Jose Antonio Albarés Alberca (jalbares@gmail.com)
David Igeño Rodríguez (exmune@yahoo.es)

June, 2016

Sentiment Analysis of Tweets

1. Introduction

How can you tell if a person feels positive or negative about an experience?
Is it possible only from a brief comment that others wrote?

Thanks to sentiment analysis we can create a model from scratch to predict a class (positive or negative) from text of reviews.
This is an example of classification, one of the areas most used in machine learning.
We will verify the validity of the model across the visualization of the feeling of the most negative and more positives tweets related to cars and flights.

2. Procedure

2.1. Data Collection

We use two datasets that contains product reviews from Amazon to perform the model.
From http://jmcauley.ucsd.edu/data/amazon:

- Babies: http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Baby_5.json.gz
- Pets: http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Pet_Supplies_5.json.gz

Also we recolect some georeferenced tweets about flights and cars from EEUU to test the model with the source file twitterStreamFilter.py (and the method that describe inside). Recolected data:

- USA-flight-tweets.json
- USA-car-tweets.json

2.1.1. Cleaning Phase

In the cleaning phase we obtain new text data used in the analysis phase. The cleaning phase contains the following processes:

- Stop words removal
- Punctuation removal
- Text lemmatization

2.1.2. Important Words

A list of important_words.json file is generated using data of review_clean column of babies+pets_reviews.csv. This file is also required in the analysis phase.

2.1.3. Defining which reviews have positive or negative sentiment

We make an arbitrary choice. Let's say that things that 4, 5 stars are things that people liked. So those are positives. Things that 1 and 2 stars are negative. We ignore all 3 star reviews.

2.2. Classifier

2.2.1. Training and evaluating the model

In classification we are talking about what inputs we get correct and which inputs we get wrong.

When we learn the classifier, we use a set of input data. So these are sentences that have been marked to say positive or negative sentiment. We split it into a training set and a test set. We feed the training set to the classifier to learn the weights for words.

And then, these weights are going to be used to score every element in the test set and evaluate how good we're doing in terms of classification.

But given those weight, how to figure out if the sentence is positive or negative? Here we use the idea of scoring.

So this is how a linear classifier works, if you know the weight of each word, and this is called a linear classifier because the output is basically the weighted sum of the input.

So in summary, given a sentence and given the learned weights for the sentence, what we do is compute the score, which is the weighted count of the words that appear in the sentence.
And then we say if the score is greater than zero, we predict a positive result. While if the score is less than zero, we predict it to be negative.

Classifiers are really trying to make decisions. Decisions as to whether a sentence is positive or negative.

2.2.2 Decision boundaries

How classifiers especially linear classifiers make decisions?
Decision boundaries are what separate the positive predictions from the negative predictions.

When you have tens of thousands of words with non-zero weight, in that case we'll call those hyperplanes, really high dimensional separators.

2.2.3 Measure of quality in classification

We use the common measure of quality in classification: accuracy.

In *accuracy*, instead of measuring the number of errors, we measure the number of correct classifications. So the ratio here is number of correct divided by total number of sentences. In terms of accuracy, the best possible value is 1, I've got all the sentences right.

2.2.3.1 What's a good accuracy?

2.2.3.1.1 If is best than Random Guessing

When you build a classifier, the first baseline comparison it should do is against random guessing.

So for example, if you have a binary classification problem like is this sentence of positive or negative sentiment, then just random guessing is gonna give you 50% accuracy on average.

2.2..3.1.2 If is best than Majority Class Prediction

Majority class prediction is just predicting most common classes. We can have amazing performance in cases where there's what's called class imbalance. (One class has much more representation than the others)

To solve it it’s very important use balanced data classes.

