’ ‘explode’) were also related to disasters and implied urgent situations.

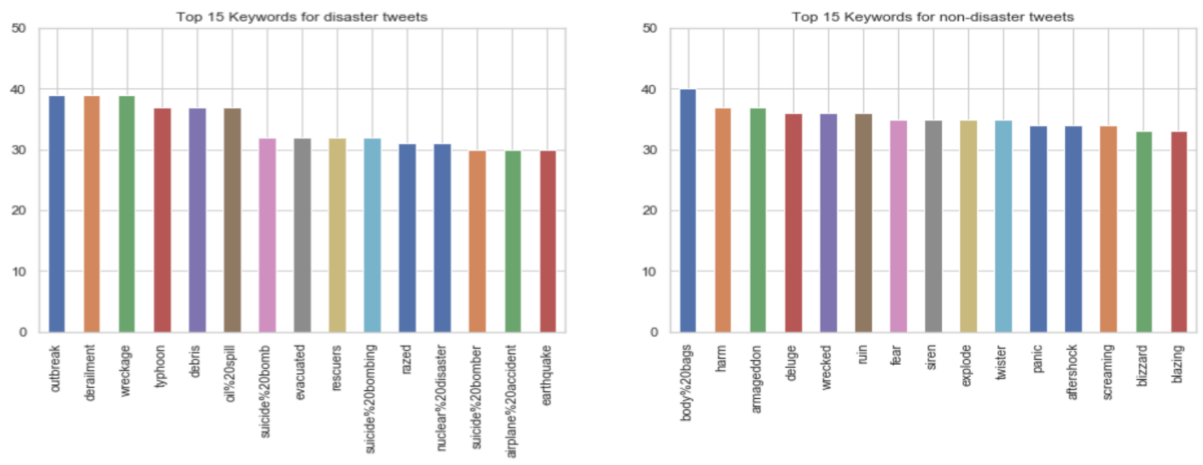


Figure 2: Top 15 keywords for disaster and non-disaster tweets

(ii) “Location” column

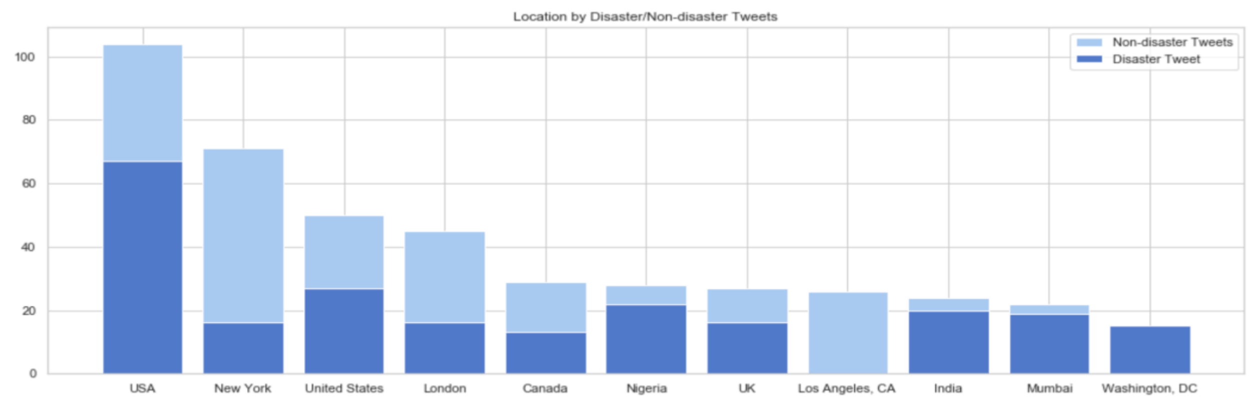


Figure 3: Distribution of locations for disaster and non-disaster tweets

Ultimately we did not use the location data as it was generally unreliable due to sparsity and data quality issues. We also believe any signal it did have might lead to overfitting, and would unlikely carry over to our test data.

