CAPSTONE PROJECT REPORT

PABLO BACHO

DEFINITION

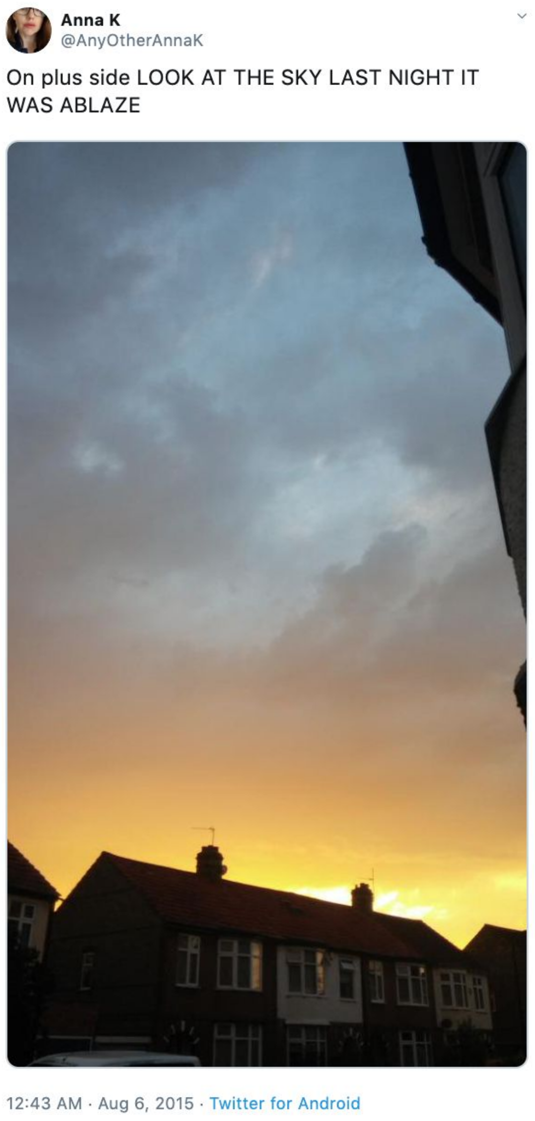
Project Overview

After finishing this program, I would like to tackle some *Kaggle* competitions to practice and keep learning. Therefore, I picked one of Kaggle's "Getting Started" competitions on Natural Language Processing for my capstone project.

The competition is **Real or Not? NLP with Disaster Tweets**, and can be found on Kaggle's website at: https://www.kaggle.com/c/nlp-getting-started/overview/evaluation

The goal of this project is to build a machine learning model that predicts which Tweets are about real disasters and which ones aren't.

To interact with the model, a website was created where tweets can be pasted to check whether they are about a real disaster or not in a similar way to the *Sentiment Analysis* project we worked on during the class.


Anna K
@AnyOtherAnnaK

On plus side LOOK AT THE SKY LAST NIGHT IT WAS ABLAZE

12:43 AM · Aug 6, 2015 · Twitter for Android

Problem Statement

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 programatically 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. Look at the example in the picture. 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.

Metrics

To measure the performance of the model we are using Kaggle's leaderboard. For this competition, the score is calculated using the F1-score method. From the Sklearn documentation:

"The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:"

The formula from Kaggle is:

```
F1 = 2 * (Precision * Recall) / (Precision + Recall)
```

Where:

```
Precision = True Positives / (True Positives + False Positives)
 Recall = True Positives / (True Positives + False Negatives)
```

Being:

- True Positive: *your prediction is 1, and the ground truth is also 1 - you predicted a positive and that's true!*
- False Positive: *your prediction is 1, and the ground truth is 0 - you predicted a positive, and that's false.*
- False Negative: *your prediction is 0, and the ground truth is 1 - you predicted a negative, and that's false.*

ANALYSIS

Data exploration

The dataset is "**Disasters on social media**", collected by the company *Figure Eight* and publicly available to download on their website:

https://www.figure-eight.com/data-for-everyone/

It comprises over 10,000 tweets that were hand classified from searches like "ablaze", "quarantine" and "pandemonium" and then noted whether they related to a catastrophic event or not.

