Michael Brughelli
CSCI 4830 Final Project
Fall 2014

Election Result Predictions via Twitter Sentiment Analysis

Problem

The problem I sought out a solution for is simple: elections can be extremely unpredictable especially in close races. So my goal was to find a way to successfully predict election results. This is an important and meaningful problem to solve because there is a ton of money and hope put into candidates running for office and people would be very interested in a technology that could correctly predict results. Obviously, there is no guaranteed method anyone could ever hope to employ that would yield correct predictions 100% of the time; but even if a developed technology could correctly predict the winner 60% of the time or even 50% of the time it would most likely be highly coveted by election campaigns and voters alike.

Previous Solutions

A team with members from a number of universities in Ireland conducted an experiment very similar to mine in which they mined tweets pertaining to the run-up to the Irish General Elections in February 2011. They used a larger corpus of tweets and used a method slightly different from mine in order to achieve their results. They used 2,624 tweets and labelled them as positive, negative, mix (both positive and negative), neutral, and non (having no mention of the related topic). Tweets that had been labelled as sarcastic were omitted from the set and the team admitted that this was a shortcoming. However, they also believe that their results would not have been as accurate had they attempted to account for sarcasm as that is a very difficult

thing for computers (even some humans) to do accurately. They achieved a 61.6% accuracy using supervised learning and a feature set consisting of subjectivity lexicon based scores, Twitter specific features and the top 1,000 most discriminative words. A big difference in this project and mine is the fact that Ireland has 5 political parties that all had at least a decent chance of winning the election, whereas in the US we have 2 parties that dominate politics. My solution only keeps track of the Republican and Democratic candidates which could account for my higher success rates.

My Solution

I picked 2 races to predict the results to during the 2014 US midterm elections- The Colorado Senate race between Cory Gardner (R) and Mark Udall (D); the Colorado Governor race between Bob Beauprez (R) and John Hickenlooper. I set a date range for which I would retrieve tweets for all data sets to be used in my project. This range was October 1, 2014 to November 3, 2014. This avoided using any data that gave away the result of either race as I conducted this study after election day (November 4, 2014). I used the advanced search tool within Twitter in order to pinpoint the date range I wanted as well as the hashtags that would give the tweets I needed. For the Senate Race, I searched the the following hashtags- #CoryGardner #MarkUdall and #cosenate. For the Governor Race, I searched the the following hashtags- #BobBeauprez #JohnHickenlooper and #cogov. The hashtags are case insensitive. I created a training set of data for each race that was 58 tweets in length and spanned a variety of issues in each campaign, respectively. I then created a test set for each race that was 200 tweets in length, respectively.

In order to train and test my data in order to yield predictions, I used a python library called TextBlob which is built over NLTK. Specifically within TextBlob I used the Naïve Bayes

Classifier which uses a feature extractor that indicates which words in the training set are contained in a document and whether or not that causes the document to lean positive or negative. My training data sets are given in the following format- tweet content.,pos/neg where I and my father went through each tweet and classified it as either positive or negative. We went through the tweets together and agreed upon the sentiment of each one. This eliminated the possibility of error on my part as we did this very meticulously. I chose to leave out neutral, unrelated, and sarcastic tweets as I believe they would have caused my predictor to be less accurate.

My driver file, Classifier.py, takes in 3 user provided arguments: the training set, the testing set and a marked testing set (that is identical to the testing set but each tweet is marked positive or negative). The testing set obviously has no human sentiment markings on it. The model trains on the training set, then uses that training in order to classify the test set as either positive or negative. I made the arbitrary decision of classifying positive things said about the Republican candidate as 'pos' and positive things said about the Democratic candidate as 'neg'. And then obviously negative things about the Republican were marked 'neg' and negative things said about the Democrat were marked 'pos'. Thus, when the test set returns it's result as 'pos' this indicates that the Republican candidate is predicted to win and if the test set returns 'neg' it indicates that the Democratic candidate is predicted to win. Finally, the marked data set is read in in order to calculate the accuracy of the test. I use the accuracy(test_data) method given by the TextBlob library to do just that. My program prints the winner of the race and then the accuracy percentage and exits.

After completing this method for both race, I wanted to try to implement something that was broader and could potentially be used on any race test set between a Republican and a Democrat during the 2014 US midterm elections. My idea was to create a much larger training set that was composed of general Republican vs. Democrat tweets. So I took to Twitter again and searched for #Republican and #Democrat from October 1, 2014 to November 3, 2014. I created a data set with 300 tweets from these results and once again went through each tweet with my father and agreed on positive or negative for each one. I then ran the test files against this new training set.

