Detecting Tweets Relating to Disasters Using Natural Language Processing

Mert Iren
4 September, 2022

Abstract

While Twitter can be a great tool to access lots of information about many different topics at once, it can be hard to parse through this data due to the nature of the vast platform; this is especially problematic if people want to gain information about a topic quickly, such as about an ongoing disaster. Using Kaggle's "Natural Language Processing with Disaster Tweets" competition, we can attempt to train models with the text data of a tweet to attempt to classify disasters. Throughout this investigation, the baseline models that were built were a support vector machine, a random forest, and a logistic regression model. Additionally, a transformer model was built with BERTweet with an accuracy of 85%, precision of 84%, recall of 81%, and an F1 score of 83%. Although the transformer model performed better than the baseline models, the difference is not very significant, indicating that it's hard to achieve much higher scores with this dataset

Word Count: 1719

Introduction

In the modern day, social media allows us to access vast amounts of information shared by many people. As a result, it can be a valuable tool to learn more about ongoing events, especially if this knowledge may be crucial such as in an ongoing disaster. Not only can this information be used by anyone wanting to gain insight or news on a recently-occured disaster to report on it, but this information can also be used by first-responders, to know what to expect in a disaster situation and since some tweets also contain location data. However, for the same reasons that make Twitter such an accessible source for a wide variety of information, it is also extremely difficult to parse through this information, especially if you don't know what exactly you're looking for. Currently, people can use Twitter for this purpose by browsing through a single hashtag or searching up key words relating to an event, yet not only would it take a long time to do this, but people may not post tweets under the same hashtag, and search terms could vary especially when the circumstances of an event are unknown.

By using natural language processing we can attempt to categorize tweets based on whether they relate to a disaster (natural or otherwise) or not in order to significantly reduce the number of tweets that people would have to parse through. As a result, it would be easier for these people to gain information on this topic, resulting in more accurate news reports and more prepared first-responders, allowing everyone to do their job more effectively. In this paper, I explore patterns in the dataset used, methods for processing data and training models, and the results obtained from the models. Therefore, I present this paper's research question, RQ1.

RQ1:How Effectively Can a Model Predict Whether a Tweet Relates to a Disaster or Not?

Data

The data for our model is taken from Kaggle's "Natural Language Processing with Disaster Tweets" competition and contains the text data of a tweet, possible keywords, the location data (if included), and a label stating whether or not the tweet relates to a disaster. However, for our models, only the text data and the label will be used.

Exploratory data analysis was conducted in order to determine how the NLP models will be built

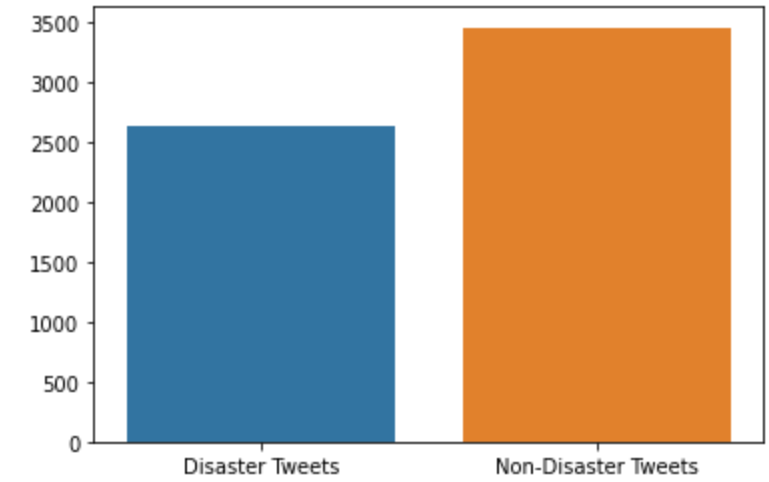


Figure 1: Disaster and Non-Disaster Tweets

By analyzing the number of both disaster and non-disaster tweets, it can be seen that both categories seem to have an approximately balanced number of tweets within them, meaning that there wouldn't need to be many steps taken in order to ensure that the model isn't heavily biased towards predicting one or the other.

