CS 229 Final Project: Bias Detector
or, Using Language Models To Identify Editorial Political Slant

Rush Moody -- rmoody@stanford.edu

December 12, 2014

1. Introduction

Text classification techniques have proven useful in a variety of domains, ranging from simple but vital tasks like spam filtering to more complex tasks like determining text authorship or performing sentiment analysis on product reviews. In this project I attempt to apply such techniques to identify the political slant of editorial articles without using any prior knowledge of the authors' work or political leanings. This is essentially a classification problem on editorial articles; our two classes in this domain are conservative-leaning and liberal-leaning editorials, and I use a variety of language models to aid in their classification. The resulting classifiers tend to perform well at classifying data that comes from the same editorial board domain as our training data, but does not generalize as well to classifying the work of several individual columnists in a separate data set.

2. Related Work

In the domain of text classification multinomial Naïve Bayes classifiers have consistently proven themselves to be remarkably effective despite their simplicity, especially when combined with data pre-processing steps such as stopword elimination and word stemming, as well as LaPlace smoothing or more sophisticated methods of handling unobserved tokens. Relaxing the Naïve Bayes assumptions to allow for n-gram Markov chains further improves general performance without introducing much further complexity, and these classes of models have achieved text classification performance that compares favorably to state-of-the-art SVM-based approaches [1]. In the area of political bias detection, existing work has seen moderate success in detecting and classifying biased opinions both in Twitter posts [2] and at the sentence level using recursive neural networks [3]. Given that these approaches generally use finer-grained class labels and/or work on classifying significantly smaller bodies of text, applying the aforementioned text classification models at the level of an editorial article seemed like a promising and logical approach.

3. Data

Our data consists of two primary data sets. The first data set consists of more than 6,000 newspaper editorial board columns published during the previous four years from four different newspapers with a known and consistent editorial bias, with equal numbers of liberal (the New York Times and the Washington Post) and conservative (the Wall Street Journal and the Washington Times) articles. We use the known biases of these institutions as an informal labeling scheme for this data, and maintain disjoint subsets of the data for training and testing our models. Although there are likely some articles in the data set that do not strictly adhere to our labeling scheme, as we proved in problem set 2 we can overcome such noise by noting that our probability of corruption, τ, is likely fairly small and by using large amounts of training data. There are also bound to be some included editorial articles that are not expressly aimed at advocating for a liberal or conservative agenda, but here again we rely on the fact that voicing such an agenda is one of the primary purposes of an editorial board column and that thus our assumption applies to the vast majority of the training examples.

Our second data set comes from of the combined editorial portfolios from the last three years of two conservative columnists (David Brooks and Ross Douthat) and two

liberal columnists (Charles M. Blow, and Thomas Friedman), from which we randomly sample 125 articles each. Here again we use the known editorial slant of each columnist as an informal labeling scheme for choosing the class label to assign to entries in this additional test set. Our labeling assumption is slightly more dubious for this data set, given that many of the articles might be touching on a non-political topic relevant to the author instead of advancing a liberal or conservative agenda, but we still use this set to help determine whether our models generalize beyond an editorial board setting.

4. Features and Models

For all of our classifiers our input features are drawn directly from the text of the articles in our dataset, each of which we tokenize into a set of distinct words. For the Naïve Bayes and N-Gram language models, the input to our model is a vector of the n words in the article $[x_1, \ldots, x_n]$, with each variable x_i taking on values [1, |V|] where V is our vocabulary. For the perceptron model our input is a vector $X \in R^{|V|}$, with each element x_i being a binary-valued variable indicating the presence or lack of word $v_i \in V$ in the input article.
Note that for all of our models we use the nltk python module to tokenize our input and remove common stopwords.

Naive Bayes

Our first model is a multinomial Naive Bayes classifier that attempts to model $p(x_1, \ldots, x_n, c)$, which is the joint probability of generating the given article for the two classes of political bias, and then selects the class with the higher probability. Using the Naïve Bayes assumption, this likelihood for each class of article is given by $p(c) * \prod p(x_i \mid c)$. We learn the individual probabilities $p(x_i \mid c)$ by calculating the sum $\sum 1\{x_i = v_{xi}\}$ for each word over every article and then normalizing each sum by dividing by the total number of observed tokens. We also apply +1 LaPlace smoothing and use the probability 1/|V| as the probability for unobserved tokens. We treat the prior class probabilities as a tunable parameter (within reasonable limits), as we have no good way of estimating the global prior likelihood of liberal versus conservative editorials.

