Twitter Sentiment Analysis on 2016 Republican Party (GOP) Presidential Debates

CSE-5523 Project

By

Zhenyu Wu

Yiran “Lawrence” Luo

Fan Bai

Hongsen Shi

The Ohio State University

2016

Instructor: Mikhail Belkin

Abstract

Using twitter data, we are introducing a novel approach to analyze the sentiment influence of public figures' behavior or speech on twitter users. In this project, we are focusing on the 2016 GOP candidates. They are D. Trump, J. Bush and T. Cruz. Among them, D. Trump is the most controversial and leading the most topics on twitter. Labeling sentiment polarity for each tweet is the most difficult part of our project. We classify sentiment into 3 polarities, positive, negative, neutral. In order to apply machine learning method, we need to extract features from tweets. Using these features, models can learn to classify tweets. Analyzing tweets with regard to a candidate on a specific debate can reveal his sentiment influence on twitter. Our main contributions are: (1) Taking the neutral sentiment polarity into account (2) Doing with negation in tweets (3) Handling typos in tweets (4) Investigating the impact of stemming (5) Introducing a multi-classifier architecture based on sentiment coexisting assumption (6) Building 7 different models (including the most powerful Deep Neutral Network model).

Keywords: GOP Debates, sentiment polarity, feature extraction, negation, stemming, dealing with typos, multi-classifier architecture, Deep Neutral Network, sentiment coexisting.

Content

List of Figure

List of Table

Introduction

1. Background

Microblogging provider Twitter is a very popular communication tools for Internet and Mobile users. It also provides an important site for expressing political opinions throughout the world. In the year leading up to the June 2016 Republican Party (GOP) presidential primaries in the United States, we have developed a tool for real-time analysis of sentiment expressed through Twitter towards three presidential candidates, Donald Trump, Ted Cruz, and Jeb Bush – the first two of them remain in the running as of this writing. With this analysis, we seek to explore whether Twitter provides insights into the unfolding of the campaigns and indications of shifts in public opinion.

With the dramatic rise of text-based social media, millions of people broadcast their thoughts and opinions on a great variety of topics. As one of the social network sites, Twitter allows users to post tweets, messages of up to 140 characters, and it has a great number of users. According to the report [1], more than 100 million users posted 340 million tweets a day in 2012. More than two thirds of U.S. congress members have created a Twitter account and many are actively using Twitter to reach their constituents [2] [3]. It is convenient for research to get a very large number of useful messages.

2016 GOP presidential primaries began on March 23, 2015. As one of the major candidates, Donald Trump has quickly emerged as the Republican nomination front-runner since he entered the race. However, remarks on him have been highly controversial among the public. So we want to develop a system using machine learning method, which can help explore how some events of GOP candidates affect public opinion, and offers the public, the media, politicians and scholars a new and timely perspective on the dynamics of the electoral process and public opinion.

2. Overview

In this project, we classify “tweets” into 3 different sentiment polarity: positive, negative and neutral. We build seven different models: Naïve Bayes, Kernel Support Vector Machines (SVM), Linear SVM, Max Entropy, and Deep Neural Network with one (DNN #1), two (DNN #2), and three (DNN #3) hidden layers. Through the cross validation process, the DNN #3 has the best

performance. Next we try to find the best feature space using DNN #3. Four different feature spaces are tested here: Raw feature space with 11,414 features, stemming feature space with 11,414 features, negation feature space with 13,239 features, and the combination of stemming and negation feature space with 11,071 features. And we find that the performance of raw feature space is the best. Then we use the best model (DNN#3) to analyze the public sentiment based on 1 thousand tweets of Trump, Bush and Cruz respectively within 08/07/2015~08/09/2015, and compare the results with the polling data of media (CNN, Quinnipiac, and etc.) investigation as the further validation part. Finally we analyze 14 thousand recent tweets towards two candidates-Trump and Cruz to predict the public sentiments on them. At the last part, the top 50 informative features are extracted, which may stand for the hottest topics of 2016 GOP debates.

3. Outlines

The rest of the paper is organized as follows. In section 2, we evaluate some previous works related with the twitter sentiment analysis. In section 3, we give details about the data collection. In section 4 we discuss our pre-processing and feature extraction techniques. In section 5 we introduce the seven classifiers used in this project and synthesizing sentiment scores. In section 6 we present procedure of experiment setups. In section 7, we discuss the experiment results and give the recent prediction of public sentiment. We conclude and give future directions of project in section 8.

