Determining Disasters from Tweets

Introduction:

We live in a day and age where 79% of the United States population has a social media profile of some kind. The average time a day spent on social media alone is almost three hours. At first glance that doesn't seem like an outrageous number, but let's look into it. On average, people sleep for eight hours a day and work/study for another eight hours a day. That's 16 hours right there, leaving only eight total hours left in the day. That means the average person is spending over a third of their remaining time on social media.

Social media has revolutionized the way in which people communicate with others. Social media applications like Facebook and Twitter might have started off as a way to connect with friends and family, but have now evolved to the point where it is used as a platform for a variety of objectives like networking ,marketing, and sharing news.

News on social media platforms spread like a fire, quickly and chaotically and that is because of how interconnected everyone is. This is especially the case on a platform like Twitter, where Twitter will send out an alert to all users when a specific 'tweet' is trending.

I believe it's because of social media platforms, that the recent news on the Coronavirus was able to be relayed to a majority of the global population in such a quick manner, as well as explain/demonstrate the impact of the virus on the entire population. What I want to do is leverage twitter data and build a model that takes the Twitter data and categorizes them by 'disaster tweets and 'not disaster tweets'. With this we can save lives, be more efficient in the way first responders react in emergency situations, and how the public can better prepare themselves.

What is the Problem I Want to Solve:

1) Response Time: I believe if there is a constant lookout for 'disaster events' on social media platforms like Twitter, response time of first responders can be greatly reduced.
2) Awareness: This gives awareness to people in the area of a disaster event letting them know how it could affect them and what precautions they should take.
3) Real Time Updates: Twitter users near the 'disaster event' will most likely keep updating on the situation of said event. This will help first responders know of any changes in situation so they are not going in blind.

Who are my Clients, and Why do they Care about this Problem? :

I believe there are two clients who will benefit from this problem being answered:

1. First Responders
    a. As I mentioned earlier, if first responders are able to react quicker to an event, it could potentially save lives.
2. General Public
    a. The general public needs to be made aware of anything that could affect their daily lives. By getting news of a disaster event unfolding they can change their plans that otherwise would have ended up getting them caught in the middle of the disaster event.

What Data am I Using? How will I Acquire the Data :

The dataset I am using is from a previous Kaggle competition. It is called Real or Not? NLP with Disaster Tweets. The Kaggle dataset contains a variety of tweets and if they are designated as a disaster event tweet.

How will I Solve This Problem:

The way I will solve this problem is by taking a bag of words or a TfIdf approach to the twitter data. My first step will be to preprocess the data and remove stopwords and tokenize and vectorize the remaining words. I'll also perform stemming/lemmatization to increase accuracy. Finally, I will train and fit on 70% of the Twitter data and test and predict on the remaining 30% using multiple classification algorithms and determine the one(s) that perform the best.

1) Initial Dataset: The data I am using is taken from one of the Kaggle competitions. It is in the form of a .csv file.

2) Pre-Process Dataset: This step involves cleaning the data by getting rid of any missing values, perform feature engineering, remove stopwords,

3) Then I will perform Exploratory Data Analysis (EDA), allowing for a better understanding of relationships between variables. This step is important because the visual representation helps with seeing what type of relationships there are between features and text.

4) After making sure we have the input variables and output variable stored in X and y respectively, we split the data into a training set and test set. Since this data involves text information, we need to transform the data into a sparse matrix of words in order to perform our learning algorithms.

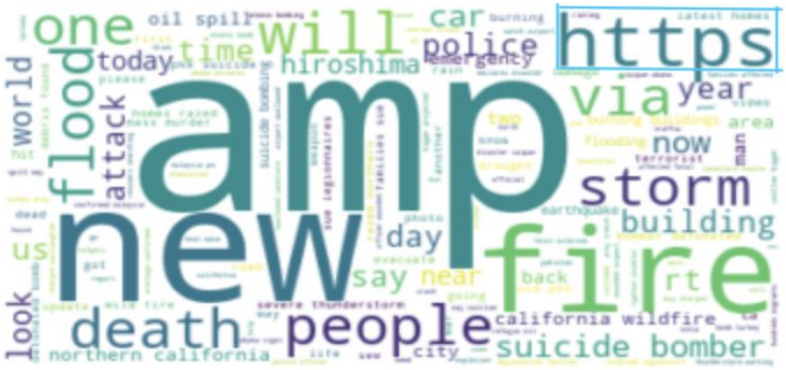
5) Finally, we need to evaluate our models' performance to figure out which model performed the best. Since this is a text classification problem, the evaluation metrics to look at would be accuracy, precision and recall.

