Predicting Political Bias in Media
CS224U Final Project

Andrew Quirk, Cody Hankins
Stanford University, Department of Computer Science
{aquirk,chankins}@stanford.edu

Abstract

We will be investigating techniques related to sentiment analysis bias extraction of news articles related to American politics. The goal is to provide an easy and reasonable metric for bias, in order to provide some context for the average reader to consider when consuming media. Many political science academics are concerned that the aggressive partisanship in American politics creates a poor environment for cooperation and progress, and our goal is to develop a methodology for improving transparency.

1 Introduction

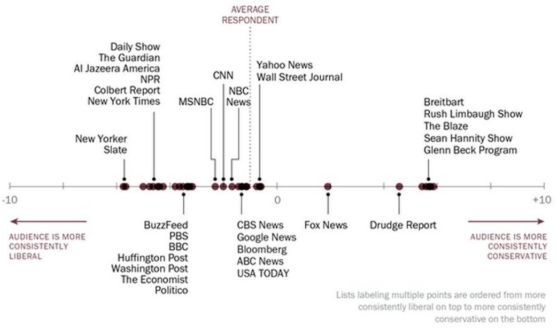
Political polarization is one of the largest problems in America today, and a large part of this issue is due to the news that Americans consume and their perceptions of it. Currently, there are some sites that analyze media bias by looking at the political affiliations of their viewers. This metric is insufficient, it relies on respondents' self-reported liberal or conservative ideology. There needs to be a more rigorous, quantitative method for rating and analyzing news articles that results in an informative metric of political slant. The end goal is to educate the user: informing them, from a non-biased model, about the content they are consuming and if it is skewed toward a more liberal or more conservative point of view.

We were interested in building a political affiliation classifier that would be able to inform regular people about what kind of material they are consuming, using deep learning NLP techniques that analyze the words of the article itself. Our initial model's goal was to inform viewers by extracting information from news articles about the bills coming through Congress and then analyze the biases of the bill. However, we found that in reality, most people don't know about bills in Congress and are not as interested in debating the nuances of policy as they are interested in stances on issues that affect them directly. Even though awareness of legislation is important, we believed that there was a more accessible metric that we could procure to provide meaningful information to constituents.

While building the political sentiment classifier of this initial model, we realized that a very useful product would be a model that takes as an input a regular news article, and outputs a bias analysis of that article. If our goal is to inform citizens of the materials they are consuming, this is a much more effective tool. Thus, we decided to focus on the main classifier and not on the sort of article summary that goes with content extraction. Our primary focus is assigning a meaningful metric of overall political bias in an average-sized news article.

This sort of knowledge could be useful in classifying data that does not come from a well-known source of generally uniform bias: for example, chat forums or blogs where content creators do not have a well-known political affiliation. A binary political classifier could generalize to new issues.

Previous attempts relied heavily on article source for determining the slant. For example, a news source like Breitbart is generally considered to be very far right, and any content from that source is assumed to have a medium to high amount of bias. We found a comprehensive mapping of the generally accepted political affiliation of news articles, included here for reference.



(Mitchell et al., 2014)

We used this chart as a general guiding heuristic for doing the calibration of our future model.

2 Prior Literature

There have been numerous explorations into extracting political affiliation through text, with varying degrees of success.

One paper that initially sparked our interest was Automatic Detection of Political Opinions in Tweets (Maynard and Funk, 2012). Maynard et.al. explored advanced techniques from inferring political slant with tiny amounts of text. The two main components built into their model were content and time frame. Content of the tweet itself was most important, and the component from which we drew the most inspiration. The model classified each token as an issue word, a position word, or an emoji (a surprisingly powerful and expressive character). By selecting a hundred prolific political twitter figures, and extrapolating out to followers, they were able to build a large corpus of training data for a political classification problem.