N-Gram Language Model

Our second model extends the Naïve Bayes classifier by loosening the independence assumptions from that model. Instead of assuming each word is generated independently, we now use a k-gram Markov chain generation model such that $p(x_1, \ldots, x_n, c)$ equals $p(c) * \prod p(x_i \mid x_{i-1}, \ldots x_{i-k+1}, c)$, for $k = \{2, 3, 4\}$. We learn the parameters for this model in a similar fashion, calculating the sum $\sum 1\{x_i, x_{i-1}, \ldots x_{i-k+1} = v_{xi}, v_{xi-1}, \ldots, v_{xi-k+1}\}$ and then normalizing by the total number of observed k-grams. Here again we apply +1 LaPlace smoothing based on the number of observed trigrams to handle unobserved tokens.

Perceptron

Our last model is a perceptron, which takes as input a vector of binary valued values $X \in R^{|V|}$ and makes a prediction using a learned weight vector W and bias term b; if $W^T X + b > 0$ we predict one class while if $W^T X + b <= 0$ we predict the other. We use uniform 0 initialization and a randomized stochastic training approach to learn W and b, and use an n-hot bag-of-words vector generated from the content of an article as our input vector. Our bias term is implicitly included in our model by adding an entry to X that is set to 1 for every article and thus is trained for every example in our training set. I experimented with several convergence criteria and settled on setting a hard limit of observing 4,000 training examples; our randomized training approach also yields significant variations in performance from run to run.

5. Results

Table 1 summarizes our results for the three models on our various data sets:

Model	Training Accuracy (~6000 articles)	Test Accuracy (Editorial Board set, 400 articles)	Test Accuracy (Columnist set, 500 articles)
Naïve Bayes	99.5%	89.5%	55.8%
N-Gram (n=2)	99.7%	90.5%	58.2%
N-Gram (n=3)	99.8%	92.0%	61.4%
N-Gram (n=4)	99.7%	88.5%	43.2%
Perceptron	98.0%	90.5%	60.2%

Table 1: Results

6. Discussion and Analysis

On the whole our results were mixed: when classifying articles from the same domain as our training set (NYT/WP/WSJ/WT editorial board columns) we consistently achieved a fairly high level of classification accuracy. Moving from a simple N.B. classifier to an n-gram model also appears to help accuracy for $n < 4$; for higher values of n our language data becomes too sparse and performance suffers. This makes intuitive sense, as the number of possible n-grams increases exponentially in n, thus requiring exponentially more training data in order to accurately estimate the probability for each n-gram. The optimal value for n seen in [1] appears to be 5-grams, which implies that we could likely achieve further improvement if we had sufficient training data to avoid the issue of sparsity.

Our perceptron model appears to offer similar performance on the first training set to the language model-based classifiers. The low degree of training error seems to imply that our training data is at least close to being linearly separable into two distinct classes, making the perceptron model a valid choice for this domain.

While unfortunate, it is relatively unsurprising that our models were less adept at classifying the works of our other selected liberal and conservative columnists. Our model displayed a consistent tendency to classify these columns as being liberally biased, thus resulting in much worse performance when attempting to classify the conservative columnists articles from our second test set. Given that each columnist is likely to have a consistent and unique pattern of language usage, this shortcoming in our classifier could not be meaningfully addressed without obtaining richer sources of training and testing data. As all of our training data currently comes from the works of biased newspaper editorial boards there is no guarantee that the trends therein will generalize well to individual columnists with a liberal or conservative bias; columnists will often inject far more first-person commentary or focus on specific issues within their area of interest or expertise, significantly affecting their word use distributions and thus causing these columnists work to diverge from the distributions learned by our models. This issue could be helped by either including a broader selection of sources of labeled training data to yield a more generalized model for liberal versus conservative language usage, or by obtaining and processing non-editorial articles from the same newspapers to help identify editorial board-specific stylistic differences that may be adversely affecting our models (although there is no guarantee that such techniques will be able to resolve the problem entirely). A broader selection of test data would also give us a much better idea of the generalization error of our models than the four columnists used for our second test set.

7. Top Features

To help analyze the performance of our systems and gain some insight into how they classify articles we will now look at the most common unigram tokens observed by our Naïve Bayes classifier for each class, as well as the top weighted words (both negative and positive) for our perceptron model. We will then use our pre-existing knowledge of the domain (political and conservative political discourse and opinion pieces) to make some comments about the results.