Data Preprocessing and EDA:

First, I took a look at the first 50 elements in the data. In doing so, I found there were quite a few entries that had a link within the text. I created a word cloud to see the most common words that occur in both a disaster tweet and a non disaster tweet. I saw that http was a word that was common in both and using regular expressions, I removed any url links from the Twitter data.

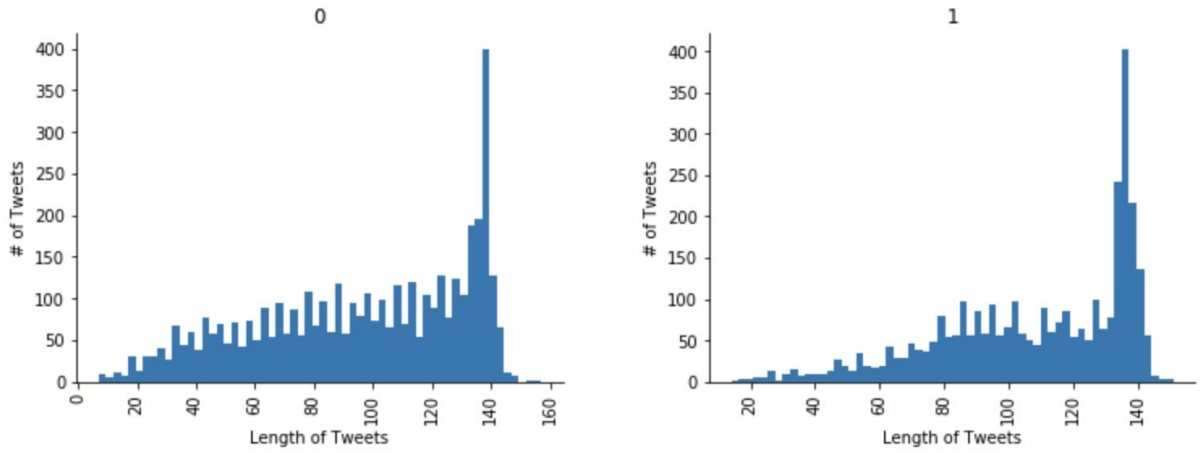
Non-Disaster Tweet WordCloud

Disaster tweet WordCloud

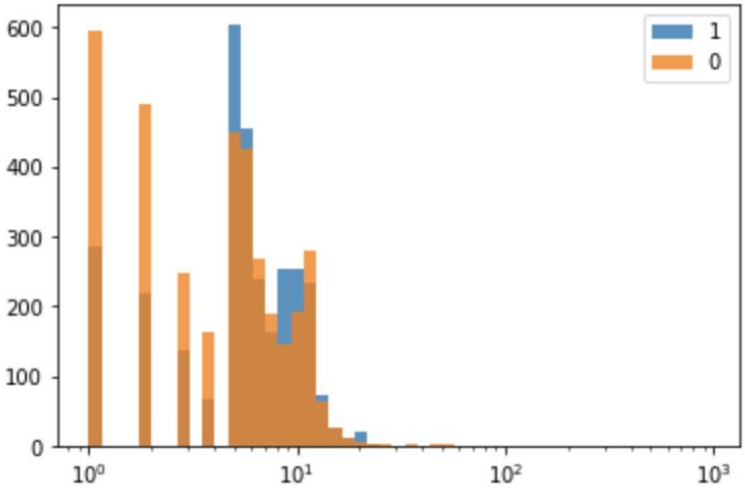


Luckily, there were no missing text or missing target values in the data. But relying on only the text and target variables is not enough to make an accurate model. I made new features that I believed might help in predictions: length(length of tweet), punctuation(number of punctuation in tweet), exclaim(number of exclamation marks in tweet). I also performed topic modeling and created a new column that separated tweets into different topics.

