Institute of Engineering & Management

Abstract

With the advancement of web technology and its growth, there is a huge volume of data present in the web for internet users and a lot of data is generated too. Internet has become a platform for online learning, exchanging ideas and sharing opinions. Social networking sites like Twitter, Facebook, Google+ are rapidly gaining popularity as they allow people to share and express their views about topics, have discussion with different communities, or post messages across the world. There has been lot of work in the field of sentiment analysis of twitter data. This survey focuses mainly on sentiment analysis of twitter data which is helpful to analyze the information in the tweets where opinions are highly unstructured, heterogeneous and are either positive or negative, or neutral in some cases. We have also discussed general challenges and applications of Sentiment Analysis on Twitter.

Background Study

Multinomial Naive Bayes classification algorithm tends to be a baseline solution for sentiment analysis task. The basic idea of Naive Bayes technique is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes. Let's have a brief look at maths.

Given the dependent feature vector *(x_1,…, xn)* and the class *Ck*. Bayes' theorem is stated mathematically as the following relationship:

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)P(x_1, \ldots, x_n \mid C_k)}{P(x_1, \ldots, x_n)}$$

According to the "naive" conditional independence assumptions, for the given class *Ck* each feature of vector *xi* is conditionally independent of every other feature *xj* for *i≠j*..

Thus, the relation can be simplified to

The

$$P(x_i \mid C_k, x_1, \ldots, x_n) = P(x_i \mid C_k)$$

$$P(C_k \mid x_1, \ldots, x_n) = \frac{P(C_k)\prod_{i=1}^{n} P(x_i \mid C_k)}{P(x_1, \ldots, x_n)}$$

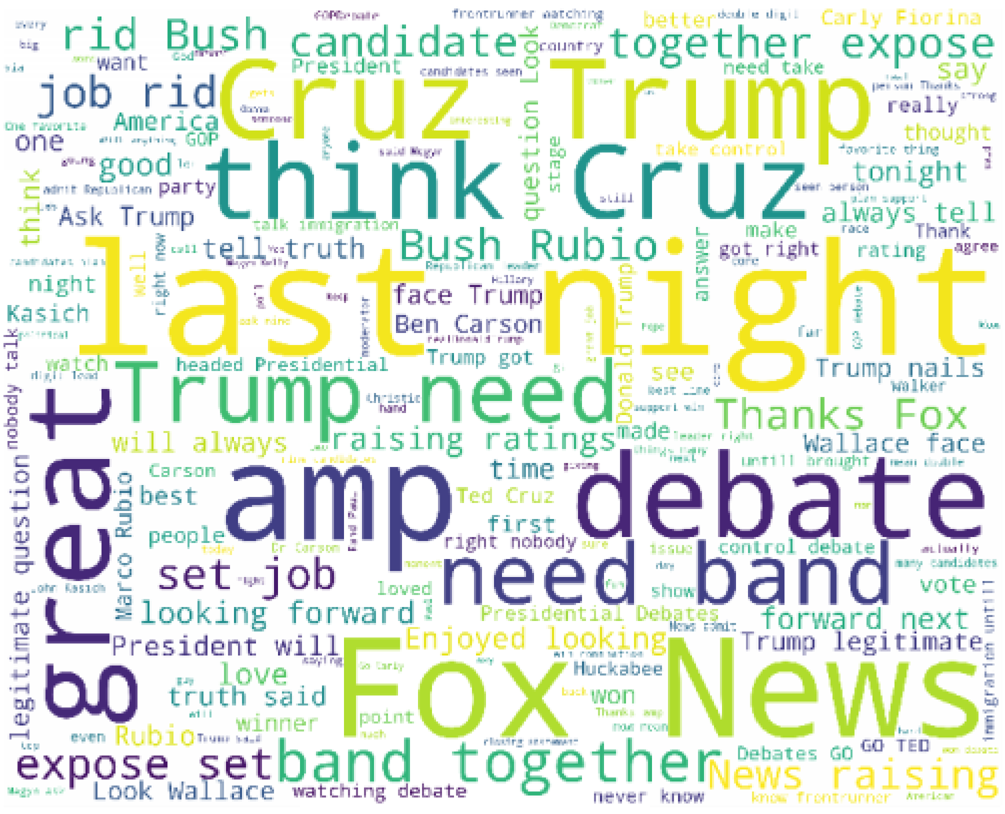
variety of naive Bayes classifiers primarily differs between each other by the assumptions they make regarding the distribution of *P(xi|Ck)*, while *P(Ck)* is usually defined as the relative frequency of class *Ck* in the training dataset.

The multinomial distribution is parametrized by vector *θk=(θk1, ...,θkn)* for each class *Ck*, where *n* is the number of features (i.e. the size of the vocabulary) and *θki* is the probability *P(xi|Ck)* of feature *i* appearing in a sample that belongs to the class *Ck*.