Literature Review

Sentiment analysis is a growing area of Natural Language Processing task at many levels of granularity. Starting from being a document level classification task (Turney, 2002 [4]; Pang and Lee, 2004 [5]), it has been handled at the sentence level (Hu and Liu, 2004 [6]; Kim and Hovy, 2004 [7]) and more recently at the phrase level (Wilson et al., 2005 [8]; Agarwal et al., 2009 [9]). However, the informal and specialized language used in tweets, as well as the very nature of the microblogging domain make Twitter sentiment analysis a very different task.

Go et al. (2009) [10] use tweets ending with positive emoticons like ":)" ":-)" as positive and negative emoticons like ":(" ":-(" as negative. They test three classifiers: Naive Bayes, Max Entropy and Support Vector Machines (SVM), and they report SVM outperforms other classifiers. In terms of feature space, they try a Unigram, Bigram model in conjunction with parts-of-speech (POS) features. They note that the unigram model is better than all other models. Specifically, bigrams and POS features do not help.

Agarwal et al. (2011) [11] build models for two classification tasks: a binary task of classifying sentiment into positive and negative classes and a 3-way task of classifying sentiment into positive, negative and neutral classes. They experiment with three single models and some model combinations: unigram model, a feature based model, a tree kernel based model, combining unigrams with the features and combining the features with the tree kernel. For the feature based model, they use uses only 100 features achieves similar accuracy as the unigram model that uses over 10,000 features. And they conclude that tree kernel and feature based models both outperform the unigram.

Gamon et al. (2004) [12] perform sentiment analysis on feedback data from Global Support Services survey. One aim of their paper is to analyze the role of linguistic features like POS tags. They perform extensive feature analysis and feature selection and demonstrate that abstract linguistic analysis features contributes to the classifier accuracy.

Data Collection

1. Training Data

In this project, a training dataset with about thirteen thousand “tweets” is provided by the Kaggle website [13]. All of these “tweets” are related with the early August GOP debate in 2015. For each tweets in this dataset, Kaggle provides time information, the sentiment label (Positive, Negative, and Neutral), and sentiment confidence. An example of the “tweet” information is shown in Fig. Based on these information, we also build a model for of classifying sentiment into positive, negative and neutral classes. And each class has a corresponding percentage of sentiment confidence. Furthermore, because all of the classifiers we used in this project are supervised methods, we randomly extract a small part of training data for the cross validation.

Table 1 Example for Training Data

Tweets	Sentiment	Sentiment Confidence
RT @GregAbbott_TX: @TedCruz: "On my first day I will rescind every illegal executive action taken by	Positive	0.6332
Me reading my family's comments about how great the #GOPDebate was http://t.co/gIaGjPygXZ	Negative	0.6957

2. Validation Data

After training process, the well-trained classifier will be used for analyzing the sentiments of 3 thousand “tweets” related with three candidates: Trump, Bush, and Cruz. Every one thousand “tweets” corresponds to one candidate, which are collected directly from twitter website within 08/07/2015~08/09/2015. This analysis results are compared with the polling data of media investigation as the further validation with “Ground Truth”.

3. Media Investigation

In further validation part, the polling data is provided by CNN [14] and Quinnipiac [15]. These two media interviewed with 1,001 adult Americans separately to investigate the public sentiments towards all of the 2016 GOP candidates within 08/13/2015~08/16/2015. We use these two polling data as our “Ground” validation standard, which are listed in Table.

Table 2 Polling Data from CNN, Quinnipiac and Bloomberg

Candidates	Media	Favorable	Unfavorable	Never Heard	No opinion
Donald Trump	CNN	36%	59%	1%	3%
Ted Cruz		27%	35%	24%	14%
Jeb Bush		34%	56%	6%	4%
Donald Trump	Quinnipiac	36%	54%	8%	2%
Ted Cruz		28%	34%	38%	0%
Jeb Bush		32%	41%	26%	1%
Donald Trump	Bloomberg	51%	40%	0%	9%
Ted Cruz		52%	26%	0%	22%
Jeb Bush		57%	31%	0%	12%

4. "Groundless" Prediction Data

Finally, the well-trained model will predict the public sentiment based on most recent "tweets", of which every one thousand "tweets" corresponds to one candidate: Trump or Cruz. These total 14 thousand "tweets" were collected from twitter directly within 04/12/2016~04/18/2016. This result stands for most recent public sentiment towards these two candidates.