(iii) Links

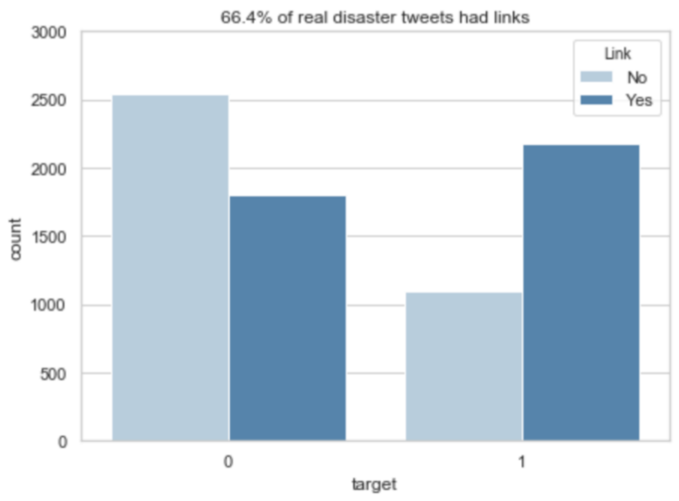


Figure 4: Distribution of tweets that had links by disaster and non-disaster tweets

The graph above shows the number of real disaster tweets with links. About 66% of real disaster tweets had links embedded in them. This seems reasonable as many tweets about natural disasters link to resource websites or news articles with updates. Therefore, we created features representing whether or not a tweet had a link and a count of links. To ensure links wouldn't be picked up as unique terms, we replaced them with strings such as "__link__" in the text of the tweet using RegEx expressions.

(iv) Most frequently used words for disaster/non-disaster tweets

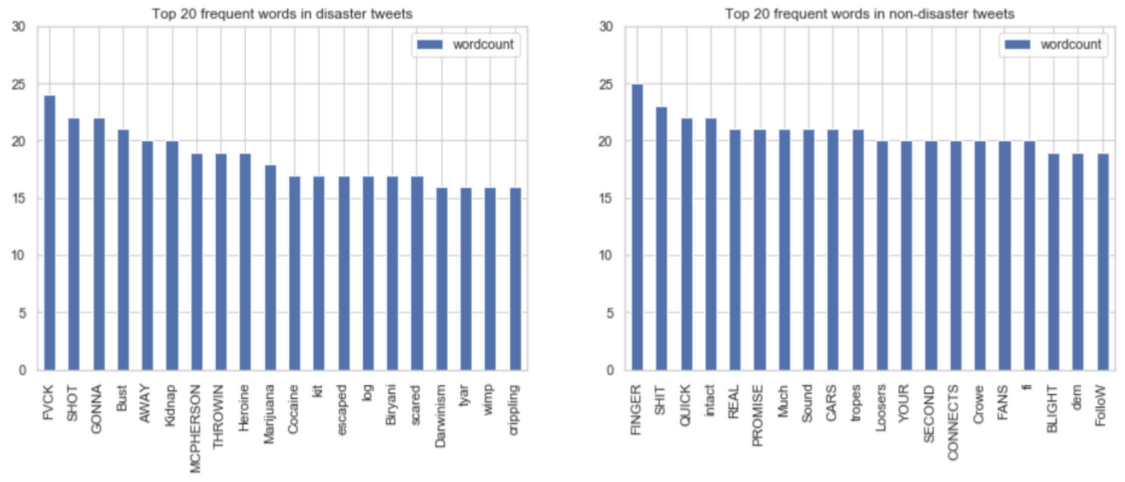


Figure 5: Distribution of words in corpus of tweets

For disaster tweets, 6 words from the top 8 frequent words are in capital letters (e.g. ‘FVCK,’ ‘SHOT’). Considering that people tend to write everything in capital letters to represent shouting or screaming, the top 8 frequent words being all capital letters show the urgency of the situation. Also, drugs were mentioned in disaster tweets, whereas there were no drug related words in non-disaster tweets. In general, the overall sentiment of words in disaster tweets was much stronger. To represent this “sentiment” in our model, we created features containing the percentage of words and characters as lowercase, uppercase, and title case.