2.2.4 Classification ML block diagram

Data is the text of the reviews, so for each review, the text is associated with a particular labeled sentiment.
From that text of the review:

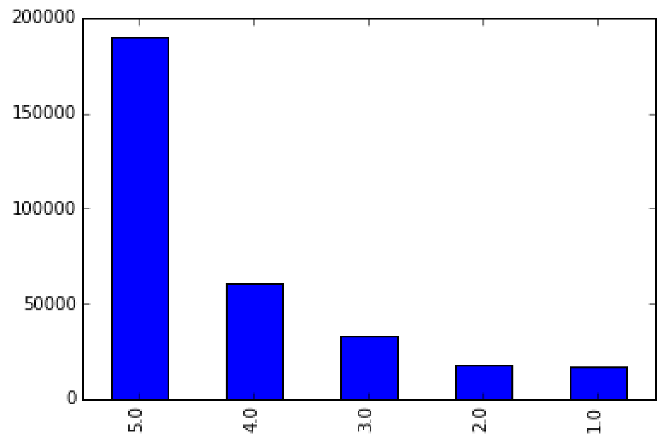
- We feed it to through a feature extraction phase which gives us x, the inputs to our algorithm.
- And this x here is going to be the word counts. So word counts for every data point, for every review.

- Machine Learning Algorithm Phase: Now our Machine Learning model is going to take that input data, the word counts, as well as some several parameters and evaluate that result and then feed it back into the algorithm to improve the parameters.

Combining these two points we're gonna output the predictions. So if the score is greater than zero, it's gonna be positive. If the score is less than zero, it's gonna be negative (the predicted sentiment)

3 Conclusions

3.1 Histogram: Overall vs number of reviews



People prefer add reviews of positive experience products.
Because of this we need to discard a lot of data to obtain a balanced dataset.

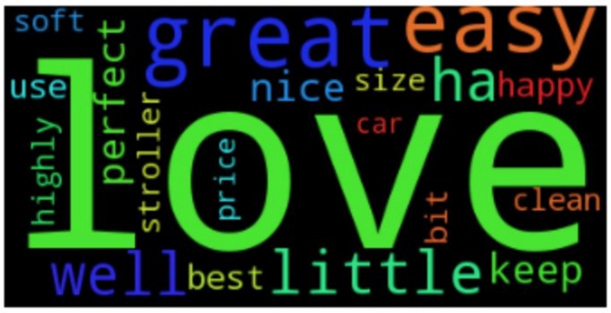
3.2 Apply the model learned

We can apply the model learned to understand the sentiment of reviews.
We're gonna sort the reviews based on the predicted sentiment and explore it.
The predicted sentiment is a much more fine grain description of whether we've used positive or negative.

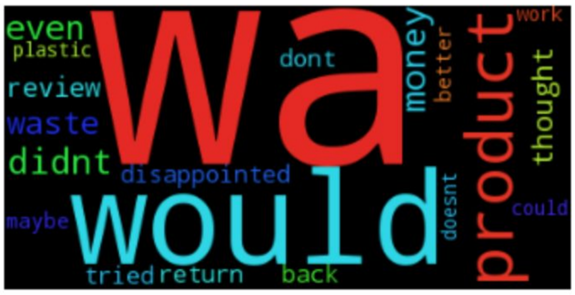
3.2.1 Word Clouds

Which words contribute most to positive & negative sentiments?