Pre-processing and Feature Extraction

1. Preprocessing (Lexical Level)

The collected raw tweets contain a lot of noise. The objective of preprocessing is to remove the noise. Generally speaking, a raw tweet contains the following kinds of noise.

i. Lower case – convert the tweets to lower case;
ii. URL – replace all URLs with generic token “url”;
iii. Punctuations and additional white spaces – get rid of white spaces while keep some stopper punctuation for dealing with negation;
iv. @username – replace with generic token “at_user”;
v. #hashtag – remove the # sign;
vi. Non-Unicode characters – get rid of them;
vii. Repeated letters – truncate more than 2 repeated letters to length of 2;
viii. Words starting with non-alphabet characters – get rid of them;
ix. Stop words – get rid of them;
x. Emoji and emoticon – useful but we can’t find a good way to deal with it, thus removing it.

These kinds of noise can be removed simply by regular expression matching. After being preprocessed, a raw tweet is turned into a bag of words.

2. Feature Extraction and Feature Space (Semantic Level)

2.1. Feature Space

The collection of features in the training set of tweets can induce a feature space. Different collection of features can induce different feature space. If we work on the tweet at semantic level, we can get more information about the tweet. So far, we have 4 different feature space. The raw feature space is simply the collection of all the features in the training data; the stemmed feature space is the collection of all features after stemming on nouns, verbs, adjectives, adverbs and so on; the negated features space is the collection of all features after dealing with negation; the stemmed-negated features space is the combination of doing stemming and dealing with negation.

2.2. Representing feature vectors

In this step, we are representing a bag of words returned by the preprocessing as a feature vector. We take the number of occurrence of a single word in the bag into account. Therefore, every single word occurred in the list is represented as a positive integer, which is its number of occurrence, at a specific position in the feature vector. Obviously, the feature vector is very sparse. Moreover, we can see that the mapping from a word of list to a feature vector is bijective. This kind of one-to-one correspondence means that we don't lose any information in the representation of the bag of words.

Table 3 Example for Feature Vectors

Tweets	"RT@TrumpIssues #GOPDebate #2016Debate elect Trump, elect Trump! Only he and @SarahPalinUSA can save America!"										
Feature	…	america	…	elect	…	gopdebate	…	save	…	trump	…
#Occurrence		1	0	2	0	1	0	1	0	2	0

2.3. Negation

Sentiment words behave very differently when under the semantic scope of negation. However, it is really hard to fully extract the semantic influence of negation.

We will use an approximating method proposed by Das and Chen (2001) [16] and Pang et al. (2002) [17]. It works as follows: append a neg suffix to every word appearing between a negation and a clause-level punctuation mark. A negation is any word matching the following regular expression:

$$(?: \text{^}(?:never|no|nothing|nowhere|noone|none|not|havent|hasnt|hadnt|cant|couldnt|shouldnt|wont|wouldnt|dont|doesnt|didnt|isnt|arent|aint)\$)|n't$$

A clause-level punctuation mark is any word matching the following regular expression:

$$\text{^}[.:;!?]\$$$

Here is one example:

Table 4 Example for Negation

Donald Trump says that he doesn't have time for political correctness. How does calling women "fat pigs" save him?
Donald Trump says that he doesn't haveneg timeneg for politicalneg correctnessneg. How does calling women "fat pigs" save him?

2.4. Stemming

Stemming is a method for collapsing distinct word forms. It could help reducing the feature space size. From the aspect of sentiment influence, Verbs of different tenses, nouns of singular or plural, adjective or adverb and so on, should give equal influence. Thus, doing some stemming can help us reduce the feature space, in the meanwhile, not changing the words' influence on the sentiment. There are three common stemming algorithms: the Porter stemmer, the Lancaster stemmer, and the WordNet stemmer. We use WordNet stemmer since it has high-precision stemming functionality. It requires (word, part-of-speech tag) pairs as input parameters, where the POS is adjective, noun, adverb, or verb. When given such pairs, it collapses tense, aspect, and number marking.

