Project Report
Twitter Sentiment Analysis

Instructor: Mekhla Sharma

Members:

- Yogita Tufchi
- Masroor Shah
- Tusheet Pal Singh
- Taposhin

1. Introduction

1.1 Context

This project has been done as a part of my course for the MSc Information Technology at Hong Kong University of Science and Technology. Supervised by Dr David Rossiter, I had three months to fulfill the requirements in order to succeed the module. Every three weeks, a meeting was organized to show and report my progress and fix the next objectives.

1.2 Motivations

Being extremely interested in everything having a relation with the Machine Learning, the independent project was a great occasion to give me the time to learn and confirm my interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Machine Learning in Finance, Medicine, almost everywhere. That's why I decided to conduct my project around the Machine Learning.

1.3 Idea

This project was motivated by my desire to investigate the sentiment analysis field of machine learning since it allows to approach natural language processing which is a very hot topic actually. Following my previous experience where it was about classifying short music according to their emotion, I applied the same idea with tweets and try to figure out which is positive or negative.

1.4 Sources

Because I truly think that sharing sources and knowledge allow helping others but also we, the sources of the project are available at the following link:
https://github.com/MIETDevelopers/107_MasroorShah_Python/tree/master/Project

2. The Project

Sentiment analysis, also refers as opinion mining, is a sub machine learning task where we want to determine which is the general sentiment of a given document. Using machine learning techniques and natural language processing we can extract the subjective information of a document and try to classify it according to its polarity such as positive, neutral or negative. It is a really useful analysis since we could possibly determine the overall opinion about a selling objects, or predict stock markets for a given company like, if most people think positive about it, possibly its stock markets will increase, and so on. Sentiment analysis is actually far from to be solved since the language is very complex (objectivity/subjectivity, negation, vocabulary, grammar,...) but it is also why it is very interesting to working on.

In this project I choose to try to classify tweets from Twitter into “positive” or “negative” sentiment by building a model based on probabilities. Twitter is a microblogging website where people can share their feelings quickly and spontaneously by sending a tweets limited by 140 characters. You can directly address a tweet to someone by adding the target sign “@” or participate to a topic by adding an hastag “#” to your tweet. Because of the usage of Twitter, it is a perfect source of data to determine the current overall opinion about anything.

2.1 Data

To gather the data many options are possible. In some previous paper researches, they built a program to collect automatically a corpus of tweets based on two classes, “positive” and “negative”, by querying Twitter with two type of emoticons:

- Happy emoticons, such as “:)”, “:P”, “:-)” etc.
- Sad emoticons, such as “:(“, “:’(”, “=(“.

Others make their own dataset of tweets my collecting and annotating them manually which very long and fastidious.

Additionally to find a way of getting a corpus of tweets, we need to take of having a balanced data set, meaning we should have an equal number of positive and negative tweets, but it needs also to be large enough. Indeed, more the data we have, more we can train our classifier and more the accuracy will be.

After many researches, I found a dataset of 1578612 tweets in english coming from two sources: Kaggle and Sentiment140. It is composed of four columns that are *ItemID*, *Sentiment*, *SentimentSource* and *SentimentText*. We are only interested by the *Sentiment* column corresponding to our label class taking a binary value, 0 if the tweet is negative, 1 if the tweet is positive and the *SentimentText* columns containing the tweets in a raw format.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL friend.....
1	2	0	Sentiment140	I missed the New Moon trailer...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! T_T
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuussssst Chillin!!
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-| TV Tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmmm.... i wonder how she my number @-)

Table 2.1.1: Example of twitter posts annotated with their corresponding sentiment, 0 if it is negative, 1 if it is positive.

In the *Table 2.1.1* showing the first ten twitter posts we can already notice some particularities and difficulties that we are going to encounter during the preprocessing steps.

- The presence of **acronyms** "bf" or more complicated "APL". Does it means apple ? Apple (the company) ? In this context we have "friend" after so we could think that he refers to his smartphone and so Apple, but what about if the word "friend" was not here ?

- The presence of **sequences of repeated characters** such as "Juuuuuuuuuuuuuuuussssst", "hmmmm". In general when we repeat several characters in a word, it is to emphasize it, to increase its impact.

- The presence of **emoticons**, ":O", "T_T", ":-|" and much more, give insights about user's moods.

- **Spelling mistakes** and "**urban grammar**" like "im gunna" or "mi".

- The presence of **nouns** such as "TV", "New Moon".

Furthermore, we can also add,

- People also indicate their moods, emotions, states, between two *such as,* *\cries*, *hummin*, *sigh*.

- The negation, "can't", "cannot", "don't", "haven't" that we need to handle like: "I don't like chocolate", "like" in this case is negative.

We could also be interested by the grammar structure of the tweets, or if a tweet is subjective/objective and so on. As you can see, it is **extremely complex** to deal with languages and even more when we want to analyse text typed by users on the Internet because people

don’t take care of making sentences that are grammatically correct and use a ton of acronyms and words that are more or less english in our case.

