Compass

Daniel Alpert, Spencer Evans, Dylan Farrell

December 8, 2017

1 Introduction

While many newspapers claim to strive for journalistic objectivity, for a long time both common and scholarly opinion have held that various news outlets have different underlying political ideologies and that these ideologies manifest themselves in the articles that the news outlets display. This belief has become more and more prevalent in recent years as American politics have grown increasingly polarizing. For example, President Donald Trump has spent the last two years railing against the "liberal media" outlets like The New York Times and The Washington Post, while many democrats have berated the perceived conservative biases of sources like Fox News and Breitbart News. We believe that an in-depth analysis of the political biases of popular newspapers is critical in today's political climate and that raising readers' awareness of the biases of their news will encourage them to explore other sides of the issues and become a more enlightened electorate.

These perceived political biases of newspapers can be broken down into two categories: selection bias–choosing to cover certain issues but not others; and framing bias–presenting issues in a certain tone. For example, Fox News has *selected* to devote lot of recent coverage to ties between anti-Trump FBI agents and FBI Director Robert Mueller, who is leading a special investigation into the Trump Adminstration's pre-election contact with Russia, in an effort to question his position as an unbiased investigator. The New York Times, on the other hand, has not devoted coverage to these ties and has instead focused on covering other aspects of the Mueller investigation. And while both newspapers have covered the content of the new Republican tax bill, Fox News has *framed* the bill in a much more positive light.

While an understanding of both selection bias and framing bias is critical to have a complete picture of digital media biases, in this project we choose to focus on quantifying the framing biases of three popular newspapers: Fox News, CNN, and The New York Times. With this end goal in mind, we use artificial intelligence algorithms to classify the political leaning of individual sentences and to create a model from these classifications that accurately places both articles and entire newspapers on the political spectrum.

2 Background and Related Work

There have been many prior studies on the political leanings of speeches, sentences, articles, and newspapers. These have usually taken one of two approaches: audience-based and content-based

analysis. Audience-based studies examine the political leanings of newspaper readers and then infer the leanings of the newspapers themselves under the assumption that most readers follow the newspapers that most closely match their own ideologies. While this approach can be effective in determining the relative ideologies of major newspapers, it does not provide much insight on quantitative differences in the magnitude of the biases between newspapers. Content-based studies, on the other hand, attempt to quantify biases using data from published articles. While these provide more direct insight on the political leanings of newspapers, classifying articles on a large scale either through machine learning sentiment analysis or through manual labor poses a significant challenge.

The previous work most similar to our project is a study conducted by Budak, Goel, and Rao called "Fair and Balanced? Quantifying Media Bias Through Crowdsourced Content Analysis" [2]. In this study, Budak, Goel, and Rao used crowd-sourcing to manually label the political bias of 10,502 political articles from 15 major US newspapers. They labeled the articles on a scale of -1 to 1 (with 1 being the most conservative). Taking the average article score for each newspaper resulted in an ordering of the major political newspapers that was in line with previous reserach and common perception. They were surprised, however, to find that the magnitude of the biases differences between major news outlets was much smaller than originally anticipated. For example, The New York Times (-0.05) and Fox News (0.11), which are commonly viewed as papers with very different political leanings, had a separation of only 0.16 on the political scale. They concluded that this was due to a number of reasons, one being that most news articles in most newspapers are actually relatively neutral in their framing of facts. They maintained, however, that while the ideological differences that they found between major news sources were small, they were still very much present and statistically significant.

Although Budak et al. used several machine learning techniques in selecting the articles for their study, they did not use sentiment analysis in the labeling of these articles, but instead scored each one manually through extensive crowd-sourcing. While our own work goes beyond just classifying individual sentences, the prior work that was most relevant to our AI-based approach to determining idealogical bias was a 2014 study conducted by Iyyer et al [3]. In this study, they created a dataset called the Ideological Books Corpus (IBC), which classifies the political bias of sentences and phrases as liberal, neutral, or conservative [4]. Using recurrent neural networks and this dataset, they were able to achieve a 69.3% classification accuracy on the political bias of individual sentences. We used this IBC dataset to train our own sentence classifiers and hoped to achieve similar classification accuracies.

