SENTIMENT ANALYSIS OF POLITICAL SPEECHES

JAY DESTORIES

Abstract

The political affiliations of politicians are no secret, but determining how extreme or moderate someone's views are is a highly opinionated task. In this paper we consider statistical techniques to place individual U.S. political speeches on a continuous scale between Republican and Democrat.

Contents

1. Background

The goal of political sentiment analysis is to gain insight into political opinions using natural language processing. In the context of this paper, we will consider political speeches, particularly State of the Union addresses, and train probabilistic classifiers to predict political party affiliations. Since probabilistic classifiers can generate probabilities of class association, we can translate the output of such a classifier into a semantic differential ranging from completely Republican to completely Democrat. We will discuss the problem of selecting a set of features which is indicative of political affiliation, as well as training classification algorithms.

1.1. **Naive Bayes.**

Naive Bayes is one of the simplest and most intuitive probabilistic classification algorithms. To assign a class to a data point, we consider all hypothesis classes and pick the one which maximizes the probability of that hypothesis given the data.

Class $\hat{c} = argmax_{h \in classes} P(h|D)$

By Bayes' Theorem:

$$P(h|D) = \frac{P(D|h) \cdot P(h)}{P(D)}$$

So:

$$\hat{c} = argmax_{h \in classes} \frac{P(D|h) \cdot P(h)}{P(D)}$$

If we have observed each h $c(h)$ times, then:

$$\forall h \in H, P(h) = \frac{c(h)}{\Sigma_{h \in H} c(h)}$$

$$\implies \hat{c} = argmax_{h \in classes} \frac{P(D|h) \cdot \frac{c(h)}{\Sigma_{h \in H} c(h)}}{P(D)}$$

And since $P(D)$ does not vary on h,

$$\implies \hat{c} = argmax_{h \in classes} P(D|h) \cdot \frac{c(h)}{\Sigma_{h \in H} c(h)}$$

Since our D is a collection of feature values $(f_1, f_2, ..., f_n)$

$$\hat{c} = argmax_{h \in classes} P(f_1, f_2, ..., f_n|h) \cdot \frac{c(h)}{\Sigma_{h \in H} c(h)}$$

But it is difficult to compute $P(f_1, f_2, ..., f_n|h)$, so we make the "Naive" assumption that $f_1, f_2, ..., f_n$ are independent, so:

$$\hat{c} = argmax_{h \in classes} \Pi_i P(f_i|h) \cdot \frac{c(h)}{\Sigma_{h \in H} c(h)}$$

This is much easier to compute, since we can just count the frequency of observing each feature value within observations that have the hypothesis class.

1.2. **ID3 Decision Tree.** Decision trees are classification algorithms which recursively partition training data based on feature values until each partition contains elements belonging to only one class or a maximum depth has been reached. At that point, the majority class in each final partition is chosen as the decision for that path. These partitions form a tree which can be used to classify unknown samples by putting the set containing that sample through the same partition scheme until it reaches a final decision. There are many possible choices for how exactly to partition the data. The partitioning algorithm we will focus on in this paper is ID3.

For ID3 tree-building, we consider splitting on one feature at a time. For each feature, we partition the current pool of observations based on the feature value, then do a weighted sum of the entropy of each new partition. The feature with the lowest total entropy is chosen for the partition.

Note: Since not all feature values may be present in the training data, when, during classification, we encounter a split in the decision tree which doesn't have a path

for a sample's feature value, we partition the sample down all paths and recurse, then make a final decision based on the majority class of the training data in all final partitions reached.

1.3. **Random Forest.** Although decision trees can have a probabilistic output when limiting depth, we can improve the granularity and accuracy of the model through ensemble learning. Random forests use a technique called bagging to combine many decision trees into a single model. The available training data is partitioned, then each partition is used to train a decision tree, which casts a vote based on its own probabilistic classification of unknown data toward the final decision made by the ensemble. In this paper, we will partition training data by partitioning the feature set into distinct sets of features, one per decision tree. Each decision tree will train on all observed samples with its own subset of the the total feature set. We will weigh each decision tree equally, so the class probabilities for any given sample for the random forest are just the average of the class probabilities over all the trees in the forest.

2. Related Work

There exists research in the area of political sentiment analysis using data from social media sources, with the motivation of improving or replacing current polling systems (Bakliwal et al. 2013). In those cases, classification techniques including naive Bayes and support vector machines have been used with some success. Notable difficulties in political sentiment analysis on social media include recognizing sarcasm (Bakliwal et al. 2013) and spelling errors. It is reasonable to expect fewer of these kinds of problems when analyzing State of the Union addresses, however the motivation of better polling no longer makes sense when analyzing speeches. If we can produce a successful probabilistic classifier for political speech, it can be used to objectively rank U.S. political speeches by party leaning, which would be interesting in order to track how individual politicians change views over time or to compare candidates in election cycles.