(v) Hashtags

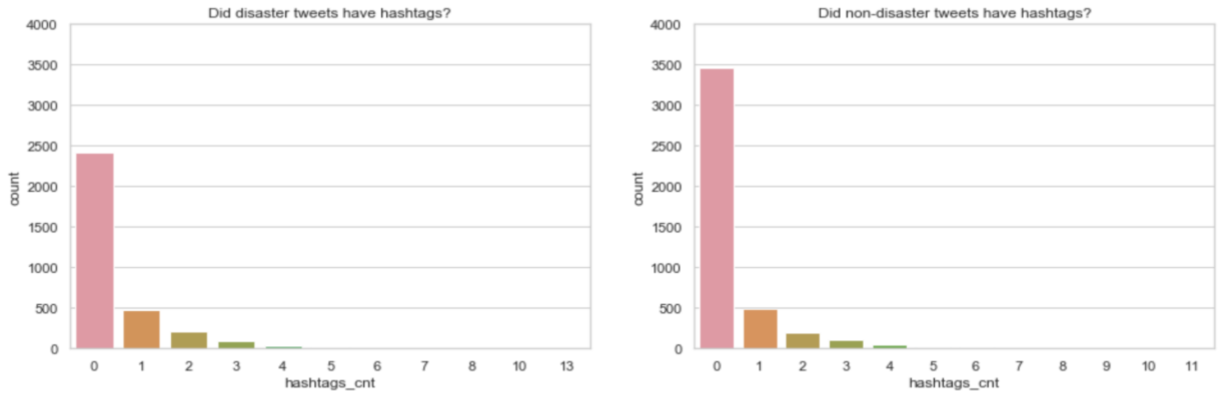


Figure 6: Distribution of tweets that had hashtags by disaster and non-disaster tweets

With hashtags, several methods were tried to unlock their predictive signal because there are potential issues with leaving hashtags as normal text in the tweet. First, a tokenizer looks at “fire” and “#fire” as different words. Second, hashtags are often utilized to emphasize a word, and we theorized that they could carry extra importance to the meaning of the tweet.

