

Predicting and Modelling Social Network

By

Utsav Maniar (150033107015)

utsavmaniar@gmail.com

Sunny Sommanek (150033107024)

sunnybuddy1995@gmail.com

Abstract

We introduce an approach for automatically classifying the sentiment of Twitter messages. These messages are classified as positive or negative with respect to a query term. This is useful for consumers who want to research the sentiment of products before purchase, or companies that want to monitor the public sentiment of their brands. We have used machine learning technique and in particular an algorithm called Naïve Bayes.

Introduction

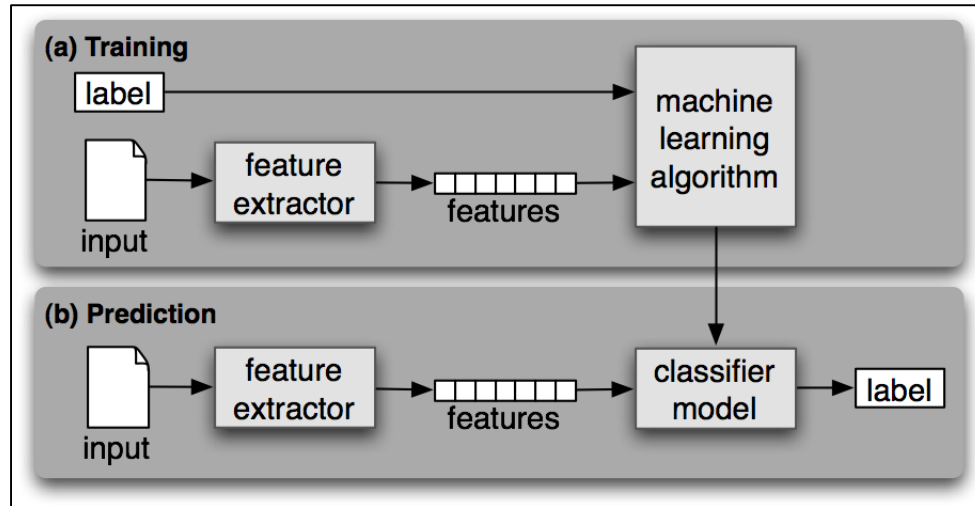
Twitter is a popular microblogging service where users create status messages (called “tweets”). These tweets sometimes express opinions about different topics. We propose a method to automatically extract sentiment (positive or negative) from a tweet. This is very useful because it allows feedback to be aggregated without manual intervention. Using this analyzer,

- Consumers can use sentiment analysis to research products or services before making a purchase. E.g. Kindle
- Marketers can use this to research public opinion of their company and products, or to analyze customer satisfaction. E.g. Election Polls
- Organizations can also use this to gather critical feedback about problems in newly released products. E.g. Brand Management (Nike, Adidas)

To help visualize the utility of the Twitter-based sentiment analysis tool, we have built a web application. This can be used by individuals and companies that may want to research sentiment on any topic.

Background

In order to build a sentiment analyzer, first we need to equip ourselves with the right tools and methods. Machine learning is one such tool where people have developed various methods to classify. Classifiers may or may not need training data. In particular, we will deal with the classifier called Naive Bayes Classifier. This classifier requires training data and hence these methods fall under the category of supervised classification. We have used Python 2.7 and NLTK in this project. Following is the flow of our approach:



Preprocess tweets

Tweets come with some irrelevant information which need to be discarded. To achieve this we have to preprocess the tweets before we can use it to gain insights. We have preprocessed tweets for following:

1. **Lower Case** - Convert the tweets to lower case.
2. **URLs** - I don't intend to follow the short urls and determine the content of the site, so we can eliminate all of these URLs via regular expression matching or replace with generic word URL.
3. **@username** - we can eliminate "@username" via regex matching or replace it with generic word AT_USER.
4. **#hashtag** - hash tags can give us some useful information, so it is useful to replace them with the exact same word without the hash. E.g. #nike replaced with 'nike'.
5. **Punctuations and additional white spaces** - remove punctuation at the start and ending of the tweets. E.g: ' the day is beautiful! ' replaced with 'the day is beautiful'. It is also helpful to replace multiple whitespaces with a single whitespace

