Analyzing the U.S. Federal Legislation Texts: A Deep Learning Approach

Hyoung Ju Seo
Department of Aeronautics and Astronautics
Stanford University
`hjseo@stanford.edu`

Hyeon Seok (Tom) Yu
Department of Political Economy
Stanford GSB
`tom.hs.yu@stanford.edu`

Abstract

This paper analyzes the U.S. federal bill preamble texts from 1973 to 2018 using various embedding and supervised classification methods to gauge the degree of partisanship among bills. In addition to nine different baseline methods from the literature, we develop and implement a CNN-LSTM architecture with a character-based word embedding model. We find that word-based embedding methods outperform character-based ones and that a single-layer LSTM outperforms all other architectures tested. Comparing the prediction accuracy over time reveals a (small) positive correlation with individual legislators' ideological data, suggesting a comparatively lower degree of partisan divide in bill preamble language. Finally, applying the trained LSTM model to a separate political ideology dataset shows a moderate degree of transferability.

1 Introduction

Have public laws enacted by the US government become polarized? While the past decade has seen a plethora of research on the topic of political polarization among mass public and elected officials, not many directly investigate the consequence of such a divide at the policy level. This gap in the literature seems to arise from the difficulty of measuring a degree of partisanship based on large and complex bill contents (e.g., bill texts, authorship, and other contextual information). This paper overcomes such methodological challenges by applying different classification methods previously employed in analyzing political speech data along with various deep learning techniques from the natural language processing ("NLP") literature. To our knowledge, this is the first paper to analyze legislation texts and uncover a trend of partisanship not well explored by political scientists. For the field of NLP, our project will identify an effective neural method for the sentiment analysis (i.e., uncovering the degree of partisanship – how Democratic or Republican a given bill is – in our context) of political text data.

2 Related Work

As aforementioned, a number of social scientists have adopted comparatively simple word embedding and classification methods to measure partisanship among politicians' use of language by analyzing speech data. Gentzkow et al. (2016) [1] and Peterson and Spirling (2018) [8], for instance, respectively analyze the U.S. Congress and the British parliament debate speech data to measure partisanship – defined as "the ease with which an observer could infer a congressperson's party from a single utterance" – and polarization – the predictive accuracy of the machine classifier. Both articles rely on a simple frequency-based bag-of-words embedding at the phrase level method to vectorize texts. This paper employs some of their methods as a baseline and extends these existing works by developing and testing much more sophisticated embedding methods.

Computer scientists have also analyzed political text data using deep learning methods particularly for sentiment analysis tasks. Iyyer et al. (2014) [4] uses recursive neural networks (“RNN") to identify political ideology/position of a given sentence. Their work motivates the inclusion of a Long Short-Term Memory (“LSTM") layer in our main architecture. In addition, we employ their political ideology sentence dataset to conduct a transfer learning experiment. Vosoughi et al. (2016) [10] develops a character-level CNN-LSTM encoder-decoder to conduct various sentiment analysis tasks using tweets. Coupled with a seminal work by Kim (2014) [5] that shows the effectiveness of convolutional neural nets in text classification tasks, these previous research inspired our use of CNNs at both embedding and classification stages. As this appears to be the first paper to analyze bill preamble texts, our main objective is to set a benchmark using state-of-the-art methods from various domains.

3 Approach

The central task of the project is to predict the bill sponsor's party affiliation based on bill preamble texts. We first employ various embedding and classification methods shown to be effective in previous research. Then, we develop a novel neural network model that combines these methods to test whether the performance could be improved. Overall, we implement three embedding methods – two word-based and one character-based – and three classification methods, which yield nine different baseline models. This section concludes with a detailed discussion on the architecture of the CNN-LSTM model using character-based word embedding we construct.