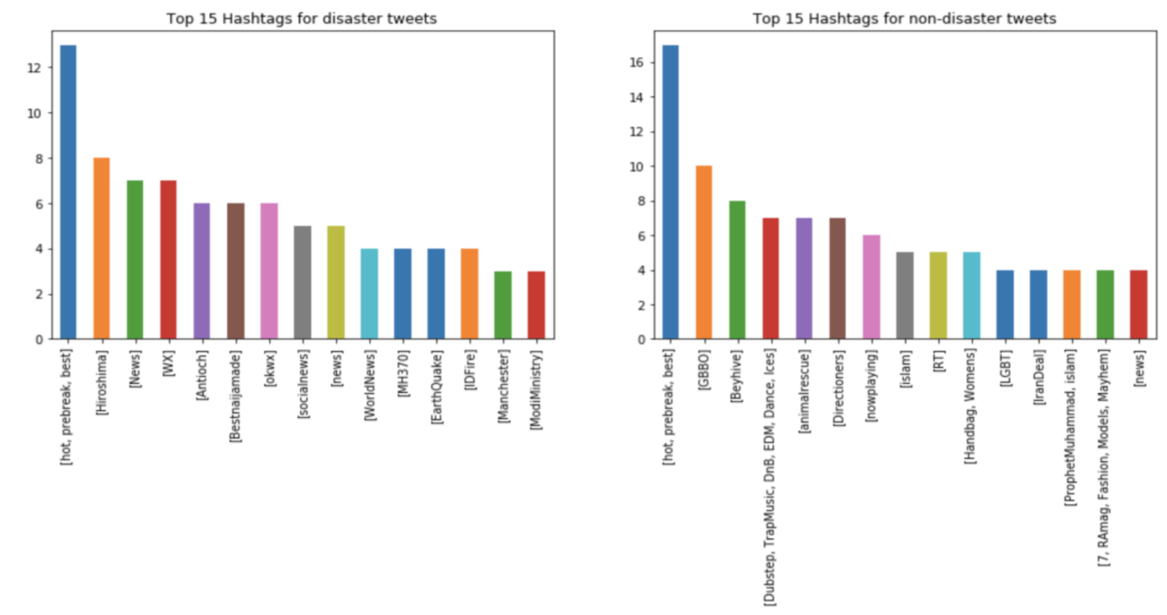


Figure 7: Top 15 hashtags for disaster and non-disaster tweets

As a result of processing the hashtag column, we observed that more than half of the tweets had no hashtags for both disaster and non-disaster tweets. For disaster tweets, it had hashtags related to news service (‘News,’ ‘socialnews,’ ‘Worldnews’) whereas non-disaster tweets had random hashtags. Using hashtags as model features were evaluated as meaningful (Anna, et al., 2014), and thus we incorporated whether or not hashtags were present and the count of hashtags in our model.

III-II. Data Processing

We used the stop words from NLTK as the baseline for data cleaning and added some frequently used symbols and characters. Then we lowercase each word and removed stopwords from each tweet. For tokenization, we determined that the spaCy and Gensim tokenizer were the best for our dataset and were what we used. Additionally, we used RegEx expressions to clean tweets into a proper format.

IV. Methodology

To translate tweets into machine-interpretable features, we tried two different NLP techniques, TF-IDF and word embeddings, to create two different training sets for our classification models.

IV-I. NLP Methods

(i) TF-IDF and Singular Vector Decomposition (SVD)

TF-IDF is an NLP technique used to emphasize terms or words relatively rare in a corpus but often repeated in each document. Therefore, by applying TF-IDF to the corpus of tweets, we aimed to capture the importance of specific words discussing natural disasters to predict natural disasters. For our dataset, TF-IDF resulted in a sparse matrix with a little under 13,000 columns representing each term in the corpus and each row representing their weight in the tweet.

Since TF-IDF produced such a large vector space, dimension reduction techniques such as SVD and Principal Component Analysis (PCA) were performed to extract the strongest signals from the features. Ultimately, after varying the number of components for each method, it was determined that these methods didn't improve any of our models' F1 or AUC scores, and as a result, were not used in the training of our models.

(ii) Word Embeddings

Word embeddings are an NLP technique used to represent words in a "multi-dimensional" vector space. Since our language corpus' size was not nearly large enough to create our own word embeddings, we used spaCy's pre-trained English (large) word embeddings to turn each valid word in a tweet into a 300-dimensional word embedding. It is important to note that when creating this feature space, words were not lemmatized to preserve the spelling of each word and our ability to represent each word as an embedding.

Finally, the mean of all the word vectors in each tweet was taken to represent each tweet into its own unique 300-dimensional space called a "tweet embedding." Using multiple word embeddings is a common approach used to capture the latent and semantic vector space of one to many sentences. This is appropriate to use when dealing with tweets because people are capped at a maximum of 240 characters per tweet, meaning they can only fit a few sentences in a tweet at most.

IV-II. Classification Methods

Once every tweet was represented in both datasets' appropriate feature space, we concatenated the features engineered earlier to both training datasets. This data was then used as the training set for five different classification models: Logistic Regression, Naive Bayes Classification* (not used for the word embedding training set), Decision Trees, Random Forest Classifier, and a Multi-Layer Perceptron (MLP) Neural Network classifier. Surprisingly, only marginal performance improvements were found across a few models. This may have been due to the engineered features' density and size compared to the TF-IDF and tweet embedding matrices. Ultimately, these features were included in our final models due to slight performance improvements.

For each dataset, the Precision, Recall, F1, Average Precisions, and AUC scores for each model were calculated. However, the F1 and AUC scores were the metrics optimized to produce the best results and compare classification models against one another. These scores were used due to the F1 score's ability to evaluate imbalanced classes (like our dataset), as it is the weighted average of the precision and recall. The

AUC is another metric that helped us understand how well our model can distinguish between disaster and non-disaster tweets.