For example,

Following are the tweets before preprocessing:

1. @PrincessSuperC Hey Cici sweetheart! Just wanted to let u know I luv u! OH! and will the mixtape drop soon? FANTASY RIDE MAY 5TH!!!!
2. @Msdebramaye I heard about that contest! Congrats girl!!
3. UNC!!! NCAA Champs!! Franklin St.: I WAS THERE!! WILD AND CRAZY!!!!!! Nothing like it...EVER <http://tinyurl.com/49955t3>

And these are the tweets after preprocessing:

1. AT_USER hey cici sweetheart! just wanted to let u know i luv u! oh! and will the mixtape drop soon? fantasy ride may 5th!!!!
2. AT_USER i heard about that contest! congrats girl!!
3. unc!!! ncaa champs!! franklin st.: i was there!! wild and crazy!!!!!! nothing like it...ever UR

Feature Vector

Feature vector is the most important concept in implementing a classifier. A good feature vector directly determines how successful your classifier will be. The feature vector is used to build a model which the classifier learns from the training data and further can be used to classify previously unseen data.

In tweets, we can use the presence/absence of words that appear in tweet as features. In the training data, consisting of tweets, we can split each tweet into words and add each word to the feature vector. Some of the words might not have any say in indicating the sentiment of a tweet and hence we can filter them out like the, to, a, an, etc., these words are known as 'Stop words'. Adding individual (single) words to the feature vector is referred to as 'unigrams' approach. We are going to filter the tweets by removing,

1. **Stop words** - a, is, the, with etc. The full list of stop words can be found at Stop Word List. These words don't indicate any sentiment and can be removed.
2. **Repeating letters** - if you look at the tweets, sometimes people repeat letters to stress the emotion. E.g. hunggrryyy, huuuuuuungry for 'hungry'. We can look for 2 or more repetitive letters in words and replace them by 2 of the same.
3. **Punctuation** - we can remove punctuation such as comma, single/double quote, question marks at the start and end of each word. E.g. beautiful!!!!!! replaced with beautiful
4. **Words must start with an alphabet** - For simplicity sake, we can remove all those words which don't start with an alphabet. E.g. 15th, 5.34am

After doing all the processes on tweets, our tweets are ready to be used as training data for the Naïve Bayes Classifier.