In order to determine whether stopwords should be removed or not from the data, the most common words can be found in a copy of the data with and without stopwords for tweets that do and don't relate to disasters. With stopwords, the most common words are words such as "the", "I", "a", "and", and "in", which carry little to no semantic meaning for both datasets. Without them, the most common words are shown below.

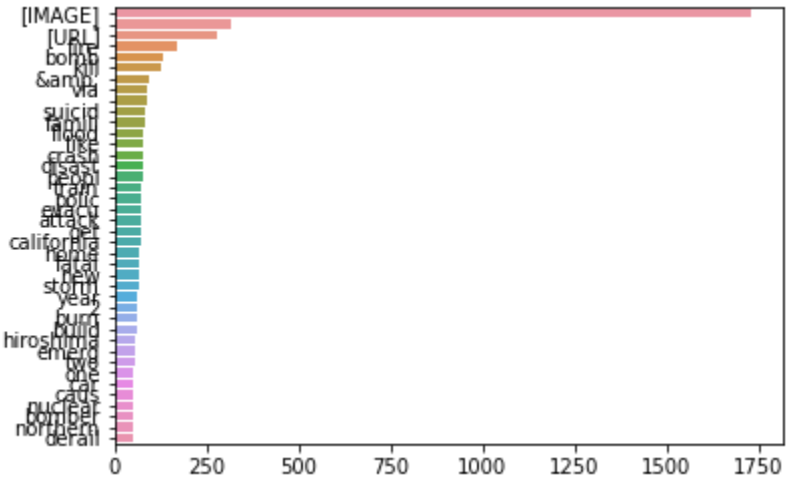


Figure 2: Common Words for Disasters

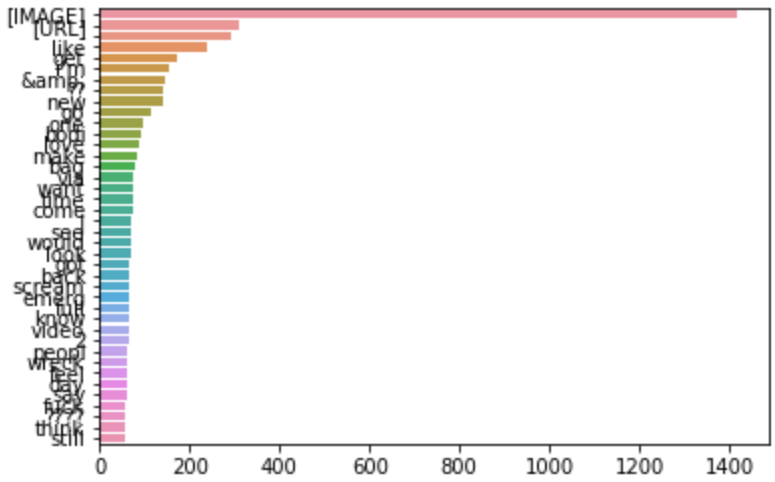
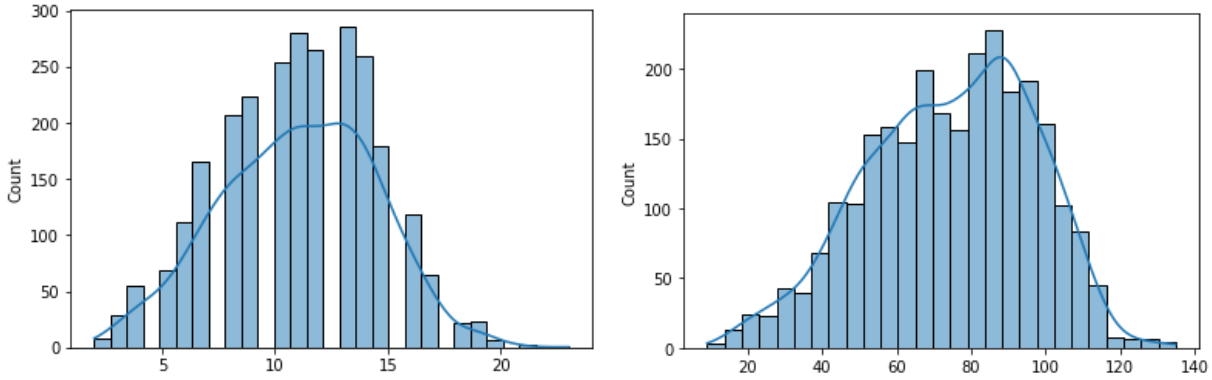