At a high level, our system performs as such. Initially, many standard preprocessing techniques are applied to the data in order to standardize the document. This includes tokenization, part of speech tagging, sentence splitting, and morphological analysis. We employed a number of these techniques in our own models. After, they employed a library which provides an automatic method for recognizing names and the type of entity that it represents. This allows for isolation of tweets based on issue or speaker, which can be useful for testing afterward. The final model itself relies on three key components

1. Affect annotations: detecting positive, negative, and neutral words

2. Issue: identifying factual or opinionated versus questions or doubtful statements

3. Context: detecting extra-linguistic clues such as smileys

From this work, we derived a few key techniques. The most important was the isolation of powerful indicator words, that were traditionally politicized and carried connotation or bias. This is accomplished in the selection of our word vector representation. The other takeaway was the volume of data, which we found to be more effective than a smaller corpus, a fairly obvious statement.

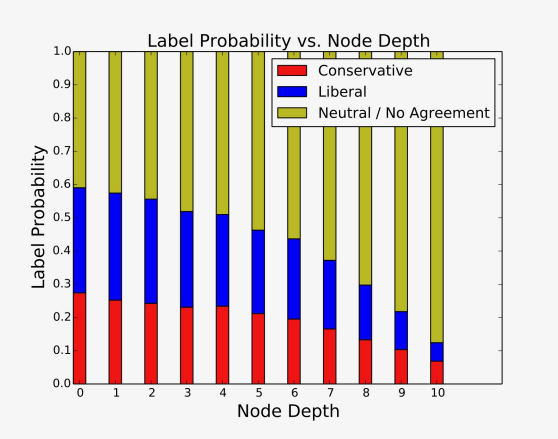
The paper we found most inspiring, though, was Political Ideology Detection using RNNs (Iyyer et al., 2014) . This research was most focused on capturing critical underlying patterns that escaped most simplistic approaches. Nuances of sentence structure and word choice tend to be more important when using complex linguistics in a political setting. The other question explored was how the model would respond to a truly politically neutral text, which would be important to see if the results from classification were particularly meaningful.

The research showed a great deal of promise: the neural net was able to capture some semblance of higher-order representation of political bias. To do so, they built a model that relied on detection at the sentence level, defined as *ideological bias* if the speaker's political affiliation is apparent from the context of any given sentence. To build data appropriate to their ends, then, they created a new corpus of training data based on sentences labeled at the sentence level, which allowed an impressive amount of granularity. This is one component which we wish were able to implement, but did not have the resources to explore.

What we did take away from their work, however, was the ways by which they captured the context of each unit. They proceeded to generate word vectors for each token in the training data, and combined them at a sentence level to capture some of the context of how they interacted. This idea, we found very powerful, and showed positive results in their analysis.

Using this setup, they then trained their model on the same corpus we did, coincidentally. One important note that they included (which we also address later) included the potential ambiguity of conflating party affiliation with political slant. However, in American politics, the stark partisanship is strong enough to justify equating party and ideological affiliation in most cases of classification.

The diagram below shows the success of their neural net as they added depth, and inspired us to create models with high node depths. In essence, the researchers ran experiments to see how high-dimensional they would be able to get the model to still capture meaning, something that seemed to taper over time. This is apparent as more neutral ratings come with more layers, as the slant is lost over larger sections of the document.



Using this model, they were able to attain up to 70% on sentence-level analysis. In their analysis, they attributed these scores to their RNN model being able to detect bias polarity switches at a higher level of the parse tree. Qualitatively, their model displayed an understanding of word associations. Conservatives were likened with synonyms for freedom and religion, while liberals focused on ideas surrounding equality and supporting the working class. It was this implicit understanding which we hoped to emulate in our model.

3 Data

The corpus we used was transcripts of House Congressional speeches from the years 1993-2012. Every recorded word from each session of Congress (each of which last two years) was recorded and labeled with the time and the speaker, and gave us quite a large dataset, composed of multiple gigabytes of pure text. Having the speaker of each line turned out to be critical, as well, as it allowed us to differentiate which speeches were associated with which party. This data, which was provided as a corpus for the class, was aggregated by scraping all of the congressional speeches on CSPAN. Twenty years of tagged and labeled JSON data on all congressional speeches is a very rich dataset that allowed us to do effective supervised learning.

To organize our data efficiently, we parsed our corpus into categories by political affiliation and session. That way, we are able to select any of the 103^{rd} to the 112^{th} congress sessions, and the speeches of each party from that period. This also allowed for some interesting analyses of the differences between the same parties over the changes in two decades, something that we will address later on. Once this data was loaded in from the JSON files, it underwent a number of steps in order to standardize the data. Stop words and neutral words are removed, in order to reduce the amount of noise in our predictions.