Naive Bayes Classifier

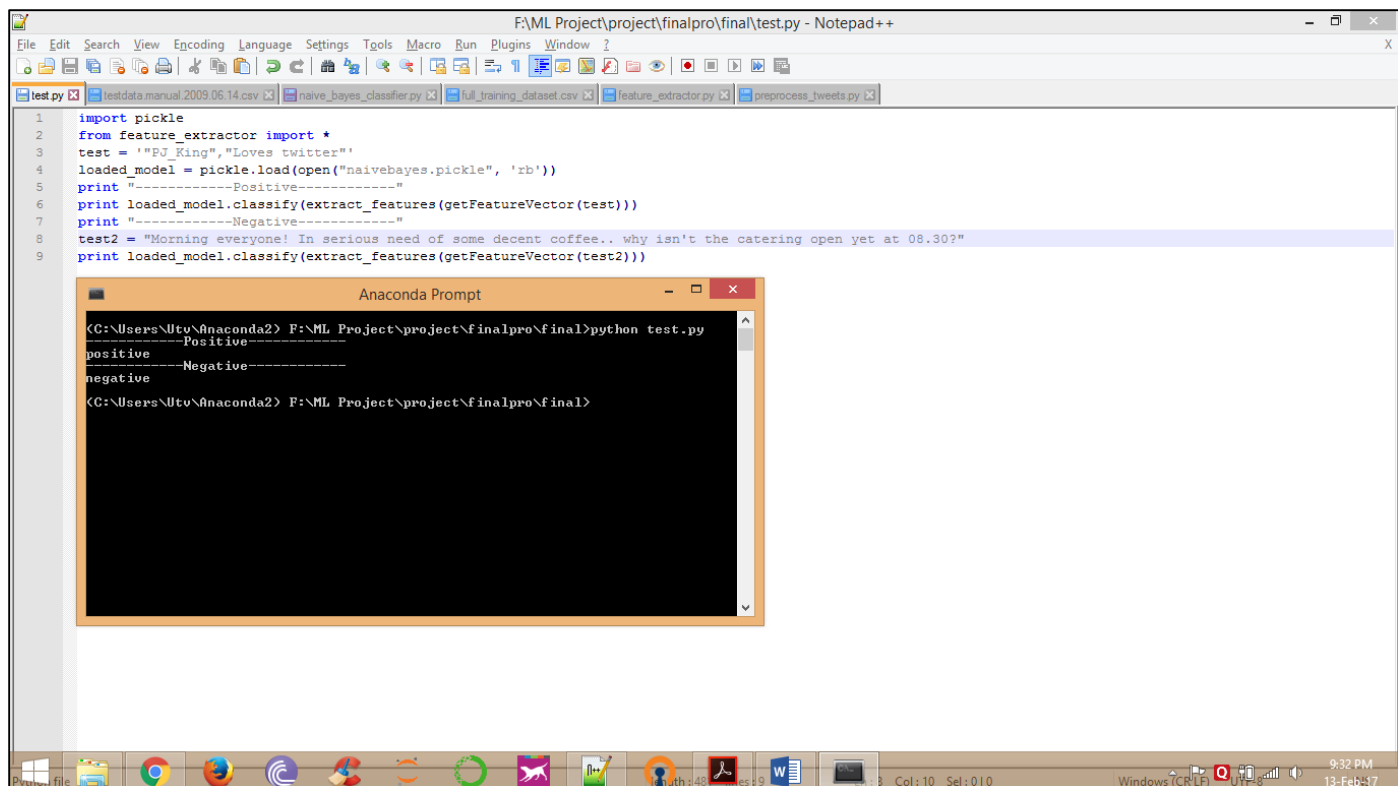
Naive Bayes is a simple model which works well on text categorization. The model is derived from the Bayes theorem. We use a multinomial Naive Bayes model. Class c^* is assigned to tweet d , where

$$c^* = \operatorname{argmax}_c P_{NB}(c|d)$$
$$P_{NB}(c|d) := \frac{(P(c) \sum_{i=1}^m P(f_i|c)^{n_i(d)})}{P(d)}$$

In this formula, f represents a feature and $n_i(d)$ represents the count of feature f_i found in tweet d . There are a total of m features. Parameters $P(c)$ and $P(f_i|c)$ are obtained through maximum likelihood estimates, and add-1 smoothing is utilized for unseen features. We used the Python based Natural Language Toolkit library to train and classify using the Naïve Bayes method.

To test whether our learned model is working correctly, we have given 2 inputs namely test & test2 to test our model. The 'test' contains the positive tweet, so the output for this should be positive. The 'test2' contains the negative tweet, so the output for that should be negative. The model returns the right sentiments for these two test inputs.

Following is the screenshot of the code and output to test the learned classifier:



