Predicting Roll Call Votes using Machine Learning Methods

Tom (Hyeon Seok) Yu [*1] **Floyd Jiuyun Zhang** [*1]

Abstract

We present an approach for predicting roll-call votes in the U.S. Congress, using bill text word embedding as well as bill and Congress member characteristics as inputs. Various prediction models are implemented, tested and finally combined using ensemble stacking. Our methods yield higher accuracy than existing methods, especially for newly elected members of Congress.

1. Introduction

Every year, the United States Congress votes on hundreds of bills, impacting individuals, social groups, and businesses in the U.S. and beyond. Predicting how each member votes on each bill is thus of enormous scholar, policy, and commercial interests. In academic research, roll call vote predictions have been applied to the testing of majority party agenda power in Congress (Ballard 2021), a long-debated hypothesis in political science (Cox and McCubbins 2005). In non-academic settings, such predictions can help organizations and business anticipate policy enactment and thereby reduce policy uncertainty; such tools would also help identify persuadable legislators as subjects of influence and activism.

Here, we present a prediction approach based on bill texts and congress member characteristics, which are taken as inputs of our algorithms. The outputs are binary voting outcomes, namely whether each member votes "yea" or "nay" on the passage of each bill. We implement a variety of prediction models (including logistic regressions, decision tree methods, and ensemble stacking) and report their accuracy. Our models achieve high accuracy rate and outperforms existing alternatives. Further, our methods perform well on predicting voting behaviors of newly elected members of Congress without relying on additional information about the members such as campaign finance and news reports, which constitutes an improvement to the existing studies.

[*]Equal contribution [1]Political Economy, Stanford Graduate School of Business, USA. Correspondence to: Tom (Hyeon Seok) Yu <tom.hs.yu@stanford.edu>, Floyd Jiuyun Zhang <floydjz@stanford.edu>.

2. Related Literature

Due to the intuitive structure of the problem as that of a classification, a number of studies have applied machine learning methods to predict roll call votes.

2.1. Predicting Roll Call Votes

Gerrish and Blei (2011) utilize the bill text data and the Bayesian ideal point topic model to predict the roll call vote and achieve about 4% accuracy gain relative to the baseline that predicts all yea votes. Taking this work as a baseline, Kraft, Jain, and Rush (2016) develop an embedding-based model – specifically, GloVe embedding proposed by Pennington, Socher, and Manning (2014) – that relies on text and past voting data. Applying the model to the 106th-111th Congress, the authors' model makes about 2% gain to the benchmark. Kornilova, Argyle, and Eidelman (2018) further extends the literature by developing a convolutional neural network model that takes bill summary text and sponsor data. More recently, Ballard (2021) proposes a text-based model using the `Doc2Vec` embedding method to achieve (to our knowledge) the current state-of-the-art result of 94.8% accuracy on the out-of-sample prediction for final passage votes in the House. One notable limit of these methods pertains to their inability to make predictions for the newly elected Congress members, who do not have any voting records, hence ideological data, necessary for training their models.

2.2. Predicting Newly Elected Members' Votes

There have been at least two attempts at closing the aforementioned gap: using supervised-learning methods, Bonica (2018) finds campaign finance data to be highly predictive of voting behavior among newly-elected members. In addition to bill text data, Patil et al. (2019) show that incorporating information from newspaper texts and other knowledge base can improve the prediction accuracy on those newly elected members. Building on the existing literature, this paper explores whether a model that utilizes bill text data and other bill and legislator-specific features can yield a better accuracy on the existing and newly elected members' voting behavior. In terms of the specific method, we emulate Nay (2017)'s ensemble staking method, which has shown high accuracy in predicting bill passage rates.