3. Method

[1]

3.1. **The Data.** We will consider every State of the Union address given between 1934 and 2016. It is reasonable to expect these speeches to contain very low rates of sarcasm and misspellings. Hopefully these speeches are sufficiently representative of speeches from the Republican and Democratic parties to create an accurate generalized model for speech classification.

3.2. **Preprocessing.** Following the standard used for preprocessing by Kummer & Savoy (2012), we tokenize the sentences and words in every speech, then replace every word occuring fewer than 4 times with an unknown token. We pad the start and end of every sentence with start and end tokens so that n-grams can represent words and phrases being at the start or end of a sentence.

3.3. **Feature Generation.** We consider n-gram, binary bag-of-words (bbow), and term frequency-inverse document frequency (tf-idf) features.
Before considering the effectiveness of any individual feature, we generate the feature values for all possible n-grams of length 1, 2, 3, or 4, all bbow features, and all tf-idf features. This is a simple task of counting all words and short sequences of words in the dataset.
On the State of the Union dataset, we end up with $974,083$ features before any filtering.

3.4. **Feature Selection.** First, we eliminate any features which are not present in at least 6 of the speeches. Although some of these features may help us get higher accuracy, they are more likely to overfit around a small number of speeches, and we want the classifier to be generalizable.
At this point, we have $37,608$ features. Now, for each feature, we find a split which maximizes the information gain of partitioning that feature at that threshold (as in a binary decision stub), and binarize the feature based on that split. As shown

[1]Code and data available at https://github.com/jwde/MLFinalProject

by Fayyad & Irani (1992), these splits must occur between two observations belonging to different classes. Therefore, we can sort the data on a feature value and calculate the entropy only at boundaries between classes. Since each information gain-maximizing split can occur at any point between the two adjacent values a, b, we (arbitrarily) pick the point half-way in between a and b.

Finally, we sort the set of binarized features by information gain and select the top 1000 features to train our models.

3.5. **Classification.**

3.5.1. *Models.* As a baseline, we classify all speeches as more Democratic. Since the majority of the speeches given between 1934 and 2016 were given by Democrats, this is the binary equivalent to a classifier considering only the prior probability of each class.

We also consider a naive Bayes model, a binary ID3 decision tree model with several different maximum depth settings, and a random forest model. The random forest model uses the binary ID3 decision tree model and one of several maximum depth settings for each tree. Oshiro, Perez, & Baranauskas (2012) suggest that the optimal number of trees in a random forest is generally between 64 and 128. We will try several values in that range.

3.5.2. *Testing.* We use 5-fold stratified cross-validation on the State of the Union dataset, and take the average accuracy over all folds for each model.

4. Results

Model	Max Depth	# Trees	Accuracy
Baseline	N/A	N/A	57.00%
Naive Bayes	N/A	N/A	50.07%
ID3 Decision Tree	5	N/A	65.05%
ID3 Decision Tree	10	N/A	58.45%
ID3 Decision Tree	20	N/A	60.47%
ID3 Decision Tree	40	N/A	57.00%
ID3 Decision Tree	80	N/A	61.93%
Random Forest	5	64	88.42%
Random Forest	5	80	81.48%
Random Forest	5	96	76.62%
Random Forest	5	112	68.71%
Random Forest	5	128	61.78%
Random Forest	10	64	92.92%
Random Forest	10	80	87.24%
Random Forest	10	96	80.16%
Random Forest	10	112	72.18%
Random Forest	10	128	65.17%
Random Forest	20	64	94.31%
Random Forest	20	80	88.42%
Random Forest	20	96	82.30%
Random Forest	20	112	72.25%
Random Forest	20	128	65.25%
Random Forest	40	64	90.64%
Random Forest	40	80	89.40%
Random Forest	40	96	77.96%
Random Forest	40	112	73.65%
Random Forest	40	128	66.09%
Random Forest	80	64	95.42%
Random Forest	80	80	86.05%
Random Forest	80	96	79.06%
Random Forest	80	112	71.28%
Random Forest	80	128	63.93%

Naive Bayes failed to outperform even the baseline, and the ID3 decision tree barely outperformed the baseline. However, the random forest model achieved accuracy above 90% with multiple settings. The best maximum depth / number of trees settings based on these results are maximum depth of 80 and 64 trees, which achieved an accuracy of 95.42%. Based on these results, random forests are clearly an excellent candidate to use to generate the semantic differential for political affiliation.