Table 5 Example for Stemming

Word	Stemmed	Word	Stemmed	Word	Stemmed
(exclaims, v)	exclaim	(proved, v)	prove	(happy, a)	happy
(exclaimed, v)	exclaim	(proven, a)	prove	(happier, a)	happy
(exclaiming, v)	exclaim	(proven, a)	proven	(happiest, a)	happy

2.5. Handling Typos

According to the word frequency count on the raw tweets collected, we find that typo is commonplace for tweets. On average, a single tweet contains 1.5 typos. Moreover, we have found that the typo is generally missing or misusing one or two letters. If we don't do with the typo, we will simply discard it when doing feature extraction from the testing data. However, doing typo correction is easy and can be achieved with high accuracy. An undirected graphical model for string edit distance and a conditional-probability parameter estimation method that exploits both

matching and non-matching pairs have been proposed by McCallum et al. [18]. We borrow the idea from the paper and implemented our own misspelling corrector with the help of Dirko [19]. We train the Hidden Alignment CRF model with string matching pairs collected from twitter. After training, it is able to give every matching a transition score. For each misspelled word, we find the closest matches from the candidates in the feature space. We adopt the matching with highest score. After handling typos, the feature vector becomes less sparse, which means that we are extracting more useful information from the tweet.

Here is an example. Suppose the model is fed with a wrong word "**'stopinng**", it is able to find top 10 candidates with smallest Levenshtein distance as the correction for this word. For each correction matching, it will give a score evaluating the matching. We use the matching having highest score as the correction of the wrong word. (stopinng --- stopping)

Table 6 Example for Handling Typos

Correction Matching	**('stopinng', 'stopping')**	**('stopinng', 'stoping')**	**('stopinng', 'stoning')**	**('stopinng', 'stoking')**	**('stopinng', 'sipping')**
Score	0.999807519	0.999617820	0.99781656	0.99537426	0.99171354

Classifiers and Sentiment Scores

1. The Multi-Classifier Scheme

Previous works involving in sentimental analysis on Twitter, such as [11], choose to conduct generalized classification that classifies multiple sentiments spontaneously. In this project, however, we present a discriminative approach to figure out sentiments from text-based features. According to [20], human emotions, even the contradicting ones like sad and happy, may mutually co-exist at the same time. For example, on the Internet it is possible to come across posts with contradicting strong emotions such as:

$$I\ used\ to\ love\ Trump\ as\ a\ TV\ host, but\ now\ he's\ just\ an\ asshole.$$

By the conventional sentiment keyword filtering, (i.e. capturing strong emotional words like love and asshole), such tweets may be classified as 50% positive and 50% negative, but by doing such, the mutuality of multiple strong emotions is lost.

Therefore, in this project, we introduce a multi-classifier architecture inspired by former works such as [21]. As is shown in Figure x, we use a group of parallel and identically structured classifiers, each of which deals with only one sentiment. The input m-dimensional feature vectors are duplicated and passed down to the N parallel classifiers. The classifiers are trained and tested independently from each other and the classifier group all together outputs a sentiment score tuple, which is described in the next section. Each single scalar score y is generated through a sigmoid function and ranges from 0 to 1.

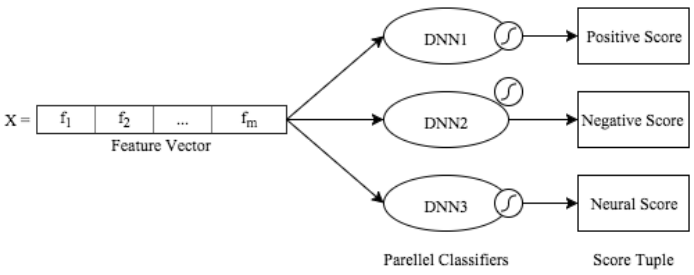

Figure 1 the Multi-Classifier Classification Scheme

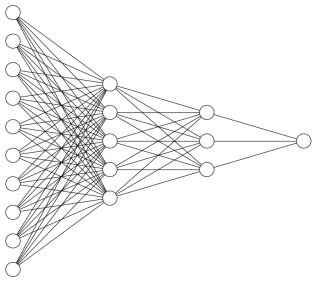

Figure 2 the 2-hidden-layer Deep Neural Network

2. Sentiment Scoring Algorithm

In this project, the test outputs of the classifier group forms a score tuple that referentially measures the dominance of each sentiment. For experimental purposes, we implement an approach to provide such sentiment scores on a finite bundle of shared-topic tweets, which are tweets about a certain presidential candidate. We assume the scenario contains N classifiers and the topic consists of K tweets. The raw output $Y_{raw,k}$ for the $k-th$ tweet in the topic corresponding to an input feature vector X_k is thus in N-dimension. The topic score tuple is calculated as follows:

$$Y_{raw,k} = \big(C_1(X_k), C_2(X_k), C_3(X_k), \dots\dots, C_N(X_k)\big) = \big(y_{1,k}, y_{2,k}, \dots\dots, y_{N,k}\big)$$

$$Y_{topic} = \sum_{k=1}^{K} Y_{raw,k} = (\sum_{k=1}^{K} y_{1,k}\,, \sum_{k=1}^{K} y_{2,k}\,, \dots\dots, \sum_{k=1}^{K} y_{N,k})$$

