Detecting Political Bias in Speeches and News Articles Final Report

Sachi Angle
sva22
Cornell Tech
New York, NY, 10044
`sva22@cornell.edu`

Varun Ganesh
vg274
Cornell Tech
New York, NY, 10044
`vg274@cornell.edu`

Pargol Gheissari
pg463
Cornell Tech
New York, NY, 10044
`pg463@cornell.edu`

Rina Schiller
rs2536
Cornell Tech
New York, NY, 10044
`rs2536@cornell.edu`

Nicolas Sempere
nls92
Cornell Tech
New York, NY, 10044
`nls92@cornell.edu`

Abstract

The political ecosystem in the United States is dynamic and complex. There are two major political parties in the United States and much of the political discourse in the country can be classified as either leaning conservative or liberal. This paper details the approach we have taken towards identifying political bias that can be inferred from text. We refer to existing research done in the area and aim to replicate two previous approaches for this classification problem: a Recursive Neural Network and Long short-term memory architecture. We then assess the transferability of the models to similar datasets. We detail the datasets used for this purpose and the metrics for evaluating the results achieved.

1 Introduction

The rise of social media over the last decade has led to a dramatic increase in sharing information online. Users actively engage in political discussions on social media, where conversations are typically fueled by a plethora of public debates, interviews and speeches. This information is captured in both structured and unstructured form at an unprecedented scale.

In the US, there are two predominant ideologies in the political atmosphere: liberal and conservative [7]. The two ideologies have opinions on different issues and are obviously biased towards them. However, identifying ideological bias in political texts, such as news, social media and journals, is difficult. In the age of big data, it becomes impractical, expensive and very challenging to manually decipher this public information. Furthermore, this bias can be localized within a small section of the document which can make it difficult to detect. Thus, for a human to be able to detect this, knowledge of the field along with annotator's ability to pick up on subtle elements of language use is required.

Most existing work on bias and ideology detection have focused on "bag-of-words", which ignores the linguistic context. However, recently with the success of deep neural networks in the field of natural language processing, there has been a growth in applying state-of-the-art models for opinion detection and sentiment analysis. For instance, recursive neural networks have been applied to parsing, sentence level sentiment analysis and paraphrase detection [8].

In this paper, we will explore classifying articles into conservative or liberal classes using recursive neural networks and long short term memory networks.

2 Related Work

In previous work, Horne et. al used political ideology as a feature in fake news detection. Furthermore, they classified whether an article is biased or unbiased. They developed a credibility search toolkit consisting of several modules, each serving a different function. The first module predicts the reliability of the user-selected news article. The second module is responsible for bias and subjectivity prediction. It consists of two independent classifiers: a random forest classifier and a Naive Bayes classifier. The random forest classifier is used for predicting hyper-partisan articles. The Naive Bayes classifier is more generic and focuses on sentence-level objectivity. The third module is built to predict which online groups are interested in a certain article and the final module analyzes the news at a source-level granularity [5].

Similar to Horne et al., Potthast et al. classified the bias in a target article as left, right or mainstream, and as hyper-partisan or mainstream. They reported a comparative style analysis of extremely one-sided news and fake news. They mainly explored classification models such as topic-based bag of words and Naive Bayes [9]. This type of bias classification at article level was also explored by Kulkarni et al. They modeled both the textual and the URL contents of the target article and used a Bayesian approach with stochastic attention units to effectively model textual cues. In order to model ambiguity and avoid overfitting, they constructed a model with three main components. First, the discriminator, denoting a probability distribution, given a hidden representation was modeled using a feed-forward network with a linear layer, followed by a ReLU for non-linearity followed by a linear layer and a final softmax layer. Next, they parameterized the latent distribution using a "multi-view" network which incorporates hidden representations learned from multiple modalities into a joint representation. Finally, they modeled the content of an article, using a hierarchical approach with attention. In particular, they computed attention at both words and sentences levels [2].

Moreover, researchers have targeted bias at different levels. For instance, Iyyer et al. have focused at phrase and sentence level, using a recursive neural network (RNN) framework to the task of identifying the political position evinced by a sentence [7]. Sim et al. measured political candidates' ideological positioning from their speeches by applying a domain-informed Bayesian HMM to infer the proportions of ideologies each candidate uses in each campaign [10]. Gerrish et al. focused on linking legislative sentiment to legislative text. They explored several models, such as logistic regression with random effects, text regression and supervised Latent Dirichlet Allocation, that connect the voting patterns of legislators to the original bill texts [4].