3.1 Baseline Methods

Embedding Method 1 – Bag of Words (“BOW")

This approach vectorizes a word by counting the number of occurrences of a given word within the corpus/vocabulary. A number of social scientists have employed the method to analyze congressional speech data [1, 8], and we adopt a simplified version of their procedures. We first limit the size of the vocabulary (i.e., fix the dimension of a word embedding) based on word frequency. Next, we apply the following standard pre-processing steps to remove words that do not deliver much signal on the party affiliation of the bill sponsor: (a) remove stop words,[1] (b) stem the words to their roots.[2]

Embedding Method 2 – Word2Vec (“W2V")

Despite its intuitive appeal, however, BOW is limited in its inability to take context, which likely carries important information about the meaning of a given word, into account. W2V is an iteration-based method that overcomes such a shortcoming by extracting contextual information; it vectorizes words by “the probability of a word given its context” and yields a distributed representation of words. We apply this embedding method by taking Google's pre-trained Word2Vec model, which includes word vectors of three million words trained on 100 billion words from a Google News dataset [6]. Specifically, for neural models (i.e., CNN and RNN), the embedding layer takes this pre-trained vector as the initial word embedding. This is motivated by Iyyer et al. (2014) who find that initializing the word embedding matrix using W2V “improves the performance of RNN models over random initializations" [4].

Embedding Method 3 – Character2Vec (“C2V")

This character-based embedding method relies on the *Tweet2Vec* method presented by Vosoughi et al. (2016)[10]. As the name suggests, the authors' method returns a character-level embedding for a tweet, and the idea behind the method remains very similar to that of W2V. However, instead of skip-gram or CBOW approach, the method relies on CNN-LSTM encoder-decoder to arrive at a character-level embedding: “the sequence of features (e.g. character and word n-grams) extracted from CNN can be encoded into a vector representation using LSTM that can embed the meaning

[1] In addition to common English stop words, we remove common legislation-specific stop words (e.g., “bill", “act", “authorize", etc.) that do not deliver much information about party affiliation of a given bill.

[2] We rely on the stemming package – NLTK's offered on `SnowballStemmer` to reduce each word in a sentence to its root form. For instance, words “purpose" and “purposeful" are reduced to “purpos" by the stemmer.

of the whole tweet." Given the similarity in the limited size and sparseness of data – 140-character limited tweet and preamble texts whose average length is around 170 characters, we conjecture this method to be potentially effective in extracting relevant features.

Classification Method 1 – Logistic Regression ("LR")

This is the off-the-shelf classifier in our experiment. The relatively simple nature of the classification task and Iyyer et al. (2014)'s adoption of LR classifier as the baseline motivated our usage.

Classification Method 2 – CNN

Kim (2014) shows that CNN models can perform well in sentence classification tasks. Just as in the original paper, we adopt a similar baseline structure where a convolution layer produces a feature map with certain sized multiple filters and a max-over-time pooling operation captures "the most important feature for each feature map" [5]. In addition to its proven effectiveness in various sentiment analysis tasks, CNN's computational advantage from possibility of computing features in parallel renders it a sensible approach to implement.

Classification Method 3 – LSTM)

The principle of semantic compositionality states that "the meaning of a (syntactically complex) whole is a function only of the meanings of its (syntactic) parts together with the manner in which these parts were combined" [7]. Such a principle is relevant to the current task because both the usage of certain words and the way in which they are used likely reflect the party affiliation of a bill sponsor. Recurrent Neural Networks ("RNN") is a family of neural networks that can model this principle. We specifically employ a variant of RNN – LSTMs – that overcomes the vanishing gradient problem [3]. This method is particularly appealing since it allows the model to learn long-term dependencies, an important attribute to retain especially for long sentences.

A LSTM model has units composed of a memory cell, an input gate, an output gate and a forget gate. It computes a relation from an input sequence $X = (x_1, \dots, x_T)$ to an output sequence $Y = (y_1, \dots, y_T)$ with following equations:

$$\begin{aligned} i_t &= \sigma(W_{ix}x_t + W_{im}m_{t-1} + W_{ic}c_{t-1} + b_i) \\ f_t &= \sigma(W_{fx}x_t + W_{fm}m_{t-1} + W_{fc}c_{t-1} + b_f) \\ o_t &= \sigma(W_{ox}x_t + W_{om}m_{t-1} + W_{oc}c_{t-1} + b_o) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g(W_{cx}x_t + W_{cm}m_{t-1} + b_i) \\ m_t &= o_t \odot h(c_t) \\ y_t &= \phi(W_{ym}m_t + b_y) \end{aligned}$$

where i, f and o are the input gate, forget gate and output gate, respectively. c and m are the activation vectors for each cell and memory block, and the weigh matrices W and bias vectors b build a connection among the input layer, output layer and memory block.[3]