$C_i(X_k)$ here stands for the output with X_k made by $Classifier_i$. By doing so, we have a scoring system that keeps traits of mutually strong sentiments and mutually week sentiments. Lastly, to display the topic score in a probabilistic format, we may normalize this topic tuple with simply dividing each component value with the sum of all components.

$$Normalized\big(Y_{topic}\big) = \left(\frac{1}{Z}\sum_{k=1}^{K} y_{1,k}\,, \frac{1}{Z}\sum_{k=1}^{K} y_{2,k}\,, \dots\dots, \frac{1}{Z}\sum_{k=1}^{K} y_{N,k}\right)$$

Where $Z = \sum_{i=1}^{N}\sum_{k=1}^{K} y_{i,k}$

Experiment Setups

1. Sentiment Coexisting Assumption

We assume that different sentiment polarities can coexist in one tweet. That is to say, each tweet has a 3-D sentiment polarity score tuple, which represents positive, negative and neutral score respectively. However, one tweet can only have one sentiment label. We use the sentiment polarity having dominant score as the label of the tweet.

2. Classifiers

We are using Naive Bayes, Max Entropy, Kernel SVM, Linear SVM, and 2-layer DNN x 3 (200-200, 500-500, 2000-500 hidden neurons, 0.5 dropout, MSE, SGD minibatch = 250). We implement Naïve Bayes and Max Entropy with Natural Language Toolkit, SVM with Scikit-Learn library and DNN with Keras, a highly modular neural network library.

3. 3-Paralleled-Classifier Structure for SVM and MLP

We use 3-parallelled-classifier to classify the 3 sentiment polarity. Each classifier is responsible for classifying one polarity. The output is a score within range [0, 1].

4. Tweet Score and Topic Score with normalization

For a single tweet, we normalize it to present dominance / probability. For a specific candidate, we sum up all its sentiment scores before normalizing.

Results and Prediction

1. Classifier Performance Comparison (5-fold cross validation)

We evaluate the performance of different classifier in raw feature space (11,414 features). As we expected, DNN#3 has the lowest training error.

Table 7 Cross Validation for Classifiers

Classifier	Max Entropy	Naïve Bayes	Kernel SVM	Linear SVM	DNN#1	DNN#2	**DNN#3**
Training Error	0.38772	0.19876	0.15119	0.06267	0.03869	0.03100	**0.02716**

2. Feature Space Comparison (5-fold cross validation)

Using the best classifier DNN#3, we compare different feature space. Unexpectedly, stemming and negation fail to give a satisfactory result, which will be analyzed later.

Table 8 Cross Validation for Feature Space

Feature Space	**Raw (11,414 Features)**	Stemming (9,425 Features)	Negation (13,239 Features)	Negation & Stemming (11,071 Features)
Training Error	**0.02716**	0.02933	0.02937	0.02877

3. Validation

We use the three thousand tweets referred to Trump, Bush and Cruz posted between 08/07/2015 and 08/09/2015 as our validation data. We use the media investigation which is the polling data provided by CNN and Quinnipiac as sentiment validation.