We were interested to see, then, how this could generalize to modern-day political rhetoric. To this end, we set up a web crawler that automatically downloaded the most recent articles from a number of popular websites, including cnn.com, foxnews.com, economist.com, newyorker.com, and more (see our data for the full list). This incredibly useful library (newspaper3k) enabled easy automated tests. Every run, the scraper would download a large number of most recent articles from each web url we provided. We process the raw text from these articles in the same manner, so that we have our testing data in the same general form as our training data (tokenizing, removing stop words, etc.). At this point, we have on the order of 500 articles in our test set. The political affiliation of the test articles relies on their source: for example, the articles downloaded from breitbart.com had an 'R' appended to them. We relied on the bias chart to gauge approximately where each source appeared on the political spectrum. Now that we had both a training and testing set, we were able to evaluate our model and see how accurately it captured the nuances in text for predicting future affiliations.

One interesting component of our data that needed extra attention was non-determinism. The scraper automatically finds the most recent content from each website, which meant that subsequent runs of our classifier on real data results in different overall outcomes aside from randomness in our model. This presented both a challenge an an opportunity. Maintaining consistency when going through different iterations of our model was difficult. However, every time we ran our most recent version, we were confident that the data would generalize fairly well to unseen testing data, because every time we were actually getting unseen testing data. We will explain further how this affected our results later in the paper.

An aside: another avenue we had explored was the efficacy of using the scraper to generate training data, as well as retrieve testing data. However, there was no way to aggregate the sheer volume of text that would be needed to accurately procure our word embeddings from news articles. Our attempts to do so resulted in a paltry training set that did not generalize well in the testing phase. We therefore opted to focus on using the gigabytes of Congressional speeches made available to us through the class.

As stated before, the corpus was labeled with POS for all tokens in the corpus, but we want to priori-

tize speed of testing the model and wanted to avoid labeling the testing set. Therefore, we worked on composing a model which would be effective regardless of whether the testing input had been labeled or not. This proved to be very effective, but a further direction would be testing on POS labeled data.

4 Model

We approached the problem from a number of perspectives, searching for the most effective way to encapsulate the higher-order information we were seeking. We will discuss our approaches, both effective and not, and emphasize our most sound approach of using aggregate word embeddings to predict political affiliation. We used four different models which are outlined below.

4.1 Vector Similarity

An effective metric between the similarity of two vectors of words is cosine similarity, as well as Jaccard similarity.

Cosine Similarity:

$$\cos\varphi = \frac{[\overrightarrow{CA'}, \overrightarrow{CB}] \cdot [\overrightarrow{CA'}, \overrightarrow{CD}]}{\left|[\overrightarrow{CA'}, \overrightarrow{CB}]\right| \cdot \left|[\overrightarrow{CA'}, \overrightarrow{CD}]\right|}.$$

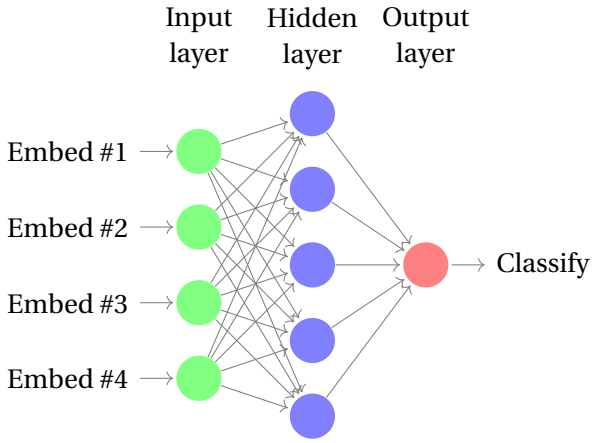
Jaccard Similarity;

$$Jaccard(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

These naive metrics yielded some very interesting results. The vectors between speeches from the same year were extremely similar (97% cosine similarity), which is logical because much of the diction in political speeches is repeated (especially in Congress where a large proportion of the text is recited daily such as the pledge of allegiance and business formalities) and this noise needed to be removed. After finding this, the data was cleaned and normalized through standard NLP practices (removal of stopwords, capitalization, punctuation), but the results only improved by 4%. The difference in cosine similarity between years, however, was very large (79%) which represented the difference in rhetoric over time. To accomplish the goal of classifying a test set from a more modern setting, we moved to a more complex model to represent the data.

