STAT154 - Kaggle Final Project

Marie-Camille Achard, Marie Douriez

Due 4/29/16

Exploratory Data Analysis

The dataset contains 1,578,627 tweets and their sentiment scores. 50,000 tweets do not have scores yet, this is our test set. Our goal is to predict the sentiment, positive or negative (respectively scores of 1 and 0) of this test set.

We worked first with the preprocessed matrix containing the word counts of the 1,000 most commonly used words. This smaller set of features helped us having shorter running times to try different algorithms.

We will therefore first focus on exploring this data set, studying for example the most significant words, and then on the choice of our algorithm.

Analysis of the 1,000 most frequent words and their influence

Length of tweets analysis

We have 440 tweets with more than 140 word occurrences. This is impossible, as the length of tweet has to be less than 140 characters. Even if we consider that these tweets have more than 140 short words, it would still be above the 140 characters limit. As 440 is extremely small compared to the size of the dataset, we consider that we can keep these data points without loosing accuracy in our analysis. On the other hand, we have 8872 tweets with none of these 1000 words, which again, seems reasonable. Furthermore, our final analysis will include all the words available in the corpus.

Distribution of word occurrences

By plotting the sorted counts of occurrences of each word, we are able to see that a few word are highly used, and most of the words are used only a few times compared to the dataset size.

Below is the distribution plot.

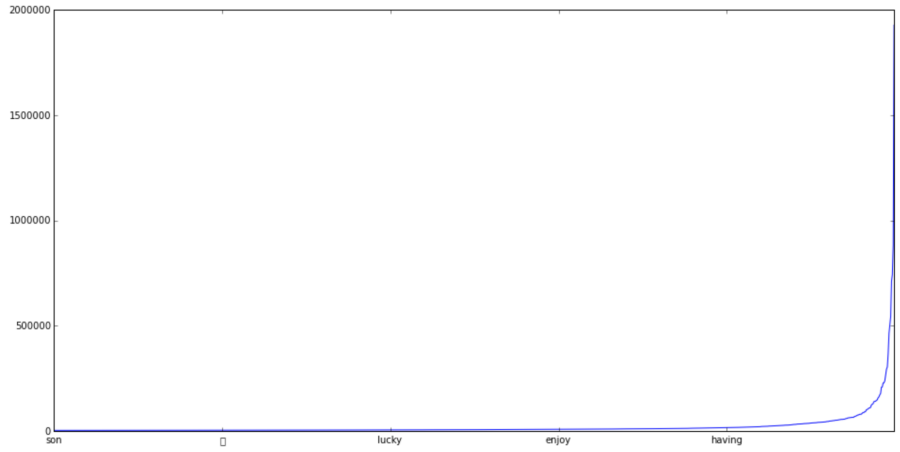

Figure 1: Word distribution

Let's have a closer look at which words are used the less and the most.

Words	Occurrences
son	2059
wishes	2072
apple	2073
exciting	2077
wine	2078
keeps	2079
case	2083
exactly	2084
annoying	2085
wit	2094
?	2094
kid	2096
re	2100
hmmm	2101
gorgeous	2102
hospital	2102
played	2106
twilight	2110
kill	2115
needed	2117

Table 1: Least used words

Words	Occurrences
me	156719
on	161381
0	170819
of	177202
in	207650
for	208333
it	227484
is	227771
?	237738
you	263756
and	292913
my	303800
a	367367
,	463684
the	504932
to	543414
"	710587
i	743042
!	883255
.	1928161

Table 2: Most used words

We can see that in the most used words, most of them are not associated strong sentiment but are simply pronouns or punctuation. Punctuation particularly can mean both: "!" stresses the importance of the sentence, whether it is positive or negative. We chose to print here only the 20 more extreme words each time, but printing more would lead to the same conclusion.

This justify our choice to later use the actual texts of the tweets and not the processed matrix.

Influence of chosen words on the sentiment: great, hate and like

"Great" is used 32,311 times in 31,231 unique tweets. Intuitively, it should be mostly related to positive tweets. We want to verify this. Below is the repartition of the tweets containing "great" between negative and positive. Indeed, three quarters of those are positive but we surprisingly have about 25% of the tweets that are negative.