Length of of non disaster and disaster tweets

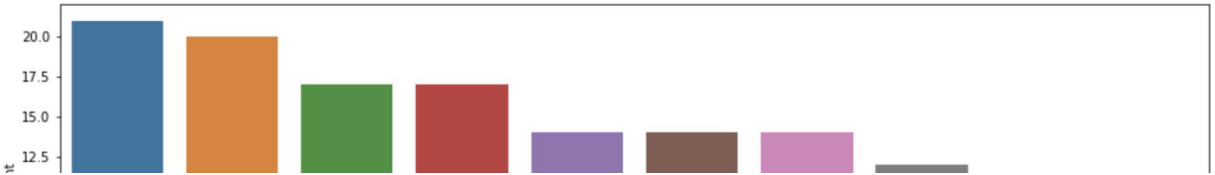


Overlap of Lengths of Tweets

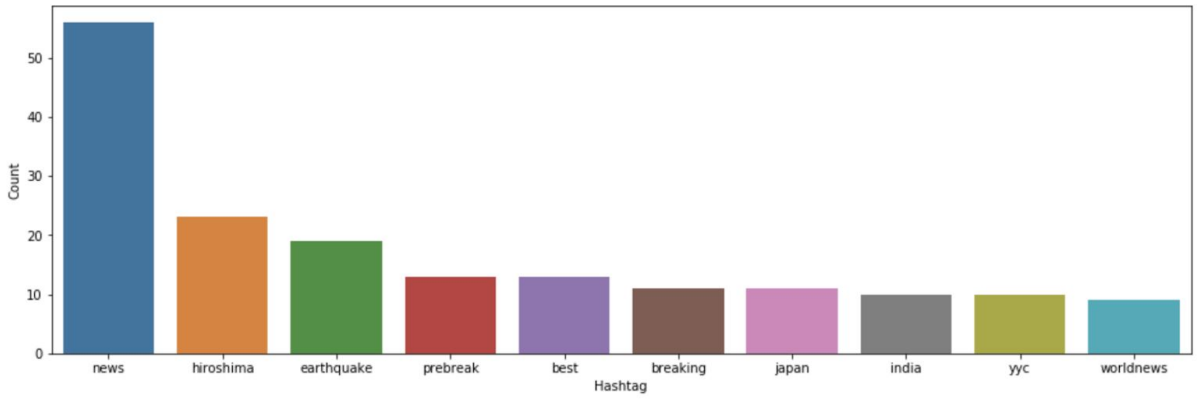


Next, I looked at words following the hashtag, to determine any information of value. This is what I found:

Non Disaster tweet



Disaster tweet

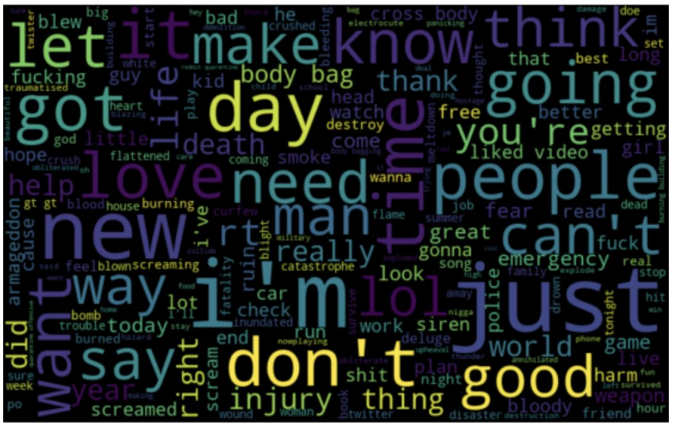


For each type of tweet, these are the 10 most common words following a hashtag.