4.2 CNN/RNN Classifiers

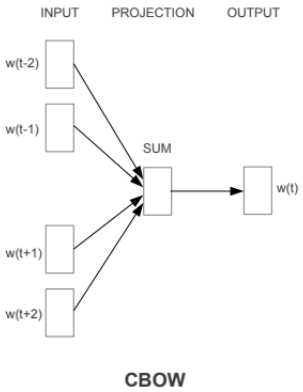
One of the main issues with the vector similarities of this corpus was the similarity in terms of the words: for any particular issue, it was highly likely that both parties would be using a highly correlated vocabulary when debating about issues. Therefore, we captured the higher-order relationships of the words using neural networks. We trained our model on the our Congressional speech corpus and built a myriad of standard deep learning models with varying hyperparameters. Our best models were only able to achieve 59% F-1 scores, though, and this led us to further explorations into areas of capturing these high-order relationships.



4.3 Word Embeddings

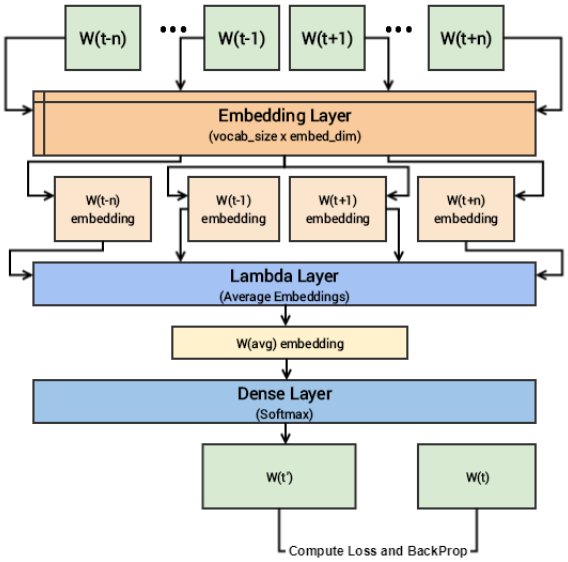
Traditional models fall short in that they are unable to represent the sequence, semantics, and structure of the words when they simplify a document to only a word count. Our first approach used a simple bag-of-words approach to see what sort of accuracy we could get from simple word-level representations of the text. While fairly trivial to implement, we were able to see reasonable accuracy that was meaningfully above random, but did not capture the level of intricacy that we had hoped to build into our model.

We built two different word embeddings, the Continuous Bag of Words Model (CBOW) and Skip-gram embeddings. The first, CBOW, relies on the context of a word for prediction. It uses a window of words around the target word as represented below:

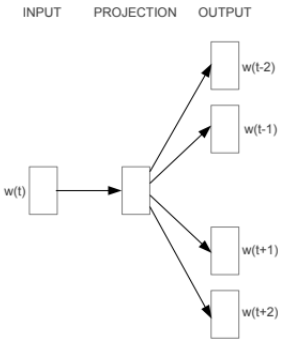


CBOW

To create the embeddings, we then ran our corpus of speeches through an embedding, lambda, and dense layer to arrive at our final embeddings.

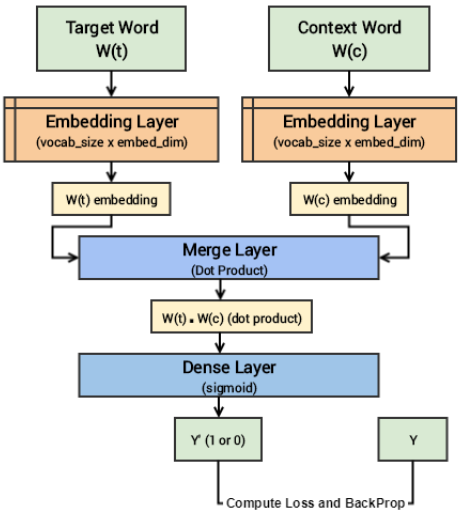


Next, we built word embeddings with a skip-gram model that worked in a similar way, but relied on a pair of words and their relevance. To create this, we used the FastText library, released in 2016 by Facebook. This provided a quick and clean way to generate accurate word embeddings without too much computing power. The model uses an unsupervised learning algorithm to obtain the word representations of the words in our corpus.



