Predicting Congressional Bill Outcomes

Zach Cain
Stanford University
zcain@stanford.edu

Pamela Chua
Stanford University
ylpchua@stanford.edu

Kristian Gampong
Stanford University
kgampong@stanford.edu

Abstract

This election season, as those before it, has demonstrated the importance of buzz words in Congressional, as well as Presidential, campaigns. We have been bombarded with vague references to jobs, family, crime, and so on, as candidates attempt to gather as many votes as possible. In this project, we attempt to explore this question: how much of an impact do these words have once those candidates take office? More specifically: can we predict the voting behavior of a Representative on a given bill using only the frequency of words in the bills text?

1 INTRODUCTION

Text classification is a well-known field of information retrieval and has been applied to many different areas as well. Although this broad area of research has been extensively explored, we did not find any substantial published work on this particular application of Congressional vote predictions. Our goal is to build a binary classifier for each Representative that predicts what he or she will vote based solely on the frequencies of words contained in a bill. We implemented three different algorithms: Multinomial Naive Bayes, K-Nearest Neighbors, and Support Vector Machines, and evaluated their strengths and weaknesses. On top of that, we also used an ensemble of the above three algorithms to predict the voting patterns of a Representative.

2 PRIOR WORK

Previous work in a similar field was done by Yano, Smith and Wilkerson [1] who explored textual features to predict bill survival in Congressional committees. Our project will focus on the individual congressman and his voting record, while Yano, Smith and Wilkerson looked strictly at Congressional committees. Furthermore, our planned methodology differs significantly from their work.

3 DATA COLLECTION AND CONVERSION

3.1 Data collection of voting records

We chose to work with the 109th Congress (2005-2006) and chose to further focus only on the House of Representatives, since it does not have anomalies like filibusters, which occur in the Senate, and has 435 voting members, as opposed to the Senates 100, providing a wider-encompassing view of Congressional voting habits. Although all Congressional histories are publicly available

online, there was a substantial amount of preprocessing required to transform this data into a usable format. We retrieved the full roll call history of the 109th House of Representatives from http://polarizedamerica.com/house109.htm, but these roll calls included votes on all administrative actions in the House between 2005 and 2006. For example, there were votes to add amendments to existing bills, to redirect failed bills to an appropriate committee, to table bills, to suspend the rules and agree, and so on. Since we were only looking at cases when a bill was being voted into a law, the overwhelming majority of these votes were not useful to us. We had to parse through the roll calls to find only those roll calls that involved voting a bill into law, and then further filtered these roll calls to focus only on the H.R. bills–those that originated in the House (as opposed to bills that originated in the Senate, were passed there, and were then being voted on in the House). We chose to ignore the bills that originated in the Senate because we wanted to avoid the possible extra noise that accompanied these bills, such as different writing style/word choice or additional party pressure to vote in a certain direction, given that such bills had already passed the Senate.

Once we determined which roll calls corresponded to votes on bills, we used the publicly available roll call vote data at http://www.govtrack.us/data/us/109/rolls/ to retrieve the vote of each Representative for each relevant roll call, using the data at http://www.govtrack.us/data/us/109/people.xml to match the votes to the name, state, and party of each individual Representative. Using this data, we constructed a matrix L of labels, with each $L(i, j)$ corresponding to the jth Representatives vote on the ith bill (1 for yes, 0 for no).

3.2 Preprocessing of text

For our feature matrix, we had to collect the full text of all the House Resolution bills passed during the 109th Congress. We obtained the corpuses from the U.S. Government Printing Office website (http://www.gpo.gov/fdsys/). Some basic preprocessing of the text included removing punctuation, setting capitalized words to lowercase, and removing stop words commonly seen in legal documents. We acquired this list of legal stop words from the California Legislature website (http://www.leginfo.ca.gov/help/stopwords.html).

To standardize the tokens, we used Porters stemmer to stem tokens in an attempt to eliminate redundancies in our tokens. For instance, the words representative, represent, and representation are all mapped to the stemmed token "repres". We then split the space-delimited processed text into tokens, sorted them and converted them into an index of unique tokens. We then generated a sparse representation of the document-word matrix T with each $T(i, j)$ corresponding to the count of the jth token in the ith bill.

4 METHOD

4.1 Multinomial Naive Bayes

Although it makes a strong assumption of the conditional independence of words given the labels, the Naive bayes algorithm has shown to produce surprisingly good results and is frequently implemented in text classification. We decided to implement the Multinomial Naive Bayes model, which uses the frequency of occurrence as a probability, as opposed to the Multivariate alternative. Improvements made to the model were made through stemming and preprocessing of the text in the bills. Other adjustments attempted including tweaking the value for Laplace smoothing. In Laplace smoothing, the constant 1 is usually added to account for words that have not been seen before. We tried to vary this value to see if the accuracy improved but it did not improve accuracy much if at all, possibly due to the spareness of the word matrix.