3 Problem Specification

In this project, we attempt to solve the NLP task of quantifying the political leaning of both individual articles and entire newspapers on a scale from -1 to 1, with -1 being the most liberal and 1 being the most conservative. In order to do this, we first considered a variety of methods for classifying sentences (Naive Bayes, Random Forest, and Recurrent Neural Networks) and then devised a method of using these classifications to score entire articles and newspapers.

4 Data

4.1 Ideological Books Corpus (IBC)

To train a supervised model for sentence classification, we need data that includes both a sentence and its label (policial leaning). We reached out to Professor Mohit Iyyer to get access to the Ideological Books Corpus data he had used while a Ph.D. student at the University of Maryland. This data includes 4,062 prelabeled sentences as ideologically liberal, neutral, or conservative.

Preliminary modeling on this data was not very accurate in part, we believe, due to the focus on having ideological sentences as opposed to partisan ones. This difference is subtle—ideological sentences focus more on policy, while partisan ones focus on party lines.

Ideological liberal: Indeed, Lind argues that high profits and high wages reinforce each other because workers then have the wherewithal to buy the products they are making.

Ideological conservative: Another major omission in this report is that when a permit holder kills an attacker in self-defense, the police often arrest them while they investigate the shooting.

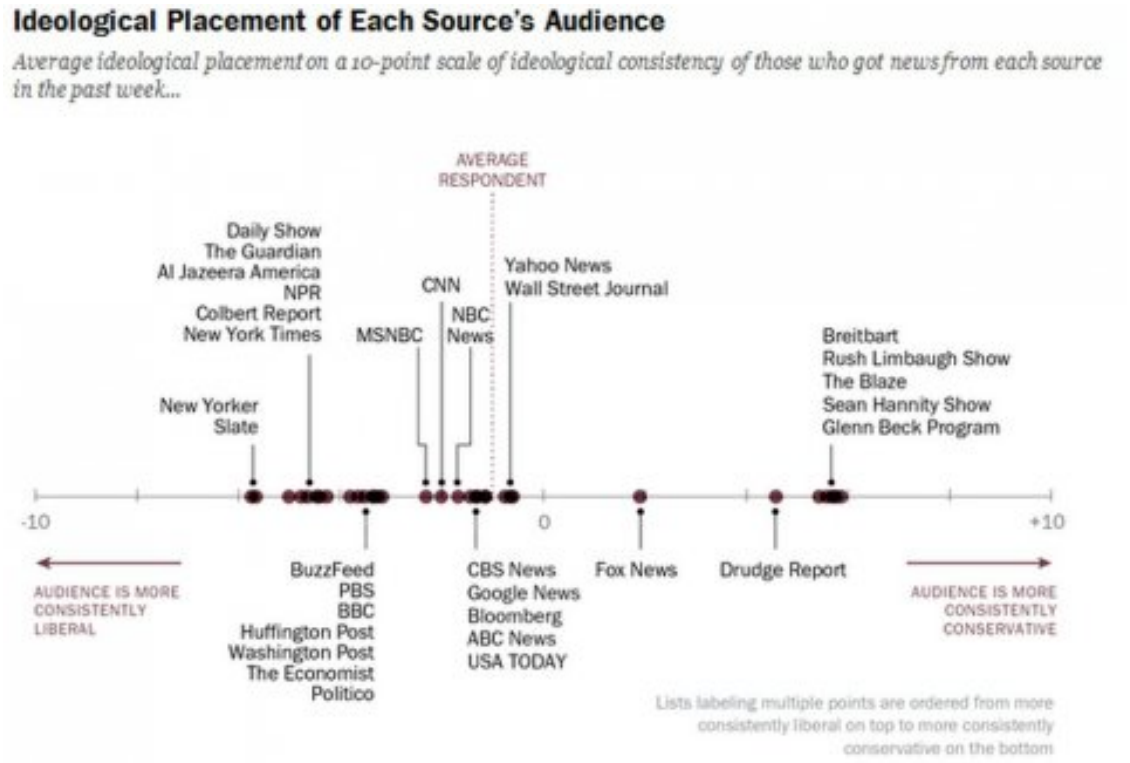
Given the vitriol of many of the news outlets we read, we wanted to include more partisan sentences as well, so we scraped many current New York Times, CNN, and Fox News articles and op-eds and manually labeled each sentence in these articles as liberal, neutral, or conservative. Over all, we added 1,235 sentences to the corpus, leaving us with 5,297 sentences to train on.