Skip-gram

Similarly, we then ran our corpus of speeches through an embedding layers, merge, and a dense layer to arrive at our final embeddings with the other two types of embeddings, and were interested in finding the relevance of specific words, given that most words hold a minimal political bias. Using this model would capture the weights to indicate whether our assumptions about neutrality are accurate, and statistically meaningful.



It turned out the CBOW model was the most effective for our purposes, so we used those weights to build a classifier for other Congressional speeches as well as news articles.

Our goal was to capture the underlying syntactical and lexical indicators that underly the sort of political narratives that were being exchanged. Our model is based on the assumption that parties tend to align and their styles are relatively similar. Being able to encapsulate this into our neural net. We knew that the word embeddings themselves captured a great deal of information, and used traditional methods of summing, averaging, and multiplying the vectors created by the embeddings to generate a classifier. We chose to use these naive methods because they were more effective on this specific dataset, but if these methods were to be expanded upon we would advise using CNNs in the classification phase for higher accuracy.

5 Results

We explored a large search space of effective models, with varying degrees of success. What we found most compelling was the word embeddings generated by FastText, which provided robust and well-developed representations of the sort of meaning we were hoping to capture. As stated before, we found more success in the specific issue we were trying to tackle in some of the more accessible, elegant approaches to this problem.

One of the first metrics we looked at to analyze the success of our embeddings was the most similar words to politicized words (hand-picked as a sample). When we did this analysis, we found that the sample represented politicized topics of the day (2012) very well and

also included more general political information.

Associations		
Word	Republican	Democrat
Immigrants	Lowskilled, wages, dropouts	Nationals, apprehended, syndicates
Vote	Votes, voting, favor	Voting, defeat, votes
Israeli	Arafat, hamas, palestinians	Palestinian, hamas, palestinians
Liberal	Elites, conservatives, philosophy	Column, conservative, columnist
Refugee	Ethiopia, Kenya, camps	Ethiopia, displaced, camps
Corruption	Disclosed, governance, indentured	Coverups, balloons, abuses
Abortion	Abortions, incest, prolife	Abortions, reproductive, prolife
Government	Intruding, grating, bureaucrats	Unfortunately, fundsdollars, individual

Some of the most interesting results from this part of the results are the words associated with refugees: in 2011 there was an enormous drought in East Africa, resulting in a refugee crisis that led to the closest associations of 'refugee' being 'Ethiopia' and 'Kenya'. If we were to run this model on Congressional speeches from today, we would probably get very different associations for that word.

Other words like 'Immigrants', 'Government', and 'Abortion' reflect generalized political feelings more accurately and are more generalized to today. For 'Immigrants', it is not extremely surprising that Republicans used 'lowskilled' more, and Democrats used 'Nationals'. Similarly for 'Government', it is logical that Republicans used words like 'intruding' and 'bureaucrats', while on 'Abortion' Democrats associated 'reproductive' (as in reproductive rights). 'Rights' did not have as high of a score, since it is used in more varied contexts.

We also looked at neutral words like 'vote', that had relatively unbiased responses. Confirming that these words did not have a meaningful slant score assigned to them was a reassuring occurrence, as we felt more confident that our model was not extracting a sense of bias in a scenario where there was not any. You can see those representations to the norm difference chart, later on in the analysis section.

Classifications		
Category	Predict Dem	Predict Rep
USA Today, Neutral	Sports, Tech, Games, Life	Nation, Money, Weather, Games
Fox News, Right	Food	Politics, US, Entertainment, Great Outdoors, Radio, Business
Mother Jones, Left	Politics, Environment, World, Crime-Justice, Food	Media

One area that we were very excited to explore was the difference in types of news and how they were classified. USA Today, one of the most consistently neutral-rated sources, had a logical split of classifications. Articles about the US and Money (traditionally topics more associated with conservative/Republican viewpoints) received that classification, while issues like Tech and Life were more likely to receive a 'Democrat' label.