We can visualize a bit more the dataset by making a chart of how many positive and negative tweets does it contains,

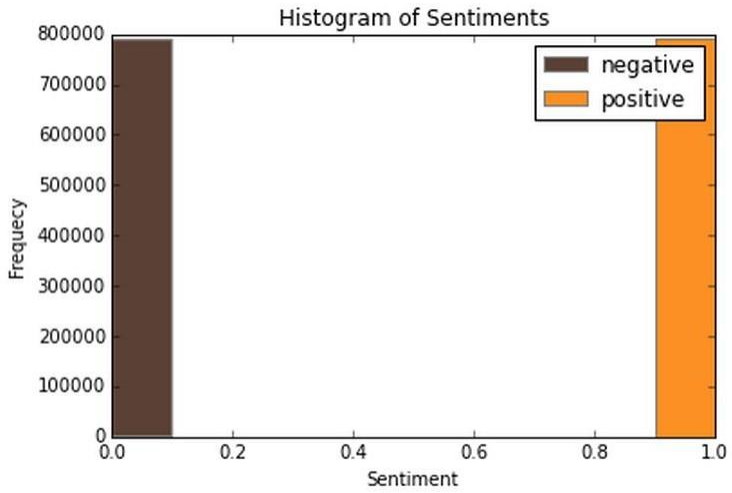

Figure 2.1.1: Histogram of the tweets according to their sentiment

We have exactly 790177 positive tweets and 788435 negative tweets which signify that the dataset is well-balanced. There is also no duplicates.

Finally, let’s recall the Twitter terminology since we are going to have to deal with in the tweets:

- Hashtag: A hashtag is any word or phrase immediately preceded by the # symbol. When you click on a hashtag, you’ll see other Tweets containing the same keyword or topic.

- @username: A username is how you’re identified on Twitter, and is always preceded immediately by the @ symbol. For instance, Katy Perry is @katyperry.

- MT: Similar to RT (Retweet), an abbreviation for “Modified Tweet.” Placed before the Retweeted text when users manually retweet a message with modifications, for example shortening a Tweet.

- Retweet: RT, A Tweet that you forward to your followers is known as a Retweet. Often used to pass along news or other valuable discoveries on Twitter, Retweets always retain original attribution.

- Emoticons: Composed using punctuation and letters, they are used to express emotions concisely, ") :) ...".

Now we have the corpus of tweets, we need to use other resources to make easier the pre-processing step.

2.2 Resources

In order to facilitate the pre-processing part of the data, we introduce five resources which are,

- An **emoticon dictionary** regrouping 132 of the most used emoticons in western with their sentiment, negative or positive.
- An **acronym dictionary** of 5465 acronyms with their translation.
- A **stop word dictionary** corresponding to words which are filtered out before or after processing of natural language data because they are not useful in our case.
- A **positive and negative word dictionaries** given the polarity (sentiment out-of-context) of words.
- A **negative contractions and auxiliaries dictionary** which will be used to detect negation in a given tweet such as “don’t”, “can’t”, “cannot”, etc.

The introduction of these resources will allow to uniform tweets and remove some of their complexities with the acronym dictionary for instance because a lot of acronyms are used in tweets. The positive and negative word dictionaries could be useful to increase (or not) the accuracy score of the classifier. The emoticon dictionary has been built from wikipedia with each emoticon annotated manually. The stop word dictionary contains 635 words such as “the”, “of”, “without”. Normally they should not be useful for classifying tweets according to their sentiment but it is possible that they are.

Also we use Python 2.7 (https://www.python.org/) which is a programming language widely used in data science and scikit-learn (http://scikit-learn.org/) a very complete and useful library for machine learning containing every techniques, methods we need and the website is also full of tutorials well-explained. With Python, the libraries, Numpy (http://www.numpy.org/) and Panda (http://pandas.pydata.org/) for manipulating data easily and intuitively are just essential.

2.3 Pre-processing

Now that we have the corpus of tweets and all the resources that could be useful, we can pre-process the tweets. It is a very important since all the modifications that we are going to during this process will directly impact the classifier's performance. The pre-processing includes cleaning, normalization, transformation, feature extraction and selection, etc. The result of pre-processing will be consistent and uniform data that are workable to maximize the classifier's performance.

All of the tweets are pre-processed by passing through the following steps in the same order.

2.3.1 Emoticons

We replace all emoticons by their sentiment polarity **||pos||** and **||neg||** using the emoticon dictionary. To do the replacement, we pass through each tweet and by using a regex we find out if it contains emoticons, if yes they are replaced by their corresponding polarity.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL friend.....
1	2	0	Sentiment140	I missed the New Moon trailer...
2	3	1	Sentiment140	omg its already 7:30 :O
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! T_T
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuussssst Chillin!!
7	8	0	Sentiment140	Sunny Again Work Tomorrow :-| TV Tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmmm.... i wonder how she my number @-)

Table 2.3.1.1: Before processing emoticons, list of tweets where some of them contain emoticons.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my APL friend.....
1	2	0	Sentiment140	I missed the New Moon trailer...
2	3	1	Sentiment140	omg its already 7:30 ||pos||
3	4	0	Sentiment140	.. Omgaga. Im sooo im gunna CRy. I've been at this dentist since 11.. I was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ||neg||
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	Juuuuuuuuuuuuuuuusssssst Chillin!!
7	8	0	Sentiment140	Sunny Again Work Tomorrow ||neg|| TV Tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmmm.... i wonder how she my number ||pos||

Table 2.3.1.2: After processing emoticons, they have been replaced by their corresponding tag

The data set contains 19469 positive emoticons and 11025 negative emoticons.

2.3.2 URLs

We replace all URLs with the tag **||url||**. There is about 73824 urls in the data set and we proceed as the same way we did for the emoticons.