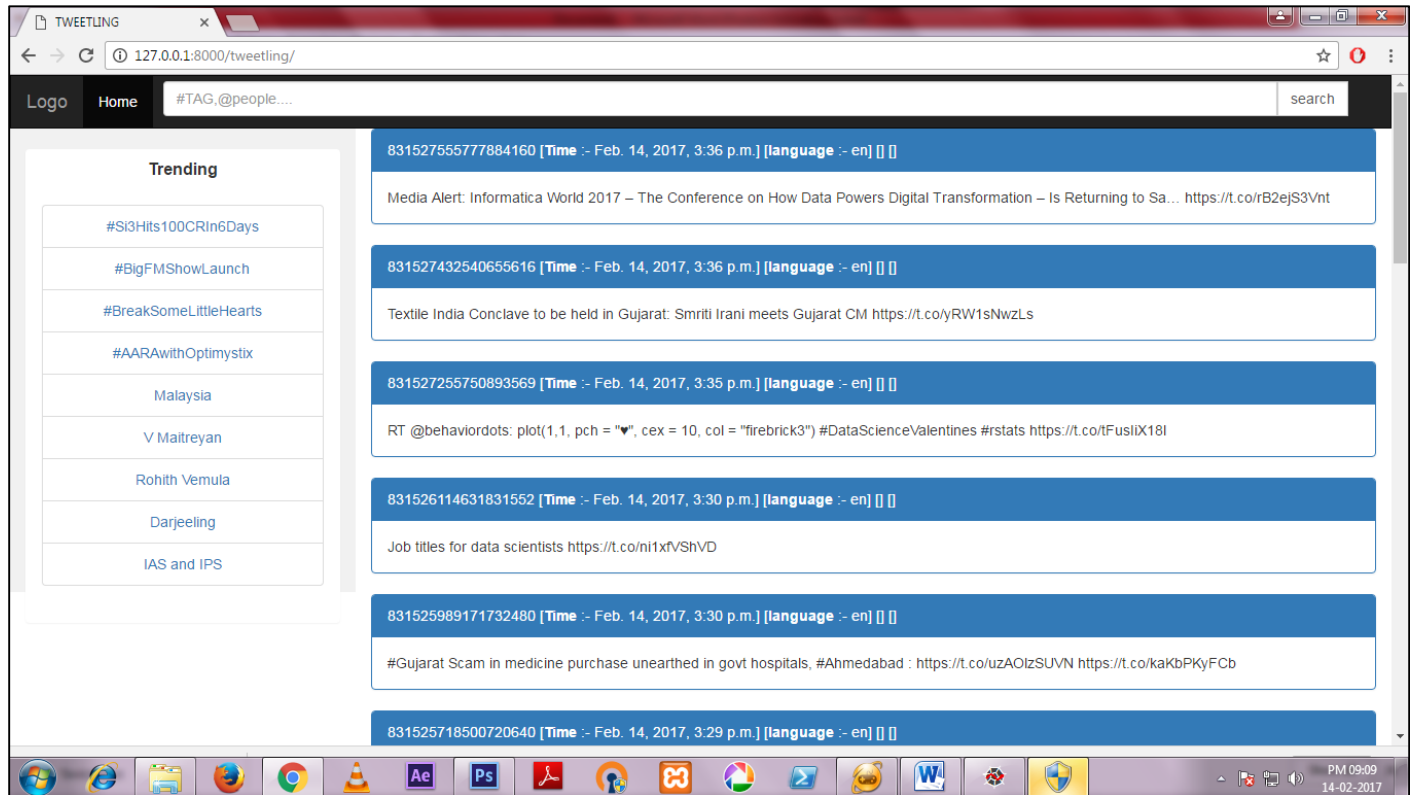
```
1 import pickle
2 from feature_extractor import *
3 test = "PJ_King","Loves twitter"
4 loaded_model = pickle.load(open("naivebayes.pickle", 'rb'))
5 print "-----Positive-----"
6 print loaded_model.classify(extract_features(getFeatureVector(test)))
7 print "-----Negative-----"
8 test2 = "Morning everyone! In serious need of some decent coffee.. why isn't the catering open yet at 08.30?"
9 print loaded_model.classify(extract_features(getFeatureVector(test2)))
```

```
<C:\Users\Utv\Anaconda2> F:\ML Project\project\finalpro\final>python test.py
positive-----
positive
negative-----
negative
<C:\Users\Utv\Anaconda2> F:\ML Project\project\finalpro\final>
```