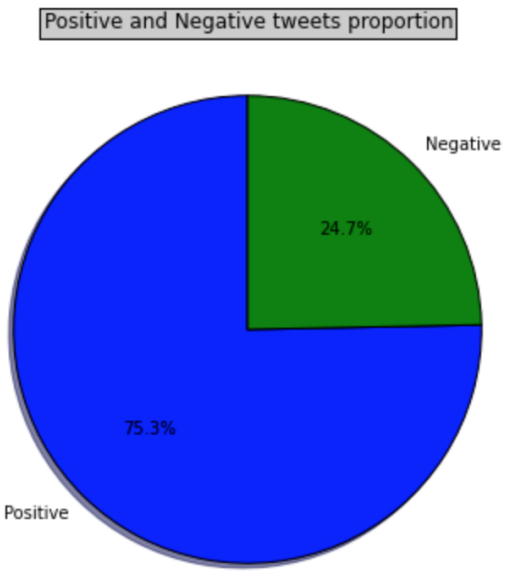

Figure 2: Word distribution

We iterate the study with the word "hate". 18,348 tweets use this word, and of those 86.9% are negative. Running this test on words intuitively extreme was often more conclusive for very negative words (higher percentage of tweets actually negative).

Words that can be used both ways are also interesting as we assume they might not be significant in the prediction. Below is the result for "like", which can be used with and without a negation. It is almost equally distributed.

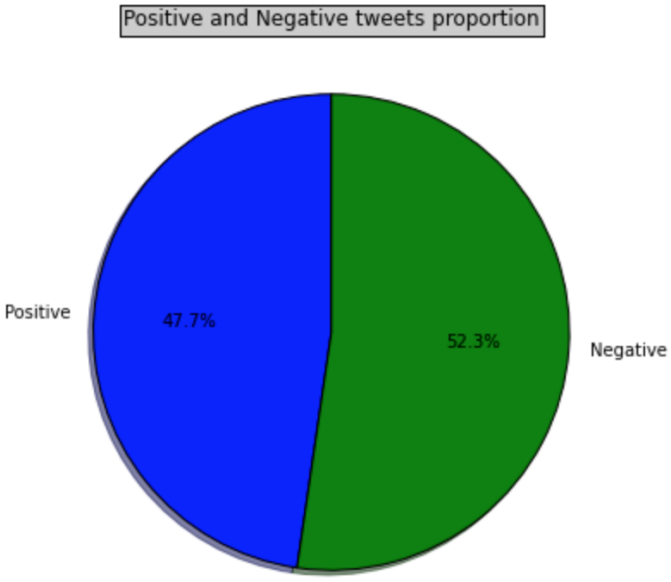

Figure 3: Word distribution

Testing different algorithms on a smaller sample (10^5)

We first tested different algorithms on a 10^5 sample set with 90% of the data as training set and 10% as validation set.

It seems that, on the test set used, Logistic Regression, Linear SVM, LDA and Gradient Boosting perform best. Naive Bayes is widely for text analysis, computes very fast but does not give a very good accuracy here. One of the advantages of Logistic Regression also is that the algorithm fits much faster to the data than other algorithms.

	Hyper-parameters	Test set accuracy	AUC	Kaggle score
Logistic Regression	C=1	0.762	0.835	0.76696
Linear SVM	C=1	0.7630		0.76636
Linear Discriminant Analysis		0.7612	0.834	
Quadratic Discriminant Analysis		0.709	0.728	
Naive Bayes		0.7382	0.813	
Decision Tree	depth = 60	0.6765	0.685	
Random Forest	$n_{estimators}$=100	0.7447	0.824	
Adaboost	$n_{estimators}$=500	0.7447	0.832	
Gradient Boosting Classifier	$n_{estimators}$=1000	0.7617	0.837	
Bagging	classifier=decision tree	0.707	0.788	

Table 3: Results for the algorithms performed on a 10^5 sample set

That is why we chose to fit a Logistic Regression later on the whole dataset.