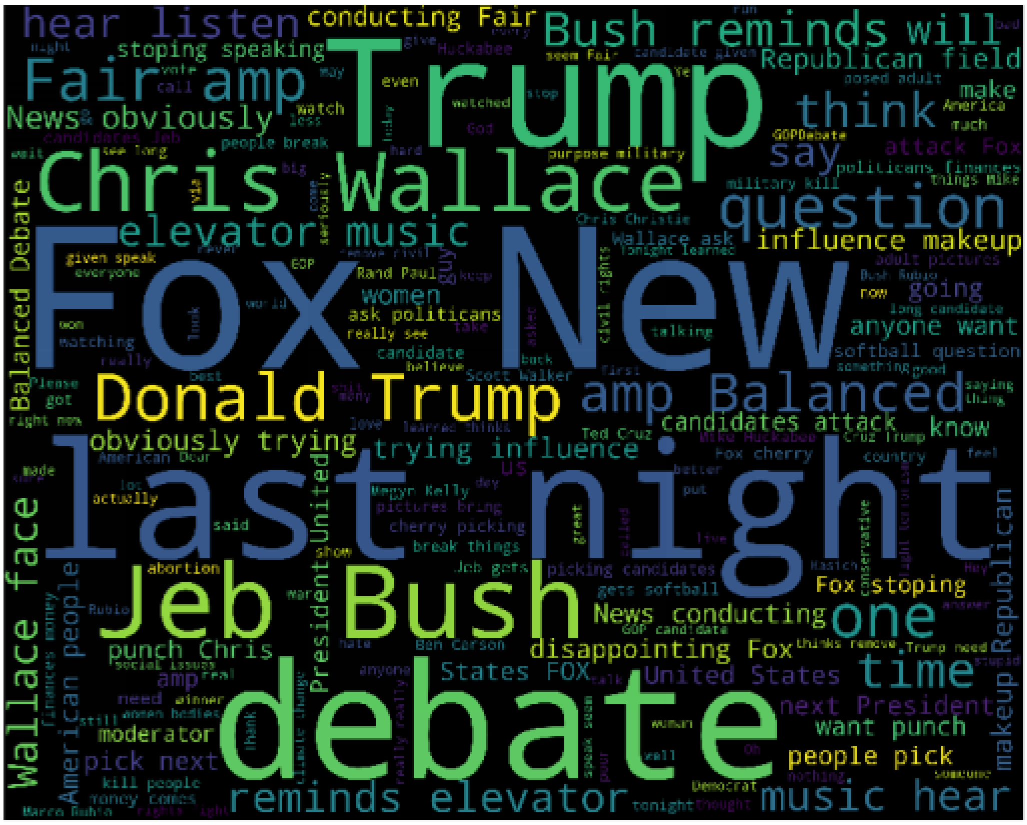
Source Code

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split

import nltk
from nltk.corpus import stopwords
from nltk.classify import SklearnClassifier

from wordcloud import WordCloud,STOPWORDS
import matplotlib.pyplot as plt
%matplotlib inline

from subprocess import check_output

data = pd.read_csv('/home/Desktop/Sentiment.csv')

data = data[['text','sentiment']]


train, test = train_test_split(data,test_size = 0.1)

train = train[train.sentiment != "Neutral"]


train_pos = train[ train['sentiment'] == 'Positive']
train_pos = train_pos['text']
train_neg = train[ train['sentiment'] == 'Negative']
train_neg = train_neg['text']

