CS229 Project
Predicting The US Presidential Election using Twitter data
by
Swathi Chandrasekar, Emmanuel Charon, Alexandre Ginet

Introduction

Twitter is becoming one of the major online platforms for expressing opinions and thoughts. As people share more and more opinion online, we believe that the way to collect data and point of view is going to dramatically change in the coming years. As an example, we would like to tackle the problem of opinion polls in the particular context of a political election. How can we use publically available data to estimate elections results? How accurate is our method in comparison to traditional opinion polls?

The fundamental goals of the project can be listed as follows:

- Analyze the sentiments from Twitter feeds containing words like 'Obama', 'Romney', 'Democrat' or 'Republican'.
- Identify whether a tweet expresses a positive or a negative sentiment about a particular candidate and achieve good accuracy on this.
- Predict the overall voting intentions using large sets of tweets.

I - Building a training/testing set

1) Collecting data

The Twitter API has restricted data pulls to 1500 tweets at a time. We can specify hash tags and words that must be contained within the tweets we pull. For this particular project, we used the following keywords: 'obama', 'romney', 'democrat', 'republican', 'mitt', 'barrack', 'michelle', '#Obama2012', '#RomneyRyan2012', 'Mitt2012' to pull 1500 tweets every 15 minutes starting 26th October. We used a software called DiscoverText which gives a free enterprise license to Stanford students to implement this. The database within this software has been building up tweets and very quickly, more than a million tweets from the week before November 6th, the day of the election was available for analysis.

2) Representing, processing and formatting the data collected

For every tweet, we only analyzed the words it contains: we didn't analyze grammar or the order in which the words appear. So our representation of a tweet is the vector of occurrences of particular words in the tweet. In the rest of this paper, we will call 'features' the words we choose to take into account.

All tweets, both for training purposes or actual prediction, were pre-processed using R and the data-mining package (tm) within it. The following pre-processing was made:

- All words in tweets were first converted to lowercase.
- All punctuation marks were removed.
- Commonly used words like 'a', 'an', 'the' which don't contribute any meaning to the tweet were removed.
- We identified the root of each word and modified each word with the same root to have the same text (e.g. attitude and attitud clearly refer to the same word and our algorithm represents these two words as attitud)

3) Building the label list

Ultimately we want to build a training/testing set - as large as possible - where for each tweet we know with great accuracy if it is Pro-Obama (label +1) or Pro Romney (label -1). Several methods were envisioned to automate the categorization of the training tweets but we finally decided to manually label each of them to achieve maximum accuracy and reliability. Each member of the team was assigned a sample of tweets from various days before the election.

In order to simplify the problem of building the training/testing set, we decided to focus on tweets that (1) mention only one candidate or party and (2) have a clear positive or negative sentiment about it. Hence, we eliminated the tweets that mention no candidate or both candidates, the tweets that have a neutral sentiment, the tweets that are neither clearly pro Obama nor pro Romney, the tweets that have a clear position for one of the candidates but with a misleading combination of candidate mention and sentiment.

We have labeled a training set of 5,300 tweets manually, out of which 2,807 passed the filter described above.

4) Building the training matrix

We then needed to build the document term matrix like in Problem Set #2 for training and testing purposes. We used two separate datasets: One set of 10,000 tweets just to build the feature list (feature set) and our previous set of 2,807 tweets for the actual matrix (training set). The point was to have a feature list as general as possible, that would generalize well to all the tweets for prediction and that would not be limited by the features in our training set. For both those datasets, we chose tweets from different days so that contextual tweeting does not bias our analysis. From the feature set, we obtained a list of 16,204 features. We decided to only retain the features that would appear at least 5 times and ended up with a final list of 1,892 features.

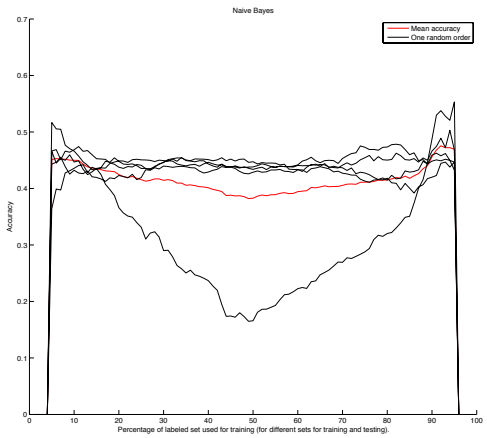
Finally the size of our training/testing matrix was 2,807 x 1,892.