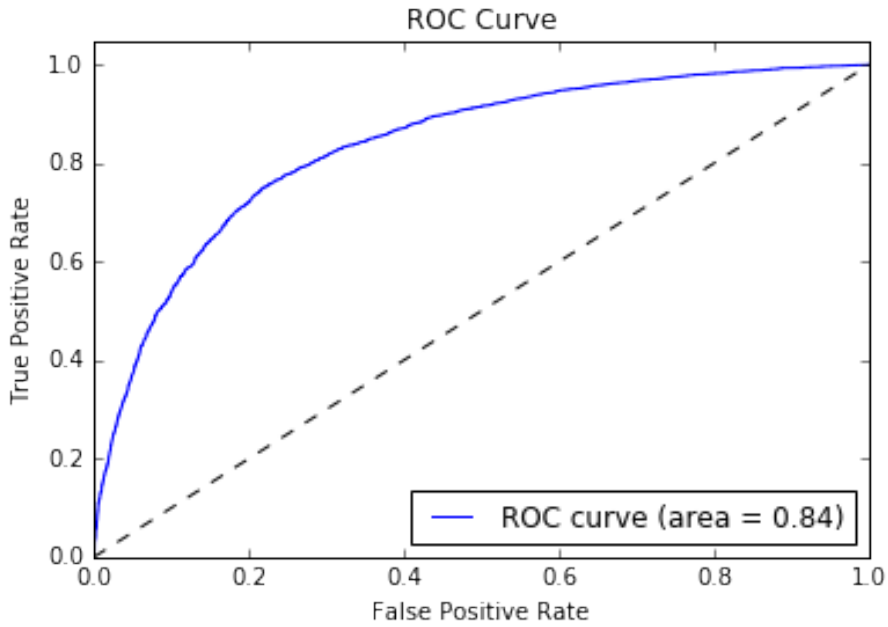

Figure 4: ROC Curve of Logistic Regression for 90,000 samples in training set and 10,000 samples in test set

```
[[3693 1316]
 [1064 3927]]
```

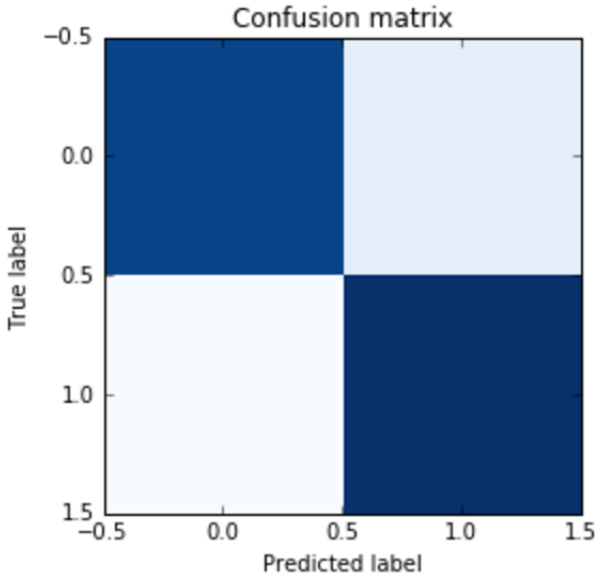

Figure 5: Confusion matrix for Logistic Regression

Processing the raw data

From now on, we will use the entire dataset. In this dataset, we have for each tweet:
- the ID
- the sentiment source (always Sentiment140)
- the actual text of the tweet
- the sentiment score

Again, 50,000 of the 1,578,628 tweets have been assigned a fake score of -1.

We want to engineer new features to improve the accuracy of the first test while still using a logistic regression.

Stop words

First, we removed the stop words in every tweet. Stop words are the common English words that usually do not carry significance or sentiment, like "some" for example.

Count of words

Using this processed text, we built a count of words matrix. We now have a matrix where the number of features equals the number of different words in the corpus. The values are the number of occurrences of this word (column) in this tweet (row). We have a set of 670,688 words.

Final logistic regression

Using the insights of our first study, we want to run a logistic regression on the matrix we built. Considering the size of the dataset (670,688*1,528,627), the running time was great, below 3 min.

Using this method and after a submission on Kaggle, we were able to have an accuracy of **0.78148**.

A potential flaw in the analysis is that we do not have the vocabulary that is not in the train set. A tweet composed of only new words will be assigned randomly to a class, probably the most frequent one.