I believe my solution is superior to previous solutions for a few key reasons. The first is my addition of general training data that is not specifically related to any one race in order to broaden my method to many races quickly. This increases usability and cuts down on time that it takes to develop a test set for individual races. In the least, it offers a second way to test data and reinforce (or contradict) the prediction of the individual set. The accuracy was lower (obviously) using this training set but it is a tradeoff. The second thing I did differently is I made sure to have uniformity in my data sets. For example, if one tweet referred to Cory Gardner as CoryGardner (1 word) and another referred to him as simply Gardner or Cory or perhaps even an abbreviation like Gard this would lead to a less accurate test. So I went through all corpuses and made this references uniform. For example, every time Mark Udall is mentioned in any tweet in my data it reads exactly, Mark Udall. This builds a stronger unity between the training and testing sets and leads to higher accuracy between them which then, in turn, leads to a higher prediction success rate. Lastly, my method of classifying tweets as only positive and negative leads to greater accuracy rates especially in our two-party dominant political arena. It took far longer creating the data sets due to omitting neutral, non-related, and sarcastic tweets from them

but in the long run, it gave me higher accuracy rates. Most tweets I read through were either very clearly positive or negative anyway which may be a referendum on the current highly divided state of politics in this country.

Results

In reality, Cory Gardner won the Senate race and John Hickenlooper won the Governor race. My solution using the training sets specific to each race correctly predicted both races. The solution using the general Republcian vs. Democrat training set correctly predicted Cory Gardner as the winner of the Senate race but incorrectly predicted Bob Beauprez as the winner of the Governor race. This may be due to the very heavy number of tweets in the last few days of the Governor election that stated Beauprez was leading in the polls that were included in the test set for the Governor race. The table below gives the accuracy percentages for each test.

RACE	**TRAIN SET**	**WINNER**	**ACCURACY**
Senate	SEN	C. Garner (correct)	73%
Governor	GOV	J. Hickenlooper (correct)	75%
Senate	R vs. D	C. Gardner (correct)	58%
Governor	R vs. D	B. Beauprez (incorrect)	47%

Comparatively to the Ireland election model which, at its best, only ever yielded a 61.6% accuracy rating, my model performs very well. The specific race training data tests performed especially well and correctly predicted the results of the elections. My Republican vs. Democrat training data tests understandably far underperformed the aforementioned test. But even still, the accuracy for the Senate race using the Republican vs. Democrat training set is only 3.6% below the Ireland model which was developed by a large team. Their number of tweets was far greater than mine however and they also classified neutral, mix, and non-related tweets which added a

lot of difficulty to their model that I eliminated from mine. I believe that choice I made was a good one considering the general high accuracy of my model.

Error Analysis

The tests that used the individual data sets for training had both high precision and high recall as 73-75% indicates high recall while correctly predicting the results indicates high precision. The Republican vs. Democrat training data tests had much lower recall at 58% and 47% but the precision of the Senate test in this category was correctly predicted so the precision is high. The Governor race performed poorly when tested with the Republican vs. Democrat training set as it incorrectly predicted the winner; this indicates low precision as well as low recall- 47%. The low recall was to be expected using the R vs. D training set.

Possible Improvements

Even though I achieved what I perceive to be a relatively high level of success for my project there are many areas that I could improve both my method and my testing. First, I could have implemented a method that classifies tweets into more categories than just simply positive or negative. From the Ireland solution, I could have used at least mixed tweets that contain both positive and negative things. For example, Democrats are thieves but Republicans are heartless; from my method of classifying tweets, I'd give the 'Democrats are thieves' portion a 'pos' but the 'Republicans are heartless' part would receive a 'neg'. There were actually a sizeable number of tweets similar to this example that I skipped over because I did not implement a solution to deal with them.

Secondly, I'd like to have tested more races on this model prior to the due date but due to time constraints was not able to. Having a test body of at least 10 or so races would indicate to be far better the true accuracy of my model's predictions. On the bright side, politics fascinates me so I will most definitely continue to build upon this project- the first step being testing it on more races with larger corpus sizes of both training and testing data. The last thing I'd really like to improve upon would be adding a 3rd parties to the model. Not because they have high chances of winning in this country but because oftentimes they steal key votes away from either the Republican or Democrat or both. This factor can oftentimes sway the result of an election and therefore would be paramount in adding further accuracy to my model.

Conclusion

Do I believe that I've created a magical technology that can tell the future? In a way, yes. I believe I have used the knowledge of natural language processing that I have been taught this semester in conjunction with many dozens of hours of work on this project doing research, reading massive amounts of political rants in 140 characters or less, coding, tweaking, and coding some more to create a model that is statistically highly accurate in predicting the results of political races. This is the goal I set out to achieve and I honestly far surpassed my own expectations of what I'd actually be able to do. With 3 of 4 tests accurately predicting the winner and accuracy rates higher than those of professionals far above my "pay" grade, I am pretty satisfied for now with what my model can do. As stated in the improvements section of this document, I have a strong desire to continue to work on this model and increase its functionality and broaden its scope even further to increase usability.