(i) Logistic Regression

Logistic regression is a binary classification model that models the probabilities between zero and one.

(ii) Naive Bayes Classifier

Naive Bayes Classifier is a classification model based on Bayes' Theorem, built on the assumption that each feature is not related to other features. This model was only applied to the TF-IDF dataset because it cannot be applied to the tweet embeddings due to negative values. With the baseline model, several 'ngram_range' parameters were applied. After testing which n-gram combination gave the best score, n-grams with a range of 1-3 were selected to be a feature for this model.

Table 1: Key metrics for different ‘ngram_range’

	(1,3)	**(1,4)**	**(1,5)**	**(1,6)**	**(1,7)**	**(1,10)**	**(1,20)**
Precision	0.85	0.86	0.86	0.86	0.86	0.86	0.86
Recall	0.62	0.60	0.60	0.59	0.59	0.59	0.59
F1	0.72	0.71	0.70	0.70	0.70	0.70	0.70
Average Precision	0.68	0.68	0.68	0.68	0.68	0.68	0.68

(iii) Decision Tree Classifier

A decision tree classifier is a non-parametric classification model that optimizes on a specific criteria and does not assume a particular distribution in the training dataset. Ultimately, the model uses rules derived from the features to classify the target variable based on a predetermined tree maximum size. For our datasets, the optimal maximum depth of the decision tree to make predictions was five, as anything below didn’t capture the full context of the tweet, and anything higher seemed to be over-fitting to our training set.

(iv) Random Forest Classifier

A Random Forest Classifier is also a non-parametric model that consists of multiple individual decision trees that operate as an ensemble. They operate by creating a number of trees, N, on bootstrapped datasets containing a smaller sample of predictors. They also attempt to de-correlate trees to over-fit them on the training set and prevent certain variables from dominating tree splits. The number of estimators was set to 100, and the minimum number of samples needed to split an internal node was 2.

(v) Multi-Layer Perceptron (MLP) Classifier

An MLP Classifier is a feedforward neural network designed to make classification predictions based on particular activation functions. The activation function used for this problem was the rectified linear unit (ReLU).

V. Results

V-I. TF-IDF

Each model's performance metrics fitted on the TF-IDF matrix plus the additional features are listed in Table 2. The ROC curves for each model on the training set can be seen in Figure 8 below. Surprisingly, logistic regression had both the highest F1 and AUC scores, although not by much. The decision tree classifier was the only model with significantly worse performance results with an F1 score of 0.60 and an AUC of 0.68. This was probably due to the classifier being overfitted on the training data, even with a max tree size of five. Although the MLP classifier did decently, it didn't outperform logistic regression, potentially due to the relatively small size of the training set.

Table 2: Precision, Recall, F1, Average Precision, and AUC scores for TF-IDF training set

	Logistic Regression	**Naive Bayes**	**Decision Tree Classifier**	**Random Forest Classifier**	**MLP Classifier**
Precision	0.79	0.85	0.60	0.88	0.73
Recall	0.71	0.62	0.61	0.59	0.72
F1	0.75	0.72	0.60	0.71	0.73
Avg Precision	0.68	0.68	0.53	0.68	0.65
AUC	0.86	0.85	0.68	0.85	0.83