For this project, we are using a modified version by Kaggle that can be found at:

https://www.kaggle.com/c/nlp-getting-started/data

Here is an example of the train dataset content:

id	keyword	location	text	target
6848	loud%20bang	NaN	I don't laugh out loud at many...	0
10461	wild%20fires	Olathe, KS	Man! What I would give to be...	1
8169	rescuers	NaN	#WorldNews\n VIDEO: 'We're...	1
9060	structural%20failure	San Francisco	[CLIP] Top-down coercion - ...	0
4388	earthquake	in the Word of God	@DArchambau THX for your...	1
4058	displaced	Ojodu,Lagos	Angry Woman Openly Accuses...	0

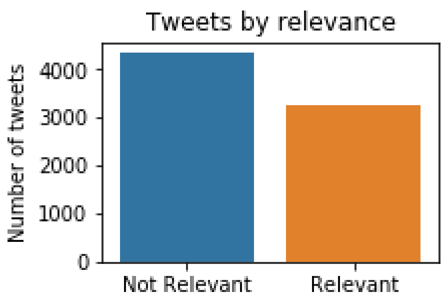
The dataset contains the following columns:

text	The text of a tweet
keyword	A keyword from that tweet (it might be blank)
location	The location the tweet was sent from (it might be blank)
target	1 for relevant to a disaster, 0 for not relevant

Exploratory Visualization

Target

This is the overall distribution of our dataset:



7,613 total tweets
3,271 disaster-related tweets
4,342 non-disaster related tweets
43.0 % percentage of disaster-related tweets

I would like to highlight that the dataset is fairly distributed, with 43% of tweets being positive and 57% negative.

Text

Some example tweets:

- *Afghan Soldier Kills US General America's Highest-Ranking Fatality Since Vietnam http://t.co/SiHQPlUIDW*
- *A look at state actions a year after Ferguson's upheaval http://t.co/vXUFtVT9AU*
- *#hot Funtenna: hijacking computers to send data as sound waves [Black Hat 2015] http://t.co/8JcYXhq1AZ #prebreak #best*

We can see "text" contains punctuation marks, links, hashtags and even special characters that will need to be cleaned up before feeding it to the algorithm.