3. Dataset and Features

We collect data for roll-call votes and congressperson (i.e., the voter)-specific features from `VoteView.com`, and bill summary text along with bill(sponsor)-specific data for the 113th-117th Congress (2013 - Present) from `GovInfo.gov`. We link roll-call votes with the passage of bills using PIPC data. [1] The unit of analysis is member-bill, and in sum, we have 620,081 examples, about 103,000 votes per congress. We apply the 70-20-10 train-validation-test split. As the chamber could vote on the same bill multiple times, only the latest votes on a given bill are retained, and bills without the force of law (e.g., simple and concurrent resolutions) are excluded from the analysis. Below lists the set of bill and sponsor-specific features employed and notes on pre-processing applied:

- Bill (sponsor)-specific features: Congress session, chamber, state, sponsor party, committee reported, total number of co-sponsors, the share of co-sponsors from the opposite party, the main topic, policy classification.[2]
- Legislator (voting member)-specific features: party, state.
- Bill summary text: 300 or 600 dimensions extracted using `Doc2Vec` after applying preprocessing.[3]

Table 1 provides the summary statistics of key features by Congress. At least three patterns stand out from the table. First, newly elected members' votes account for about 15% of the total votes. Second, House votes make up almost all the votes, which results from the fact that most of the Senate's votes pertain to nominations and that much fewer bills go through roll calls in the Senate. Lastly, the share of sponsors from the Democratic Party jumped in the 116th Congress, as the Democrats retook the majority that year, but regardless of the majority status, the share of co-sponsors from the opposite party has remained relatively stable at about 30%.

[1] https://www.ou.edu/carlalbertcenter/research/pipc-votes

[2] The policy classification is made by the Congressional Research Service ("CRS") and represents the major policy area to which the given bill pertains.

[3] These summary texts are provided by CRS, and we applying the following pre-processing: (1) remove all digits, punctuations, section indicators, stop words, (2) stem each word, and (3) require the minimum count of 15 (across all bills) to be included in the dictionary. This leaves 7,030 unique words. Details on feature extraction using `Doc2Vec` can be found in the Methods section.

Table 1. Summary Statistics by Congress

Variables	113	114	115	116	117
Total Votes	146,349	162,251	185,413	122,818	34,733
Total New Member Votes	25,309	22,301	23,842	24,283	4,628
House Share	0.98	0.97	0.98	0.96	0.97
Democrat Share	0.46	0.43	0.45	0.54	0.51
Senate Democrat Share	0.53	0.44	0.46	0.45	0.48
Bill-Specific Data					
Sponsor Dem. Share	0.17	0.14	0.15	0.81	0.86
Avg. # of Cosponsors	34	28	20	50	61
Avg. Share of Opp. Party	0.30	0.27	0.31	0.29	0.26

4. Methods

4.1. Logistic Regression

Logistic regressions model output as distributed according to a sigmoid function where the natural parameter is a linear function of the inputs. Formally:

$$P(y = 1|x; \theta) = h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$

The cost function to minimize is given by:

$$J_\theta = \frac{1}{2} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})^2$$

The function is optimized via the Newton Conjugate Gradient Method, which combine Newton's Method with a line search to find a step size for each iteration along the "Newton direction". This line search is intended to find step sizes more suitable for non-quadratic objective functions, of which the logistic likelihood function is a member. One drawback of "Newtonian" method is that it cannot distinguish a saddle point from a local optimum, but this is not a problem here given GLMs have convex loss functions.

4.2. Regularized logistic regressions

The data is high-dimensional. It is plausible that the underlying problem is sparse as many dimensions of the large word-embedding vector may not contain useful information for predicting votes. To avoid possible overfitting, I also implement regularized versions of logistic regression by adding L1 penalization.

$$J_\theta^{reg} = \frac{1}{2} \sum_{i=1}^{N} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{k=1}^{d} ||\theta_k||$$

Inputs are re-scaled/normalized for the implementation of regularization. Due to its non-differentiability, standard Newton approaches don't work well with this function. Instead, the function is minimized using SAGA, an incremental gradient descent algorithm that is both fast and robust to non-differentiability