	ItemID	Sentiment	SentimentSource	SentimentText
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though
55	56	0	Sentiment140	David must be hospitalized for five days end of July (palatine tonsils). I will probably never see Katie in concert.
56	57	0	Sentiment140	friends are leaving me 'cause of this stupid love http://bit.ly/ZoxZC
57	58	1	Sentiment140	go give ur mom a hug right now. http://bit.ly/azFwv
58	59	1	Sentiment140	Going To See Harry Sunday Happiness
59	60	0	Sentiment140	Hand quilting it is then...

Table 2.3.2.1: Tweets before processing URLs.

	ItemID	Sentiment	SentimentSource	SentimentText
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though
55	56	0	Sentiment140	David must be hospitalized for five days end of July (palatine tonsils). I will probably never see Katie in concert.
56	57	0	Sentiment140	friends are leaving me 'cause of this stupid love ||url||
57	58	1	Sentiment140	go give ur mom a hug right now. ||url||
58	59	1	Sentiment140	Going To See Harry Sunday Happiness
59	60	0	Sentiment140	Hand quilting it is then...

Table 2.3.2.2: Tweets after processing URLs.

2.3.3 Unicode

For simplicity and because the ASCII table should be sufficient, we choose to remove any unicode character that could be misleading for the classifier.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu SpÃ¤t' by Die Ã„rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top â€“ I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab... â™« ||url||
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight. ||url||

Table 2.3.3.1: Tweets before processing Unicode.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu Spt' by Die rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab... ||url||
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight. ||url||

Table 2.3.3.1: Tweets after processing Unicode.

2.3.4 HTML entities

HTML entities are characters reserved in HTML. We need to decode them in order to have characters entities to make them understandable.

```
'   Cannot get chatroom feature to work.   Updated Java to 10, checked ports, etc. I can see video, but in the &quot;chat,&quo
t; only a spinning circle.   '
```

Figure 2.3.4.1: A tweet before processing HTML entities.

```
u'   Cannot get chatroom feature to work.   Updated Java to 10, checked ports, etc. I can see video, but in the "chat," only a
spinning circle.   '
```

Figure 2.3.4.2: A tweet after processing HTML entities.

2.3.5 Case

The case is something that can appears useless but in fact it is really important for distinguish proper noun and other kind of words. Indeed: “General Motor” is the same thing that “general motor”, or “MSc” and “msc”. So reduce all letters to lowercase should be normally done wisely. In this project, for simplicity we will not take care of that since we assume that it should not impact too much the classifier’s performance.

	ItemID	Sentiment	SentimentSource	SentimentText
1578592	1578608	1	Sentiment140	'Zu Spt' by Die rzte. One of the best bands ever
1578593	1578609	1	Sentiment140	Zuma bitch tomorrow. Have a wonderful night everyone goodnight.
1578594	1578610	0	Sentiment140	zummie's couch tour was amazing....to bad i had to leave early
1578595	1578611	0	Sentiment140	ZuneHD looks great! OLED screen @720p, HDMI, only issue is that I have an iPhone and 2 iPods . MAKE IT A PHONE and ill buy it @micro...
1578596	1578612	1	Sentiment140	zup there ! learning a new magic trick
1578597	1578613	1	Sentiment140	zyklonic showers *evil*
1578598	1578614	1	Sentiment140	ZZ Top I Thank You ...@hawaiibuzz .....Thanks for your music and for your ear(s) ...ALL !!!! Have a fab... ||url||
1578599	1578615	0	Sentiment140	zzz time. Just wish my love could B nxt 2 me
1578600	1578616	1	Sentiment140	zzz twitter. good day today. got a lot accomplished. imstorm. got into it w yet another girl. dress shopping tmrw
1578601	1578617	1	Sentiment140	zzz's time, goodnight. ||url||

Table 2.3.5.1: Tweets before processing lowercase.

	ItemID	Sentiment	SentimentSource	SentimentText
0	1	0	Sentiment140	is so sad for my apl friend.....
1	2	0	Sentiment140	i missed the new moon trailer...
2	3	1	Sentiment140	omg its already 7:30 ||pos||
3	4	0	Sentiment140	.. omgaga. im sooo im gunna cry. i've been at this dentist since 11.. i was suposed 2 just get a crown put on (30mins)...
4	5	0	Sentiment140	i think mi bf is cheating on me!!! ||neg||
5	6	0	Sentiment140	or i just worry too much?
6	7	1	Sentiment140	juuuuuuuuuuuuuuuussssst chillin!!
7	8	0	Sentiment140	sunny again work tomorrow ||neg|| tv tonight
8	9	1	Sentiment140	handed in my uniform today . i miss you already
9	10	1	Sentiment140	hmmmm.... i wonder how she my number ||pos||

Table 2.3.5.2: Tweets after processing lowercase.

2.3.6 Targets

The target correspond to usernames in twitter preceded by "@" symbol. It is used to address a tweet to someone or just grab the attention. We replace all usernames/targets by the tag **||target||**. Notice that in the data set we have 735757 targets.

	ItemID	Sentiment	SentimentSource	SentimentText
45	46	1	Sentiment140	@ginaaa <3 go to the show tonight
46	47	0	Sentiment140	@spiral_galaxy @ymptweet it really makes me sad when i look at muslims reality now
47	48	0	Sentiment140	- all time low shall be my motivation for the rest of the week.
48	49	0	Sentiment140	and the entertainment is over, someone complained properly.. @ruptureapture experimental you say? he should experiment with a me...
49	50	0	Sentiment140	another year of lakers .. that's neither magic nor fun ...
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though

Table 2.3.6.1: Tweets before processing targets.