4.2 Global Vectors for Word Representation (GloVe)

Developed by the Stanford NLP group, GloVe is an "unsupervised learning algorithm for obtaining vector representations for words." Made available under the Public Domain Dedication and License, we accessed a file of over 400,000 pre-trained word vectors. Each of these words is represented by a 50-dimension vector. These vectors have interesting functionality, like finding the most similar word to a given word (most similar to "excited" is "thrilled"), understanding semantics ("king" + "woman" - "man" = "queen"), and BLANK. Thus, instead of using a bag-of-words approach to classify sentences, we could also explore a vector-based approach using this data.

4.3 Articles from Major US Newspapers

We scraped 192 political articles from Breitbart News, Fox News, CNN, The New York Times, and The New Yorker Slate. While we would have liked to label each article manually, we did not have time to go through them one by one. Instead we labelled them on a scale of -1 to 1 (1 being the most conservative) using rough estimates of the political leanings of each of the five newspapers from the chart below. These estimates were as follows: Breitbart, 0.7; Fox News, 0.3; CNN, -0.1; The New York Times, -0.3; The New York Slate, -0.5.



5 Sentence Classification

The meat of this project lies in correctly classifying sentences based on their political leanings. With the IBC we trained many classifier models, weighing the capabilities, drawbacks, and accuracy of each. We explored Naive Bayes, Random Forest, and various Recurrent Neural Networks. With the GloVe data, we had the option to represent the words in each sentence as vectors instead of strings or bags-of-words.

5.1 Basic Classifiers

5.1.1 Naive Bayes Classifier

The first, and most basic, approach that we took was training a Naive Bayes classifier. Naive Bayes is a great approach if there are certain words or combinations of words that tend to appear frequently in sentences of one classification but not the others. Sentences are fed into this model represented as a bag-of-words, where the sentence is a vector of the number of appearances each word makes in the sentence. This model does not, however, take into account the ordering of the words so, for example, it would interpret the sentence "Republicans are smarter than Democrats" the same as "Democrats are smarter than Republicans" despite these sentences having very opposite meanings.

Before training this model, we removed all Python "stopwords"—words that are common (like "the" and "and") which have no political leaning and may muddle the interpretation of a sentence

more than help it.

We trained this model on two important parameters: Neutral threshold and ngrams. Because our data undersamples neutral sentences, our classification probabilities continually underpredict the probability of the sentence being neutral. Because of this, we tuned our model to see if there was an optimal probability that we should set as a threshold to predicting a neutral sentence. This parameter was tuned to be 0.17, meaning if a sentence's probability of being neutral is predicted as at least 0.17, we classify the sentence as netural. Ngrams is how many words in a row we include as a word. A ngram value of 3 means that all individual words, sets of 2 words in a row, and sets of 3 words in a row would count as features in the Naive Bayes model.

On this fully tuned model, we achieved an average accuracy of 57.0% accuracy.