The fact that topics like 'Food' stayed on the 'Dem' side of the split for both news sources is very interesting: the embeddings associated with those words are more consistently similar to Democratic speeches than Republican ones. Likewise, the labels around topics like the Outdoors were associated with Republican affiliation. This sort of behavior was observed over a range of news sites, showing how subject-specific topics can have a specific slant that the model captures. Therefore, the analyses we do should be concerned specifically with a sort of "neutral topic" like politics, where each party is represented uniquely.

Another interesting takeaway from the embeddings was the most politicized words from the embeddings. This metric involves the greatest of the absolute value between the norm of the word embedding representations of the Republican and Democrat trained models. This can be expressed as

$$d = |\|w_d\| - |w_r\||$$

We know, because of how the word embeddings were generated, that the vectors are the same dimension, so different overall sizes should not be an issue. We wanted some sort of absolute metric to

see which issues pointed in the most opposite direction along party lines.

Word	Norm Difference
Wall	166.34
Street	133.35
Wars	132.35
Aisle	128.15
Cybersecurity	125.33
Texas	118.34
Mexico	112.97
Afghanistan	111.39
...	...
Robotics	.012
Imagery	.0102
Poetry	.0008
December	.0002

This part of the model validated the accuracy of our word embeddings, as many of these words are very logically biased topic of conversation in the speeches. The first two words 'Wall' and 'Street' also show some of the short-comings of a unigram model, and some phrases like their result, "Wall Street", need to be inferred from the results. However, we found from prior research in this area that unigrams are the most accurate way to represent these word embeddings and we found similar results with our model.

The fact that words like 'Texas', 'Mexico', and 'cybersecurity' were very politicized in 2012 is logical, and words like 'Poetry' and 'December' had very low scores because of their lack of political connotation makes sense.

6 Analysis

One of the largest issues with trying to analyze bias in news is that there is no gold standard for which articles are actually biased. When we were looking at the success of the classifier in our, we used the heuristic of the audience of these sources. While this is very accurate for far-right and far-left sources, it is less so for moderate sources and it is hard to tell from a day of news (that we are testing our model on) which ones are actually biased and which ones are not. For Congressional speeches, we used which party they were labeled with.

However, we do know that our model was completely accurate on news sources. Given that we had ten distinct sessions of data to compare against, we found our model, trained on the most recent session of Congress, to be effective in predicting some of the other sessions of congress.

Self-Prediction Scores		
Years	Correct D	Correct R
2009-2011	0.758	0.762
2007-2009	0.742	0.765
2005-2007	0.761	0.775
2003-2005	0.727	0.751
2001-2003	0.727	0.751
1999-2001	0.727	0.719
1997-1999	0.743	0.725
1995-1997	0.731	0.710
1993-1995	0.705	0.714
Average	0.742	0.746

One interesting thing to note about these predictions is the gradual decline in accuracy, which makes sense. Because the training was done for session 112 of Congress, as time gets further from that initial training point, there will be a decline in similarity between the data of the training and testing corpus.

Overall, however, we were happy with the meaning which our word embeddings were able to capture. From the relationships between polarizing words, we are confident that the system was able to extract which of the politically telling words were closely related to each other. Moreover, we also found it compelling how it was able to properly select words that would have a great deal of controversy surrounding the issue, and isolate those terms accordingly. Likewise, it was able to capture which issues were not very politically polarizing (poetry) which also manifested itself in the testing phase. Testing articles with a small or zero degree of bias displayed, overall, a much more centered classification.

One notable concern that we encountered when working with this corpus was generalizing to modern data: it became apparent, at many steps of the process, that the corpus did not align very well with some of the more modern political rhetoric found in media today. The drop in performance we observed, then, was most likely a result of our testing data addressing completely different issues, which confused the model. One way to ameliorate this issue would be to align the training and testing data to a stronger degree, an interesting avenue for future work to explore. However, in our assessment, this was one of the bigger concerns to provide context to our results. If we had a stronger corpus of labeled news articles, I think we would be able to apply this technique to most modern news articles.