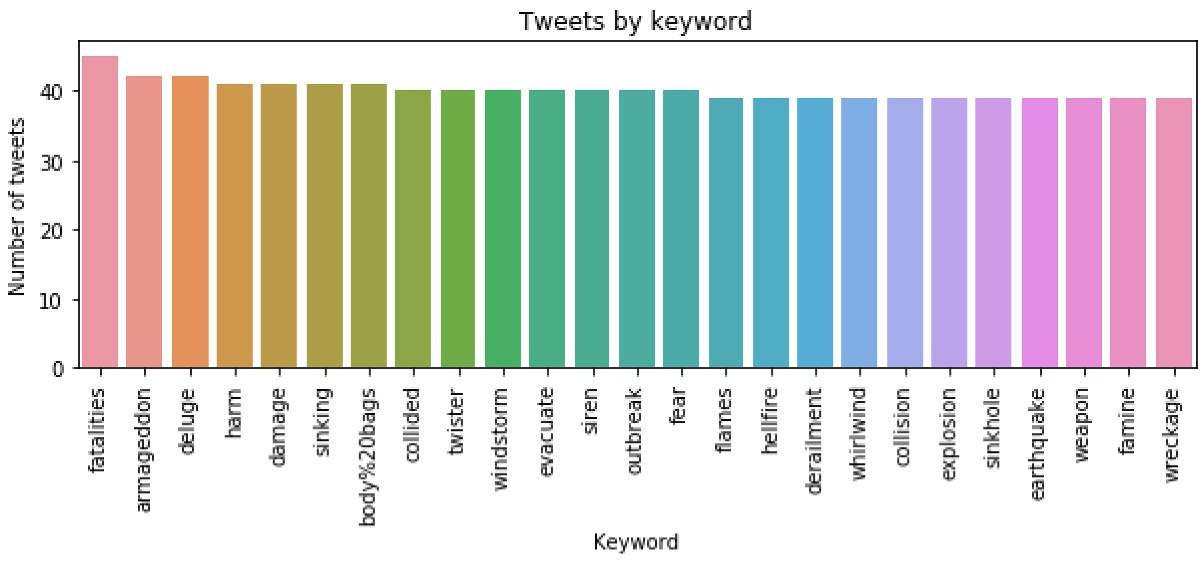
Keyword

Keywords are the search terms used to retrieve the tweets to build the dataset. This keyword is NOT part of a tweet and it will not be available when we use our model to classify new tweets. For this reason I did not use it to build the model.

Let's take a look at it nonetheless to see what the feature looks like.

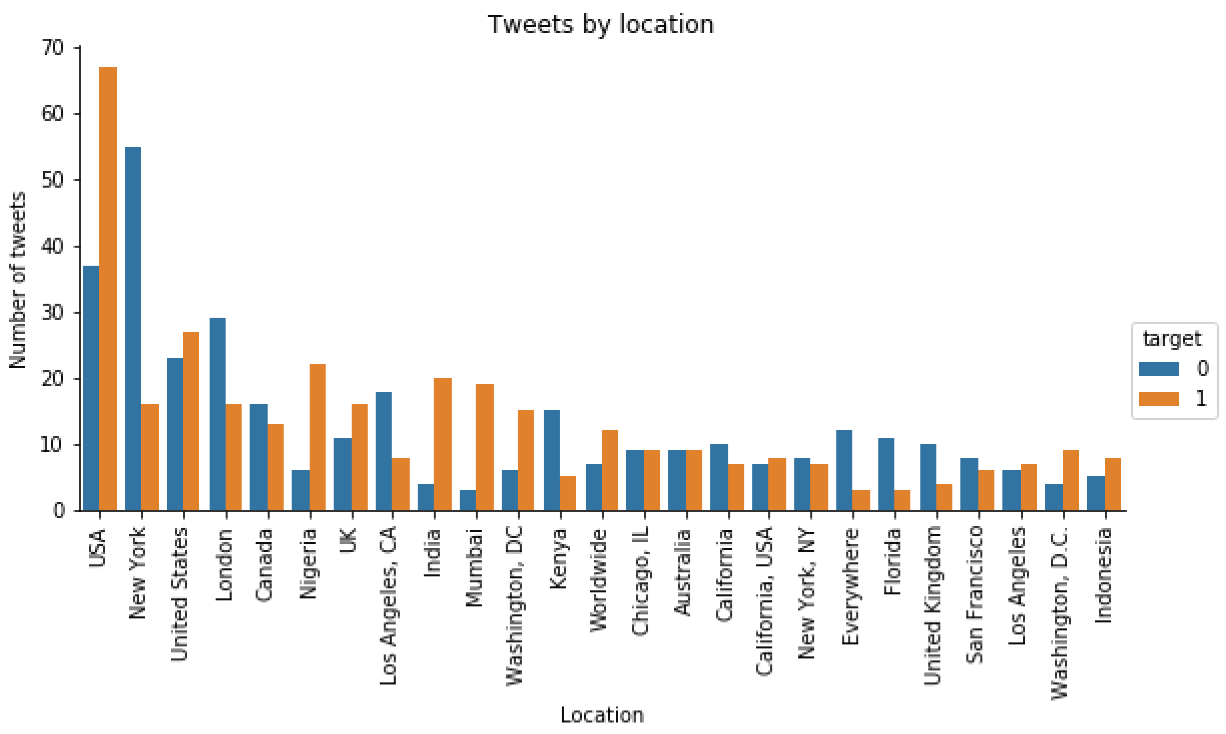
Total different keywords	221
Average number of tweets per keyword	34.2
Standard deviation	1.39
Number of tweets with no keyword	61

We can see there are a total of 221 different keywords in the dataset. Each keyword has an average of 34.2 tweets and a standard deviation of 1.39, so they are pretty evenly distributed.