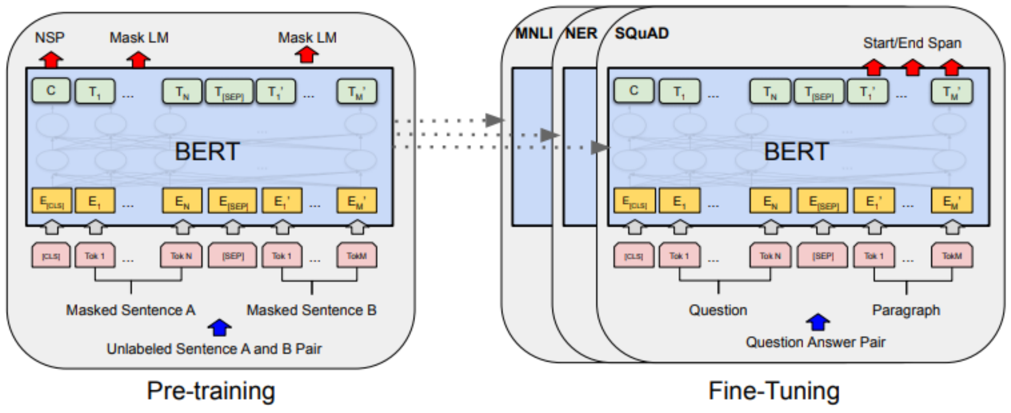

Figure 1: Pre-training and fine-tuning processes in BERT have similar architectures [3]

The natural language modeling tasks mentioned above can be improved by language model pre-trainning. Natural language inference and paraphrasing, which aim to predict the relationships between sentences by conducting a holistic analysis, as well as token-level tasks such as question answering, where models are required to produce fine-grained output at the token level, are examples of these tasks. The Bidirectional Encoder Representations from Transformers (BERT), a new language representation model developed by researchers at Google AI Language, has been able to achieve a deeper sense of language context and flow. This conceptually simple yet empirically powerful model, is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. Thus, this model can be fine-tuned by just adding one additional output layer; which can consequently create state-of-the art models for a wide range of tasks, such as question answering and language inference, without substantial task specific architecture modifications. In this model, excluding the output layer, the architectures are the same in both pre-training and fine-tuning[3]. The procedures of fine-tuning and pre-training can be seen in figure 1 above.

Our work focuses primarily on detecting bias in both political speeches and news articles using a recursive neural network. In the following sections, we will discuss our methods and provide details on the datasets used.

3 Dataset

As mentioned, the rise of social media over the past few years has lead to a dramatic increase in the data sets containing public debates and discussions in this space. Consequently, many models focusing on detecting bias and ideology take advantage of such data sets. In this project, we used two different data sets: the Convote data set and the data set generated by Budak et al.

As our model is inspired by the work of Iyyer et al, we chose to use the same data set for benchmark comparison. This data set is the Convote data set[11]. Convote is a set of speech segments from congressional debates circa 2005. It was sourced from raw transcripts from the House of Representatives, encoded as a series of HTML documents at `govtrack.us`. These transcripts, which represent contiguous debates regarding a bill and feature multiple speakers, are broken apart into segments at the speaker level. Each segment is stored within text document that is labelled according to the bill being debated, the speaker, that speaker's political party, and information relevant to the segment's location in the original transcript.

