NLP Disaster Tweets

Nicholas Bergeland
Northwestern University MSDS 422 – Dr. Fulton

Summary:

For the assignment this week I worked to construct a model which would determine if a tweet was related to a natural disaster event. Given the seemingly recurrent frequency of natural disasters, a model which combs tweets could serve as a potential useful indicator. This is due to the rise in use of social media.

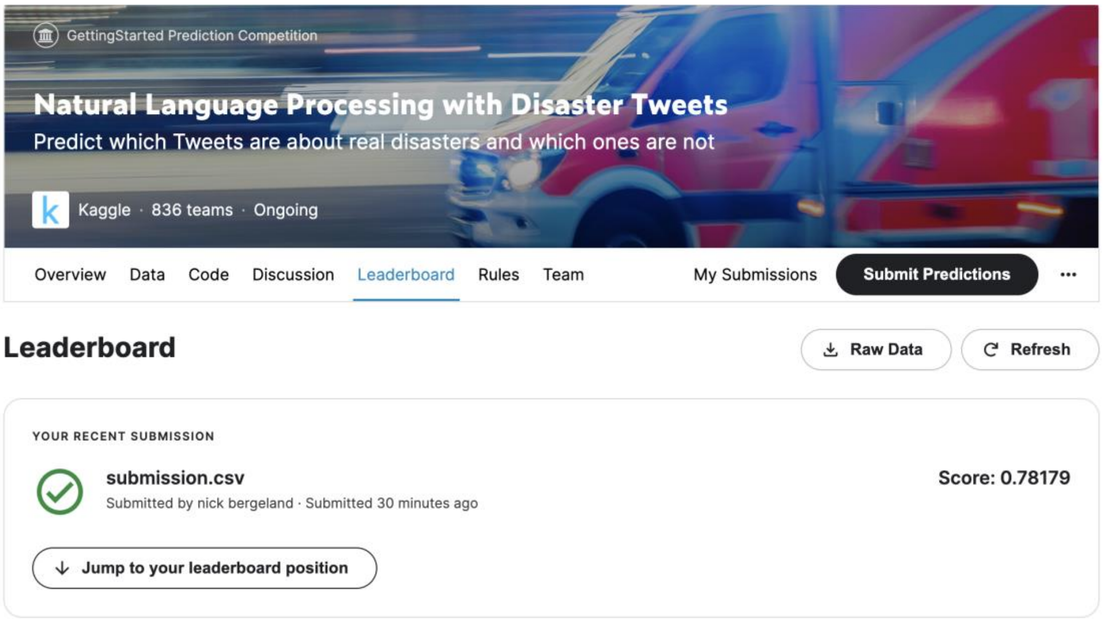
Recently we have seen cases of natural disaster, in which individuals engage in activities such as "going live" on Facebook or Instagram, as well as "tweeting" about current events. This new means of communication by individuals can also potentially serve to inform and aid others in times of crisis. To test this hypothesis, a machine learning model could prove to be a useful tool.

To start with the process, I first needed to import my libraries. For this model, I used a number of packages located inside of Python (Numpy, Pandas, and Sklearn). Once the programs were imported into my workframe, I loaded in both the training and test datasets. From this I point, I ran a couple of checks to make sure the data was correct. When running head tests on the datasets, terms which indicated disaster were returned in the quick search.

With seeing the model return wording that is consistent with disaster indicates the model is on the right track. From this point I took some steps to further develop my model. These included adding ridges and vectors. By doing so I was able to make the model more accurate in its analysis of the tweets being analyzed.

After a few rounds of smoothing and manipulating the model, I was left with what I felt to be a passable model. The result correctly predicted nearly 79% of tweets related to natural disasters! While this is far from perfect, it is still much better than paying a team to sift through and identify tweets (in my opinion). After saving my results to CSV and submitting the results to Kaggle, my score may be found below. Code is submitted as an appendix.