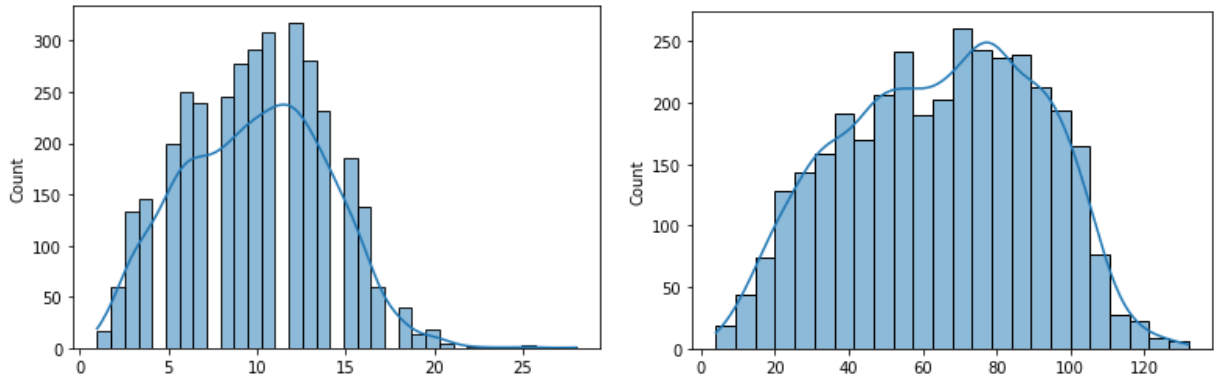
Figure 3: Common Words for Non-Disasters

So, for the models, stopwords will be removed since they may confuse our models without providing any additional meaning to tweets.

Additionally, the character and word count for both disaster and non-disaster tweets was also found as shown below.



Figures 4 & 5: Word and Character Count for Disaster Tweets



Figures 6 & 7: Word and Character Count for Non-Disaster Tweets

Figures 4-7 demonstrate to us that non-disaster tweets tend to contain both less words and characters than disaster tweets. This information can also be used when building the model.

Splitting the Data

The data goes through a train-test split using the Sci-Kit Learn Python library. The test size is 20% of the entire dataset and the random state for the split is 1

Pre-processing the Data for Baseline Models

The data was pre-processed using Reg-Ex in order to remove all image links, and replace them with the tag '[IMAGE]', to remove all other URLS and replace them with the tag '[URL]' and all hashtag symbols were removed. Additionally, the words were stemmed using the nltk library's Porter Stemmer, so that the model can recognize words that come from the same stem.

Pre-processing the Data for Transformer Models

For the transformer models, the data was pre-processed simply by tokenizing the tweets with the BERTweet tokenizer, then converting these encodings into a torch-compatible dataset. Additionally, 20% of the training set was allocated to validation set, (or 16% of the entire data)

Baseline Models

Baseline models were built to see how effectively our data could be analyzed with a simple model. Each model was evaluated using 5 fold cross validation, in order to see how they perform in an independent dataset. To train the baseline models, they were given TF-IDF vectors of each of the tweets as well as whether or not they were a disaster, represented by a 1 or 0.

Support Vector Machine:

The hyperparameters used for Support Vector Machine are available in table 1.

Table 1. Support Vector Machine Hyperparameters

Parameters	Values
C	0.1, 1, 10, 100
Gamma	1, 0.1, 0.01, 0.001
Kernel	RBF, Sigmoid, Poly
Degree (for poly kernel only)	1, 2, 3

Logistic Regression:
The hyperparameters used for Logistic Regression are available in table 2.