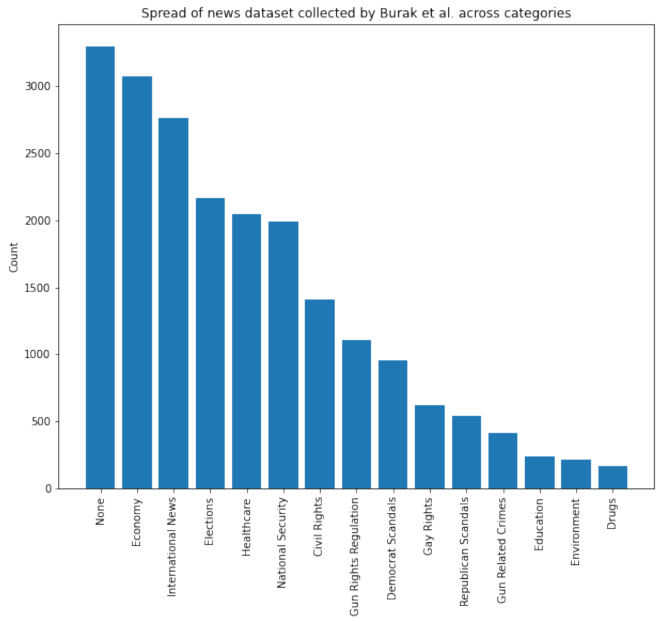

Figure 2: Spread of articles used in the dataset of Burak et al. across categories

The second data set that we used is the news article data set developed by Budak et al [1]. Their dataset consists of news articles from 15 popular news outlets and political blogs published online in 2013. From an original set of over 800,000 articles, they distilled 115,000 articles deemed as being political in nature. Their labels were created through crowdsourcing efforts, such as using workers from Amazon Mechanical Turk. Each article was given two labels, corresponding to the degree to which the workers thought the article was positive, neutral, or negative toward the Republican and Democratic parties, respectively. The distribution of articles across different topics and different

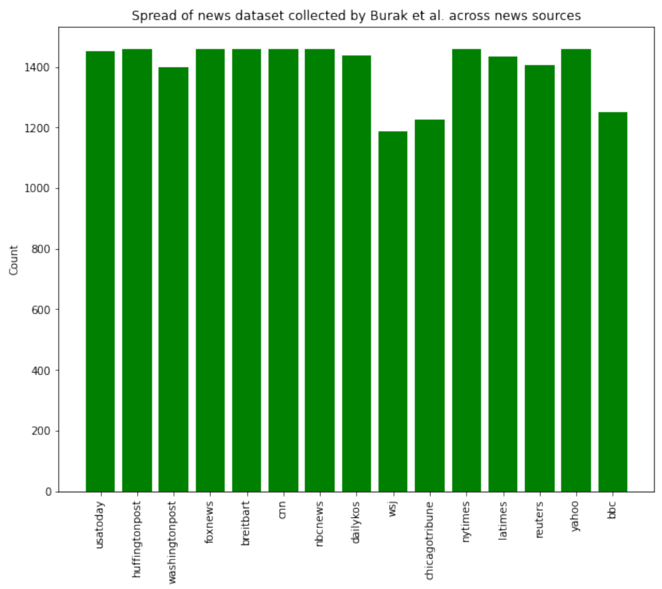

Figure 3: Sources of articles used in the dataset of Burak et al.

news sources can be seen in figures 2 and 3. As seen, there is an even spread of news articles across different sources; however, the distribution of articles across topics shows that most articles are either not grouped in a specific category or belong to the topics of economy and international news.

4 Methods

4.1 Pre-Processing

In order to apply our model to these datasets, we first needed to preprocess the data such that each document, which contains segments of a speech, is represented as a numerical feature vector. This process consists of retrieving the articles and their labels, removing stop words, and performing lemmatization and tokenization.

In order to preprocess the Convote dataset, we first downloaded the set of speech segments. The dataset does not have associated labels; rather, each speech file has a specific file-naming convention that includes the party associated with the speech along with other data. We wrote a script to save only the text of the speech with the political party as the label.

On the other hand, the dataset collected by Burak et al. contained URLs, labels for topics and labels for votes. The votes correspond to the ideological leaning of the article for each party and consisted of any of these values: Positive, Somewhat Positive, Neutral, Somewhat Negative, Negative. These were encoded on a scale ranging from positive to negative as –1, –0.5, 0, 0.5, 1 for the Democratic party. Analogously, for the Republican Party, the same scale is encoded as 1, 0.5, 0, –.0.5, –1. The final score is calculated as the average over the two values. For example, an average score of –1 indicates that the article is very positive toward Democrats. The final label is obtained by applying a threshold at 0.

Moreover, as the second data set contained only the URLs and not the actual news articles, we built a scraper that visits the webpages and retrieves the text content. A Python library called BeautifulSoup was leveraged for this purpose. The text content was targeted using HTML tags. While the generic paragraph tags worked to retrieve content for most websites, some needed custom inspection to identify the classes of tags that contained the relevant content. This was done through a manual process.