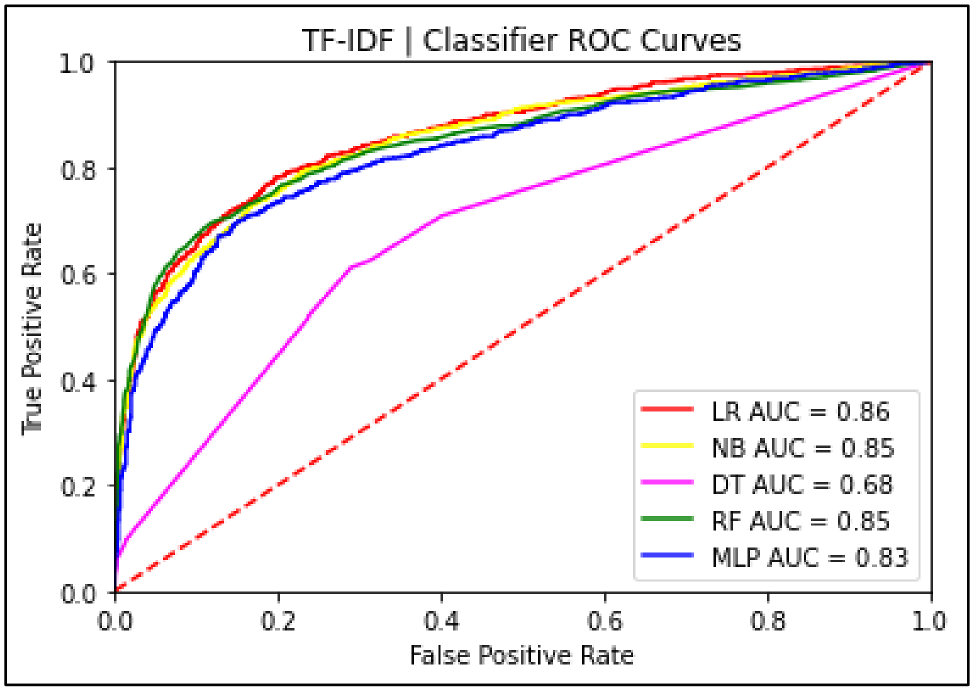


Figure 8: ROC curves for models using TF-IDF training set

V-II. Word Embeddings

Each model's performance metrics fitted on the tweet embedding matrix plus the additional features are listed in Table 3. The ROC curves for each model on the training set can be seen in Figure 9 below. Overall, the logistic regression model performed the best again and fairly well with an F1 score of 0.75. The model's AUC score of 0.86 suggests that the model is good at distinguishing between disaster and non-disaster tweets. Although the Decision Tree Classifier was once again the worst performing model, it did a better job than the same model trained on the TF-IDF training set with a drastic improvement of 0.03 to the F1 score and 0.09 to the AUC.

Table 3: Precision, Recall, F1, Average Precision, and AUC scores for word embeddings training set

	Logistic Regression	**Decision Tree Classifier**	**Random Forest Classifier**	**MLP Classifier**
Precision	0.78	0.70	0.82	0.73
Recall	0.73	0.58	0.65	0.73
F1	0.75	0.63	0.72	0.73
Avg Precision	0.68	0.58	0.68	0.64