What produced really interesting results for this model was training the Representatives separately based on whether they were Republicans or Democrats. The model for the Republicans achieved a high mean accuracy (0.8377) while Democrats performed poorly (0.5638) even though the number of Republicans is about the same as the number of Democrats in the training set, hence pointing to the possibility of an improved model based on party lines.

4.2 K-nearest neighbors

We implemented two slightly different versions of KNN, one using Jaccard similarity, and the other using cosine similarity. The process behind the two, however, was the same: for each bill in the test set, we searched the training set to find the k most similar bills to the test bill based on the given similarity metric. Then, for each Representative, we looked at the votes of the top k bills. If the Representative voted yes on a majority of the k, we predicted yes for that Representative, and predicted no otherwise. We then checked the actual vote on that test bill, and updated the error accordingly. Through experimentation, we found the optimal value of k to be 5:

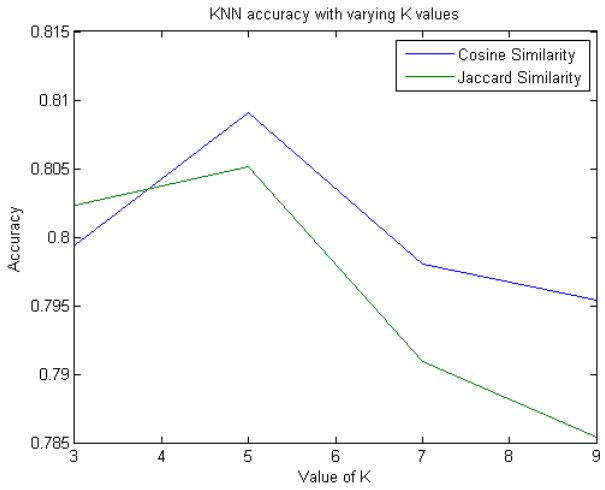

Figure 1: KNN accuracy with varying K values

4.3 Support Vector Machines

We also used Support Vector Machines (SVMs) to solve our text classification problem. From class, we expect the SVM classifier to have relatively higher error on very small training sets like ours, but perform asymptotically better than Naive Bayes. This is because generative learning algorithms (such as Naive Bayes) have smaller sample complexity than discriminative algorithms (such as SVMs) but generative algorithms may also have higher asymptotic error.

We used the library sliblinear (http://www.csie.ntu.edu.tw/ cjlin/liblinear/) with default parameters to run our SVMs for each Representative but we first converted our matrices to the requisite liblinear format by converting classes to [-1, 1].

4.4 Evaluation of models

In order to evaluate how well our classifiers are performing, we decided to use 10-fold cross validation: randomly dividing our data into 10 sets, and for each iteration, using one of the sets as our test set and the other nine as our training set. We chose to do 10-fold cross validation instead of hold-out cross validation because of our relatively smaller data set. Leave-one-out-cross-validation (LOOCV) was not considered because training would be computationally expensive.

4.5 Improvements made to models

Though our initial results were very promising, we thought of several areas where we could make improvements to the accuracies of our models. One of our main ways of doing so was improving the relevance of the tokens in our token matrix by removing tokens that were erroneous and did not help with our predictions.

To eliminate unhelpful tokens, we calculated the inverse document frequency (idf) scores for all tokens in order to determine which tokens are more important. We define the idf score of a token t

to be:

$$idf_t = log(N/df_t),$$

where N is the number of bills and df_t is the number of bills that contain the token t.

Thus, the idf of a rare term will be high while the idf of a frequent term will be very close to zero. We ranked the tokens based on the idf-scores and eliminated those that can be found in all bills (idf-scores equal to zero).

Another way we boosted our accuracy was by improving the label matrix of congressional votes. One major issue we faced was that Congressional votes are not mandatory; Representatives can, and often do, miss votes for a variety of reasons: illness, campaigning, family emergencies, and other circumstances that may or may not be under that congressmans control.

At first, we experimented with a three-way classifier, that would predict whether a congressman would vote yes, vote no, or be absent from the vote on a given bill. However, we found that the votes missed by congressman were more or less random, just as we suspected. Therefore, we began treating not present votes as no votes for all of our training and testing data. Unfortunately, since these absences could be totally unrelated to the text of the bill under consideration, it was extremely hard to predict the behavior of Representatives who were absent for many votes. Therefore, the improvement we made was to remove habitually absent Representatives from consideration, creating cutoffs for the Representatives we would consider based on how many votes they cast during their time in office. Out of 109 total votes that we tracked, there were very few Representatives who missed more than 30 of those votes, but most Representatives missed at least a few (¡= 10) votes. We experimented with different cutoffs between 30 and 10 missed votes, and as expected, removing Representatives who missed many votes noticeably improved our classification accuracies across all models. Our best results occurred with the strictest cutoff of 10 missed votes, meaning we only considered Representatives who voted on at least 99 out of 109 bills. These results matched up quite well with our expectations.

