Sentiment Analysis of Twitter Data

15-383 Course Project

By : John Naguib
Under Supervision of Prof. Behrang Mohit

Introduction:

There are millions of tweets that are posted every day about different topics. From the artificial intelligence and machine learning point of view this is a great source of data to be able to make computers understand humans. The objective of this work, is to try to classify tweets into positive, negative or neutral using a training set and different features collected from the tweet.

First Trial for classification

First, I used the Naive bayes to get the prediction of the class of a tweet. This include 2 phases, the training phase and the testing phase. In the training phase, I read from the training file the class of the tweet either positive, negative or neutral, and the tweet. Then add the words of the tweet to a dictionary based on their class so for this task there were 3 dictionaries, dpositives, dnegatives and dneutral. Where the keys of each of the dictionaries are the words in that class and the values are their frequencies in that specific class.

After adding the words to the dictionaries from all the tweets, I got the probability of each class by counting the number of tweets that are in that class divided by the total number of tweets. For the words probability, I divide the frequency of the word by the sum of the frequencies of words in that class.

For the testing phase, The system gets a file as an input and read the tweet and then get the sum of the log of the probability of each word, and then add it to the log of the probability of the class. The system does that for each class and gets the maximum probability of the three classes then this is the predicted class.

After training the system for task A and testing it with the development set the result produced by the scorer was 0.0911 for the average fscore. After doing the same for task B, the result was 0.3960 for the average fscore.

The Improved System

I then focused on task A to improve the score. First I tried to come up with more features other than the bag of words (depending on the probabilities of the words in the tweet for each class).

There are different features in the system that I use to classify the tweet, The features are listed below.

The features are:

1- Type of Word:

- The tweet contains positive words
- The tweet contains negative words

2- Emoticons

I. There is an emoticon in the tweet

II. The emoticon is positive

III. The emoticon is negative

3- Hash tags

I. There is a hash tag in the tweet

II. The hash tag is positive

III. The hash tag is negative

4- Not

I. The tweet contains not

II. The word after not is positive

III. The word after not is negative

5- Elongated Words (at the beginning or the end of the word)

I. There is an elongated word in the tweet

II. The word is positive

III. The word is negative

6- Upper cased words

I. There is an upper cased word in the tweet

II. The word is positive

III. The word is negative

7- User

I. There is an @'user' in the tweet

8- Bag of Words:

- Presenting words as binary feature whether the words exist in tweet or not.

Details on the Features:

1- Type of Word:

One of the basic features that can help in the prediction of the class, is the type of words in the tweet, If there is a positive or a negative word in the tweet.

In the system, there is a function responsible for checking if there is a positive/ negative word in the tweet. Using a list of positive and negative words from the NRC-CANADA team. The function returns two values one for the existence of positive words and one for the existence of negative words. For each tweet I check if it contains any of these words, If yes, it returns true for the corresponding value (positive or negative).

2- Emoticons:

Many users use emoticons to help express their emotions in a tweet. Based on that, I though this can help in identifying the class of the tweet by checking the type of the emoticons that are in the tweet whether positive, negative or neutral if they exist.

Based on work done by DataGenetics, They produced a list of the top frequency emoticons used in twitter, I got some of the highest frequencies and I added some that I thought they are frequent but was not in DataGenetics list. Moreover, I added to them some from wikipedia because there are different types of emoticons for the same emotion (Eastern and Western for example). After that I put all the gathered emoticons in a file where I classified them to positive, negative or neutral since they are not classified by DataGenetics.

The system uses this list of emoticons to check for the emoticons in the tweet and returns 3 binary values, one for the existence of an emoticon in the tweet, one for existence of positive emoticon, and one for the existence of the negative emoticon.

3- Hash tags:

Hash tag is a very important component in Twitter that contains important data that can help significantly in the process of classification. Moreover, positive or negative word in the hash tag gives a higher significance that this tweet is positive or negative compared to a positive or negative word in the body of the tweet.

The system uses a function that checks if there is a hash tag in the tweet, and if there is, check whether it is positive or negative, using the list obtained from NRC-CANADA 's work. But that will not show

the higher significance since it is in the hash tag. So the function adds the words that are in the hash tag, to the tweet as a normal words. So if for example there is a positive word, then it will be counted in the first feature too, so it will be counted twice. This is how I implemented the higher significance of the words that are in hash tags.
However there was a problem in hash tags that they are not words split by spaces, so I simplified this problem so that I split the words by the upper case letters, and check the words. For example,
#GoodGame → Good , Game

4- Not

Using not in a tweet can be very tricky, since not good is not the same as good, it even sometimes means the opposite.
Based on that, The system uses a function that returns 3 binary values. One for the existence of not in the tweet, The second is checking the word following not whether it is positive or not and the third is checking whether the word following not is negative or not. However as a pre-process all the “isn't” , “wasn't” .. etc are converted to “is not” or “was not”.

5- Elongated words

People tends to use elongated word to emphasize on specific words or emotions. The system uses this information to help in the classification process.
The system uses a function that checks if the word is elongated or not and returns 3 binary values, Whether the word is elongated, If the word is positive or not, and the word is negative or not. Checking all possible elongations is a very difficult problem because there might be different elongations in the same word with different letters, to get all the possibilities will create a huge list that you need to check. As a simplification for this problem, The system only checks the elongation in the beginning or the end of the word where letter is repeated more than two times. If the letter is repeated more that two times, It checks for the word in the positive/negative words list with word where the repeated letter exists once or twice and removing the other repetitions of the letter.

6- Upper Cased words

Sometimes people use upper cased letters to emphasize on some words. By checking these words it can help in the classification process since the user is emphasizing on this word.
The same concept as the hash tag should be applied here, There should be some way to show the emphasizing that the user is making.
The system uses a function that checks if there is an upper cased word in the tweet, And If the upper cased word is positive or negative. Moreover the positive or negative words will be detected too by feature 1.