AUC	0.86	0.77	0.85	0.85

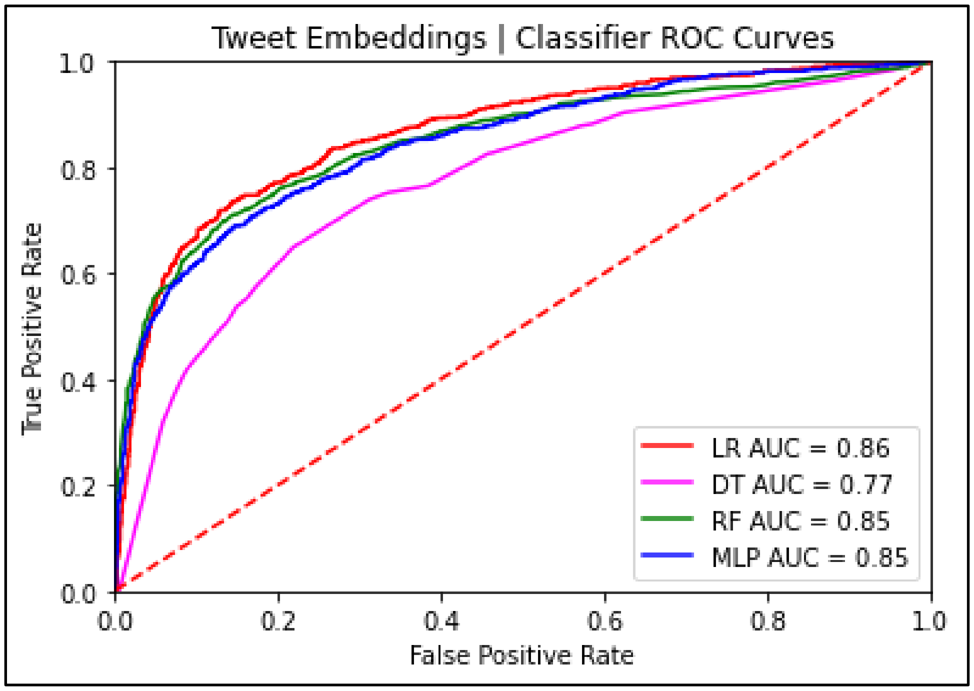


Figure 9: ROC curves for models using tweet embeddings training set

VI. Conclusion

This study aimed to identify real disaster tweets from non-disaster tweets using several NLP methods and machine learning models. From our results, the logistic regression models built on either training set performed the best with an AUC score of 0.86 and an F1 score of 0.75. However, the Naive Bayes, Random Forest, and MLP models were not far behind with F1 and AUC scores within 0.3 points on both training sets. Ultimately, the different training sets didn't prove to improve our ability to differentiate between the two classes in the target variable, but did improve the predicting power of models other than linear regression.

Although our model performed pretty well in identifying disaster tweets from non-disaster tweets, there were a few limitations in our analysis. Additional features associated with tweets such as retweet counts and like counts could be a useful index in differentiating tweets. Additionally, since Twitter data is unique due to the lack of traditional English structure used, some translations of words to embeddings were lost because words were not contained in spaCy's pre-trained embeddings. This could've potentially left out keywords from being included in each tweet embedding, and decreased model performance.

Taking our research and integrating it into a Twitter API might be useful future expansion for our work. An automated system for identifying real disaster tweets could be meaningful to social media users affected by natural disasters by providing them with relevant and useful real-time information on their Twitter feeds and provide value to first responders and media outlets with more information.

VII. References

Castillo, Carlos, Marcelo Mendoza, and Barbara Poblete. "Information credibility on Twitter." Proceedings of the 20th international conference on World wide web. 2011.

Gao, Zheng, and John Wolohan. "Fast NLP-based Pattern Matching in Real Time Tweet Recommendation." TREC. 2017.

Go, Alec & Bhayani, Richa & Huang, Lei. (2009). Twitter sentiment classification using distant supervision. Processing. 150.

Hunt, Kyle, Puneet Agarwal, Ridwan Al Aziz, and Jun Zhuang. "Fighting Fake News during Disasters." 6 Jan. 2020. Web. 17 Nov. 2020.

Kai Shu, Amy Sliva, Suhang Wang, Jiliang Tang, Huan Liu, New York, NY, United States
Fake News Detection on Social Media: A Data Mining Perspective Association for Computing Machinery September 2017

Li, Hongmin, et al. "Comparison of word embeddings and sentence encodings as generalized representations for crisis Tweet classification tasks." en. In: New Zealand (2018): 13.

Soroush Vosoughi, Deb Roy, Sinan Aral, The spread of true and false news online, Science 09 Mar 2018: Vol. 359, Issue 6380, pp. 1146-1151 DOI: 10.1126/science.aap9559

Stavrianou, Anna, et al. "NLP-based feature extraction for automated tweet classification." Interactions between Data Mining and Natural Language Processing 145 (2014).

Wang, William. (2017). "Liar, Liar Pants on Fire": A New Benchmark Dataset for Fake News Detection. 422-426. 10.18653/v1/P17-2067.

Yadav, Omprakash, Abhay Shinde, and Omkar Palkar. "Tweet Summarization using NLP and Sentiment Analysis." (2008).