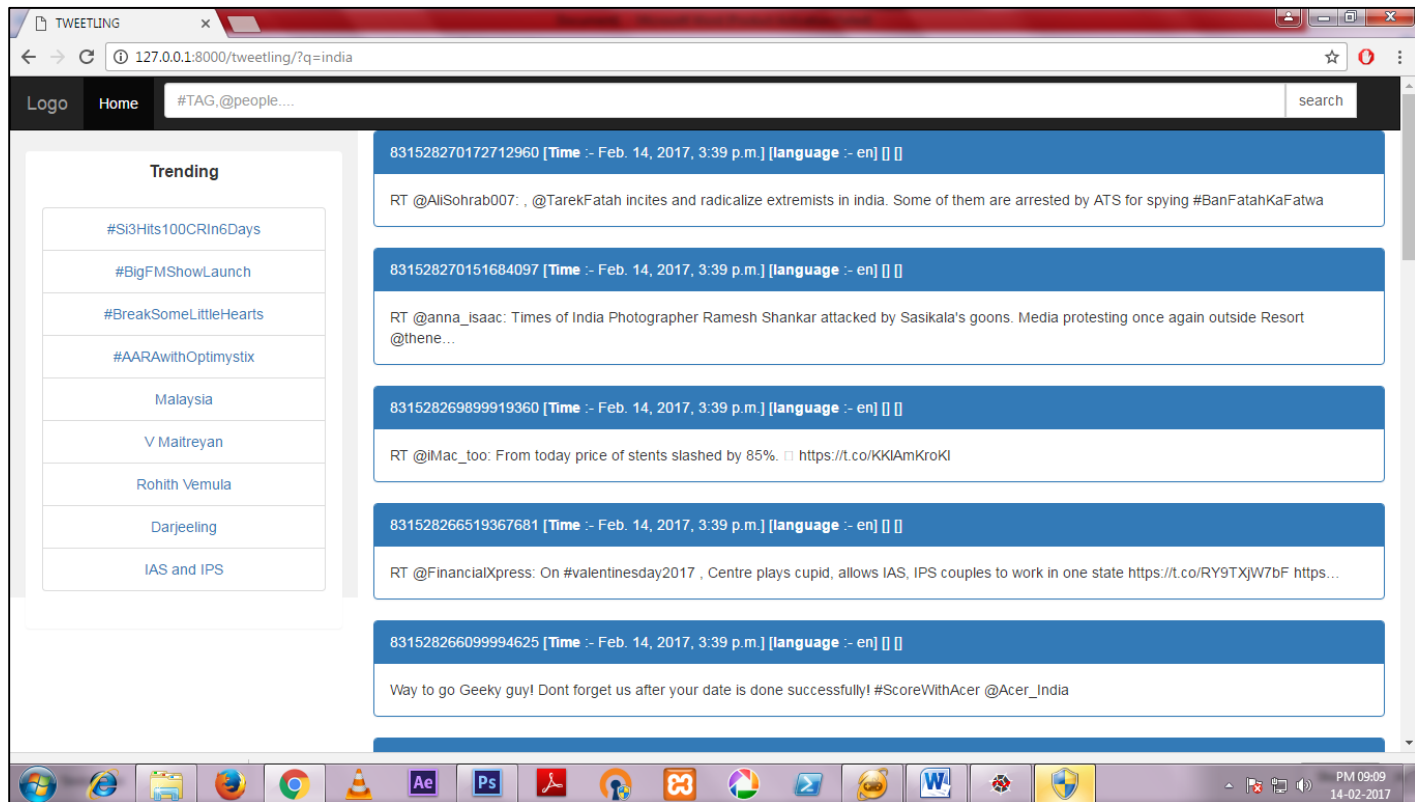
Web Application Interface

Following are the screenshots of web application we have built to visualize the results in user-friendly way.

When you start the web application, by default public tweets will be shown with user ID, Timestamp of posted tweet, type of language and the tweet.