After retrieving the contents and the labels, we used the BERT pre-trained model to produce the feature vectors. This process consisted of four main steps: removing stop words, lemmatization, tokenization and converting to feature vectors. The stop words were defined using the nltk.corpus package. The BERT transformer was originally trained on sentence entities, and its maximum sequence size is 512. These constraints are not directly compatible with entire documents, which contain numerous sentences. To structure our data in a way that allowed us to use BERT effectively, we broke down each speech into an array of sentences and then the stop words were removed. Next, we tokenized the dataset using the pre-trained BERT tokenizer. As BERT was trained using the WordPiectokenization with a 30,000 token vocabulary, the words were broken down into more than one sub-word. The first token of every sequence starts with the [CLS] special token and sentences are first seperated using the [SEP] special token. Under the condition that a word is not present in the vocabulary the [UNK] special token is used. In order to ensure that all our input are of the same size, we padded to the size of 512. After the data is tokenized we create a word-by-word matrix of feature vectors with size of $Sentence\ Length \times 768$, where $Sentence\ Length$ is fixed across all the documents in the dataset. After completing this final step, the documents are ready to be passed through our model.

4.2 Model

We designed our models based on the methods implemented by Iyyer et al. and Misra et al. We explored an RNN model similar to that of Iyyer et al. and a LSTM model similar to Misra et al.

Since RNN models are able make predictions based on what was learnt from the prior input, they can model semantic composition. This is the principle that a phrase's meaning is a combination of the meaning of the words within that phrase. Most languages follow this principle, except for sarcasm and idioms [7]. In the RNN model, each computed state is used to update a hidden state, which influences the next output. Furthermore, in this model, each word in a sentence is associated to a vector representation. These words form phrases; similar to the words, each of these phrases are also associated to a vector of the same size as the vector representation of the words [7]. As seen in figure 4, the phrases can then be recursively used to generate vectors at higher-level nodes and merge into the sentence. The vector representation of these sentences is trained such that the meaning of the entire sentence is retained. As Iyyer et al had data labelled at the phrase level, an RNN was able to effectively learn the political bias in each sentence. Their RNN model predicted an output for each sentence by learning from the words contained in it. In comparison, our approach predicted the bias present in each document by learning from the sentences within. The final output layer will employ a softmax classifier:

$$\hat{y}_{article} = softmax(W_{cat} \cdot x_{article} + b_{bias}) \tag{1}$$

where W_{cat} is a matrix of weights such that $W_{cat} \in \mathbb{R}^{k \times d}$ where k denotes the number of ideological label types.

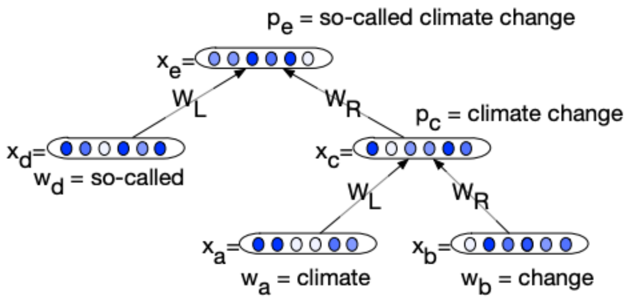

Figure 4: Example RNN for the phrase "socalled climate change" [7]

As phrase level annotations of data are not always available, we also implemented a Long Short Term Memory (LSTM). As shown by Misra et al, Long Short Term Memory (LSTM) Networks are effective in capturing more information with lesser data. These models are able to capture long range correlation between sentences to output a final label for the document. In our model, a LSTM unit is described by a sigmoid activation function and a binary cross entropy loss is calculated at the final time step of the network.

In the following section, we show the results from the RNN and LSTM models and compare their performances on the Convote datatset. To resolve the issue of maximum sequence size in the word embedding step, we explored truncating sentences vs picking certain sentences and its effect on the model. We also explored the effect of depth on the model.