We are able to look at the most influential words for classifying each label. The top 10 words that signify a sentence will be a given class are below:

Table 1: Most important words per class

	Liberal	Neutral	Conservative
1	economic	make	state
2	new	good	new
3	energy	us	federal
4	health	even	make
5	public	people	economic
6	people	jerusalem	one
7	government	new	free
8	care	one	people
9	tax	said	would
10	would	would	government

The advantages and pitfalls of Naive Bayes can be seen by looking at this table. The advantages: the model is very interpretable. There are certain words that are most important in indicating the leaning of an article. Seeing the word "energy" for example may tip us off that the sentence is left-leaning. The glaring pitfall: many of these words show up as influential for multiple classes. The most important liberal word, "economic," is the fifth most important conservative word. "New" is second most important for both. As alluded to in the example earlier, simply put, liberals and conservatives write about the same topics and use the same words.

5.1.2 Random Forest—Bag of Words

A random forest model could also work well with this data. A random forest classifier is an ensemble method, taking the average classification of many decision trees. This helps correct for a decision tree's tendency to overfit the data. Decision trees perform variable screening for us—the word features that are unimportant will not be in the tree and those that are more important will generally be split higher up in the tree. Decision trees are easily interpretable too. At each feature, we choose which direction of the tree to go down depending on the value of the feature. Tuning for number of trees in the ensemble, max depth, maximum features, and neutral threshold, we achieved an average accuracy score of 56.4%, which is very similar to that of the Naive Bayes classifier. While we had expected an increase in accuracy, there are a few reasons for this underperformance. For

one, random forests require training numerous parameters. For each parameter, we held all other parameters constant while tuning. There may, however, be an interaction between parameters that we did not pick up on. This would be too expensive—timewise and computationally—for this project. Also, with our limited dataset, we may have "more random" forests that lead each of the trees in the ensemble being greatly different.

5.1.3 Random Forest—GloVe

In the GloVe dataset, each word is represented as a 50-dimensional vector of floats. A sentence can be represented by combining the word vectors in some way. For our input into the random forest model we chose to represent a sentence as the weighted average of the words in the sentence. We used the TF-IDF algorithm to get the log weights of the words in the overall corpus and properly weight the words in the sentence. We used the weighted average as opposed to the regular average because we felt as though more niche words that appear less may be more politically charged and should thus be weighted more highly. This method would yield well if certain politically charged words were similar dimensionally. For example, if more liberal words tended to have higher values for the first few dimensions, a sentence could be classified as liberal by having those same features.

Using a 50-dimensional sentence vector as input, we went through the same tuning process as with the earlier random forest model (excluding tuning for maximum features–we used all 50). We ultimately got an average accuracy of 51.2%. This result could imply a few things. For one, a vector representation of the sentence may not have been the best representation. Another would be that the weighted average was not the best combination. We could instead have trained on different linear combinations of the individual words to predict a political leaning, thus keeping each individual word in the model instead of aggregating them.

5.2 Neural Networks

The main algorithm we wanted to try and were most excited about was a recurrent neural network (RNN). While most of the other algorithms we've discussed thus far consider each word independently, a task as difficult as determining the political ideology of a sentence, something that even educated people can struggle with, requires a more nuanced approach to be effective. While there are certain words that have strong correlations with political leaning - "estate tax" vs. "death tax" for example - it is the phrase level combination of words, taking into account the meaning of each word in the phrase and the syntax that links each of them, that best models the meaning of the sentence. Thats why RNN's are a natural fit for this type of problem. Similar to how humans process the world around us, RNN's pass on their belief state at time t to the state at time t+1. This ability to take past states into account when modeling its current state should allow it to capture the semantic meaning of full sentences. RNN's have achieved state-of-the-art accuracy on many NLP problems due to its natural proficiency for understanding sequential data and we believe it is a great fit for this type of problem as well.

We began by trying to implement an RNN model by scratch, without using any libraries. While this proved very pedagogical and we made some solid progress, we were getting too close to the deadline to continue down that path and instead started using the keras library. We started out testing a standard RNN with Long Short-Term Memory (LTSM). We used the LSTM layer because while RNN's in theory should be able to remember important features from many states ago, in

reality, they often are not able to as they suffer from the vanishing gradient problem. LSTM cells can be thought of as additional memory that is preserved through each step of the input sequence. They allow RNN's to remember more detail and context of the sequence than they otherwise would be able to. Our baseline RNN with LSTM achieved a 29.6% accuracy on the IBC data. This was a disappointing first result as this was worse than guessing randomly between liberal, conservative, and neutral. To make sure that we hadn't implemented the RNN improperly we trained and fit the model on movie review data from a kaggle competition that did binary sentiment analysis classification (positive or negative review). Our model achieved 89.1% accuracy on that dataset so this confirmed that we have a solid model but need to configure it better to our data.