Table 2. Logistic Regression Hyperparameters

Parameters	Values
C	0.1, 1, 10, 100
Penalty	L2, None
Tolerance	0.00001, 0.0001, 0.001, 0.01, 1

Random Forest:
The hyperparameters used for Random Forest are available in table 3.

Table 3. Random ForestHyperparameters

Parameters	Values
Number of Estimators	10, 100, 1000
Max Features	Log2, Square-Root, None

XGBoost:
The hyperparameters used for XGBoost are available in table 4.

Table 4. XGBoost Hyperparameters

Parameters	Values
Number of Estimators	100, 1000
Learning Rate	0.05, 0.1, 0.2
Max Depth	3, 6, 9

Transformer Models

The transformer models were given the BERTweet encodings as well as whether the tweet is a disaster or not with the following parameters

The hyperparameters used for the transformer model are available in table 5.

Table 5. BERTweet Hyperparameters

Parameters	Values
Epochs	3, 4, 5
Train Batch Size	16, 32
Weight Decay	0.01, 0.3
Learning Rate	2e-5, 3e-5, 5e-5

Results

The best* parameters and the results for each of the tested models are as follows:

Table 6. Transformer Results

Model Name	Parameters	Accuracy	Precision	Recall	F1
Support Vector Machine	C = 100 Gamma = 0.1 Kernel = Sigmoid	0.79	0.79	0.68	0.73
Logistic Regression	C = 2 Penalty = l2	0.79	0.78	0.70	0.74
Random Forest	Max Features = Square-Root Number of Estimators = 1000	0.79	0.83	0.63	0.72
XGBoost	Max Depth = 6 Learning Rate = 0.2 Number of Estimators = 1000	0.79	0.79	0.68	0.73
BERTweet	Epochs = 3 Training Batch Size = 16 Weight Decay = 0.3 Learning Rate = 5e-05 Evaluation Batch Size = 8	0.85	0.84	0.81	0.83

**models are evaluated on their F1 score*

It's clear from our results that the BERTweet model functions the best out of all of the models built, consistently doing better across all of the performance metrics defined above