Kaggle Results:


GettingStarted Prediction Competition
Natural Language Processing with Disaster Tweets
Predict which Tweets are about real disasters and which ones are not
Kaggle · 836 teams · Ongoing
Overview
Data
Code
Discussion
Leaderboard
Rules
Team
My Submissions
Submit Predictions
Raw Data
Refresh
Jump to your leaderboard position

Leaderboard

YOUR RECENT SUBMISSION

submission.csv
Submitted by nick bergeland · Submitted 30 minutes ago

Score: 0.78179

Appendix:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
from sklearn import feature_extraction, linear_model, model_selection, prepro
cessing
```

In [3]:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [4]:

```
train_df = pd.read_csv("train.csv")
test_df = pd.read_csv("test.csv")
```

In [5]:

```
train_df[train_df["target"]==0]["text"].values
```

Out[5]:

```
array(["What's up man?", 'I love fruits', 'Summer is lovely', ...,
       'These boxes are ready to explode! Exploding Kittens finally arrived!
gameofkittens #explodingkittens\x89Û_ https://t.co/TFGrAyuDC5',
       'Sirens everywhere!',
       'I just heard a really loud bang and everyone is asleep great'],
      dtype=object)
```

In [6]:

```
train_df[train_df["target"] == 0]["text"].values[1]
```

Out[6]:

```
'I love fruits'
```

In [7]:

```
train_df[train_df["target"] == 1]["text"].values[1]
```

Out[7]:

```
'Forest fire near La Ronge Sask. Canada'
```

In [8]:

```
train_df[train_df["target"]==1]["text"].values
```

Out[8]:

```
array(['Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all
',
       'Forest fire near La Ronge Sask. Canada',
       "All residents asked to 'shelter in place' are being notified by offic
ers. No other evacuation or shelter in place orders are expected",
       ...,
       'M1.94 [01:04 UTC]?5km S of Volcano Hawaii. http://t.co/zDtoyd8EbJ',
       'Police investigating after an e-bike collided with a car in Little Po
rtugal. E-bike rider suffered serious non-life threatening injuries.',
       'The Latest: More Homes Razed by Northern California Wildfire - ABC Ne
ws http://t.co/YmY4rSkQ3d'],
      dtype=object)
```

In [9]:

```
#Building Vectors using scikitlearn CountVectorizer
count_vectorizer = feature_extraction.text.CountVectorizer()

## let's get counts for the first 5 tweets in the data
example_train_vectors = count_vectorizer.fit_transform(train_df["text"][0:5])
print(train_df["text"][0:5].values)
print(count_vectorizer.get_feature_names())
print(count_vectorizer.vocabulary_)
print(example_train_vectors)
['Our Deeds are the Reason of this #earthquake May ALLAH Forgive us all'
 'Forest fire near La Ronge Sask. Canada'
 "All residents asked to 'shelter in place' are being notified by officers. N
o other evacuation or shelter in place orders are expected"
 '13,000 people receive #wildfires evacuation orders in California '
 'Just got sent this photo from Ruby #Alaska as smoke from #wildfires pours i
nto a school ']
['000', '13', 'alaska', 'all', 'allah', 'are', 'as', 'asked', 'being', 'by',
'california', 'canada', 'deeds', 'earthquake', 'evacuation', 'expected', 'fir
e', 'forest', 'forgive', 'from', 'got', 'in', 'into', 'just', 'la', 'may', 'n
ear', 'no', 'notified', 'of', 'officers', 'or', 'orders', 'other', 'our', 'pe
ople', 'photo', 'place', 'pours', 'reason', 'receive', 'residents', 'ronge',
'ruby', 'sask', 'school', 'sent', 'shelter', 'smoke', 'the', 'this', 'to', 'u
s', 'wildfires']
{'our': 34, 'deeds': 12, 'are': 5, 'the': 49, 'reason': 39, 'of': 29, 'this':
50, 'earthquake': 13, 'may': 25, 'allah': 4, 'forgive': 18, 'us': 52, 'all':
3, 'forest': 17, 'fire': 16, 'near': 26, 'la': 24, 'ronge': 42, 'sask': 44, '
canada': 11, 'residents': 41, 'asked': 7, 'to': 51, 'shelter': 47, 'in': 21,
'place': 37, 'being': 8, 'notified': 28, 'by': 9, 'officers': 30, 'no': 27, '
other': 33, 'evacuation': 14, 'or': 31, 'orders': 32, 'expected': 15, '13': 1
, '000': 0, 'people': 35, 'receive': 40, 'wildfires': 53, 'california': 10, '
just': 23, 'got': 20, 'sent': 46, 'photo': 36, 'from': 19, 'ruby': 43, 'alask
a': 2, 'as': 6, 'smoke': 48, 'pours': 38, 'into': 22, 'school': 45}
  (0, 34)	1
  (0, 12)	1
  (0, 5)	1
  (0, 49)	1
  (0, 39)	1
  (0, 29)	1
  (0, 50)	1
  (0, 13)	1
  (0, 25)	1
  (0, 4)	1
  (0, 18)	1
  (0, 52)	1
  (0, 3)	1
  (1, 17)	1
  (1, 16)	1
  (1, 26)	1
  (1, 24)	1
  (1, 42)	1
  (1, 44)	1
  (1, 11)	1
  (2, 5)	2
  (2, 3)	1
  (2, 41)	1
  (2, 7)	1
  (2, 51)	1
  :	:
  (2, 32)	1
  (2, 15)	1
  (3, 21)	1
  (3, 14)	1
  (3, 32)	1
  (3, 1)	1
  (3, 0)	1
  (3, 35)	1
  (3, 40)	1
  (3, 53)	1
  (3, 10)	1
  (4, 50)	1
  (4, 53)	1
  (4, 23)	1
  (4, 20)	1
  (4, 46)	1
  (4, 36)	1
  (4, 19)	2
  (4, 43)	1
  (4, 2)	1
  (4, 6)	1
  (4, 48)	1
  (4, 38)	1
  (4, 22)	1
  (4, 45)	1
```

In [10]:

```
## we use .todense() here because these vectors are "sparse" (only non-zero e
lements are kept to save space)
print(example_train_vectors[0].todense().shape)
print(example_train_vectors[0].todense())
(1, 54)
[[0 0 0 1 1 1 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 1 0
  0 0 0 1 0 0 0 0 0 0 0 0 0 1 1 0 1 0]]
```

In [11]:

```
#create vectors for all tweets
train_vectors = count_vectorizer.fit_transform(train_df["text"])
```

```
## note that we're NOT using .fit_transform() here. Using just .transform() m
akes sure
# that the tokens in the train vectors are the only ones mapped to the test v
ectors -
# i.e. that the train and test vectors use the same set of tokens.
test_vectors = count_vectorizer.transform(test_df["text"])
```

In [12]:

```
#Our Model
#As we mentioned above, we think the words contained in each tweet are a good
indicator of whether they're about a real disaster or not. The presence of pa
rticular word (or set of words) in a tweet might link directly to whether or
not that tweet is real.
## Our vectors are really big, so we want to push our model's weights
## toward 0 without completely discounting different words - ridge regression
## is a good way to do this.
clf = linear_model.RidgeClassifier()
```

In [13]:

```
#Metric for completion is F1.  Testing here
scores = model_selection.cross_val_score(clf, train_vectors, train_df["target
"], cv=3, scoring="f1")
scores
```

Out[13]:

```
array([0.60355649, 0.57580105, 0.64485082])
```

In [14]:

```
#predictions on train set and model for competition
clf.fit(train_vectors, train_df["target"])
```

Out[14]:

```
RidgeClassifier(alpha=1.0, class_weight=None, copy_X=True, fit_intercept=True
,
                max_iter=None, normalize=False, random_state=None,
                solver='auto', tol=0.001)
```

In [16]:

```
sample_submission = pd.read_csv("sample_submission.csv")
```

In [17]:

```
sample_submission["target"] = clf.predict(test_vectors)
```

In [18]:

```
sample_submission.head()
```

Out[18]:

	id	target
0	0	0

	id	target
1	2	1
2	3	1
3	9	0
4	11	1

In [19]:

```
sample_submission.to_csv("submission.csv", index=False)
```

Works Cited:

https://github.com/MahalavanyaSriram/Natural-Language-Processing-with-Disaster-Tweets/blob/master/Jupyter%20Notebooks/baseline.ipynb