def wordcloud_draw(data, color = 'black'):
   words = ' '.join(data)
```

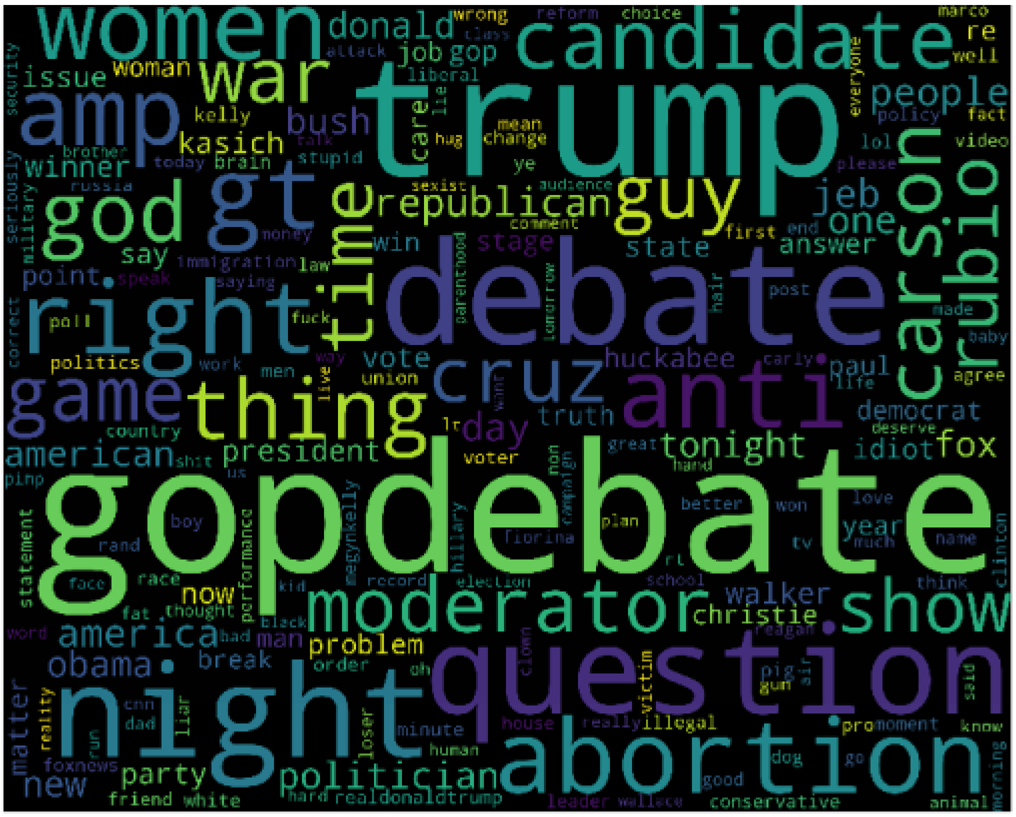
```
    cleaned_word = " ".join([word for word in words.split()
                            if 'http' not in word
                                and not word.startswith('@')
                                and not word.startswith('#')
                                and word != 'RT'
                            ])
    wordcloud = WordCloud(stopwords=STOPWORDS,
                      background_color=color,
                      width=2500,
                      height=2000
                     ).generate(cleaned_word)
    plt.figure(1,figsize=(13, 13))
    plt.imshow(wordcloud)
    plt.axis('off')
    plt.show()

print("Positive words")
wordcloud_draw(train_pos,'white')
print("Negative words")
wordcloud_draw(train_neg)
```



```
tweets = []
stopwords_set = set(stopwords.words("english"))

for index, row in train.iterrows():
    words_filtered = [e.lower() for e in row.text.split() if len(e) >= 3]
    words_cleaned = [word for word in words_filtered
        if 'http' not in word
        and not word.startswith('@')
        and not word.startswith('#')
        and word != 'RT']
    words_without_stopwords = [word for word in words_cleaned if not word in stopwords_set]
```



```
tweets.append((words_without_stopwords, row.sentiment))

test_pos = test[ test['sentiment'] == 'Positive']
test_pos = test_pos['text']
test_neg = test[ test['sentiment'] == 'Negative']
test_neg = test_neg['text']



def get_words_in_tweets(tweets):
    all = []
    for (words, sentiment) in tweets:
        all.extend(words)
    return all

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    features = wordlist.keys()
    return features
w_features = get_word_features(get_words_in_tweets(tweets))

def extract_features(document):
    document_words = set(document)
    features = {}
    for word in w_features:
        features['contains(%s)' % word] = (word in document_words)
    return features
wordcloud_draw(w_features)
```



```
training_set =
nltk.classify.apply_features(extract_features,tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)

neg_cnt = 0
pos_cnt = 0
for obj in test_neg:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Negative'):
        neg_cnt = neg_cnt + 1
for obj in test_pos:
    res =  classifier.classify(extract_features(obj.split()))
    if(res == 'Positive'):
        pos_cnt = pos_cnt + 1

print('[Negative]: %s/%s '  % (len(test_neg),neg_cnt))
print('[Positive]: %s/%s '  % (len(test_pos),pos_cnt))

[Negative]: 864/819
[Positive]: 225/87
```

Conclusion and Future Work:

Twitter acts as a utility where one can send their SMSs to the whole world. It enables people to instantaneously get heard and get a response. Since the audience of this SMS is so large, many a times responses are very quick. So, Twitter facilitates the basic social instincts of humans. By sharing on Twitter, a user can easily express his/her opinion for just about everything and at anytime. Friends who are connected or, in case of Twitter, followers, immediately get the information about what's going on in someone's life. This in turn severs another human emotion—the innate need to know about what is going on in someone's life. Apart from being real time, Twitter's UI is really easy to work with. It's naturally and instinctively understood, that is, the UI is very intuitive in nature.

Each tweet on Twitter is a short message with maximum of 140 characters. Twitter is an excellent example of a microblogging service. As of July 2014, the Twitter user base reached above 500 million, with more than 271 million active users. Around 23 percent are adult Internet users, which is also about 19 percent of the entire adult population. If we can properly mine what users are tweeting about, Twitter can act as a great tool for advertisement and marketing. But this not the only information Twitter provides. Because of its non-symmetric nature in terms of followers and followings, Twitter assists better in terms of understanding user interests rather than its impact on the social network. An interest graph can be thought of as a method to learn the links between individuals and their diverse interests. Computing the degree of association or correlations between individual's interests and the potential advertisements are one of the most important
applications of the interest graphs. Based on these correlations, a user can be targeted so as to attain a maximum response to an advertisement campaign along with followers' recommendations.