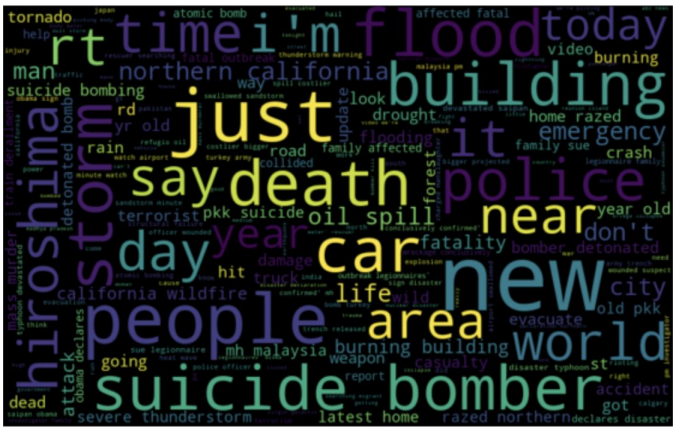
ML Model:

I found that the most accurate model was when I performed TfIdf on my twitter text data. This involved calculating the term frequency and inverse document frequency in the form of a sparse matrix of all words, each word giving a value depending on if the word is in the tweet. But before that, I tokenized each word in every tweet, performed stemming, which got the stem of each word, and removed all the stopwords, reducing the number of irrelevant words. The following word clouds are a result of removing stop words.

Non-Disaster WordCloud



Disaster WordCloud



Once the text is ready for analysis, it is time to split the data into a train and test set. Then, we take the training set and create a sparse matrix of all the words in the training set which is the format we need to train on. We do the same process on the test set, and then predict on the test set. The metrics used to evaluate this model are accuracy, F1, precision, and recall, the latter three can be taken from a confusion matrix.

I started off building three models. After the tokenizing and vectorizing process, I trained the vectorized model on three different learning algorithms: LinearSVC, Naive Bayes Classifier, and Logistic Regression algorithm.

Linear SVC model:

	precision	recall	f1-score	support
0	0.81	0.84	0.83	1446
1	0.77	0.73	0.75	1067
avg / total	0.79	0.80	0.79	2513

Naive Bayes Classifer:

	precision	recall	f1-score	support
0	0.80	0.90	0.85	1446
1	0.84	0.69	0.76	1067
avg / total	0.81	0.81	0.81	2513

Logistic Regression:

	precision	recall	f1-score	support
0	0.81	0.91	0.85	1318
1	0.85	0.70	0.77	966
avg / total	0.82	0.82	0.82	2284

We see that the Logistic Regression model performed the best all around. My next step is to see if I can improve this model in any way. What I did was take the top words from each tweet and created a new TFIDF vector with only the top five words from each of the tweets. In doing so, we slightly improved the precision while slightly decreasing the recall.

	precision	recall	f1-score	support
0	0.92	0.79	0.85	1539
1	0.67	0.86	0.75	745
avg / total	0.84	0.81	0.82	2284

Assumptions and Limitations:

1. This data contains only text information. I had to create more features myself, but none had a big correlation on the target variable. I assume if we had found another feature that had a great bearing on the target variable, the precision and accuracy of the model would be a lot higher.
2. The assumption for this was that all tweets were different and there were specific words that corresponded to a tweet being classified as a disaster tweet or a non disaster tweet.
3. Our tweets covered a wide range of topics so it limited us in getting too much information from topic modelling.

Future Work:

There are many things you can do to classify tweets. Here I relied mainly on vectorizing using a TFIDF model. However, with more time, some things I would like to add in order to get a more efficient model would be doing sentiment analysis, trying a stacked model approach, doing some sort of event recognition, and trying to reduce the number of words used as features.

Right now it is difficult for the general public to interpret or decipher these results. Since I said previously that this project is tailored to help first responders and the general population take action on disasters quicker, I wanted to create some sort of

application that allows the public to use what I've implemented. With enough time and resources I would have liked to work with a software developer to deliver this app.

Conclusion:

In this project I set out to classify if a particular tweet is a disaster tweet (e.g. "There is a shooting on A Street") or a non disaster tweet (e.g. "This concert is awesome"). I wanted to build a foundation that allowed first responders to react quicker to disaster events. The process I took allowed me to build a model that classified tweets correctly with > 82% accuracy.

During this project, I learned that the bulk of NLP problems are the data preprocessing steps and feature engineering. Being able to extract information from text is critical in order to build a model with a high accuracy rating. In this project in particular, I tried to create many different types of features to see if they were strong drivers behind classifying disaster tweets. Once we establish everything suggested in the 'Future Work' section, I believe this project will be ready for production.

What Will be My Deliverables?:

I will include my Jupyter Notebook with all my code and comments, a paper, or if time allowing a blog post.