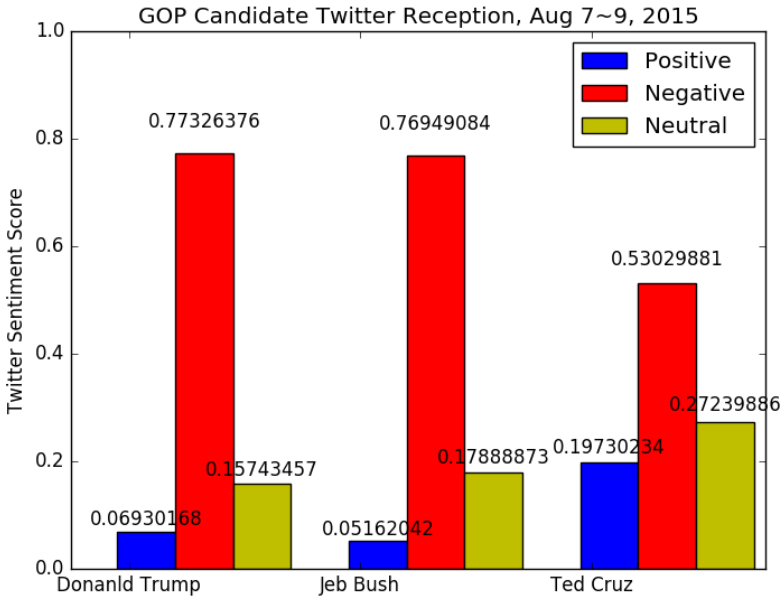

Figure 3 GOP Candidate Twitter Reception from Classifier

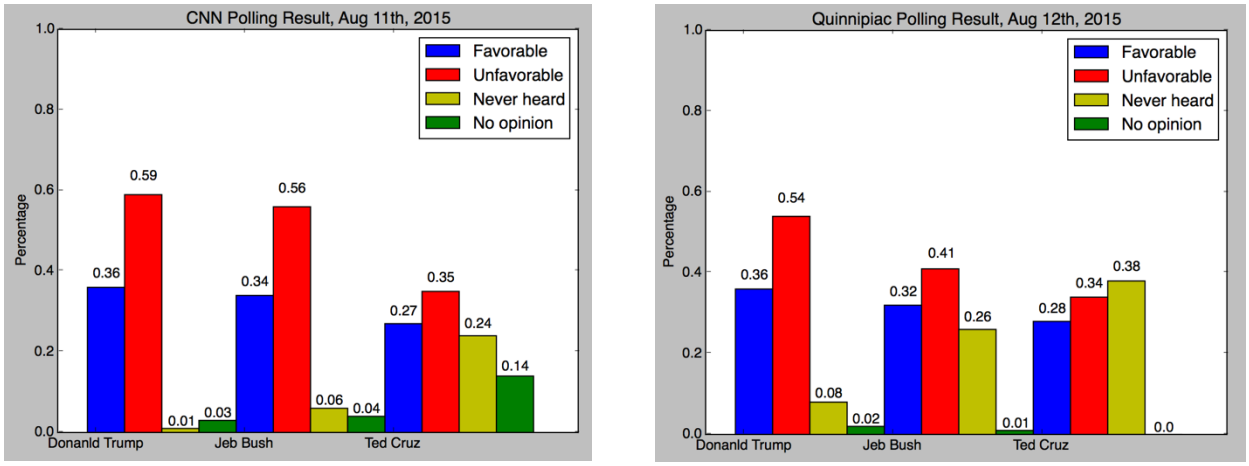

Figure 4 Polling Data from CNN and Quinnipiac

4. Prediction

We collect 14 thousand tweets referred to Donald Trump and Ted Cruz posted between 04/12/2016 and 04/18/2016 for prediction.