5 Results and Discussion

In this section we describe the results of our experiments classifying ideological leanings as liberal or conservative using the RNN and LSTM models on the Convote dataset and the dataset collected by Burak et al. The Convote dataset was used to compare our performance against literature and the Budak et al. news dataset was used to test the generalizability of our model. Furthermore, we compare our results against the results achieved by Iyyer et al. and Misra et al. Iyyer et al achieved an accuracy score of 0.702 using an RNN approach on the Convote data. Misra et al achieved an F1 Score of 0.718 using an LSTM approach using the Ideological Book Corpus (IBC) dataset [8].

In order to compare our model against literature, we first experimented with our RNN model on the Convote dataset. We assessed several different variations. First, we compared the accuracy of our model by modifying the method of generating word embeddings. We tested both BERT and fastText; with both variations, with a learning rate of 0.001 and over 25 epochs, we achieved an accuracy of 51% and a loss of 4.15. Although the accuracy of our model did not change with either of these methods, the time for training was much faster using fastText. We believe this is due to the fact that the size of the embeddings are different; BERT has a size of 768 while fastText is 300. Next, while keeping the learning rate and number of epochs same as before (0.001 and 25 respectively), we assessed the effect of number of hidden layers on the accuracy of our model: we experimented with 1, 2, 4, 8, 16 and 32 layers. The accuracy and loss were fairly consistent with every condition. We achieved an accuracy of 54% using 2 hidden layers and an accuracy of 45% using 32. Thus, we chose to have 2 hidden layers. Next, we modified the learning rate from 0.001 to 0.005 and assessed the performance over 25 epochs. The accuracy and loss did not change significantly under both conditions, the loss vs number of epochs can be seen in figure 5 below. The learning rate of 0.001 performed slightly better: it resulted in 54% accuracy while the learning rate of 0.005 resulted in 51% accuracy. We also modified the batch size but did not see a significant improvement and the accuracy remained the same. However, the loss varied for different batch sizes, the loss increased with decreasing the batch size. We experimented with batch sizes of 32, 256 and 512. The loss was about 7.5 with batch size of 32 and 5.2 with batch size of 256. We then experimented with the number of epochs being set to 25, 50 and 100. However, the loss remained fairly constant at 4.15 after the 25th epoch. As a final experiment, we wondered if essential information was being discarded by truncating the documents that were longer than $Sentence\ Length$ to that length. We experimented with an approach where we used a pretrianed nltk Vader model [6] to identify the sentiment score of each sentence in a document. We did this as we concluded that the sentences in speeches and articles that were more sentimental than the neutral sentences would contain information relevent to the political bias of the document. This model assigns a positive sentiment score, negative sentiment score and a neutral sentiment score to a sentence. The $Sentence\ Length$ number of sentences in the document that had the largest sums of positive and negative sentiment scores were used as the data input into our political bias classifier. However, even this model achieved an accuracy of 54% and a loss of 4.15.

Therefore, this model achieved best performance over 25 epochs when using fastText method for word embeddings, a learning rate of 0.001, a batch size of 256 and 2 hidden layers.

Next, in order to compare our RNN model against LSTM, we ran LSTM on the Convote dataset and assessed several variations: vanilla LSTM, three-layer LSTM and three-layer bidirectional LSTM. The performance remained fairly similar for all variations of LSTM. The loss and accuracy remianed fairly constant at about 51% and 16 respectively. As it can be seen, the performance of RNN and LSTM were similar on this dataset.

As mentioned, we also evaluated the generalizabilty of the LSTM model by running it on the articles dataset collected by Burak et al. Under similar conditions as before, learning rate of 0.001 and 25 epochs, the lowest loss achieved was 1.9. We experimented with different variations of the parameters,

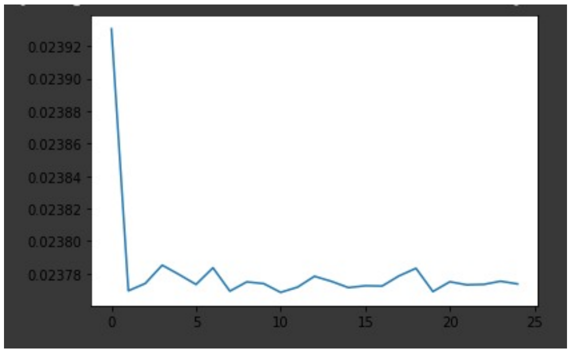

Figure 5: Loss of RNN model over 25 epochs with learning rate of 0.001 and 2 hidden layers

with 1-2 layers, hidden dimensions ranging from 25-100, learning rate ranging from 0.0001 to 0.1 and batch size varying from 128 to 512. All approaches achieved similar results which did not yield an accuracy that was better than random chance. These results indicate that the model is not generalizable to other datasets.