II - Training with different algorithms

We implemented several algorithms on our current training/testing set. The goal was to determine which algorithm will perform the best and then apply it to our entire dataset in part III. We first implemented Naïve Bayes and a regular SVM. We computed the accuracy for different sizes of training and testing sets. For a given percentage of training data used, we computed the accuracy several times: one time with the given set of tweets (chronological order, a few from each day) and 4 times by randomly shuffling the order of the tweets.

In the following plots, the X axis is the percentage of the labeled set used for training (using all the rest of labeled tweets for testing), and the Y axis is the accuracy (%).

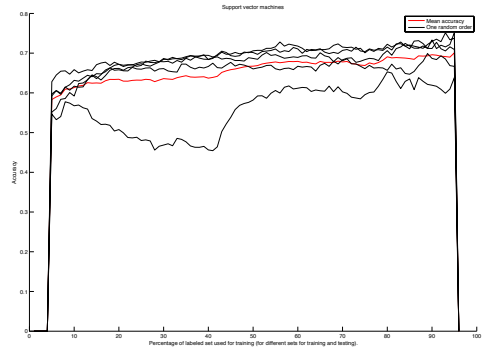
1) Naives Bayes



Naive Bayes model

We can see that for Naïve Bayes, the accuracy is quite bad (about 50%, i.e. random). This was expected, since the Naïve Bayes assumption is wrong in our problem. Take the following example: given that a tweet is Pro-Obama, the correlation of the words 'Obama' and 'good' is not zero.

2) SVM



SVM model

First, the SVM performs much better than Naïve Bayes, reaching accuracies of around 69% when we use 80% of the labeled set for training and the 20% left over for testing.

Furthermore, we see that the set taken to be the training set leads to different accuracies in both models. In particular, we notice in each graph one extraneous curve: it corresponds to when we do not shuffle the tweets. In these curves, the accuracy is globally smaller. This is due to the temporal correlation between tweets. We predict better when the training and testing sets are mixed in time (and not separated like in the original set of labeled tweets). In other words, the

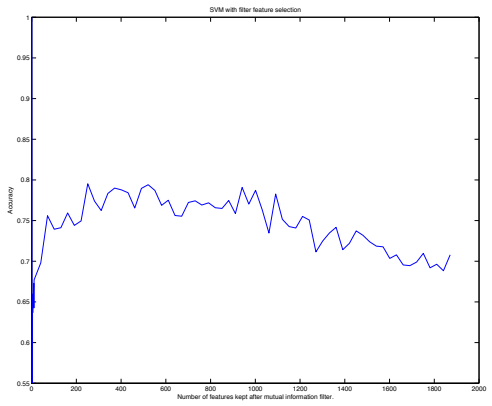
training set must contain tweets of any time for better overall accuracy. Hence, we implemented the random shuffling of tweets before all what follows. In the next steps, we tried to improve the accuracy of SVM using feature selection.

3) Feature selection for SVM

We used three different feature selection techniques described in class: best mutual information, PCA (Principal Components Analysis) and forward search.

3a) Mutual information:

We implemented the filter feature selection described in the lecture notes. Based on the 2807 selected tweets, we ranked the 1892 features in function of their mutual information with the labels. Then, we computed the accuracy resulting form a SVM (using 60% of the labeled tweets for training and 40% for testing) using the k best features, for k from 1 to 1892:



Filter feature selection for SVM

In the previous figure, we can see an increase and then a decrease in the accuracy in function of k. When there are too few features, the accuracy is low because the meaning of many tweets is not captured. When there are too many features, lots of the features can be seen as 'noise' because they are not informative on the opinion of a tweet. Finally, we reach the maximum accuracy of 80% at about 350 features: this is an increase of 11 points compared to no feature selection at all. Note that the 80% accuracy is a mean: the accuracy varied between 79% and 81% for different runs, because of the random shuffling of tweets, but consistently occurred around 350 features selected.