We then reviewed some of the literature on the topic to see how we might be able to get better results, and read an article by Yin et al. entitled "Comparative Study of CNN and RNN for Natural Language Processing," which demonstrated that Convolutional Neural Networks and RNNs provide complementary information [5]. "While the RNN computes a weighted combination of all words in the sentence, the CNN extracts the most informative ngrams for the relation and only considers their resulting activations." We layered the CNN and RNN such that the CNN received the input, which processed it then passed its output to a pooling layer, which then passed its output to the RNN with LSTM which then ran on the CNN's processed data. With this configuration we achieved 34.8% accuracy. While certainly an improvement on our previous version, we were still not very excited about this result as it was not significantly different from randomly guessing.

We then adopted the GloVe pre-trained vector representation for words into our model. This dataset represents each word in their dataset as a vector of 100 integers, obtained by crawling through every wikipedia page to learn the relationship between every word and another. We then converted each of our words into the GloVe vector representation of the word and then passed the vector into our model. We believed that with this richer representation of each word, we could account for our smaller dataset by allowing our model to understand the relationship between similar words. In theory, this would allow our model to understand the meaning of a word it has never seen before if it is similar in some way to a word that is has seen before. One trivial example would be the word 'republican'. Even if our model has never seen the word 'republican' before, if it has seen the word 'conservative', it should be able to understand that these are intrinsically related and should be able to interpret them in a similar manner. Testing with the GloVe implemented as well, we achieved 31.2% accuracy on our basic RNN with LSTM and 51.3% with our RNN and CNN - however we have not been able to consistently reproduce the 51.3% score (mean reproducible score is 37%).

We then removed all the stopwords from our data under the hypothesis that this might remove some of the noise in our data. The rationale for this was discussed in an earlier section. This did not improve our model.

We then considered switching from a LSTM cell to a Gated Recurrent Unit. They address the vanishing gradient model in a similar manner, storing information in cells that get passed from one RNN state to the next. The IBM Research paper indicated that GRUs can perform better "when sentiment is determined by the entire sentence or a long-range semantic dependency rather than some local key-phrases." We hypothesized that with political orientation, key phrases, while still considering the semantics, could be more predictive of political leaning than longer-range semantic

meaning. Examples include, 'obamacare' vs. 'Affordable care act'. With hyper-parameter tuning (discussed more in depth later), we achieved a 39.9% with the GRU cell instead of the LSTM cell.

We then did extensive hyper-parameter testing on all of our models. To do this, we built a RNN class with fit, predict, evaluate, and scoreAccuracy methods. We tried to integrate it with sklearn's API so that we could use their cross-validation and model selection tools; however, after much frustration, punctuated with brief moments of elation, followed by frustration, we abandoned the sklearn integration and built our own randomized grid search algorithm. We did some basic research on hyper parameter tuning methods and after reading a 2012 paper by James Bergstra and Yoshua Bengio, (particularly section 2 which discusses random vs grid search for optimizing neural networks) we choose a randomized grid search rather than a normal grid search because of its ability to search a larger parameter space and its ability to give a more texturized view of the space [1]. We used randomized grid search to identify which areas of the parameter space were promising before zeroing in on them. The parameters we tested on include number of epochs, batch size, validation split, loss function, dropout rate, whether to use an optimizer, which optimizer to use (either adam or RMSprop), whether to include the CNN layer, the kernel size of the CNN, and whether to use a GRU or LSTM RNN cell. Some of our notable findings include boosting our LSTM cell to 41.0% accuracy, which suggests there isn't a significant difference between LSTM and GRU cells for our use case.