6 Conclusion

In this project, we applied two different models to detect political ideology: RNN and LSTM. The previous work in this domain heavily rely on bag-of-words and are not able to capture the deep understanding of the structure of language. We have shown that our approach detects bias similar to the models implemented by Iyyer et al. and Misra et al; however, as the same level of accuracy was not achieved, it can be concluded that the models developed by them are not perfectly replicable.

Moreoever, we explored two different datasets with different contents. The Convote dataset included speech segments from the congressional debates in 2005 and the dataset developed by Burak et al. consisted of news articles from popular news outlets in 2013. Our results have shown that the RNN and LSTM models perform similarly on the Convote dataset. Furthermore, the evaluation of the LSTM model on the news dataset collected by Burak et al. shows that the LSTM developed by Misra et al. is not generalizable to other datasets.

7 Future Work

There are a few directions in which our work can be expanded. First, as indicated by the results, we can experiment with more sophisticated RNN models to improve the accuracy of classifying bias. Furthermore, we can compare and explore the performance of models at sentence-level versus document-level.

Second, we can consider political ideologies beyond liberal and conservative. For instance, we can investigate a finer-grained dataset that include neutral annotations which could result in showing more subtle distinctions between the two ideologies.

As the models we tried did not sufficiently learn the distinction between liberal and conservative text, we can investigate topic modelling as a part of the feature engineering to assist the model in achieving higher classification accuracy. We can also investigate methods to better represent sentences as vectors, as it is possible that by using sentences to make predictions, we lose out on essential word-level information.

Moreover, we can also explore the use of pre-built sentiment analysis models to identify and remove sentences from each document that are generic or neutral towards both parties. This can help remove noise from our dataset. After removing the neutral sentences, we can proceed to representing each sentence as an average of the word embeddings of each word in the sentence. Removing this noise can help improve the performance of both models.

8 Appendix

The work for this project was split as following:

- Deep Blue Data Scraping - done by Varun, Sachi
- Literature Survey - done by Pargol
- Preprocessing and incorporating BERT into our model - done by Rina, Nick
- Model Experimentation - All

References

[1] Goel-S. Budak, C. and J. Rao. Quantifying news media bias through crowdsourcing and machine learning dataset [data set]. university of michigan - deep blue. Melbourne, Australia, 2019. University of Michigan - Deep Blue.

[2] Sohan De Sarkar, Fan Yang, and Arjun Mukherjee. Attending sentences to detect satirical fake news. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3371–3380, Santa Fe, New Mexico, USA, August 2018. Association for Computational Linguistics.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[4] Sean M. Gerrish and David M. Blei. Predicting legislative roll calls from text. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, page 489–496, Madison, WI, USA, 2011. Omnipress.

[5] Benjamin D. Horne, William Dron, Sara Khedr, and Sibel Adali. Sampling the news producers: A large news and feature data set for the study of the complex media landscape, 2018.

[6] Clayton J Hutto and Eric Gilbert. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*, 2014.

[7] Mohit Iyyer, Peter Enns, Jordan L. Boyd-Graber, and Philip Resnik. Political ideology detection using recursive neural networks. In *ACL*, 2014.

[8] Arkajyoti Misra and Sanjib Basak. Political Bias Analysis. pages 1–8.

[9] Martin Potthast, Johannes Kiesel, Kevin Reinartz, Janek Bevendorff, and Benno Stein. A stylometric inquiry into hyperpartisan and fake news. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 231–240, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[10] Yanchuan Sim, B.D.L. Acree, Justin Gross, and N.A. Smith. Measuring ideological proportions in political speeches. *EMNLP 2013 - 2013 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, pages 91–101, 01 2013.

[11] Matt Thomas, Bo Pang, and Lillian Lee. Get out the vote: Determining support or opposition from Congressional floor-debate transcripts. In *Proceedings of EMNLP*, pages 327–335, Department of Computer Science, Cornell University Ithaca, NY 14853-7501, 2006.