Location

Below there is a graph with the 25 most popular locations in our dataset grouped by their label.



There are a total of 3,341 different locations for 7,613 tweets.

We need to keep in mind that locations can be very inaccurate:

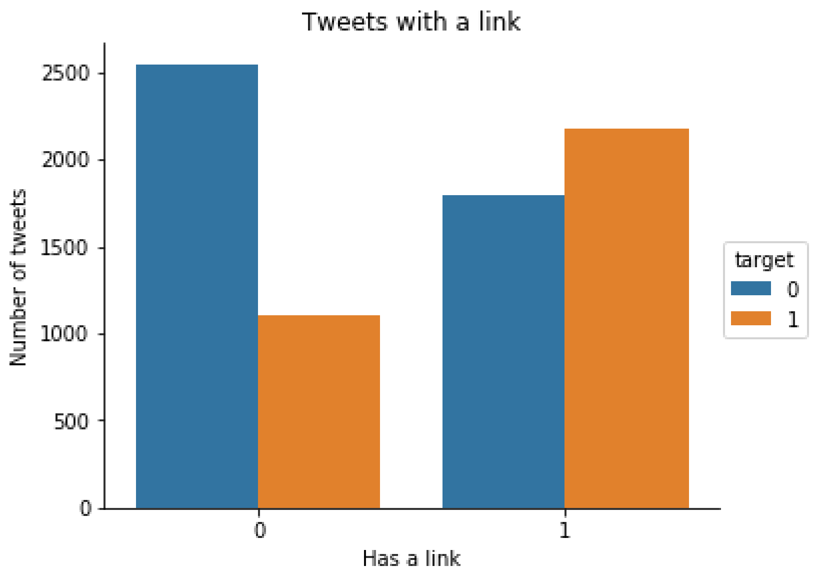
- They are user defined, and the user can write absolutely anything.
- They are strings with no meaning. "USA" and "United States" are different locations, "California" is not contained in "USA", neither is "Los Angeles", and "Los Angeles, CA" is a completely different location.
- Having 3,341 locations for 7,613 tweets implies that most locations will be irrelevant for not having enough tweets from them to extract patterns.

Looking at the graph above it looks like there is a tendency of some places to produce more or less "disaster" tweets than the global average. However, to include this “metadata” in my predictions I would have to do the natural language processing part first, turning the tweet text into a probability of it being a disaster, and then adding the location as a number and feeding both to a different model.

I would like not to over-complicate this first solo project and I did not use location that way. Nonetheless, on my tests I tried adding the location as another word to the text without getting any performance gain.

Hyperlinks

Hyperlinks are unique in each tweet and do not provide any meaningful information as they are. What might provide information is the fact that the tweet includes a hyperlink.



From the graph above some correlation can be found. Tweets that don't have a hyperlink are more likely to not be about a disaster.

Similarly to location, *having a hyperlink* is metadata that would best be used in a second model after the NLP part. However, to keep things simple I just replaced all links with an 'islink' keyword.

Algorithms and Techniques

I used techniques seen in class for *Natural Language Processing*:

1. I start with *PyTorch* and *bag of words*, following what I learnt working on the *Sentiment Analysis* project.
2. Then, I tried to improve the performance of the model by introducing 2-grams. The score using 2-grams ended up being lower than using single words.
3. Searching for a better score, I used BlazingText, one of the built-in algorithms in Sagemaker, which gave me the best results.