3.2 Main Model & Layer Details

In addition to nine different methods, we develop a CNN-LSTM model with a character-based word embedding. On the embedding, while largely motivated by the C2V method, our method – denoting it as "W2C2V" – applies the character-embedding method at the word level instead of turning the entire sentence into a sequence of characters. More concretely, we follow the procedure below:

1. Tokenize each sentence in the corpus to obtain a list of list of words.
2. For each word in a list (single sentence), transform it to a sequence of characters, and then vectorize each character into a one-hot vector.
3. Concatenate these vectors within a word, then apply the 1D convolution to extract features and use max-pooling.

[3]Further on notations, $\odot$ indicates the scalar product of two vectors, $\sigma(\cdot)$ represents the sigmoid function, and $g(\cdot)$, $h(\cdot)$ and $\phi(\cdot)$ are the input, cell output and network output activation function.

4. Finally, pass the output from step 3 to a highway network based on Srivastava et al. (2015) [9] to derive a character-based word embedding.

After vectorizing the bill preamble texts, we use the CNN-LSTM architecture designed to exploit the advantage of each method; CNN layer can extract important features from the input data, the output of which LSTM layer can then use to capture long-term dependencies that might be relevant in classifying bills. Figure 1 illustrates the model,[4] and a formal description of the process that uses three different kernel sizes is as follows:

$$
\begin{aligned}
x_{conv_1} &= \text{Conv1D}(x_{embed}) \\
x_{lstm_1} &= \text{LSTM}(x_{conv_1}) \\
x_{conv_2} &= \text{Conv1D}(x_{embed}) \\
x_{lstm_2} &= \text{LSTM}(x_{conv_2}) \\
x_{conv_3} &= \text{Conv1D}(x_{embed}) \\
x_{lstm_3} &= \text{LSTM}(x_{conv_3}) \\
x_{merged} &= \text{Concatenate}(x_{lstm_1}, x_{lstm_2}, x_{lstm_3}) \\
x_{dropout} &= \text{Dropout}(x_{merged}) \\
\hat{y}_{pred} &= \sigma(W x_{dropout} + b)
\end{aligned}
$$

Finally, loss is computed by applying the following binary cross entropy loss layer:

$$\mathcal{L} = -(y\text{log}(\hat{y}) - (1 - y)\text{log}(1 - \hat{y}))$$

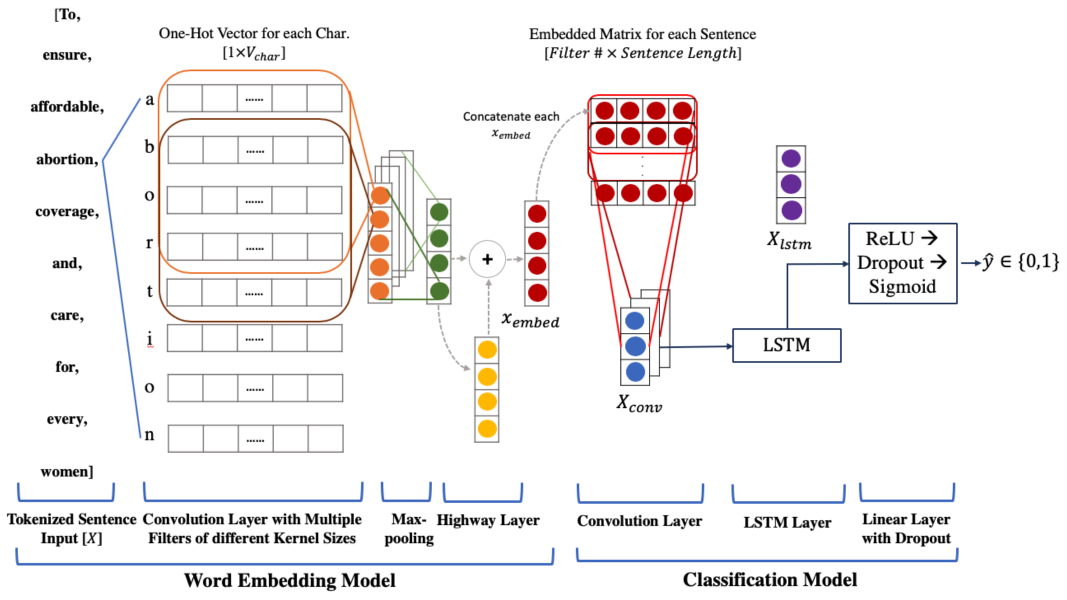

Figure 1: Main Model: CNN-LSTM with Word2Char2Vec Embedding

4 Experiments

The first stage of the experiment analyzes the results from implementing the nine baseline methods on bill preamble texts. Then, we implement the main CNN-LSTM model. Finally, upon finding that a single-layer LSTM with W2V embedding outperforms all models, we conduct a hyperparameter-tuning exercise to obtain the highest accuracy score.

[4]The figure itself only shows one set of CNN and LSTM layers to conserve space. Note that we apply three different sets for each kernel size employed.

4.1 Data

Bill Preamble Text Data

Every U.S. federal legislation contains a preamble that states its purpose. To gauge the partisanship of a given bill, we consider such preamble texts instead of the entire bill texts; most bills contain procedural texts/labels and numeric digits that NLP models cannot exploit. Instead of analyzing the entire texts that are computationally expensive and potentially abundant of insignificant words, we focus on the portion that reflects intent of a bill, which is presumably influenced by a bill sponsor's party affiliation. Examples of abortion-related bills authored by politicians from the two parties are provided in Table 1. We have collected bill preamble texts from the 93rd to the 115th congress (1973 to 2018). Each congress session has approximately 12,800 bills on average, and the average length of a bill in the dataset is 163 characters, close to that of a tweet (140 characters). Overall, we analyze preamble texts of 295,324 bills sponsored by either Republican or Democratic politician in the US federal legislatures.[5] More detailed summary statistics of key variables in the dataset are provided in the Appendix Table 3.

Annotated Ideological Book Corpus ("IBC")

This is a dataset of ideologically labeled – "liberal", "conservative", or "neutral" sentences constructed by Gross et al. (2013) [2] and annotated by Iyyer et al. (2014) [4]. Gross et al. (2013) build the data by collecting texts from political blogs, magazine articles, and books by authors with known political ideologies from 2008 to 2012. Iyyer et al. (2014) then cull politically biased sentences using a classifier and rely on a crowdsourcing platform to label each sentence. In sum, there are 3,726 annotated sentences comprised of 2,025 liberal and 1,701 conservative sentences. We employ this dataset to implement the transfer learning framework. Examples of abortion-related sentences annotated as either "liberal" or "conservative" are provided in Table 1.

Table 1: Data Text Examples on Abortion

Bill Preamble Text	
Democrat [S.Res.526–115]	"A resolution expressing the sense of the Senate that politicians should not interfere with a woman's personal health care decisions or attempt to prevent providers from offering their full medical recommendations to their patients."
Republican [S.2311 – 115]	"A bill to amend title 18, United States Code, to protect pain-capable unborn children, and for other purposes."
Ideological Book Corpus	
Liberal	"Reformers began by attacking laws that prevented women whose lives were threatened or who were pregnant as a result of rape or incest from getting abortions."
Conservative	"The disturbing effect of partial-birth abortion on medical personnel, and the guilt it engenders in those who must live with its consequences, are additional state interests that justify the prohibition."

4.2 Evaluation Metric

To gauge the effectiveness of each method, we adopt a simple measure of accuracy on the target of predicting the bill sponsor's party affiliation, similar to that of Peterson and Spirling (2018) [8]:

$$\text{Prediction Accuracy} = \frac{|\text{True Positives}| + |\text{True Negatives}|}{|\text{True Positives}| + |\text{True Negatives}| + |\text{False Positives}| + |\text{False Negatives}|}$$

[5]Note that bills authored by Independent politicians are not included. Bills authored by these politicians comprise approximately 0.5% of all bills included in the dataset, meaning the exclusion should not affect the analysis. This also reduces the number of categories to be classified, thereby simplifying the analysis to the binary classification.