Furthermore, it is interesting to see what the most informative features are. Here are the first 20 (in order): 'benghazi', 'obama', 'romney', 'tcot', 'mitt', 'libya', 'romneyryan2012', 'female', 'field', 'impeach', 'sticker', 'obama2012', 'bitch', 'seal', 'christian', 'hurricanesandy', 'thedailyedge', 'attack', 'administratrtion', 'security'.
Among them, we can distinguish different categories: words in relation to the candidates (like 'romneyryan2012'), words that have a clear positive or negative meaning (like 'bitch'), and words that are related to the particular context of this election (like 'hurricanesandy'). Most of the significant words actually belong to the third category, which confirms our global approach on the problem: we did not try to determine if a tweet had a general positive of negative implication, but directly if it expressed an opinion on a candidate. It happens that only mentioning 'benghazi' is strongly in favour of Mitt Romney. Hence, we abandoned the idea of using the dictionary of affects in language (that gives an opinion score of individual words [3]) to classify tweets, since very informative words like 'benghazi' would have been given a neutral score and thus wouldn't have helped classify the tweets.

3b) PCA

We implemented and tested PCA with SVM, but for 5 principal components and more, the SVM does not converge (ends at 1000 iterations and gives about 50% accuracy i.e. random predictions). It seems that combinations of features do not help. More importantly, it is an indication that unsupervised learning has little chances to give good results in our problem: the distances between all tweets are small and only few dimensions actually separate the data (like the value of a tweet with features 'obama' or 'romney'). Thus, separating the data based only on the occurrence of words didn't work. That's why we did not try to implement other unsupervised learning techniques like k-means clustering or the EM algorithm. Algorithms like SVD could also have helped us identify clusters of tweets, but since we had only a limited number of clusters of interest, we expected the accuracy to be low and did not pursue this method.

3c) Forward search/ Wrapper feature selection:

We followed the same procedure as in 3a) but with 'forward search' using SVM to select meaningful features. For a small amount of features, artifacts of features present in too few tweets led to poor performance. Hence, we decided to begin with the 10 features having the best mutual information, and then we improved the accuracy step-by-step using forward search. We reach about 78% accuracy with only 20 features, so just a bit less than in 3a). Unfortunately, time limitation and the high computational complexity

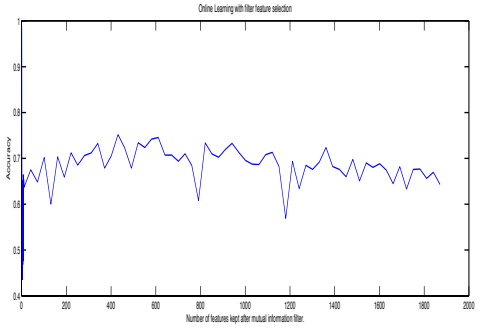
of forward search prevented us from performing it on more features. The first ten words it choose (except for the top 10 words of 3a) are:
'president', 'vote', 'win', 'my', 'endorse', 'breitbartnew', 'play', 'me', '2012', 'get'.

Surprisingly enough, all the words are different from the words 11 to 20 of 3a). Denote the appearance of possessive words like 'my', which express a connection to the candidates.

4) Online Learning

Last but not least, we had good reasons to believe online learning would yield good results in our problem. First, the SVM (with feature selection) gives about 80% accuracy: we argue that this good SVM result implies the samples are almost separable (with margin \gamma), which is the condition for Block's theorem to hold. Second, the D=max(norm(xi)) is quite small in our problem: a tweet has about 10 to 20 words so we roughly have D=20. Now using Block's theorem, the number of errors in online learning should be bounded by (D/gamma)^2. This bounded number of errors implies that with a big training set, the accuracy can become very high.

Hence, we tried online learning. It yielded about 67% accuracy compared to 69% for SVM. We then used filter feature selection with online learning and it gave the following accuracies (in function on the number of features used):



Filter feature selection with Online Learning

We see some improvement (up to an accuracy of 75% for 420 features) but the increase and decrease are not as well delimited as with the SVM. Some features introduce a fast decrease in accuracy at 800 and 1200 features.

Finally, our best accuracy was reached with the SVM using filter feature selection (80% with 350 features) and this is what we used to train on the whole labeled set and then make our predictions in part III.