7- User

From my observation on the tweets, the tweets that have @USER in them tends to be positive, so I added this as a feature so that it can better classify the tweet.

8- Bag of Words

Getting all the words that are in a specific class and check which of them exist in the tweet required to be tested.

The Calculation of the probability:

As seen in the Details of Features, All the features are represented with one or more binary values.
The system split the tweet to words after preprocessing the tweet. For each tweet, It checks for the previous features on the preprocessed tweet. After that It gets the sum of the logs of the probabilities of features after smoothing and add to it the log of the probability of the class. The maximum value of the three probabilities (the three classes) is the predicted class.
The probability of the class is calculated by dividing the total number of tweets in a specific class by the total number of the tweets.
For the probability of the features, since they are binary values, for each feature, it is counted how many times it is 1 and divide that by the total number of 1's in the class.

Another Classifier

For the other classifier, I chose SVM, Support Vector Machine. It is a classifier that takes each instance as a point in a graph whose dimension is the number of features. Moreover it takes the class that this point (instance) belong to. For example, for the sentiment analysis, each point represent a tweet where the dimension of the graph is the number of features in the system, which in my sentiment analysis system is a very high dimensional graph, and the class is either positive, negative or neutral.

The procedure for classifying in SVM:

Training in SVM produces a model, which is a high dimensional graph with each instance as a point on that graph where it also saves the class of the instance. It then tries to create a line or a hyper plane that separate the different classes. However, usually, it is not that simple, so It uses a mapping function called a kernel function, that maps the points in a way that is easier to be separated.
For the prediction, it converts the test instance to a point on the model (the graph of the trained data). It then checks the closeness of the test point to the trained data, or in another words in which sector of classes this point belongs. The closest class is the class that SVM produces as prediction.

SVM is helpful in my system because in the system, there are a lot of features and with the help of SVM it can represent them together in a high dimensional graph that will make the classification more accurate. Moreover with the help of the kernel function it makes it more accurate.

RESULTS:

All results are produced by the scorer script provided.

1- Naive Bayes:

I. Development set:

Result with all features and bag of words:

Score:

class	prec	recall	fscore
Positive	(552/814) 0.6781	(552/648) 0.8519	0.7551
Negative	(209/320) 0.6531	(209/430) 0.4860	0.5573
Neutral	(0/1) 0.0000	(0/57) 0.0000	0.0000

average(pos and neg) **0.6562**

Result with Bag of Words only:

Score

class	prec	recall	fscore
Positive	(306/538) 0.5688	(306/648) 0.4722	0.5160
Negative	(178/478) 0.3724	(178/430) 0.4140	0.3921
Neutral	(7/119) 0.0588	(7/57) 0.1228	0.0795

average(pos and neg) **0.4540**

II. Test Set:

Result with all features and bag of words:

Score

class	prec	recall	fscore
Positive	(2389/3266) 0.7315	(2389/2734) 0.8738	0.7963
Negative	(777/1166) 0.66641	(777/1541) 0.5042	0.5741
Neutral	(0/3) 0.0000	(0/160) 0.0000	0.0000

average(pos and neg) **0.6852**

Result with Bag of Words only:

Score

class	prec	recall	fscore
Positive	(1225/1984) 0.6174	(1225/2734) 0.4481	0.5139
Negative	(703/1978) 0.3554	(703/1541) 0.4562	0.3995
Neutral	(8/473) 0.0169	(8/160) 0.0500	0.0253

average(pos and neg) **0.4594**

2- SVM

I. Development set:

Result with all features and bag of words:

Score

class	prec	recall	fscore
Positive	(544/764) 0.7120	(544/648) 0.8395	0.7705
Negative	(242/360) 0.6722	(242/430) 0.5628	0.6127
Neutral	(6/11) 0.5455	(6/57) 0.1053	0.1765

average(pos and neg) **0.6916**

Results with Bag of Words only

Score

class	prec	recall	fscore
Positive	(425/768) 0.5534	(425/648) 0.6559	0.6003
Negative	(103/312) 0.3301	(103/430) 0.2395	0.2776
Neutral	(8/55) 0.1455	(8/57) 0.1404	0.1429

average(pos and neg) **0.4390**

II. Test Set:

Result with all features and bag of words:

Score

class	prec	recall	fscore
Positive	(2281/2948) 0.7737	(2281/2734) 0.8343	0.8029
Negative	(955/1433) 0.6664	(955/1541) 0.6197	0.6422
Neutral	(9/54) 0.1667	(9/160) 0.0563	0.0841
average(pos and neg)			**0.7226**

Results with Bag of Words only

Score

class	prec	recall	fscore
Positive	(1824/2990) 0.6100	(1824/2734) 0.6672	0.6373
Negative	(411/1236) 0.3325	(411/1541) 0.2667	0.2960
Neutral	(4/209) 0.0191	(4/160) 0.0250	0.0217
average(pos and neg)			**0.4667**

From the previous results, We can see that only considering the bag of words is not enough to get a decent prediction of the class of a tweet. By adding the different features, we can see that the fscore increased by around 0.25. Moreover, As expected, the SVM was more accurate than the Naive Bayes with around 0.04 increase in the fscore.

Papers and tools used:

The main work that I studied and got many ideas from was the work of NRC-Canada, the winning team in sentiment analysis conference. Moreover, I used their list of words. Most of the features and the ideas I got from their work.

Another work that helped me was the work of DataGenetics where I got the list of the most frequent emoticons.

Bibliography:

1- Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu, NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets

2- DataGenetics, from : http://datagenetics.com/blog/october52012/index.html retrieved on : Dec 12 2013