	ItemID	Sentiment	SentimentSource	SentimentText
45	46	1	Sentiment140	||target|| <3 go to the show tonight
46	47	0	Sentiment140	||target|| ||target|| it really makes me sad when i look at muslims reality now
47	48	0	Sentiment140	- all time low shall be my motivation for the rest of the week.
48	49	0	Sentiment140	and the entertainment is over, someone complained properly.. ||target|| experimental you say? he should experiment with a melody...
49	50	0	Sentiment140	another year of lakers .. that's neither magic nor fun ...
50	51	0	Sentiment140	baddest day eveer.
51	52	1	Sentiment140	bathroom is clean..... now on to more enjoyable tasks.....
52	53	1	Sentiment140	boom boom pow
53	54	0	Sentiment140	but i'm proud.
54	55	0	Sentiment140	congrats to helio though

Table 2.3.6.2: Tweets after processing targets.

2.3.7 Acronyms

We replace all acronyms with their translation. An acronym is an abbreviation formed from the initial components in a phrase or a word. Usually these components are individual letters (as in NATO or laser) or parts of words or names (as in Benelux). Many acronyms are used in our data set of tweets as you can see in the following bar chart.

At this point, tweets are going to be tokenized by getting rid of the punctuation and using split in order to do the process really fast. We could use nltk.tokenizer but it is definitely much much slower (also much more accurate).

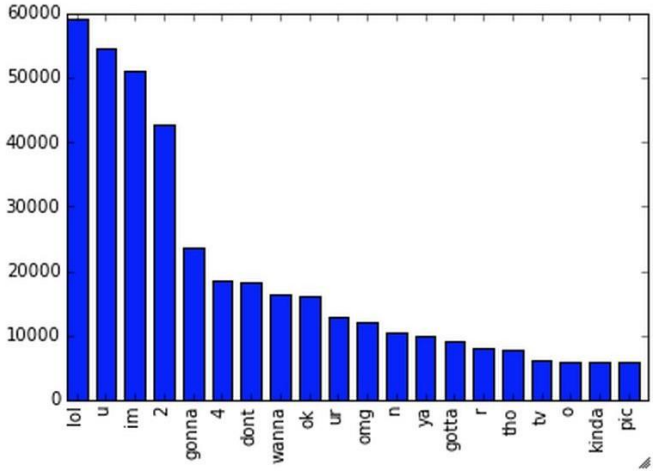

Figure 2.3.7.1: Top 20 of acronyms in the data set of tweets

As you can see, “lol”, “u”, “im”, “2” are really often used by users. The table below shows the top 20 acronyms with their translation and their count.

```
1) lol => laughing out loud : 59000
2) u => you : 54557
3) im => instant message : 51099
4) 2 => too : 42645
5) gonna => going to : 23716
6) 4 => for : 18610
7) dont => don't : 18363
8) wanna => want to : 16357
9) ok => okay : 16104
10) ur => your : 12960
11) omg => oh my god : 12178
12) n => and : 10415
13) ya => yeah : 9948
14) gotta => got to : 9243
15) r => are : 8132
16) tho => though : 7696
17) tv => television : 6246
18) o => oh : 6002
19) kinda => kind of : 5953
20) pic => picture : 5945
```

Table 2.3.7.1: Top 20 of acronyms in the data set of tweets with their translation and count

2.3.8 Negation

We replace all negation words such as “not”, “no”, “never” by the tag **||not||** using the negation dictionary in order to take more or less of sentences like "I don't like it". Here like should not be considered as positive because of the "don't" before. To do so we will replace "don't" by ||not|| and the word like will not be counted as positive. We should say that each time a negation is encountered, the words followed by the negation word contained in the positive and negative word dictionaries will be reversed, positive becomes negative, negative becomes positive, we will do this when we will try to find positive and negative words.

```
['i', "didn't", 'realize', 'it', 'was', 'that', 'deep', 'geez', 'give', 'a', 'girl', 'a', 'warning', 'atleast']
```

Figure 2.3.8.1: A tweet before processing negation words.

```
['i', '||not||', 'realize', 'it', 'was', 'that', 'deep', 'geez', 'give', 'a', 'girl', 'a', 'warning', 'atleast']
```

Figure 2.3.8.2: A tweet after processing negation words.

2.3.9 Sequence of repeated characters

Now, we replace all sequences of repeated characters by two characters (e.g: "helloooo" = "helloo") to keep the emphasized usage of the word.

	ItemID	Sentiment	SentimentSource	SentimentText
1578604	1578620	1	Sentiment140	[zzzz, no, work, tomorrow, yayyy]
1578605	1578621	1	Sentiment140	[zzzzz, time, tomorrow, will, be, a, busy, day, for, serving, loving, people, love, you, all]
1578606	1578622	0	Sentiment140	[zzzzz, want, to, sleep, but, at, sister's, in, laws's, house]
1578607	1578623	1	Sentiment140	[zzzzzz, finally, night, tweeters]
1578608	1578624	1	Sentiment140	[zzzzzzz, sleep, well, people]
1578609	1578625	0	Sentiment140	[zzzzzzzzzz, wait, no, i, have, homework]
1578610	1578626	0	Sentiment140	[zzzzzzzzzzzz, whatever, what, am, i, doing, up, again]
1578611	1578627	0	Sentiment140	[zzzzzzzzzzzzzzzzzz, i, wish]

Table 2.3.9.1: Tweets before processing sequences of repeated characters.