3.2.1.1 Most Positive Words



3.2.1.2 Most Negative Words



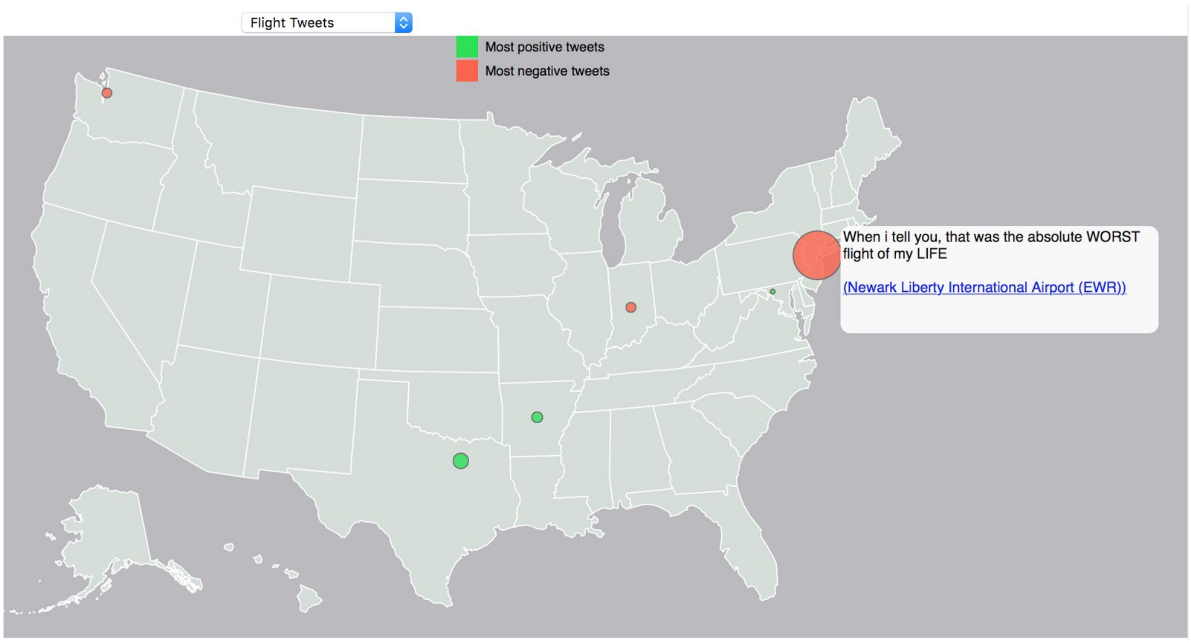
3.2.2 Predicted Sentiment in tweets of EEUU

For the analysis presented here we selected only some georeferenced tweets from EEUU, about cars and flights.
The radius of circles is proportional to the score of the tweet.
Each tooltip point has the text of the tweet and the city relative to the tweet, that it's a link to google maps with the coordinates.

The results show that the model works fine but we only detect few tweets, those with a very definite sentiment. This is the reason because we only show a little set of points.

Using this type of models will help you gather, analyze, and manage conversations about for example brands and this can then provide real insights and learnings on the levels of engagement via content that your audiences are creating.