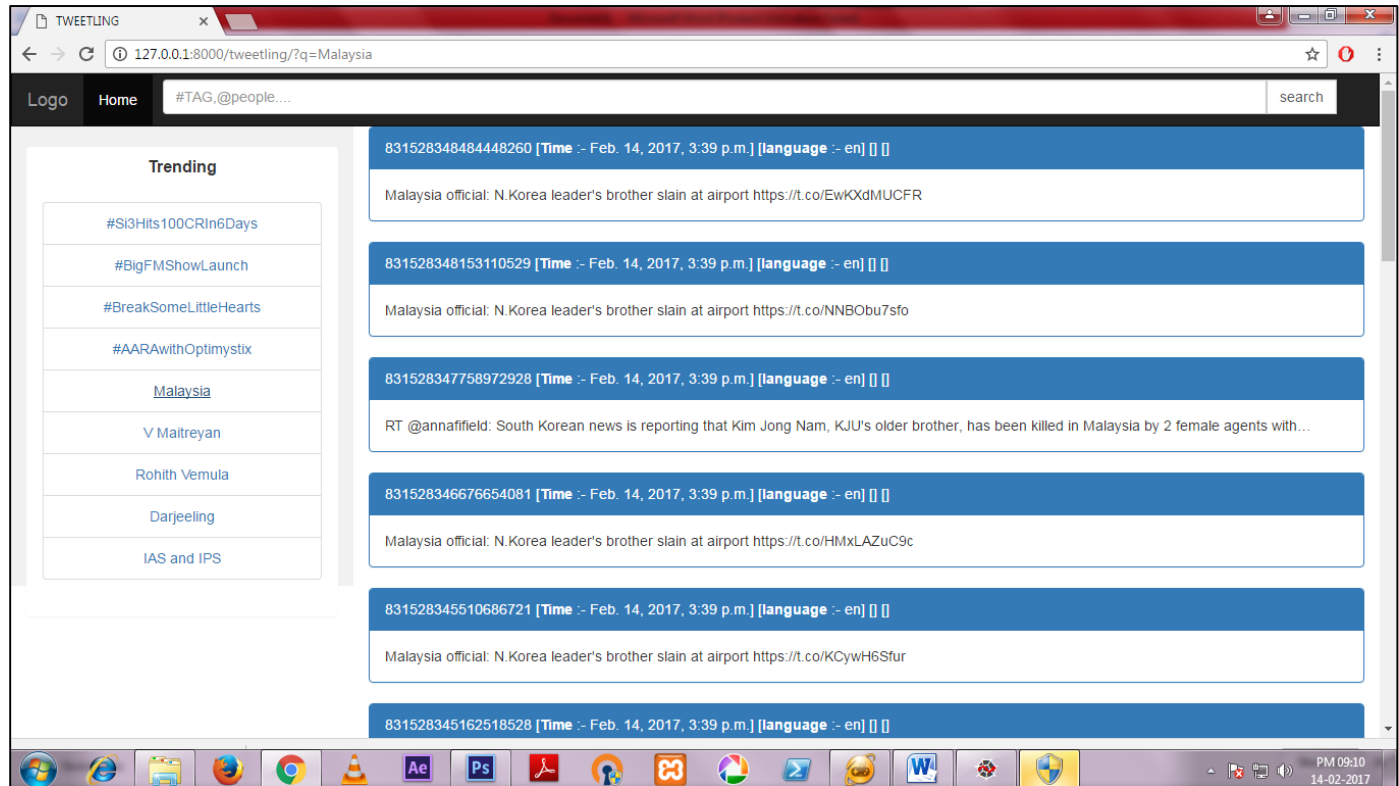


You can search a keyword and tweets related to that keyword will be displayed. Here, we have used keyword 'India'.



On the left side trending topic are given of particular region based on weoid mechanism.

Here we have chosen “Malaysia” as the region and therefore trending topics related to Malaysia have been shown:



Future Work

Machine learning techniques perform well for classifying sentiment in tweets. We believe the accuracy of the system could be still improved. Below is a list of ideas we think could help the classification:-

- **Internationalization** Currently, we focus only on English tweets but Twitter has a huge international audience. It should be possible to use our approach to classify sentiment in other languages with a language specific positive/negative keyword list.
- **Bigger Dataset** The training dataset in the order of millions will cover a better range of twitter words and hence better unigram feature vector resulting in an overall improved model. This would vastly improve upon the existing classifier results.
- **Multiclass Classification** Currently we are classifying the tweets in terms of positive and negative i.e. binary class classification. In future we will try to classify the in terms of positive, negative and neutral i.e. multiclass classification.
- **Different Algorithms** In our approach we have only used Naïve Bayes algorithm. Although there are other algorithms available such as Maximum Entropy, Support Vector Machines, etc. We can use those algorithm to get more accuracy.

Conclusion

Using our approach we have shown that machine learning algorithms can be used to classify the sentiment from twitter or any other microblogging sites.