	ItemID	Sentiment	SentimentSource	SentimentText
1578604	1578620	1	Sentiment140	[zz, no, work, tomorrow, yayy]
1578605	1578621	1	Sentiment140	[zz, time, tomorrow, will, be, a, busy, day, for, serving, loving, people, love, you, all]
1578606	1578622	0	Sentiment140	[zz, want, to, sleep, but, at, sister's, in, laws's, house]
1578607	1578623	1	Sentiment140	[zz, finally, night, tweeters]
1578608	1578624	1	Sentiment140	[zz, sleep, well, people]
1578609	1578625	0	Sentiment140	[zz, wait, no, i, have, homework]
1578610	1578626	0	Sentiment140	[zz, whatever, what, am, i, doing, up, again]
1578611	1578627	0	Sentiment140	[zz, i, wish]

Table 2.3.9.2: Tweets after processing sequences of repeated characters.

2.4 Machine Learning

Once we have applied the different steps of the preprocessing part, we can now focus on the machine learning part. There are three major models used in sentiment analysis to classify a sentence into positive or negative: SVM, Naive Bayes and Language Models (N-Gram). SVM is known to be the model giving the best results but in this project we focus only on probabilistic model that are Naive Bayes and Language Models that have been widely used in this field. Let's first introduce the Naive Bayes model which is well-known for its simplicity and efficiency for text classification.

2.4.1 Naive Bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive)independence assumptions between the features. Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Maximum-likelihood training can be done by evaluating a closed-form expression (mathematical expression that can be evaluated in a finite number of operations), which takes linear time.
It is based on the application of the Baye's rule given by the following formula:

$$P(C = c|D = d) = \frac{P(D = d|C = c)P(C = c)}{P(D = d)}$$

Formula 2.4.1.1: Baye's rule

where D denotes the document and C the category (label), d and c are instances of D and C and $P(D = d) = \sum_{c \in C} P(D = d|C = c)P(C = c)$. We can simplify this expression by,

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

Formula 2.4.1.2: Baye's rule simlifed

In our case, a tweet d is represented by a vector of K attributes such as $d = (w_1, w_2, ..., w_k)$. Computing $P(d|c)$ is not trivial and that's why the Naive Bayes introduces the assumption that all of the feature values w_j are independent given the category label c. That is, for $i =/ j$, w_i and w_j are conditionally independent given the category label c. So the Baye's rule can be rewritten as,

$$P(c|d) = P(c) \times \frac{\prod_{j=1}^{K} P(w_i|c)}{P(d)}$$

Formula 2.4.1.3: Baye's rule rewritten

Based on this equation, maximum a posterior (MAP) classifier can be constructing by seeking the optimal category which maximizes the posterior $P(c|d)$:

$$c* = arg\max_{c \in C} P(c|d)$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \frac{\prod_{j=1}^{K} P(w_j|c)}{P(d)} \right\}$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \prod_{j=1}^{K} P(w_j|c) \right\}$$

Formula 2.4.1.4: Classifier maximizing the posterior probability P(c|d)

Note that $P(d)$ is removed since it is a constant for every category c.
There are several variants of Naive Bayes classifiers that are:

- The **Multi-variate Bernoulli Model**: Also called binomial model, useful if our feature vectors are binary (e.g 0s and 1s). An application can be text classification with bag of words model where the 0s 1s are "word does not occur in the document" and "word occurs in the document" respectively.

- The **Multinomial Model**: Typically used for discrete counts. In text classification, we extend the Bernoulli model further by counting the number of times a word w_i appears over the number of words rather than saying 0 or 1 if word occurs or not.

- the **Gaussian Model**: We assume that features follow a normal distribution. Instead of discrete counts, we have continuous features.

For text classification, the most used considered as the best choice is the Multinomial Naive Bayes.

The prior distribution $P(c)$ can be used to incorporate additional assumptions about the relative frequencies of classes. It is computed by:

$$P(c) = \frac{N_i}{N}$$

Formula 2.4.1.5: Prior distribution $P(c)$

where N is the total number of training tweets and N_i is the number of training tweets in class c.

The likelihood $P(w_j|c)$ is usually computed using the formula:

$$P(w_j|c) = \frac{1 + count(w_j, c)}{|V| + N_i}$$

Formula 2.4.1.6: Likelihood $P(w_j|c)$

where $count(w_j, c)$ is the number of times that word w_j occurs within the training tweets of class c, and $|V| = \sum_j w_j$ the size of the vocabulary. This estimation uses the simplest smoothing method to solve the **zero-probability problem**, that arises when our model encounters a word seen in the test set but not in the training set, **Laplace** or add-one since we use 1 as constant. We will see that Laplace smoothing method is not really effective compared to other smoothing methods used in language models.

2.4.2 Baseline

In every machine learning task, it is always good to have what we called a baseline. It often a "quick and dirty" implementation of a basic model for doing the first classification and based on its accuracy, try to improve it.

We use the Multinomial Naive Bayes as learning algorithm with the Laplace smoothing representing the classic way of doing text classification. Since we need to extract features from our data set of tweets, we use the **bag of words model** to represent it.

The bag of words model is a simplifying representation of a document where it is represented as a bag of its words without taking consideration of the grammar or word order. In text classification, the count (number of time) of each word appears is a document is used as a feature for training the classifier.

