POLITICAL SENTIMENT ANALYSIS USING TWITTER

MARCH 16, 2016

AMANDEEP SHARMA/ROSHAN SHETTY

POLITICAL SENTIMENT ANALYSIS USING TWITTER

BACKGROUND:

We were having a discussion on how we could put some machine learning techniques that we learnt to some practical use. Recently, Donald Trump is making a lot of headway in the Republican Caucus. And that has created a lot of buzz in Twitter. We decided to try out a machine learning algorithm to figure out the sentiment of Twitter users around Trump's election campaign. Thus, we started this project.

DATA EXTRACTION AND CLEANING:

We using #Trump keyword to download tweets from Twitter. Initially, we were planning to automatically label tweets as positive and negative based on the smileys used in the tweets. Unfortunately, the number of smileys used in tweets was very small. Also, the cases where sarcasm was involved in the tweets could not be handled well. Hence, we spent a few days manually labelling tweets as positive and negative. It was a painstaking process. In the end, we manually labelled 556 tweets as positive and 600 tweets as negative. We used 400 positive and negative tweets each for training purposes. The rest were set aside for testing. A standard dictionary obtained online[2] was used. It contained words along with their corresponding positive and negative scores.

Each tweet was cleaned as per the following. Words containing multiple consecutive characters were trimmed down. For e.g., "cooool" was converted to "coool", not "cool" for added emphasis. '@' handler words and url's were removed. Special characters were removed but the words were kept. Hashtag character was removed from the hastag words. Politician names like Ted Cruz, Hillary Clinton, Donald Trump and Marco Rubio were also removed. Smileys were removed after that. We got rid of all words with less than 4 characters. We replaced words with their closest matches from the dictionary to convert words like 'willing' to 'will'. That does cause an issue by converting words like 'daring' to to 'dar'. Using Python's built-in difflib.get_close_matches function was considered but it caused other issues. For e.g. it converted 'MakeAmericaGreatAgain' to a common English word. That would cause discrepancies in classifying tweets. Stop words were removed using a stop word dictionary found. Finally, all the white spaces are stripped off the final tweet.

NAIVE BAYES MODELING:

Before we started implementing a particular algorithm we looked into various algorithms we could use for classification. Based on our research and inspired by few papers [4, 5] we decided to implement naive bayes model for classifying the sentiment of the tweet.

Approach:

1. We first cleaned the data based on the steps mentioned in data extraction and cleaning section

2. Manually classified 600 negative and 556 positive tweets
3. Used 800 tweets (400 positive + 400 negative) to train our naive bayes classifier and used the others to test our algorithm

Classifier

1. First we created a python dictionary using sentiment dictionary[2] with each word having the positive score, negative score and the part of speech it belongs to
2. Using our training data, for each word present in our corpus we calculated:
    a. PosProb[a] = Probability of the word occurring in a positive tweet
    b. NegProb[b] = Probability of the word occurring in a negative tweet
3. If the word is present in the tweet but not in the dictionary e.g. MakeAmericaGreatAgain, we assign those words a positive score of 0.5 and negative score of 0.5 and PosProb/NegProb based on the calculation below
4. If the word is present in the dictionary we keep the positive score and negative score that is already provided by the dictionary and append the PosProb and NegProb to the word

a. PosProb = $\frac{Occurrence\ of\ the\ word\ in\ positive\ tweets}{total\ number\ of\ positive\ tweets\ in\ training\ set}$

Here the numerator increases in count by only one even if a word may be present in the tweet more than once. The reason we did this was to clean spam in our tweet, where a user might have written the same word again without adding significant meaning to the sentiment
E.g. I have done a great great Job, I am a great person. In this scenario we are incrementing count of great by 1 and not by 3

Similarly, we calculate the negative probability as:

b. NegProb = $\frac{Occurrence\ of\ the\ word\ in\ negative\ tweets}{total\ number\ of\ negative\ tweets\ in\ training\ set}$

Once the above step is completed we get our final dictionary with positive score (PosScore), negative score (NegScore), positive probability (PosProb) and negative probability (NegProb) Additionally, in case we encounter a word in a tweet that is not present in our training corpus we assign them probability of 0.0001

The final calculation of sentiment is based on the following calculation:

Here we multiply probability of each word in the tweet with the weight (in our case the positive score/negative score) to get the positivity/negativity.

Measure of positivity:

When word is not present in training corpus:

`positive_score = positive_score` $\times$ `.0001` $\times$ $E^{\mathbf{PosScore}}$

When word is present in the training corpus:

$$\texttt{positive\_score} = \texttt{positive\_score} \times \texttt{PosProb} \times E^{\textbf{PosScore}}$$

Measure of negativity:

When word is not present in training corpus:

$$\texttt{negative\_score} = \texttt{negative\_score} \times 0.0001 \times E^{\textbf{NegScore}}$$

When word is present in the corpus:

$$\texttt{negative\_score} = \texttt{negative\_score} \times \texttt{NegProb} \times E^{\textbf{NegScore}}$$

If measure of positivity is greater than measure of negativity then it is positive else it is negative. If they both are same then we mark the sentiment as neutral

TESTING

We performed 3-fold validation on the data by having 800 total positive and negative training tweets and 356 total positive and negative testing tweets. The following accuracies were found on the 3 testing sets:

1. 75.84%
2. 78.08%
3. 80.61%

The average cross validation accuracy score comes out to be 78.17%

CONCLUSION

We modified Naive Bayes model with weights of positivity and negativity of the word and it was able to give us an accuracy > 78 %. Classifying more tweets and running the model on a larger training set might help us improve the accuracy of the model.
In future, we might apply high degree models and narrow down our feature vector to include only the most relevant ones.

REFERENCES

[1] http://www.laurentluce.com/posts/twitter-sentiment-analysis-using-python-and-nltk/
[2] SentiWordNet v3.0.0
http://sentiwordnet.isti.cnr.it
[3] http://www.webconfs.com/stop-words.php
[4] Agarwal, A., Boyi, X., Vovsha, I., Rambow, O. and Passonneau, R. 2011. Sentiment Analysis of Twitter Data. *In the proceedings of Workshop on Language in Social Media, ACL, 2011*
[5] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. Technical report, Stanford.