To try to address our shortage of data, we then broke down the IBC data into each relevant phrasing and then training on each phrase, including the original sentence. The breakdown into relevant phrasing was built into the dataset by professor Iyyer who we got the dataset from. So each sentence was modeled as a tree, with the full sentence as the root and subphrases branching off. Training on this we achieved a score of 35.9%. However, this was one of the last things we came to and an area that we would like to do more extensive hyper parameter tuning and cross validation on.

One last thing we tried was changing our model from accepting single sentences to entire articles. We hypothesized that with many types of individual sentences it can often be easy to tell that there is a definite political orientation but difficult to tell in which direction due to a lack of context. For example, the sentence, "They were misleading the public and refused to work across party lines to reach a compromise" clearly has a political orientation; however, without the context of knowing which group 'they' refers to, its impossible to know whether this is a liberal or conservative sentence. Because these results were taken from many different news outlets that we represented on a scale of -1 to 1 (with -1 being the most liberal and 1 being the most conservative), we don't have an accuracy score to report, but our mean squared error was 0.19.

5.3 Model Comparison

Table 2: Classifier Comparison

Classifier	Accuracy
Naive Bayes	0.570
Random Forest-Bag of Words	0.564
Random Forest-GloVE	0.512
RNN + LSTM	0.296
RNN + LSTM + CNN	0.348
RNN + LSTM + GloVe	0.312
RNN + GRU + GloVe	0.399
LSTM + CNN + GloVe	0.410

6 Article Classification

We classified articles using an unsupervised approach taking the average classification of the individual sentences in the article (liberal = -1, neutral = 0, conservative = 1). We scraped 1511 total articles from The New York Times, CNN, and Fox News. In addition to giving a classification as liberal, neutral, or conservative, our model also gives probabilities of the sentence belonging to each class. We used a weighted probability of $-1 * P(liberal) + 0 * P(neutral) + 1 * P(conservative)$ which simplifies to $P(conservative) - P(liberal)$, the difference between the probability of conservative and liberal. The most liberal score possible is -1 and the most conservative is 1. In this scoring system, the neutral score need not be high to get a totally neutral score of 0. If liberal and conservative are equal, we get a score of 0 regardless of the probability of neutral.

Table 3: Unsupervised political score by news outlet

News Outlet	Political Score
New York Times	-0.134
CNN	-0.118
Fox News	-0.108

Running this algorithm on the articles from these three news sources confirmed out hypothesis about the ordering of the outlets. Of the three, NYT was the most liberal, then CNN, and Fox News was the most conservative. The closeness of their scores–all within .026 of each other was surprising to us. While all three of these differences were statistically significant to the 0.05 level, the differences in their means is negligible. This either indicates that our model is not very good or that the media is not as polarized as we had thought. We were also surprised that all three—Fox News included—had slight liberal biases according to our model.

7 Discussion

Overall, we tried eight different models to classify our sentences under a political ideology and along party lines. We had expected Naive Bayes to be our most simple, and therefore baseline, model, with more advanced methods utilizing Recurrent Neural Networks performing best. We

were disappointed to find that our Neural Networks did not beat our baseline, but we attribute a lot of that to a lack of data—more and better labeled data may have vaulted our RNN above the other models, as a similar approach by Budak et al mentioned earlier got 69% accuracy with a RNN.

Given the limitations of the accuracy of our sentence classifiers, it is difficult to make conclusions on our article scores with a high degree of confidence. That said, our results in this area aligned in many ways with both common perception and with the results found in Budak et al's study. Of the three news outlets we examined, we found that Fox News is the most conservative, followed by CNN and then The New York Times, which was the most liberal. We were surprised to find that the ideological differences between the three were so small, but this is consistent with the results from Budak et al's study.