4.6 Ensemble method

Another way that we sought to improve our accuracy is through an ensemble method. We looked at the predictions of KNN (cosine similarity), SVM, and Naive Bayes, and then for a given bill and Representative, we predicted yes if 2 out of the 3 models predicted yes, and no otherwise. The ensemble approach ultimately performed better than SVM and Naive Bayes, but slightly worse than KNN.

5 RESULTS

As can be seen from Figures 2 and 3, the peak accuracy is achieved by the KNN model using a cosine similarity metric with k = 5, a minimum of 99 out of 109 votes cast for each representative, and the stemmed token matrix adjusted for token IDF scores.

6 CONCLUSION

We faced several challenges in this project. One was a relatively small set of training examples. Since the House changes its members every 2 years, each set of representatives has a very short period in office. For example, the 109th House of Representatives voted on just over 100 bills, and those from other recent years had around the same number of votes. Another challenge, as discussed previously, was the large number of votes missed by some representatives for reasons completely unrelated to the text of the bills they missed. Furthermore, our input features consisted solely of word frequencies, without any measure of whether a bill was for or against the words it contained.

Despite these challenges, we were quite pleased with the results of this project. Our goal was to determine whether or not there was any useful link between the word frequencies of bills and voting patterns in Congress. Since our classifiers use only this data, and were able to achieve relatively high accuracies, we have demonstrated that there is a definite and useful connection between word frequencies and voting behavior. Therefore, in possible future work, it seems likely that word frequencies could be used as a single feature in a larger classifier along with other features such as

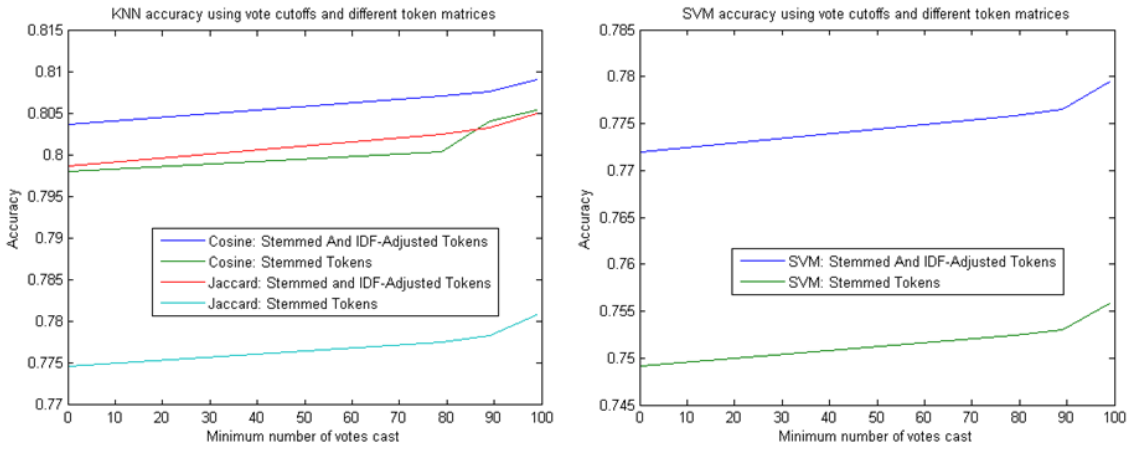

Figure 2: Accuracy for KNN and SVM using Stemmed and IDF Adjusted tokens

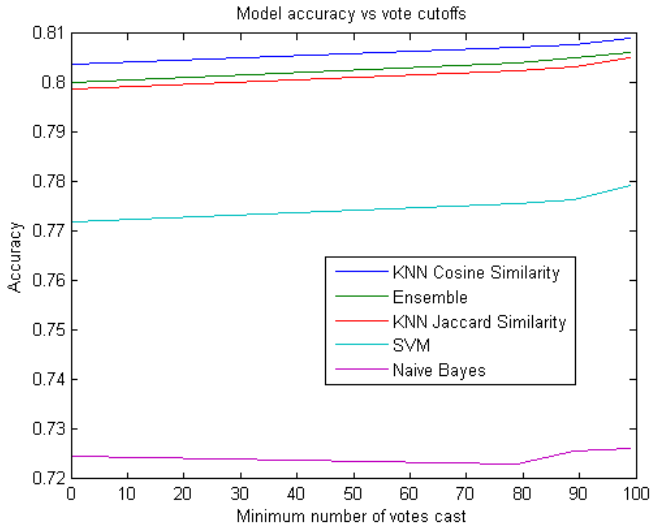

Figure 3: Main plot of accuracies for different models implemented

party of the voting congressman, party of the bills sponsors and co-sponsors, and so on. An alternate approach for future work would be to apply more sophisticated language parsing to bill text to identify not only the frequency of political buzz words, but also the bills sentiments towards them, and use these as features to make predictions.

ACKNOWLEDGMENTS

We would like to thank Professor Andrew Ng for his invaluable teaching and guidance that made this project a possibility.

REFERENCES

[1] Yano, T., Smith, N.A. & Wilkerson, J.D. (2012) Textual Predictors of Bill Survival in Congressional Committees.