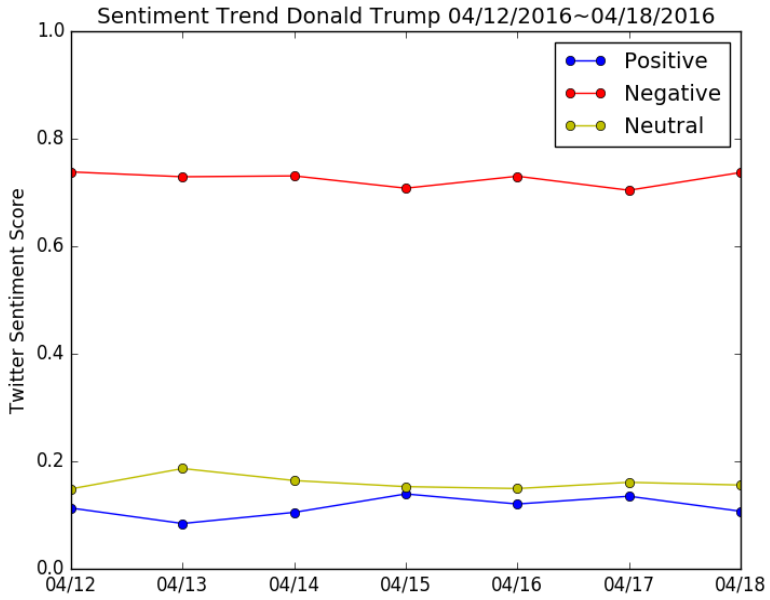

Figure 5 Prediction Results for Donald Trump

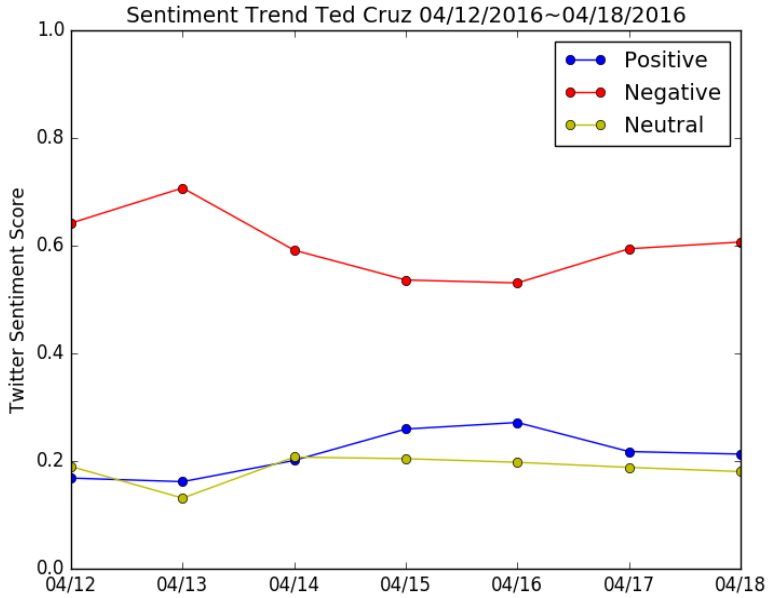

Figure 6 Prediction Result for Ted Cruz

5. Top50 Informative Features

We find the 50 most informative features for sentiment classifying using Naïve Bayes classifier. As is seen from the picture, the informative features don't need to be words expressing sentiment.

Figure 7 Top50 Informative Features

Conclusion and Future Works

1. Conclusion

In this project, we are mainly focusing on doing sentiment analysis of tweets. The steps can be summarized as data collection, preprocessing, feature extraction, labeling by multi-classification and computing sentiment score with regard to the candidate. There are some unexpected results in our experiment. The feature spaces induced by dealing with negation or doing stemming give a lower training accuracy than the raw space using the same classification model. Also, our neural network model has an amazing performance, which is beyond our expectation. With each tweet represented as an extremely sparse feature vector, literature shows that neural network models are not suitable for dealing with short texts using bag-of-words features. We have several meaningful findings from this project:

i. **The topic of tweets matters a lot**

   Different topics will bring about different feature space. In order to gain better performance, the training and testing data should focus on a more specific topic. There is no single trained model can have good performance over a vast choice of topics.

ii. **The training data and testing data should have the same feature space, otherwise not comparable**

   For example, if we use the tweets with regard to T. Cruz's opinion on "Abortion" as testing data, and use tweets regarding D. Trump's opinion "Mexico Wall" as training data, the result would be pointless.

iii. **Complex Model doesn't always give better performance**

   As is seen from the performance comparison among models, Max Entropy has a very bad performance. The training time for max entropy with only 3 iterations on 13k tweets takes us 6 hours while the training for DNN is less than half an hour. But DNN outperforms the Max Entropy a lot.

iv. **Preprocessing and Feature Extraction is really important**

   The principle here is to remove noise and extract feature as much as possible. However, it is not appropriate to intuitively treat something as noise or feature. We not only need

proof from linguistic research, but also need proof from real experiments. It is wrong to simply trust others' claim or follow intuition.

2. Future Works

2.1. Perfecting Preprocessing

The principle of preprocessing is removing the noise while retaining the noise. Nevertheless, we discard all the emojis and emoticons in the preprocessing. However, they do have influence on the sentiment of tweets. If we can find an appropriate way to handle them, the accuracy will be hopefully improved.

2.2. Improving Feature Extraction

In the feature extraction step, we use stemming to compress feature space and deal with negation to extract more features. However, neither of them give us an improvement in accuracy. Our negation strategy has flaws. We cannot handle tweets without stopping punctuations. Also, since the stemming is collapsing the original word, it will definitely lead to collision. There are a few examples of collision between two words with opposite sentiment polarity. Lastly, the training data for the HACRF model doing with typos is not rich. We can collect some string matching pairs from the training 13k tweets as training for the model to improve the accuracy.