Benchmark

Since this is a Kaggle competition, I can test my results on Kaggle's leaderboard. We need to keep in mind that the full dataset is publicly available fully labeled, so there will be cheaters that will distort the results. However, we can use the leaderboard to draw some conclusions from it to assess the performance of the model.

Also, there is an “AutoML benchmark” entry on the leaderboard set at a score of 0.81083, so I can compare my model to that one provided by Kaggle that uses Google AutoML

METHODOLOGY

Data Preprocessing

Tweets are tokenized in a multi-step process:

1. Remove links: I have replaced them with the keyword “islink”.
2. Numbers: I have replaced all numbers with the keyword “isnumber”.
3. Punctuation: I have stripped the tweets of any non-alphanumeric character.
4. Case: I have made all tweets lowercase.
5. Split the tweet into different words.
6. Remove stopwords.
7. Stem words.

Example tokenized tweet:

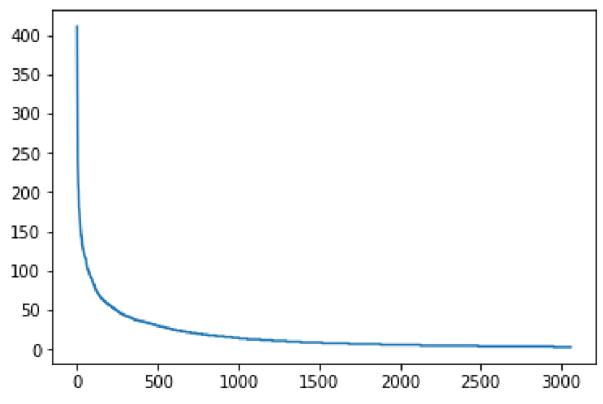
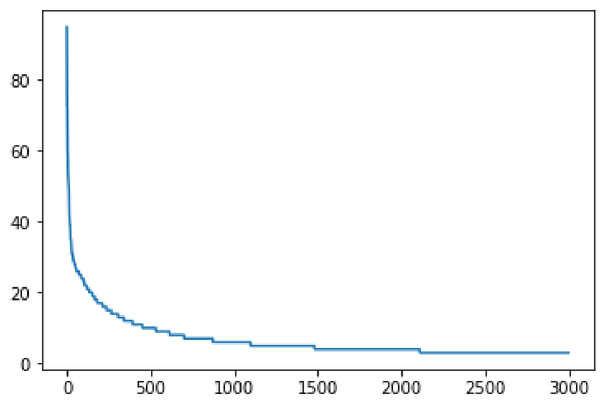
```
[deed, reason, earthquak, may, allah, forgiv, us]
```

Implementation

I created different models searching for the right one:

1. PyTorch NN using the bag of words technique like in the *Sentiment Analysis* project.
2. PyTorch NN using 2-grams, introducing concepts from the *Plagiarism Detector* project.
3. BlazingText algorithm, built into AWS Sagemaker.

For the models based on PyTorch, I needed to build a vocabulary. Below are their vocabulary distribution graphs:

Using single words	Using 2-grams
	

Total number of words: 13,066 Words found 4 times or more: 2,956	Total number of 2-grams: 47,374 2-grams found 4 times or more: 2,111

In order to save on memory and processing time, I decided to ignore words found 3 times or less, replacing them with the "INFREQ" keyword. Measuring this I could objectively define my vocabulary size.

