Real or Not? NLP with Disaster Tweets

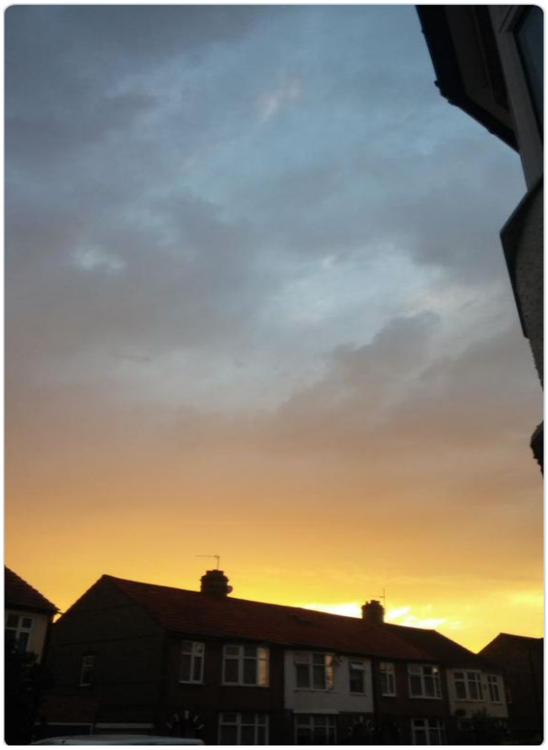
- Vishv Patel (17BCP039)
- Zeel Prajapati (17BCP042)
- Prerak Shah (17BCP044)

Introduction

Twitter has become an important communication channel in times of emergency.

The ubiquitousness of smartphones enables people to announce an emergency they're observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies).

But, it's not always clear whether a person's words are actually announcing a disaster. Take this example:



The author explicitly uses the word "ABLAZE" but means it metaphorically. This is clear to a human right away, especially with the visual aid. But it's less clear to a machine.

This dataset was created by the company figure-eight and originally shared on their 'Data For Everyone'.

Literature Survey

1. Tweet Analytics using NLP [9]
    - By studying the link we came to know about the common steps to follow in order to implement an NLP project.

2. Automated-keyword-extraction-from-articles-using-nlp[8]
    - From this, we came to know about the extraction of keywords from the text.

3. Scikit Learn APIs

4. After studying various kaggle notebooks we came to know about various different techniques.

Proposed System

Dataset Details :

Files

Files	Columns					No. of rows
train.csv	id	text	location	keyword	target	7613
test.csv	id	text	location	keyword		3263
submission.csv	id	target				3263

Columns

- id - a unique identifier for each tweet
- text - the text of the tweet
- location - the location the tweet was sent from (may be blank)
- keyword - a particular keyword from the tweet (may be blank)
- target - in train.csv only, this denotes whether a tweet is about a real disaster (1) or not (0)

Feature Extraction

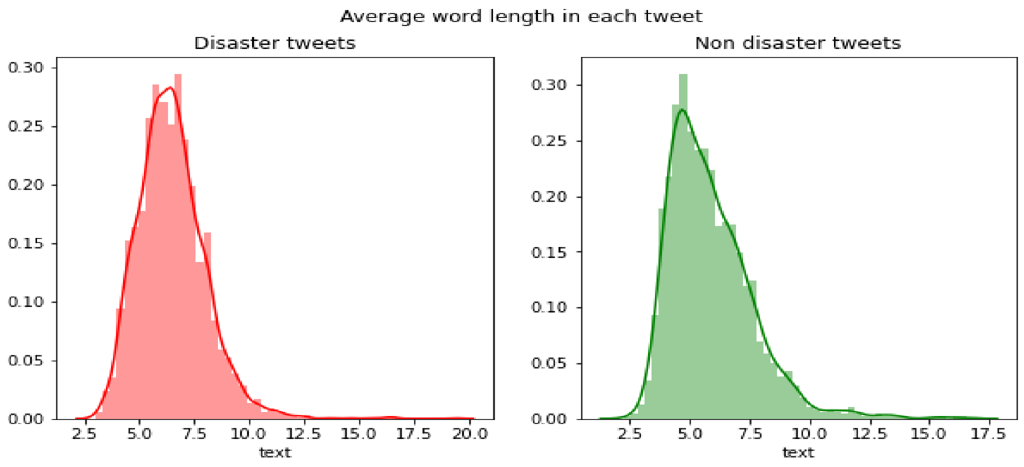
- We used scikit learn for feature extraction from the dataset.[1]
- There were 3 choices available for feature extraction of text
    - Count Vectorizer [2]
    - Hashing Vectorizer [3]
    - Tfidf Vectorizer [4]
- We tried with different ngrams:
    - Unigram
    - Bigram
    - Trigram
    - Combination of Unigram and Bigram
    - Combination of Unigram, Bigram, and Trigram