However, while we and Budak et al found that the news is not as polarizing as is perceived, we still believe that the news really is polarizing but potentially not too significantly at a sentiment level. For one, news sources are inherently showing us different news. In the last three days we have followed the news and seen all news outlets report on Alabama Senate candidate and accused sexual assaulter Roy Moore. There have been seven different articles on the front page of the New York Times that have reported on these grave accusations. Breitbart News has only run two articles in that time: one detailing how Moore will still win and the other attacking Moore's accusers. In short, we believe the polarization of outlets in large part comes from selection bias of articles—what the sites choose to write about, but not as much what the sentiment behind the articles is.

Additionally, most articles that were not op-eds we actually thought were written in a fairly neutral way. While there are certainly ideological differences, these sentences are often few and far between. Additionally, sources generally quote the actors in the other party, adding "opposite" politically charged sentences to their outlets. When the New York Times, for example, says something like "Trump tweeted that this bill will cut taxes for the majority of middle-class Americans," the sentence may be perceived as conservative in a more liberal outlet. We also noticed that without functionality to link pronouns to known political figures, we lose the ability to identify many very politically charged sentences—"He could not have made a worse choice."—because the sentence itself is not inherently liberal or conservative without knowing who the "he" is. To a reader, this polarization is clear, but to our sentence classifier model it is not.

Lastly, our models are unable to pick up on sarcasm. While not all too prevalent, op-eds often contain some form of sarcasm. Sentences like "Wow, the president really knocked it out of the ballpark on that policy" are perceived as conservative, while it would be clear to a New York Times reader that this sentence in the context of a liberal op-ed is meant in jest.

7.1 Further Steps

While we explored many different classifiers and worked extensively to tune these classifiers, ultimately we believe our results were significantly limited by the size and quality of our data. We are confident this is the case since when we used our algorithms on other data sets, such as a movie review data set with 50,000 reviews, we got accuracies as high as 89% For this reason, if we were to do further work on this project, we would definitely want to get much more data and be much more selective about choosing which articles we want to use to get this data. For example, if we had a

data set similar to Budak, Goel, and Rao's we believe we would have much higher classification accuracy and be able to draw more confident conclusions from our results.

Another extension to the project which we believe would be particularly useful is to examine how our classifications differ across different news categories and topics. For example, opinions articles are much more strongly worded than general news articles and thus are more likely to contain biased views. While newspapers might have very similar political leanings on average, they might have very different leanings on particular subject matter or news category.

The macro-goal behind this project is providing a platform for news readers to gain awareness of the biases of the articles they read and to expand their horizons to other sides of the political spectrum. To this end, we ultimately would like to create an app that allows users to track the political biases of the articles they read, encourages them to set profile goals for the political distribution of the news that they consume, and recommends articles from all parts of the political spectrum to achieve these goals.

A System Description

Check README.md in github for system description and directions.

B Group Makeup

B.1 Daniel

Worked on data cleaning and wrangling for sentence and GloVe data. Implemented and scored all articles for Naive Bayes and Random Forest.

B.2 Dylan

Worked on reading the background and related work, scraping and processing all the news articles, designing the poster, helping Daniel with the Naive Bayes and Random Forest Classifiers, and writing the paper.

B.3 Spencer

Implement all versions of the Recurrent Neural Network. Wrote a randomized grid search to effectively tune the many hyperparameters simultaneously. Wrote a baseline Naive Bayes algorithm from scratch.

References

[1] Bergstra, James and Bengio, Yoshua. Random Serch for Hyper-Parameter Optimization. *Journal of Machine Learning Research* (2012).

[2] Budak, Goel, and Rao. Fair and Balanced? Quantifying Media Bias Through Crowdsourced Content Analysis. *Public Opinion Quarterly* (2016).

[3] Iyyer, Enns, Boyd-Graber, and Resnik. Political Ideology Detection Using Recursive Neural Networks. *Association for Computational Linguistics* (2014).

[4] Sim, Acree, Gross, Smith. Measuring Ideological Proportions in Political Speeches. *Empirical Methods in Natural Language Processing* (2013).

[5] Yin, Kann, Yu, Schutze. Comparative Study of CNN and RNN for Natural Language Processing.