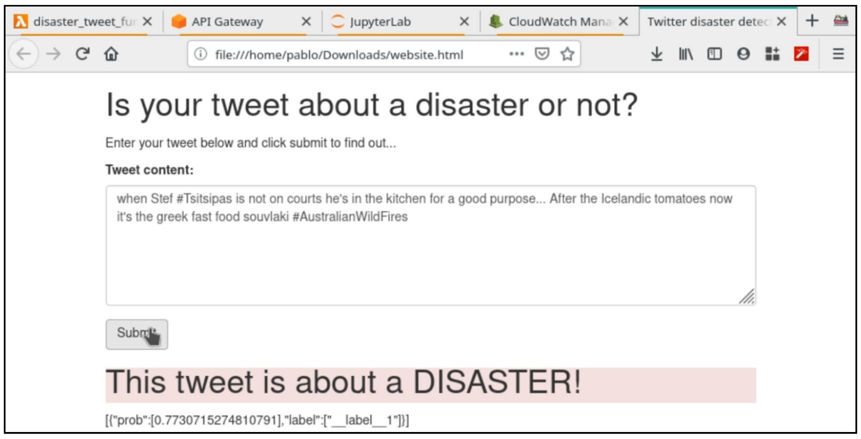
Also, the PyTorch model needs all tweets to have the same length, so I needed to choose a fixed tweet length and pad those that are shorter with "NOWORD" or truncate those that are longer. To calculate this objectively, I decided to cover 99.9% of tweets in my dataset:

Using single words the calculated length of a tweet is 26 words. Being the longest tweet 32 words long	**Using 2-grams** the calculated length of a tweet is 25 words. Being the longest tweet 31 words long

BlazingText on the other hand is so much easier to setup and use. All it needs is a text file with a tweet text per line, including the keyword `__label__` followed by the entry label, 0 or 1. For example:

```
__label__1 deed reason earthquak may allah forgiv us
```

After getting satisfactory results, I built a website based on the Sentiment Analysis project to test other tweets.



Is your tweet about a disaster or not?

Enter your tweet below and click submit to find out...

Tweet content:

when Stef #Tsitsipas is not on courts he's in the kitchen for a good purpose... After the Icelandic tomatoes now it's the greek fast food souvlaki #AustralianWildFires

Submit

This tweet is about a DISASTER!

```
[{"prob":[0.77307152748107911],"label":["__label__1"]}]
```

Refinement

I tweaked some parameters on all three models I implemented. For example, I tried different combinations of completely removing links, numbers or and adding locations with no noticeable change in performance.

I left out the use of other features that could be extracted from the text, such as the use of mentions, hashtags, exclamation marks or uppercase letters. These features could be added as metadata of the text and then processed by a second model designed for the task.

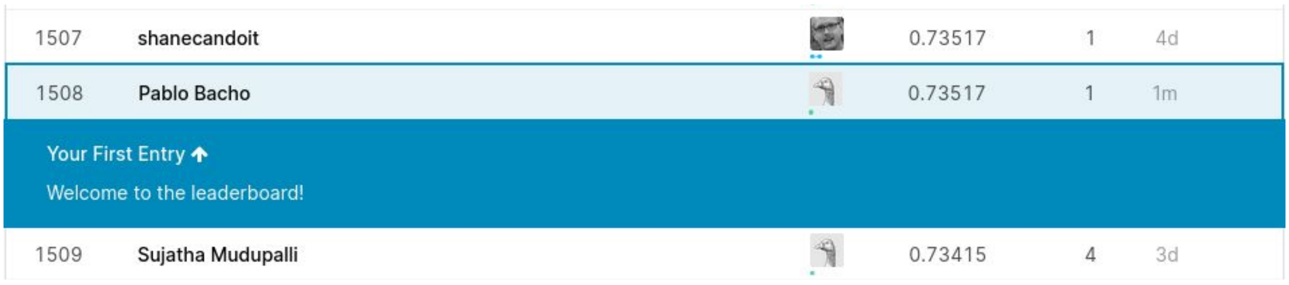
I got the best score with BlazingText using single words. The default setting of using 2-grams performed slightly worse.

RESULTS

Model Evaluation and Validation

For evaluation, I am using Kaggle's leaderboard. The score is calculated using the F1-score metric.