Firstly, we divide the data set into two parts, the training set and the test set. To do this, we first shuffle the data set to get rid of any order applied to the data, then we from the set of positive tweets and the set of negative tweets, we take $3/4$ of tweets from each set and merge them together to make the training set. The rest is used to make the test set. Finally the size of the training set is 1183958 tweets and the test set is 394654 tweets. Notice that they are balanced and follow the same distribution of the initial data set.

Once the training set and the test set are created we actually need a third set of data called the **validation set**. It is really useful because it will be used to **validate our model against unseen data and tune the possible parameters** of the learning algorithm to avoid underfitting and overfitting for example. We need this validation set because our test set should be used only to verify how well the model will **generalize**. If we use the test set rather than the validation set, our model could be **overly optimistic and twist the results**.

To make the validation set, there are two main options:

- Split the training set into two parts (60%, 20%) with a ratio 2:8 where each part contains an equal distribution of example types. We train the classifier with the largest part, and make prediction with the smaller one to validate the model. This technique works well but has the disadvantage of our classifier not getting trained and validated on all examples in the data set (without counting the test set).

- The **K-fold cross-validation**. We split the data set into k parts, hold out one, combine the others and train on them, then validate against the held-out portion. We repeat that process k times (each fold), holding out a different portion each time. Then we average the score measured for each fold to get a more accurate estimation of our model's performance.

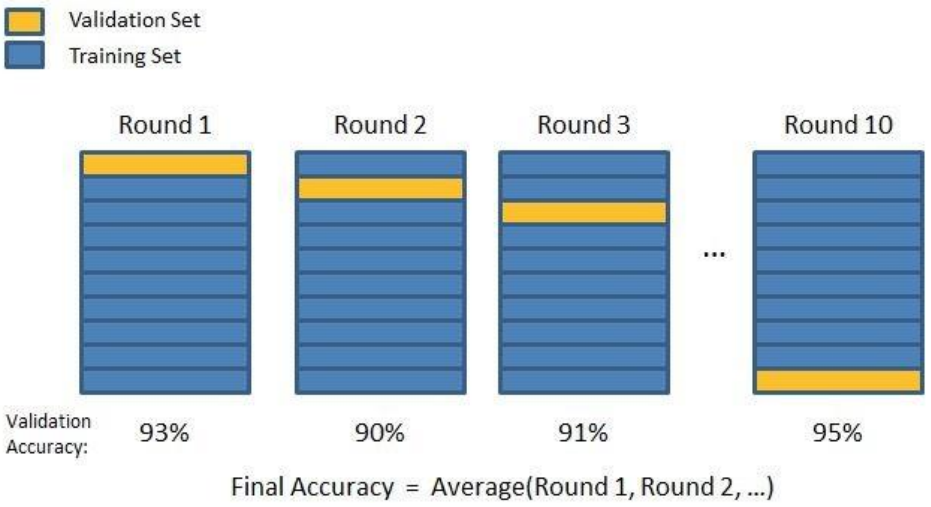

Figure 2.4.2.1: 10-fold cross-validation

We split the training data into 10 folds and cross validate on them using scikit-learn as shown in the figure 2.4.2.1 above. The number of K-folds is arbitrary and usually set to 10 it is not a rule. In fact, determine the best K is still an unsolved problem but with lower K: computationally cheaper, less variance, more bias. With large K: computationally expensive, higher variance, lower bias.

We can now train the naive bayes classifier with the training set, validate it using the hold out part of data taken from the training set, the validation set, repeat this 10 times and average the results to get the final accuracy which is about **0.77** as shown in the screen results below,

```
Total tweets classified: 1183958
Score: 0.77653600187
Confusion matrix:
[[465021 126305]
 [136321 456311]]
```

Figure 2.4.2.2: Result of the naive bayes classifier with the score representing the average of the results of each 10-fold cross-validation, and the overall confusion matrix.

Notice that to evaluate our classifier we two methods, the F1 score and a confusion matrix. 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. It a measure of a **classifier's accuracy**. The F1 score is given by the following formula,

$$F1 = \frac{2 \times (precision \times recall)}{(precision + recall)}$$

Formula 2.4.2.1: F1 score

where the precision is the number of true positives (the number of items correctly labeled as belonging to the positive class) divided by the total number of elements labeled as belonging to the positive class,

$$Precision = \frac{TP}{TP + FP}$$

Formula 2.4.2.1: Precision

and the recall is the number of true positives divided by the total number of elements that actually belong to the positive class,

$$Recall = \frac{TP}{TP + FN}$$

Formula 2.4.2.1: Recall

A precision score of 1.0 means that every result retrieved was relevant (but says nothing about whether all relevant elements were retrieved) whereas a recall score of 1.0 means that all relevant documents were retrieved (but says nothing about how many irrelevant documents were also retrieved).

There is a **trade-off between precision and recall** where increasing one decrease the other and we usually use measures that combine precision and recall such as F-measure or MCC.

A **confusion matrix** helps to visualize how the model did during the classification and evaluate its accuracy. In our case we get about 156715 false positive tweets and 139132 false negative tweets. It is "about" because these numbers can vary depending on how we shuffle our data for example.

Figure 2.4.2.3: Example of confusion matrix

Notice that we still didn't use our test set, since we are going to tune our classifier for improving its results.