4.3 Hypotheses

1. **Political**: If bill preamble texts reflect ideology of a bill sponsor's political ideology, then classification models' prediction accuracy across years will be higher (lower) in the time of high (low) ideological polarization.
2. **Model Performance**: If the actual meanings of words matter in determining a bill sponsor's party affiliation, then a word-base embedding (e.g., W2V) will be the most effective.
3. **Transfer Learning**: If party affiliation is strongly correlated with political ideology, then classification models trained on bill preamble texts will perform well in classifying ideological sentences in the annotated IBC dataset.

4.4 Experimental Details

We conduct all experiments in Tensorflow that were trained on single Tesla M60 GPUs with 56GB memory each and a batch size of 500 with 10 iterations. These were trained with the Adam optimizer with a learning rate of 1×10^{-3}. For all models involving CNN, three different kernel sizes ([3,4,5]) were used, and each layer returns 100 features. For all models involving LSTM, the number of hidden layers is 64. For all models, we use dropout function with a rate of 0.5 to mitigate the potential over-fitting issue. For both bill preamble texts and IBC data, we achieve balance by ensuring the composition to be 50% Democrat (liberal) and 50% Republican (conservative).[6] As for the size of the training, validation, and test sets, we first separate out the test set as the 10% of all data. Then, we set 10% of the remaining data as the validation set, and rest constitutes the training set.

4.5 Results and Analysis

Table 2 reports the baseline and main experiment results. First, the relatively poor performance of conventional BOW and C2V embedding models relative to the word-based embedding method confirms our initial hypothesis on performance. The classifiers relying on embedding from BOW might not have enough or good information to learn features that set Republican bills apart from Democratic ones. For the latter, it is important to note that such a character-based embedding method was developed to handle "noise and idiosyncracies in tweets" [10]. As bill texts use formalized grammatical English, the general advantage character-based embedding has over the word-based method might not be applicable to our setting.

Another notable result includes LSTM outperforming other methods when using W2V. The clear improvement in performance relative to the off-the-shelf classifier is likely attributable to LSTM's ability to extract and exploit semantic information in a sentence.

Table 2: Baseline and Main Results

Method	Prediction Accuracy
BOW – LR	0.51
BOW – CNN	0.50
BOW – LSTM	0.50
W2V – LR	0.50
W2V – CNN	0.67
W2V – LSTM	**0.71**
C2V – LR	0.50
C2V – CNN	0.50
C2V – LSTM	0.50
W2C2V – CNN-LSTM	0.50
W2V – CNN-LSTM	0.69
Overfitted Result (LSTM)	0.81

Our CNN-LSTM model with character-based word embedding performs just as poorly as the other C2V methods. A similar line of reasoning from above could be applied to explain this poor

[6]This means we exclude any bills sponsored by independent politicians and IBC sentences labeled as "neutral."

performance; even if the embedding process narrows its focus to the word-level, character-level relationships within a word may not be informative enough to extract meaningful features for the given task. This interpretation is further bolstered by the model's comparable performance when using a typical W2V instead of our W2C2V. Interestingly, adding a LSTM layer to CNN improves the performance, but it still does not outperform a single LSTM, which seems to suggest that the two methods are not complementary.

Based on the results above, we now conduct a parameter-tuning exercise on the LSTM model to identify the best parameters. As indicated in figure 2, we first test with three different word-embedding methods – Google's W2V, Stanford GloVe, and our own W2V trained on bill preamble texts. As we find that the last method yields the highest accuracy, we tune three different aspects of the model: (a) numbers of hidden units in the LSTM layer and (b) dropout rates, total of nine different combinations. We find that our initial configuration of setting 64 hidden units and 0.5 dropout rate yields the highest accuracy. Finally, we experiment with the number of LSTM layers to determine whether "deepening" the learning network improves the results, but we find null results; albeit comparable, neither outperforms the single-layer model.[7] All results are reported in Table 4 in the Appendix. In sum, the best LSTM model yields the accuracy score of **0.714**.