PyTorch using single words

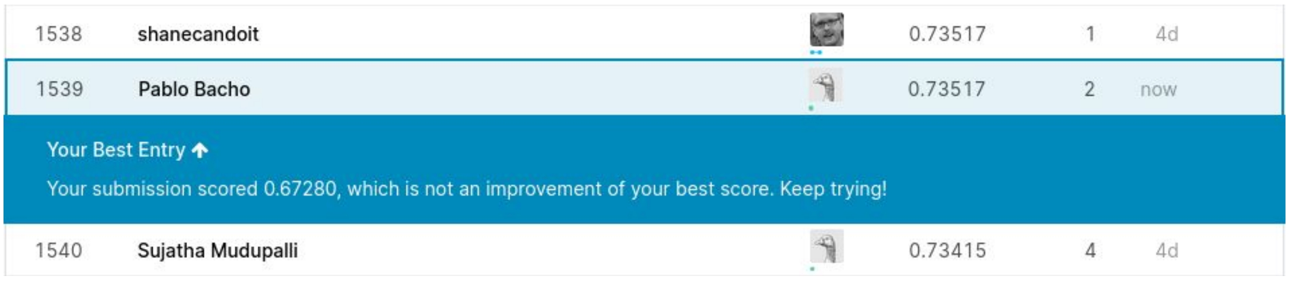


1507	shanecandoit	0.73517	1	4d
1508	Pablo Bacho	0.73517	1	1m
1509	Sujatha Mudupalli	0.73415	4	3d

Your First Entry
Welcome to the leaderboard!

Score: 0.7317

PyTorch using 2-grams

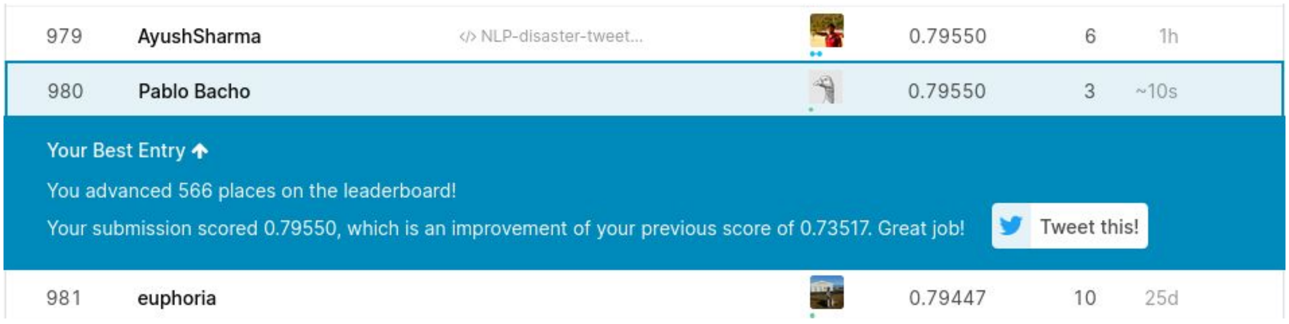


1538	shanecandoit	0.73517	1	4d
1539	Pablo Bacho	0.73517	2	now
1540	Sujatha Mudupalli	0.73415	4	4d

Your Best Entry
Your submission scored 0.67280, which is not an improvement of your best score. Keep trying!

Score: 0.6728

BlazingText using 2-grams



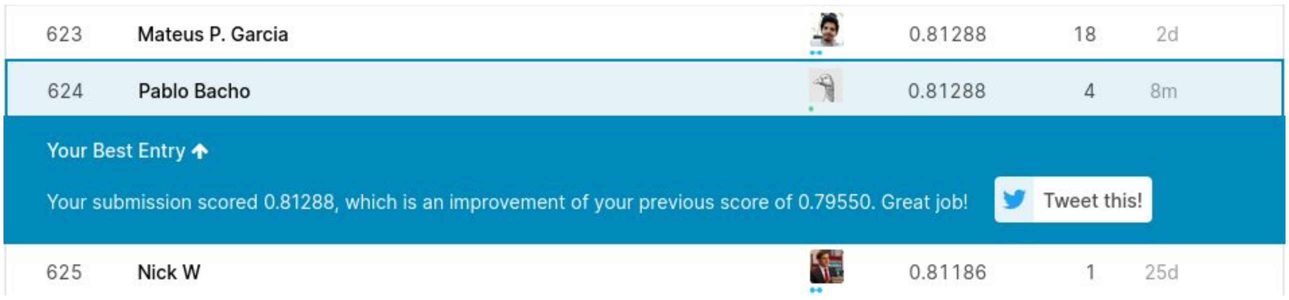
979	AyushSharma	NLP-disaster-tweet...	0.79550	6	1h
980	Pablo Bacho		0.79550	3	~10s
981	euphoria		0.79447	10	25d