A result we found surprising was the level of specificity that certain issues could have on the

rhetoric surrounding a particular issue. For example, take the words most correlated with abortion from the list of associated words. Of the tens of thousands of words in our corpus, the model extracted the term *hoyergreenwood*, a tokenized form of the Hoyer-Greenwood proposal of 2003. This specific movement was composed by Steny Hoyer, Democratic Whip of the Senate at the time, and was a corollary to Roe v. Wade. These sorts of details, we believe, are what make the returned data both unique and fragile. Our model is able to capture a strong degree of delicacy in working with complex topics and movements. However, we also know that this could present troubles in generalization, as a specific proposal from seatings of Congress a few sessions ago does not provide a great deal of meaningful information for the future. We explore ideas related to generalization in our conclusion of the paper, where we outline potential avenues for improving the performance of future models.

One of the other areas we looked at was how the model performed on articles that contained the title "Trump" or "politics". We expected the results to be extremely slanted, because of the obvious political nature of these articles. The result was that these articles followed the supposed "slant" of the news source to a high degree. For example, Fox News articles that contained the word "Trump" in the title had a much more 'Republican' score. In general, extracting if a news article is "about politics" turned out to be an interesting task in itself and was beyond the scope of this project so we simply checked the url for the indicators mentioned above.

7 Conclusion

Capturing the intricacies of bias in political text is a formidable task. It is one that has to resolve a number of ambiguities, and does not have a clear-cut resolution. However, we are confident that we were able to capture some of the underlying features of political bias in our model.

There is still a great amount of work to be done. This field is relatively untouched, and for a solid reason: programmatically capturing these sorts of nuances in political speech is an extremely difficult problem to tackle, with accuracies maxing out at about 70% for some of the most robust models. This means this sort of system deployed in practice will have some difficulties in providing an objective and accurate metric to the user.

Areas we find especially compelling to explore include training on news sources that have been hand-labeled. Right now, the linguistics and the speaker's party affiliation are the only two substantiative features captured in our model. There is a huge amount of rich data available: speaker's background, neutral rhetoric around issues, and temporal context can all provide another layer of understanding.

Another possible avenue to explore is to harness data sets that do not exist uniformly in practice. One prime example is talk shows. On the major networks, there are notable speakers (Anderson Cooper, Bill O'Reilly) who are some of the most influential partisan figures in modern media today. Capturing their work and incorporating it into the model has potential for capturing a great deal of clues that do not exist in the potentially hyperprofessional environment of a Congress floor meeting. These sorts of settings do not have the raw, day-to-day rhetoric around these polarizing issues in American politics. Diversifying training data, then, would be another particularly compelling avenue to explore when building a stronger model.

The end goal for this sort of system would consist of two things: one, there would be a form of automation that was able to procure modern testing data as new issues came to the forefront of the political scene. This is one component of our data that we found potentially troublesome (especially since Twitter and other social media data has become more difficult to procure). The issues at the forefront of Congressional debate over the last decade are rapidly changing, and our model (which incorporates issue-specific words) can have troubles generalizing to understand discussions in the modern day.

Recall our primary impetus for exploring this issue: increasing civic engagement through a informed citizenry. Having a sort of wrapper that made this model available in modern web applications could be a strongly empowering tool. Imagine a browser extension which automatically tested the text content of a webpage, detected to be concerned with political issues, and output a prediction of bias. This could provide a meaningful metric for a large subsection of the population, who could now examine new material with a more meaningful context. This knowledge, hopefully, would represent a reduction of the echo-chamber mentality that web-distributed news has made readily available to the population.

In conclusion, we believe that politics is a rich space for NLP to show some dramatic improvements, with the potential to impact the workings of the American political system in a positive way.

References

Mitchell, A.; Gottfried, J.; Kiley, J.; Matsa, K. E. Political Polarization and Media Habits. 2014.

Maynard, D.; Funk, A. Automatic Detection of Political Opinions in Tweets. Proceedings of the 8th International Conference on The Semantic Web. Berlin, Heidelberg, 2012; pp 88–99.

Iyyer, M.; Enns, P.; Boyd-Graber, J.; Resnik, P. Political Ideology Detection Using Recursive Neural Networks. Association for Computational Linguistics. 2014.