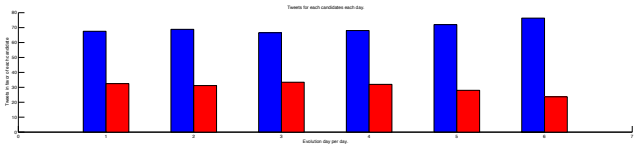
III – Predicting the Elections

We selected the SVM with the top 350 features from Mutual Information selection to analyze our entire dataset. This algorithm proved to be the best performing with about 80% of accuracy on the testing set. Although forward search achieved almost similar accuracy with only 20 features (about 78%) and would have probably achieved better results with more features, it was too time consuming to compute it.

1) Daily reports

First, we decided to apply our SVM to tweets from a given day to obtain the overall sentiment of voters on Twitter that day. We used our data from the 5 days preceding the elections as well as the Election Day itself. Each day we analyzed a total of 30,000 tweets out of the 160,000 collected for that day.

We obtain the following results, expressed as a percentage of analyzed tweets:



Percentage of electoral tweets pro-Obama (blue) and pro-Romney (red)

There are two interesting things to note here:

(1) The tweets are highly favorable to Obama

By comparing our output to actual election results and polls during the week preceding the election, we understand that our tool was able to identify a strong bias pro Obama among Twitter users. Twitter is definitely not an objective representation of the entire population sentiment. It is already a very interesting conclusion in itself. Yet, as Twitter keeps expanding its user base, we can imagine that its data will become more and more representative of the actual sentiment.

(2) The trend before the week of the election was in favor of Obama

This is a very interesting result and we understand the power of our tool. We are actually capable of identifying trends and shifts in opinion at a very granular level. We can see from just one day to another

how the overall sentiment of Twitter users has evolved. No other existing opinion poll solution is able to come up so rapidly with precise results on such a large user base.

2) Overall results of the elections

In order to compute the overall result of the elections, two different strategies could be applied here. We could either simply use the Tweets from the Election Day or compute an average over data from the last week. We have decided to adopt the second approach to account for the fact that: (1) People make a decision based on their overall impression during the campaign, not just their last word (2) Since users don't necessarily tweet everyday, the data from the whole week might be a representation of more users.

In the end, our tool predicted that Obama would collect 69% of votes. Our accuracy being roughly 80%, in the worse case the percentage of tweeters supporting Obama is at least 55%. Comparing to the actual result (50.6%) confirms our intuition that the Twitter user base is not representative of the actual Nation's sentiment.

IV – Conclusions

Our study leads to multiple conclusions. First, we believe our tool can be used to complement the traditional polls. On the one hand, our major disadvantage is the bias in the tweeter users (there are 6 times more people following Barrack Obama than Mitt Romney). Finding a way to balance the tweeter bias would be a great improvement to our algorithm. But polls also have arguably representative samples of the population (who are the people who take the time to answer to a pool?). On the other hand, our major advantage is the large number of tweets used in the predictions, way bigger than what pool institutes are able to do. This gives us a much more precise interval of confidence (even taking accuracy into account) in the final prediction.
Then, we also discovered what were the most important words in an election. It turns out they are the very context-specific ones. Goods news is that our tool finds those topics automatically. Among the different events and debates that occurred during the campaign, the incident in Benghazi and hurricane Sandy were the most indicative of an opinion. Based on this observation, using constraint weight SVM would yet be another way of improving our accuracy (along with labeling more tweets and running the forward search feature selection on a supercomputer).
Lastly, the final result of the election is not based on the nationwide opinion but on the state-by-state opinion and the number of Electoral College votes won in each state. We could improve our algorithm by running it on different states and giving a state-by-state prediction, especially in the swing states.

Bonus

As a reward to the reader, here are a few tweets we encountered during labeling:

"Mitt's thoughts and prayers go out to the people of Ohio as they watch TV coverage of Hurricane Sandy."

"I don't support #Obama because I need a job or healthcare. I support him because he shines the light on the future I want for my children."

"Bands will make her dance, Food Stamps will make her twerk. But if Romney becomes president, yall hoes will have to work."

"Wtf is Obamas last name? Does anyone know?"

References

[1] CS229 Lecture Notes, *Andrew Ng*, Stanford University

[2] Sentiment strength detection in short informal text, *Mike Thelwall & Al.*, 2010.

[3] Sentiment Analysis of Twitter Data, *Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow and Rebecca Passonneau,* 2011