Your Best Entry
You advanced 566 places on the leaderboard!
Your submission scored 0.79550, which is an improvement of your previous score of 0.73517. Great job!

Score: 0.7955

BlazingText using single words

Since using PyTorch I got a better score by using single words rather than 2-grams, I thought it would be worth trying out BlazingText using single words as well.



623	Mateus P. Garcia	0.81288	18	2d
624	Pablo Bacho	0.81288	4	8m

Your Best Entry ↑

Your submission scored 0.81288, which is an improvement of your previous score of 0.79550. Great job!

625	Nick W	0.81186	1	25d

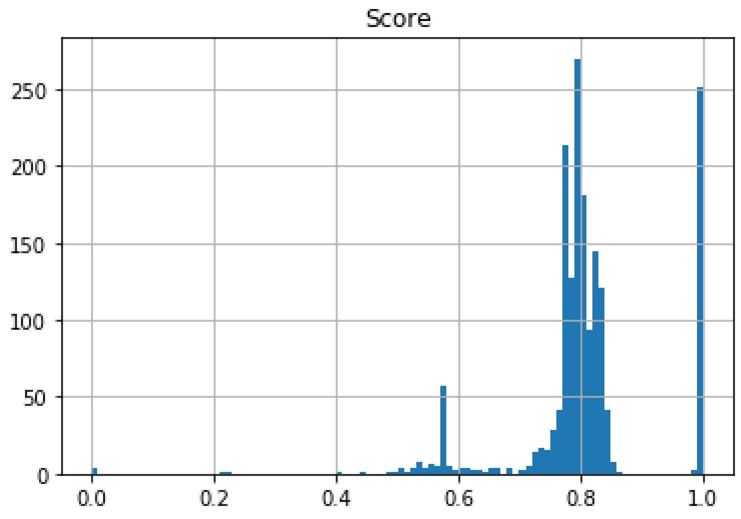
Score: 0.81288. It got an even better score!

Using this model I recorded a video of the website: https://youtu.be/lzGzzTPXokE

Justification

Kaggle allows you to download the whole leaderboard as a CSV file, which is very useful to benchmark results.

Taking a look at the histogram of the leaderboard, we can see the bulk of participants scored around 0.8.



There is a gap with 0 scores from almost 0.90 to almost 1.0 and then a sharp spike. That spike corresponds to scores by "cheaters". This is a "Getting Started" competition, and the whole dataset is publicly available online, so we can expect the leaderboard to have plenty of cheaters that submit based on the actual dataset and not on prediction, or train their algorithm using the whole dataset (and then failing to predict just some outliers). For the following calculations I omitted entries above 0.95.

My score	0.81288
Percentile	74.3%
Global average	0.77632

We can see I scored 0.81288, my algorithm did better than 74.3% of participants and the performance is slightly above the average of 0.77632. Also, the “AutoML benchmark” entry on the leaderboard is set at a very close score of 0.81083.

This is my first solo project on Machine Learning, and I am very happy with the results. Looking at the leaderboard distribution, my model’s performance is within reasonable values, where most participants are, including the benchmark entry.

There is still room for some improvement, maybe using the features discussed earlier on this document, but climbing to the top of the leaderboard in any competition will not be easy. I have a long way to get there, if it ever happens. But as of today, I am very happy with the results of the first one.