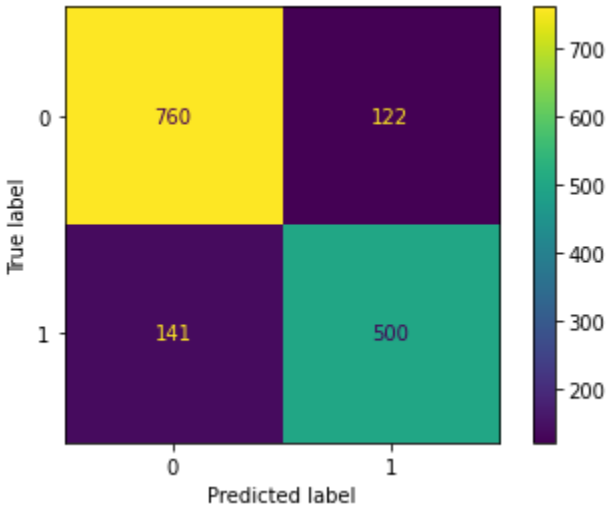


Figure 8: BERTweet Model Confusion Matrix

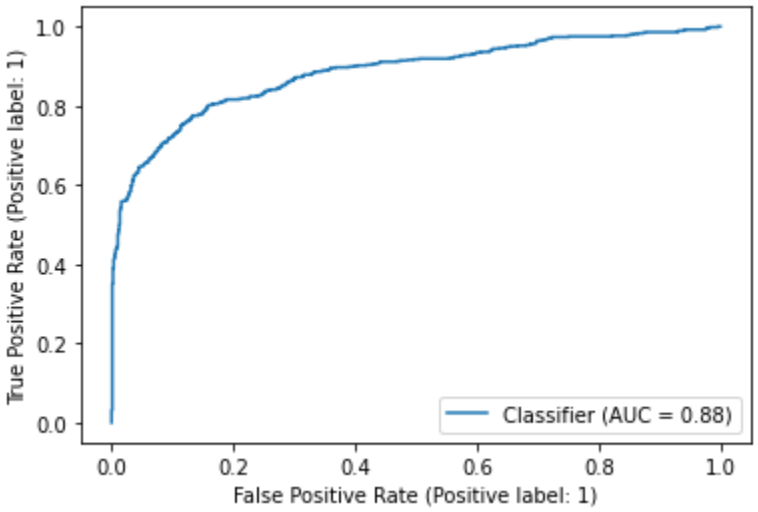


Figure 9: ROC Curve

We can gather information about the effectiveness of the model at different thresholds using the ROC curve. By finding the point with the greatest difference between the true positive rate and the false positive rate, we can see that the threshold of 0.293 is the best. Additionally, the area under the curve gives us an understanding of the overall performance of the model, which in this case is 0.88.

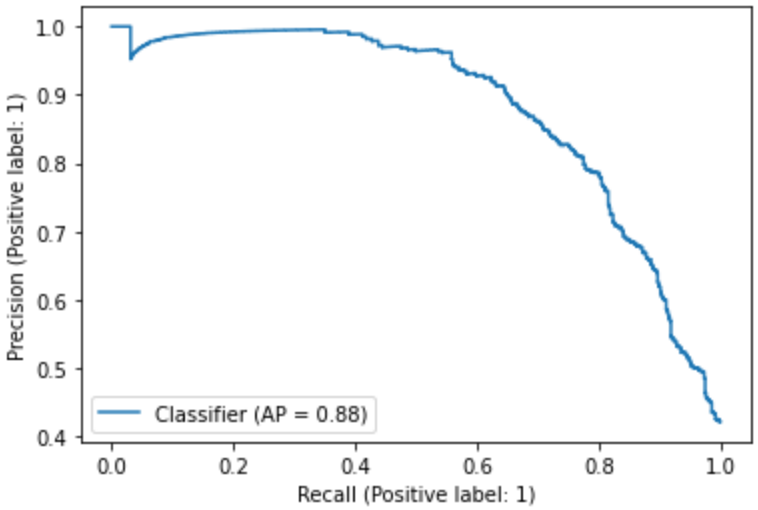


Figure 10: Precision-Recall Curve

The precision-recall curve shown in Figure 10 also shows us the tradeoff between precision and recall of the model at different thresholds. The average precision for the model is 0.88

Limitations

Despite the relatively high performance of the BERTweet model, there are still some limitations to be considered. To begin with, the transformer has only been trained on English, meaning that it is only likely to be useful in English-speaking countries whilst only approximately 32% of all tweets are in English, leaving the majority of available tweets unable to be processed. In fact, the model is not trained to recognize whether a tweet is in English so may be severely hindered if it attempts processing tweets that are in another language.

Additionally, language evolves over time and as the English language slowly deviates from the English spoken today, the model's performance is expected to decrease, especially in a platform like Twitter, where the language spoken tends to be informal and uses lots of slang. So, any model attempting to achieve the same goal would need to be retrained every few years with new tweets in order to be able to keep up with the gradual change of language.

Finally, due to the volatile and dangerous nature of the situation that the model is trying to predict, it is not recommended to actively use a model such as this one unmoderated until higher levels of performance are achieved.

Conclusion

Overall, as demonstrated by this paper, it is definitely possible to classify tweets depending on whether they relate to a disaster or not, to an extent and many distinguishing features between tweets relating to disasters and those not relating to disasters makes this possible, achieving an

F1 score of 83%. However, there are some limitations, such as the model only working on English tweets, and that English itself evolves over time. It should be noted that the performance of the transformer model is not significantly better than that of the baseline models, indicating how difficult it may be to build a very high performing model on this dataset. It should also be mentioned that a lot of the misclassified tweets were very ambiguous. A higher score may be possible by training a model by giving it access to more than just the text data, such as the location and the images attached as well. Additionally, models' performance should also improve as AI technology improves, possibly making a task like this viable and safe for governments.