The confusion matrix of the naive bayes classifier can be expressed using a color map where dark colors represent high values and light colors represent lower values as shown in the corresponding color map of the naive bayes classifier below,

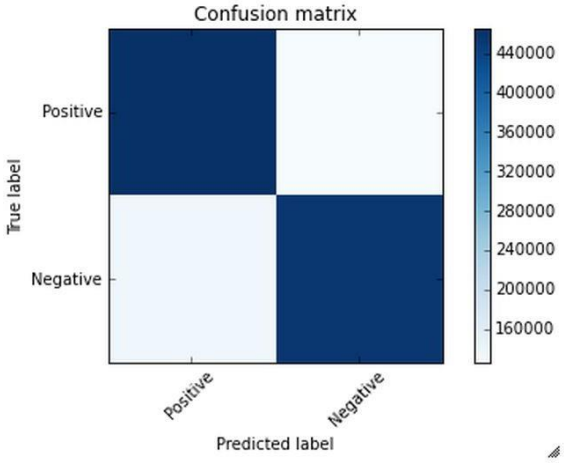

Figure 2.4.2.4: Colormap of the confusion matrix related to the naive bayes classifier used.

Hopefully we can distinguish that the number of true positive and true negative classified tweets is higher than the number of false and positive and negative tweets. However from this result we

try to improve the accuracy of the classifier by experimenting different techniques and we repeat the same process using the k-fold cross validation to evaluate its averaged accuracy.

2.4.3 Improvements

From the baseline, the goal is to improve the accuracy of the classifier, which is 0.77, in order to determine better which tweet is positive or negative. There are several ways of doing this and we present only few possible improvements (or not).

First we could try to removed what we called, stop words. Stop words usually refer to the most common words in the English language (in our case) such as: "the", "of", “to” and so on. They do not indicate any valuable information about the sentiment of a sentence and it can be necessary to remove them from the tweets in order to keep only words for which we are interested. To do this we use the list of 635 stopwords that we found. In the table below, you can see the most frequent words in the data set with their counts,

```
[('||target||', 780664),
 ('i', 778070),
 ('to', 614954),
 ('the', 538566),
 ('a', 383910),
 ('you', 341545),
 ('my', 336980),
 ('and', 316853),
 ('is', 236393),
 ('for', 236018),
 ('it', 235435),
 ('in', 217350),
 ('of', 192621),
 ('on', 169466),
 ('me', 163900),
 ('so', 158457),
 ('have', 150041),
 ('that', 146260),
 ('out', 143567),
 ('but', 132969)]
```

Table 2.4.3.1: Most frequent words in the data set with their corresponding count.

We can derive from the table, some interesting statistics like the number of times the tags used in the pre-processing step appear,

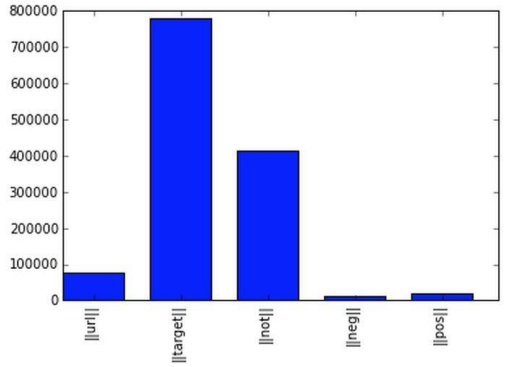

Figure 2.4.3.1: Tags in the data set with their corresponding count.

Recall that ||url|| corresponds to the URLs, ||target|| the twitter usernames with the symbol “@” before, ||not|| replaces the negation words, ||pos|| and ||neg|| replace the positive and negative smiley respectively. After removing the stop words we get the results below,

```
Total tweets classified: 1183958
Score: 0.758623708326
Confusion matrix:
[[437311 154015]
 [136343 456289]]
```

Figure 2.4.2.2: Result of the naive bayes classifier with stopwords removed.

Compared to the previous result, we lose 0.02 in accuracy and the number of false positive goes from 126305 to 154015. We conclude that stop words seem to be useful for our classification task and remove them do not represent an improvement.

We could also try to stem the words in the data set. **Stemming** is the process by which endings are removed from words in order to remove things like tense or plurality. The stem form of a word could not exist in a dictionary (different from Lemmatization). This technique allows to unify words and reduce the dimensionality of the dataset. It's not appropriate for all cases but can make it easier to connect together tenses to see if you're covering the same subject matter. It is faster than **Lemmatization** (remove inflectional endings only and return the base or dictionary form of a word, which is known as the lemma). Using the library NLTK which is a library in Python specialized in natural language processing, we get the following results after stemming the words in the data set,

```
Total tweets classified: 1183958
Score: 0.773106857186
Confusion matrix:
[[462537 128789]
 [138039 454593]]
```

Figure 2.4.2.2: Result of the naive bayes classifier after stemming.

We actually lose 0.002 in accuracy score compared to the results of the baseline. We conclude that stemming words does not improve the classifier’s accuracy and actually do not make any sensible changes.

2.4.4 Language Models

Let's introduce language models to see if we can have better results than those for our baseline. Language models are models assigning **probabilities to sequence of words**. Initially, they are extensively used in speech recognition and spelling correction but it turns out that they give good results in text classification.

The quality of a language model can be measured by the empirical perplexity (or entropy) using:

$$Perplexity = T\sqrt{\frac{1}{P(w_1,\ldots,w_T)}}$$
$$Entropy = log_2 Perplexity$$

Formula 2.4.4.1: Perplexity and Entropy to evaluate language models.