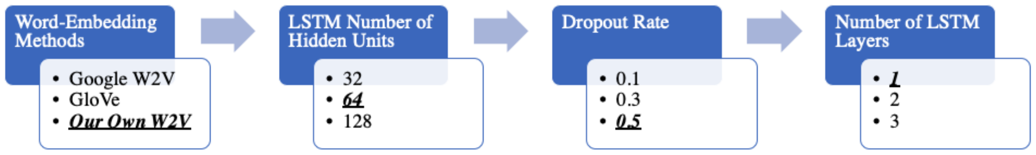

Figure 2: Hyperparameters and values selected for LSTM. Bolded values exhibited the highest accuracy, hence were selected for the final model.

On the political hypothesis, figure 3 shows a similar trend between the prediction accuracy and a measure of ideological polarization based on roll-call votes; the latter suggests that both the House and the Senate have become increasingly polarized, and the prediction accuracy of the LSTM model exhibits a similar pattern. This suggests that while legislators' voting behaviors, the basis of the ideology measure, might have become increasingly polarized, the language they use in bills has not followed as stark of a trend. We restrain from making too general of a claim on such trends, but they suggest that Republicans and Democrats are not using highly distinguishable language in their bills.

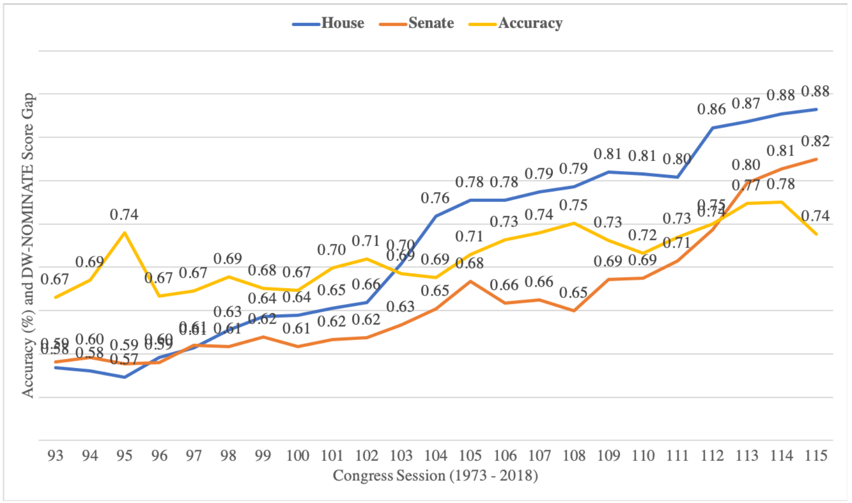

Figure 3: Comparison of Prediction Accuracy and Ideological Polarization in the U.S. Congress

[7]While not reported, we also built and added an attention layer. Contrary to initial expectation, we find that the attention layer actually inhibits the learning process of the model and yields a much lower accuracy score.

5 Transfer Learning

We apply the single-layer LSTM model, which showed the highest prediction accuracy, trained and tested on the bill preamble text data to IBC data and test the aforementioned “transfer learning" hypothesis. The idea is as follows: while there is a close correlation between a political party and ideology (e.g., Republicans are generally thought to be conservative and Democrats liberal), the mapping is not necessarily one-to-one. That is, a Democratic politician could presumably hold conservative ideology and vice versa. If, however, there is a strong correlation, our model trained to identify differentiating factors for party affiliation could presumably perform well on labeling ideologically-charged sentences.

The model manages to correctly classify **57%** of the sentences. When compared to the authors' original results that range from 62% to 69%, this result cannot be deemed high. Albeit only suggestive, the fact that the LSTM model trained to classify bill texts by their sponsors' party affiliation can correctly classify nearly 60% of the ideologically-charged sentences aligns with the general consensus that party affiliation is correlated with political ideology. In sum, while we cannot claim that our initial hypothesis on transfer learning is confirmed, we believe that our model learned features not restricted to the domain of the U.S. federal legislation texts.