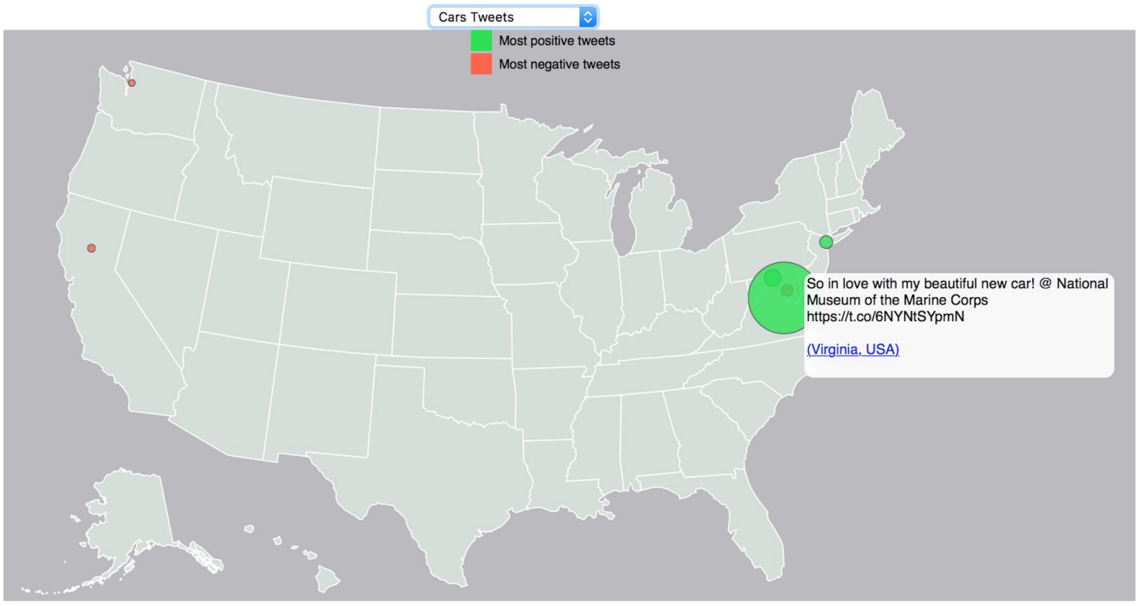
3.2.2.1 About Flights



When i tell you, that was the absolute WORST flight of my LIFE

(Newark Liberty International Airport (EWR))

3.2.2.1 About Cars



So in love with my beautiful new car! @ National Museum of the Marine Corps https://t.co/6NYNtSYpmN

(Virginia, USA)

4 Guide

4.1 Steps

There are two IPython Notebooks: Data Engineering.ipynb and Logistic Regression Model.ipynb

Data Engineering.ipynb notebook generates the required files of the analysis phase that we make in the second notebook (Logistic Regression Model.ipynb)

- ✓ First of all it´s necessary download and extract amazon reviews from http://jmcauley.ucsd.edu/data/amazon
  This web contains Amazon reviews filtered by categories whose data have been choosen in order to train and test the model. The chosen categories are:

  - Babies: http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Baby_5.json.gz
  - Pets: http://snap.stanford.edu/data/amazon/productGraph/categoryFiles/reviews_Pet_Supplies_5.json.gz

  Both files must be uncompressed in the same directory as Data Engineering.ipynb notebook.

- ✓ Use Data Engineering.ipynb notebook to create the file baby+pets_revies.csv which is the beginning of the whole process in the Logistic Regression Model.ipynb notebook.

  This file is generated from the concatenation of reviews_Baby_5.json and reviews_Pet_Supplies_5.json files and adding a new column: review_clean.

- ✓ Use Data Engineering.ipynb notebook to create the list of important_words.
  The list is generated from most frequent words from text_clean columns of baby+pets_revies.csv data set.

    - important_words_1500.json

- ✓ Use Data Engineering.ipynb notebook to processing gathered USA-flight-tweets.json and USA-car-tweets.json files to convert them to csv add a text_clean column.The result must be the following files:

    - USA-flight-tweets.csv
    - USA-car-tweets.csv

- ✓ Finally follow Logistic Regression Model.ipynb and enjoy with the EEUU map of tweets about flight and cars.

5. Software tools you'll need for this project

5.1 Python & iPython Notebook

Python is a simple scripting language that makes it easy to interact with data. Furthermore, Python has a wide range of packages that make it easy to get started and build applications, from the simplest ones to the most complex. Python is widely used in industry, and is becoming the de facto language for data science in industry. (R is another alternative language. However, R tends to be significantly less scalable and has very few deployment tools, thus it is seldomly used for production code in industry.)

We will also encourage the use the IPython Notebook. The IPython Notebook is a simple interactive environment for programming with Python, which makes it really easy to share your results.

5.2 Data manipulation libraries

- Pandas

- SFrame: an open-source, highly-scalable Python library for data manipulation: https://github.com/dato-code/SFrame

5.3 Natural Language Processing

- NLTK

- TextBlob

5.4 Matrix operations

For matrix operations, we strongly recommend Numpy, an open-source Python library that provides fast performance, for data that fits in memory.

5.5 Data Visualization

- Matplotlib, an open-source Python library with extensive plotting functionality

- Wordcloud: https://github.com/amueller/word_cloud. A little word cloud visualization in Python

- D3.js

5.6 Machine Learning algorithms

We implemented an algorithm from scratch.

5.7 Github

For those interested, the code used in this project is available in a public Github repository:

https://github.com/jaalbares/ProyectoFinMasterDataScience