The goal is to **minimize the perplexity which is the same as maximizing probability**.

An **N-Gram model** is a type of probabilistic language model for predicting the next item in such a sequence in the form of (n - 1) order Markov Model. The Markov assumption is the probability of a word depends only on the probability of a limited history (previous words).

$$P(w_i|w_1,\ldots,w_{i-1}) = P(w_i|w_{i-n+1},\ldots,w_{i-1})$$

Formula 2.4.4.2: General form of N-grams.

A straightforward maximum likelihood estimate of n-gram probabilities from a corpus is given by the observed frequency,

$$P(w_i|w_{i-n+1},\ldots,w_{i-1}) = \frac{count(w_{i-n+1},\ldots,w_i)}{count(w_{i-n+1},\ldots,w_{i-1})}$$

Formula 2.4.4.2: MLE of N-grams.

There are several kind of n-grams but the most common are the unigram, bigram and trigram. The **unigram model** make the assumption that every word is independent and so we compute the probability of a sequence using the following formula,

$$P(w_1,w_2,\ldots,w_n) = \prod_i P(w_i)$$

Formula 2.4.4.3: Unigram.

In the case of the **bigram model** we make the assumption that **a word is dependent of its previous word**,

$$P(w_i|w_1, w_2, \ldots, w_{i-1}) \approx P(w_i|w_{i-1})$$

Formula 2.4.4.3: Bigram.

To estimate the n-gram probabilities, we need to compute the **Maximum Likelihood Estimates**.
For Unigram:

$$P(w_i) = \frac{C(w_i)}{N}$$

Formula 2.4.4.4: MLE for unigram.

For Bigram:

$$P(w_i, w_j) = \frac{count(w_i, w_j)}{N}$$
$$P(w_j|w_i) = \frac{P(w_i, w_j)}{P(w_i)} = \frac{count(w_i, w_j)}{\sum_w count(w_i, w)} = \frac{count(w_i, w_j)}{count(w_i)}$$

Formula 2.4.4.5: MLE for bigram.

Where N is the number of words, C means count, w_i and w_j are words.

There are two main practical issues:

- We compute everything in log space (log probabilities) to avoid underflow (multiplying so many probabilities can lead to too small number) and because adding is faster than multiplying $(p_1 \times p_2 \times p_3) = (log_{p_1} + log_{p_2} + log_{p_3})$

- We use smoothing techniques such as Laplace, Witten-Bell Discounting, Good-Turing Discounting to deal with unseen words in the training occurring in the test set.

An N-gram language model can be applied to text classification like Naive Bayes model does. A tweet is categorized according to,

$$c* = arg\max_{c \in C} P(c|d)$$

Formula 2.4.4.6:Objective function of n-gram.

and using Baye's rule, this can be rewritten as,

$$c* = arg\max_{c \in C} \{P(c)P(d|c)\}$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \prod_{i=1}^{T} P(w_i | w_{i-n+1}, \ldots, w_{i-1}, c) \right\}$$

$$c* = arg\max_{c \in C} \left\{ P(c) \times \prod_{i=1}^{T} P_c(w_i | w_{i-n+1}, \ldots, w_{i-1}) \right\}$$

Formula 2.4.4.7:Objective function rewritten using baye's rule of n-gram.

$P(d|c)$ is the likelihood of d under category c which can be computed by an n-gram language model.

An important note is that n-gram classifiers are in fact a generalization of Naive Bayes. A unigram classifier with Laplace smoothing corresponds exactly to the traditional naive Bayes classifier.

Since we use bag of words model, meaning we translate this sentence: "I don't like chocolate" into "I", "don't", "like", "chocolate", we could try to use bigram model to take care of negation with "don't like" for this example. Using bigrams as feature in the classifier we get the following results,

```
Total tweets classified: 1183958
Score: 0.784149223247
Confusion matrix:
[[480120 111206]
 [138700 453932]]
```

Formula 2.4.4.8: Results of the naive bayes classifier with bigram features.

Using only bigram features we have slightly improved our accuracy score about 0.01. Based on that we can think of adding unigram and bigram could increase the accuracy score more.

```
Total tweets classified: 1183958
Score: 0.795370054626
Confusion matrix:
[[486521 104805]
 [132142 460490]]
```

Formula 2.4.4.9: Results of the naive bayes classifier with unigram and bigram features.

and indeed, we increased slightly the accuracy score about 0.02 compared to the baseline.

3. Conclusion

Nowadays, sentiment analysis or opinion mining is a hot topic in machine learning. We are still far to detect the sentiments of s corpus of texts very accurately because of the complexity in the English language and even more if we consider other languages such as Chinese.

In this project we tried to show the basic way of classifying tweets into positive or negative category using Naive Bayes as baseline and how language models are related to the Naive Bayes and can produce better results. We could further improve our classifier by trying to extract more features from the tweets, trying different kinds of features, tuning the parameters of the naïve Bayes classifier, or trying another classifier all together.

4. References

- Alexander Pak, Patrick Paroubek. 2010, Twitter as a Corpus for Sentiment Analysis and Opinion Mining.
- Alec Go, Richa Bhayani, Lei Huang. Twitter Sentiment Classification using Distant Supervision.
- Jin Bai, Jian-Yun Nie. Using Language Models for Text Classification.
- Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, Rebecca Passonneau. Sentiment Analysis of Twitter Data.
- Fuchun Peng. 2003, Augmenting Naive Bayes Classifiers with Statistical Language Models