2.3. A more comprehensive training data

In our 13 thousand training tweets, only 782 tweets have negation. The training data lacks learnable pattern for the classifiers, which may account for the lowered accuracy after handling negation. Moreover, the posting date of our 13k training tweets is 08/06/2015, thus covers very limited topics. It is a fact that different topic will leads to different feature space. In order to provide more learnable pattern and do better prediction, we need a more comprehensive training data. The training data should cover many topics, obtained within a long period and contains rich negation examples.

2.4. Taking the confidence of labels into account

Our 13 thousand training data is obtained from Kaggle online competition. Each tweet is hand labeled by many people. Therefore, they are labeled polarity by majority vote, which results in the

confidence of each labeling on the tweet. It is very risky to simply neglect the confidence and reduce each label to binary when we are doing DNN classification. For example, label with confidence 1 is totally different from label with confidence 0.34. (0.34 > 1/3)

2.5. Ensemble Classifiers

In the process of classification, we are using single classifier. However, bagged and boosted classifiers tend to give better performance. Bagged classifiers doing classification base on majority vote, while boosted emphasize the data points being misclassified. It is worth a try to implement bagged and boosted classifiers for sentiment analysis.

Reference

1. Twitter (March 21, 2012). "Twitter turns six". Twitter.
2. Lassen, D. S., & Brown, A. R. 2010. Twitter: The Electoral Connection? Social Science Computer Review.
3. Tweet Congress. 2012. Congress Members on Twitter Retrieved Mar 18, 2012, from http://tweetcongress.org/members/
4. P. Turney. 2002. Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. ACL.
5. B. Pang and L. Lee. 2004. A sentimental education: Sentiment analysis using subjectivity analysis using subjectivity summarization based on minimum cuts. ACL.
6. M Hu and B Liu. 2004. Mining and summarizing customer reviews. KDD.
7. S M Kim and E Hovy. 2004. Determining the sentiment of opinions. Coling.
8. C M Whissel. 1989. The dictionary of Affect in Language. Emotion: theory research and experience, Acad press London.
9. Apoorv Agarwal, Fadi Biadsy, and Kathleen Mckeown. 2009. Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams. Proceedings of the 12th Conference of the European Chapter of the ACL (EACL 2009), pages 24–32, March.
10. Go, Alec, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford 1 (2009): 12.
11. Agarwal, Apoorv, et al. 2011. Sentiment analysis of twitter data.Proceedings of the workshop on languages in social media. Association for Computational Linguistics.
12. Michael Gamon. 2004. Sentiment classification on customer feedback data: noisy data, large feature vectors, and the role of linguistic analysis. Proceedings of the 20th international conference on Computational Linguistics.
13. First GOP Debate Twitter Sentiment. Analyze tweets on the first 2016 GOP Presidential Debate: https://www.kaggle.com/crowdflower/first-gop-debate-twitter-sentiment.
14. Polling interview by ORC International on August 13-16, 2015, CNN: http://i2.cdn.turner.com/cnn/2015/images/08/17/rel8a.-.gop.2016.pdf.
15. Polling interview by Quinnipiac University. For release: August 27, 2015: https://www.qu.edu/images/polling/us/us08272015_Ueg38d.pdf.

16. Das, S. R., and Chen, M. Y. 2007. Yahoo! for Amazon: Sentiment extraction from small talk on the web. Management Science, 53(9), 1375-1388.
17. Pang, B., Lee, L., and Vaithyanathan, S. 2002. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10 (pp. 79-86). Association for Computational Linguistics.
18. Andrew, M. and Kedar, Bellare, Fernando, P. A Conditional Random Field for Discriminatively-trained Finite-state String Edit Distance.
19. Dirko Coetsee. https://github.com/dirko/pyhacrf.
20. Williams, P., and Aaker, J. L. 2002. Can Mixed Emotions Peacefully Coexist? Journal of Consumer Research, pp. 636–649, Apr 28, 2002.
21. Batista, L. B. and Ratte, S. 2012. A Multi-Classifier System for Sentiment Analysis and Opinion Mining. In Proceedings of the 2012 International Conference on Advances in Social Networks Analysis and Mining (ASONAM 2012) (ASONAM '12). IEEE Computer Society, Washington, DC, USA, pp. 96-100.