Liberal	Conservative
United	*U.S.*
people	*tax*
I	*Mr.*
years	*year*
Mr.	*last*
Obama	*Iran*
law	*government*
said	*n't [not]*
could	*President*
percent	*Obama*

Table 2: N.B. Top Tokens

Positive (Lib.)	Negative (Cons.)
need	*everyone*
decade	*U.S.*
rights	*Administration*
support	*n't [not]*
citizens	*better*
community	*President*
also	*security*
published	*today*
United	*liberal*
short	*China*

Table 3: Top Perceptron Features

Naive Bayes

Table 2 contains the most common words observed for our two classes, not including common stopwords that were removed during preprocessing. While some much-discussed words were likely to occur in either class of opinion article, which unsurprisingly includes the tokens "Mr." and "Obama", the differences in the remaining tokens are somewhat revealing. The conservative tokens include the words "tax", "Iran", and "government", all of which seem fairly consistent with the conservative talking points from the past few years of decrying the intrusion of government into people's lives and of highlighting the dangers posed by our ideological enemies abroad. The liberal tokens include words like "people" and "law", which may reflect the liberal emphasis on collectivism. Additionally, the conservative tokens include the negation "n't", which does not appear among the top 30 tokens for the liberal class, possibly reflecting a conservative preference for negative language in the current political climate.

Perceptron

Table 3 contains the top 10 most positive and negative weighted words from our perceptron classifier, which are the words most strongly indicative of liberal and conservative editorials respectively. Interestingly there appears to be significant overlap between the Naïve Bayes top features and those learned by our perceptron model. Our earlier insight about the preference by conservatives for negative language is validated by the inclusion of the familiar "n't" token in the top negative weighted features, and the presence of "security" and "China" further touches on the conservative talking points mentioned previously. On the liberal side, the words "citizens", "community", and "rights" seem to strongly related to the "people" and "laws" tokens that we observed in the liberal Naïve Bayes language model.

Although it is dangerous to read too closely into the meaning and content of these top features, the consistency between both models as well as our shallow analysis of their semantics seems to validate our insight that a language-based approach can be used to differentiate between the two classes.

8. Conclusions

Given the limitations in the data at hand we were still fairly pleased with our overall results, especially those within the editorial board domain. Our top extracted features seem to be consistent between models and seemed to make intuitive sense for the classes that we are attempting to model. The noisy

and imperfect sources of data posed serious challenges when attempting to evaluate our models' ability to generalize beyond the domain of our training set; however, we believe that this is an area ripe for further work, given the large number of published editorials in existence and the general effectiveness of language models in text classification problems. On the whole this proved to be an interesting attempt at applying machine learning to a useful problem and provided valuable insights into the importance of comprehensive and clean data in supervised learning problems.

9. Future Work

As touched upon previously, the first and most vital step towards validating and improving our results would be to obtain cleaner and more comprehensive sources of data by either hand-selecting a diverse selection of strongly biased articles or constructing a broader-sourced training corpus. Either approach would give us a training set that more closely approximates the true liberal and conservative language distributions by eliminating noise and reducing the impact of newspaper-specific stylistic guidelines on language usage. With an improved training and test set we could then re-run our existing models to see if they generalize beyond just the editorial domain. Additionally, given the success of neural net approaches to text classification such as those outlined in [4], we would experiment with modifying our perceptron to be a multi-layer neural network to help pick up more nuanced trends from the training data without having to rely purely on statistical word counts or hand-engineered features. We could also experiment with features that involve the title of the article, or features such as average word length that explore other properties of the text besides the actual words themselves.

10. References

1. F. Peng. (2003). *Augmenting Naïve Bayes Classifiers with Statistical Language Models* [Online]. Available: http://scholarworks.umass.edu/cgi/viewcontent.cgi?article=1090&context=cs˙faculty˙pubs

2. D. Maynard and A. Funk. (2011). *Automatic detection of political opinions in Tweets* [Online]. Available: https://gate.ac.uk/sale/eswc11/opinion-mining.pdf

3. M. Iyyer, P. Enns, J. Boyd-Graber, and P. Resnik. (2014). *Political Ideology Detection Using Recursive Neural Networks* [Online]. Available: http://www.aclweb.org/anthology/P/P14/P14-1105.xhtml

4. J. Nam, J. Kim, E. L. Mencfa, I. Gurevych, and J. Furnkranz. (2014). *Large-scale Multi-label Text Classification — Revisiting Neural Networks* [Online]. Available: http://arxiv.org/pdf/1312.5419v3.pdf