Classification Algorithm

- For classification, we had two major options from scikit learn:
    - Linear model
        - Logistic Regression with a pipeline.[5][6]
    - Non-Linear Model
        - MLPClassifier (Multi-Layer Perceptron Classifier)[7]

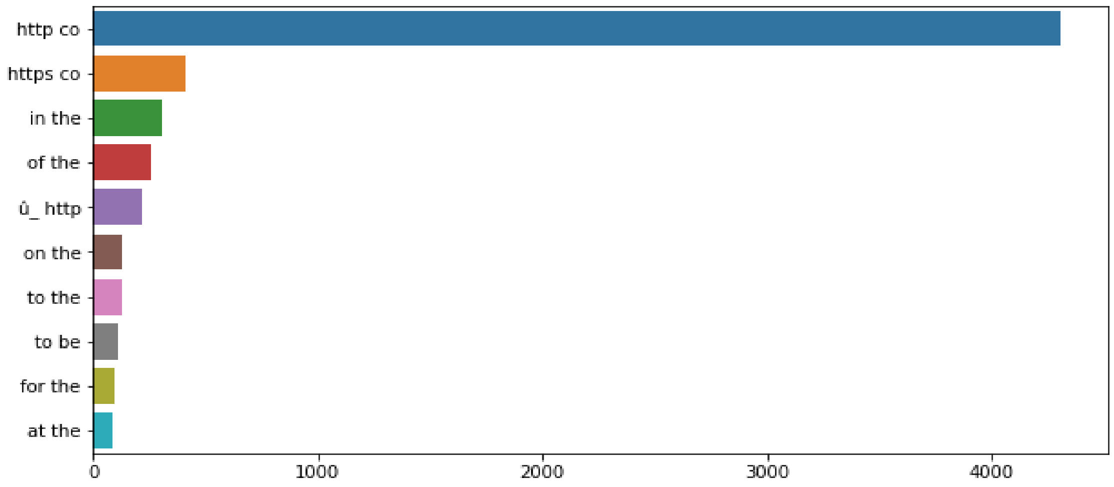
Implementation and Results

- Analysis of Data :



We also found the common words present in a tweet. From this, we came to know that we would need to remove punctuation, URLs, and emojis.

Next, we performed ngram analysis and found that bigram would work best for this dataset.



- Method Selection

    - Cleaning -
        - The empty cells (NaN values) are filled with "."
        - The table contains the accuracy for the following three cases :

Model	Raw	Raw with stop words enabled	Raw after removing punctuation, emoji or URL with stop words enabled
Count Vectorizer + Linear Regression ngram(1,2)	79.5586 %	80.3555 %	79.7119 %

    - Model Selection - From the above results we decided to work with the model 'Raw with stop words enabled'. Now the below table contains 2 different vectorizers and linear and non-linear classifiers.

Vectorizer	Linear Regression ngram(1,2)	Linear Regression ngram(2,2)	MLPClassifier ngram(1,2)	MLPClassifier ngram(2,2)
Count Vectorizer	80.3555 %	74.7472 %	79.1296 %	77.2908 %
Tfidf Vectorizer	78.8538 %	74.2262 %	79.6506 %	76.5860 %

Conclusion

From the results shown in previous tables, we came to know that for our dataset we would get the best accuracy when we use Countvectorizer as a feature extraction method, Linear Regression as a classifier, and ngram range as (1,2).
Also, we came to know that the preprocessing of data and the removal of stop words affect the accuracy of our model.

References

[1]https://scikit-learn.org/stable/modules/classes.html#module-sklearn.feature_extraction

[2]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html#sklearn.feature_extraction.text.CountVectorizer

[3]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.HashingVectorizer.html#sklearn.feature_extraction.text.HashingVectorizer

[4]https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html#sklearn.feature_extraction.text.TfidfVectorizer

[5]https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html#sklearn.linear_model.LogisticRegression

[6]https://scikit-learn.org/stable/modules/classes.html#module-sklearn.pipeline

[7]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

[8]https://medium.com/analytics-vidhya/automated-keyword-extraction-from-articles-using-nlp-bfd864f41b34

[9]https://towardsdatascience.com/tweet-analytics-using-nlp-f83b9f7f7349