6 Conclusion and Future Work

This paper analyzed the U.S. federal bill preamble texts from 1973 to 2018 using various embedding and deep learning methods. Overall, we find that a single-layer LSTM with W2V embedding method performs the best with the prediction accuracy of 71%. Comparing the trend of prediction accuracy across years to that of politicians' ideology scores reveals a positive correlation; the prediction of accuracy of our model is higher in years of greater ideological cleavage. Finally, applying our trained model to a different dataset comprised of ideologically-labeled sentences exhibits a moderate degree of transferability, which hints at the external applicability of our model.

The current project can be extended on multiple fronts. One obvious extension is to consider the entire bill text data, which will require a much more involved text preprocessing and computing power/time. Another possible venue is to train the model to predict different characteristics of a bill (e.g., the ideology score of a given bill sponsor's ideology score, the probability of reaching a different legislative stage, etc.). Indeed, training the model to learn aspects closer to political ideology should result in a higher degree of transferability. It is our hope that this paper will serve as a meaningful benchmark for analyzing legislation texts.

7 Additional Information

- Mentor: **Amita Kamath**
- External Collaborator (Advisor): **Prof. Andrew Hall** from the Department of Political Science.

Acknowledgments

We thank Chris Becker for helpful suggestions on pre-processing the bill text data and Prof. Mohit Iyyer for sharing his annotated IBC data. We also thank Prof. Chris Manning and Amita Kamath for thoughtful comments and profitable guidance.

References

[1] Matthew Gentzkow, Jesse M Shapiro, and Matt Taddy. Measuring Group Differences in High-Dimensional Choices: Method and Application to Congressional Speech. Working Paper 22423, National Bureau of Economic Research, July 2018.

[2] Justin H. Gross, Brice Acree, Yanchuan Sim, and Noah A. Smith. Testing the Etch-a-Sketch Hypothesis: A Computational Analysis of Mitt Romney's Ideological Makeover During the 2012 Primary vs. General Elections. In *American Political Science Association 2013 Annual Meeting*. American Political Science Association, 2013.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, November 1997.

[4] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. Political Ideology Detection Using Recursive Neural Networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1113–1122, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[5] Yoon Kim. Convolutional Neural Networks for Sentence Classification. *EMNLP*, August 2014. arXiv: 1408.5882.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781 [cs]*, January 2013. arXiv: 1301.3781.

[7] Francis Jeffry Pelletier. The Principle of Semantic Compositionality. *Topoi*, 13(1):11–24, March 1994.

[8] Andrew Peterson and Arthur Spirling. Classification Accuracy as a Substantive Quantity of Interest: Measuring Polarization in Westminster Systems. *Political Analysis*, 26(1):120–128, January 2018.

[9] Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway Networks. *arXiv:1505.00387 [cs]*, May 2015. arXiv: 1505.00387.

[10] Soroush Vosoughi, Prashanth Vijayaraghavan, and Deb Roy. Tweet2vec: Learning Tweet Embeddings Using Character-level CNN-LSTM Encoder-Decoder. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 1041–1044, New York, NY, USA, 2016. ACM. event-place: Pisa, Italy.

Appendix

Table 3: Summary Statistics – U.S. Federal Legislation (1973 – 2018)

	Bill Preamble Texts (N = 295,324)
Bill Preamble Length	
min	9
max	1794
mean (sd)	162.58 ± 87.97
Republicans Party	
mean (sd)	0.42 ± 0.49
Number of Cosponsors	
min	0
max	432
mean (sd)	11.42 ± 30.70
Bill Sponsor Gender (Male)	
mean (sd)	0.90 ± 0.30

Table 4: LSTM Hyperparameter Results

Hyperparameter Combinations (Hidden Layer #, Dropout)	Prediction Accuracy
32, 0.1	0.686
32, 0.3	0.690
32, 0.5	0.688
64, 0.1	0.688
64, 0.3	0.698
64, 0.5	**0.714**
128, 0.1	0.695
128, 0.3	0.701
128, 0.5	0.686
2-layer LSTM, 64, 0.5	0.700
3-layer LSTM, 64, 0.5	0.699