5. Applications

Now that we have an accurate probabilistic classifier for U.S. political affiliation, we can use it to place speeches on a scale between Republican and Democrat. If we define Republican as 0 and Democrat as 1, then the affiliation value $a(s)$ of each speech s is given by $a(s) = P(Democrat|s)$. This makes sense because as the probability of the speech being given by a Democrat increases, $a(s)$ gets closer to 1. Since the classifier is a random forest, it doesn't matter if we define $a(s)$ this way or as $a(s) = 1 - P(Republican|s)$ because $P(Democrat|s) + P(Republican|s) = 1$. One interesting application of the $a(s)$ statistic is to compare U.S. polititians in an election. This model provides an unbiased heuristic for the degree to which a polititian's rhetoric aligns with either the Democratic or Republican party.
Another interesting use of this statistic is to compare a polititian's rhetoric at one point in time with another point in time. By looking at $a(s)$, or the average of $a(s)$ over multiple speeches during certain periods of time, we could see how consistent individual polititians are. We could compare rhetoric during and out of election cycles as well as over larger periods of time to get a measure of rhetorical consistency or development.

6. Future Work

These results are encouraging, but would be more interesting given a comprehensive dataset of political speeches to apply the model to. The clear next step is to compile such a dataset and evaluate the affiliation of speeches.

Although we achieved high accuracy with a random forest model, it would be interesting to see if this could be improved further with tweaks to the random forest voting or partitioning techniques. In this paper, each tree has an equal vote. This does not necesarrily have to be true. We could weigh votes based on tree accuracy in predicting party affiliations in the training set. We could also try other partitioning criteria such as perceptrons.

We train the naive Bayes model using maximum likelihood estimates for our probabilities, however it may be advantageous to instead use a markov chain monte carlo (MCMC) technique such as Gibbs sampling for training. It would be interesting to see if results with naive Bayes improve when training with Gibbs sampling. Other research on sentiment analysis often considers support vector machines (SVM) as a classifier. It would be worth trying SVM in the future to see if we can beat the accuracy of the random forest.

7. Conclusion

The performance of random forests at political sentiment classification of State of the Union addresses is very encouraging and warrants further exploration into applications of the random forest model and others in semantic differential analysis of political speeches. Given a lack of unbiased data on how relatively moderate/extreme different speeches are, it is hard to evaluate that application, but its results are nevertheless interesting as a heuristic.

References

[1] "State Of The Union Addresses 1790-2016". State Of The Union. N.p., 2016. Web. 6 Mar. 2016.

[2] Bakliwal, Akshat et al. "Sentiment Analysis Of Political Tweets: Towards An Accurate Classifier". *Proceedings of the Workshop on Language in Social Media (LASM 2013)* (2013): 49-58. Print.

[3] Chan, Jonathan Cheung-Wai, and Desir Paelinckx. "Evaluation of Random Forest and Adaboost Tree-based Ensemble Classification and Spectral Band Selection for Ecotope Mapping Using Airborne Hyperspectral Imagery." *Remote Sensing of Environment 112.6* (2008): 2999-3011. Web.

[4] Fayyad, Usama M., and Keki B. Irani. "On the Handling of Continuous-valued Attributes in Decision Tree Generation." *Mach Learn Machine Learning 8.1* (1992): 87-102. Web.

[5] Ho, Tin Kam. "Random Decision Forests." *Proceedings of 3rd International Conference on Document Analysis and Recognition*. Web.

[6] Hsu, Chih-Wei, Chih-Chung Chang, and Chih-Jen Lin. "A Practical Guide to Support Vector Classification." (2010): n. pag. Web. 3 May 2016.

[7] Kummer, Olena, and Jacques Savoy. "Feature Selection In Sentiment Analysis". *CORIA* (2012): 273-284. Print.

[8] Oshiro, Thais Muyami, Pedro Santoro Perez, and Jos Augusto Baranauskas. "How Many Trees in a Random Forest?" *Machine Learning and Data Mining in Pattern Recognition Lecture Notes in Computer Science* (2012): 154-68. Web.

[9] Resnik, Philip, and Eric Hardisty. "Gibbs Sampling For The Uninitiated". (2010): 1-22. Print.

[10] Strobl, Carolin, James Malley, and Gerhard Tutz. "An Introduction to Recursive Partitioning: Rationale, Application, and Characteristics of Classification and Regression Trees, Bagging, and Random Forests." *Psychological Methods 14.4* (2009): 323-48. Web.