4.3. Random Forest

Another baseline model we adopt based on existing literature is random forest, a bagging (boostrap-aggregating) method (Breiman 2001; Bonica 2018; Nay 2017). In bagging, an individual prediction $\hat{y}_i$ is made by taking the average of predictions from each tree:

$$\hat{y}_i = \hat{f}(x_i) = \frac{1}{B}\sum_{b=1}^{B} f_b(x_i)$$

where B is the number of bootstrap samples (trees). Since this is a classification problem, the final prediction on a given example is made by taking the majority vote. One common issue of the method is that individual trees are highly correlated with each other, and the random forest method addresses this problem. By randomly sampling observations from training data and growing a decision tree on each sample while only considering a subset of features, trees in random forests share low correlation with other trees, thereby reducing variance (James et al. 2013). Given this design that de-correlates trees, this method can be effective when features are highly correlated with each other.

4.4. XGBoost

In contrast to bagging methods like random forests, boosting methods grow trees sequentially: each tree is built relying on information of trees from the previous iteration. One critical aspect of the method pertains to the weighting of examples. Specifically, it assigns greater weights to misclassified examples, thereby allowing the model to improve in each iteration. Instead of a simple boosting model, we implement the XGBoost model, which has shown higher efficiency and built-in regularized structure that mitigates the overfitting issue. More formally, given the label $y_i \in \mathbb{R}$ and a vector of m features $X_i \in \mathbb{R}^m$, the prediction $\hat{y}_i$ is made as:

$$\hat{y}_i = \phi(X_i) = \sum_{k=1}^{K} f_k(X_i), f_k \in \mathcal{F}$$

where f_k corresponds to an independent tree structure q and leaf weights w. The following regularized objective function is optimized (minimized) to learn these functions:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \quad \Omega(f) = \gamma T + \frac{1}{2}\lambda||w||^2$$

where T is the number of trees in each leaf node. This objective function is optimized using the second-order approximation (Chen and Guestrin 2016).

4.5. Bill Summary Text-Augmented Models (`Doc2Vec`)

For all baseline models described, we extend the data with the word-embedding features obtained from implementing

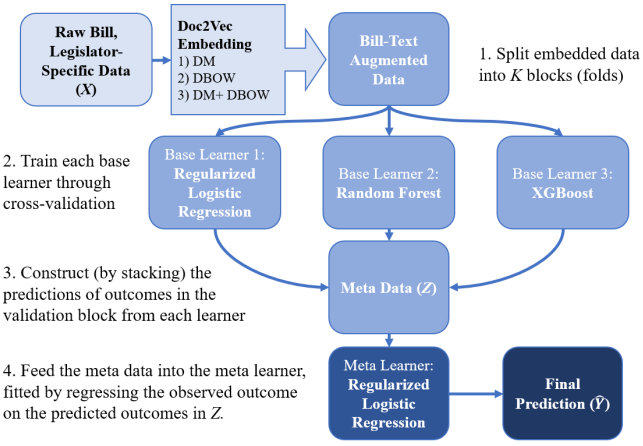

Figure 1. Ensemble Stacking

`Doc2Vec` (Le and Mikolov 2014). Similar to `Word2Vec`, a vector representation of words, `Doc2Vec` trains an unsupervised algorithm that "represents each document by a dense vector which is trained to predict words in the document." There are two different methods of obtaining these paragraph vectors: distributed memory ("DM") and distributed bag of words ("DBOW"). The former "considers the concatenation of the paragraph vector with the word vectors to predict the next word in a text window." In essence, this paragraph vector serves "as a memory that remembers what is missing" from the given context, representing a "topic" of the document/paragraph. DBOW, on the other hand, does not take any ordering of the word into account and trains by predicting words randomly sampled from the given paragraph. Although it forgoes contextual information, one advantage is the efficiency gain in memory, as it does not require learning word vectors. Ballard (2021) applies this method on bill texts and adopts a concatenated version of these two embedding methods. We implement all three variants to see which embedding yields the best performance.

4.6. Ensemble Stacking

Aforementioned ensemble methods – random forests and XGBoost – combine weak learners to generate a strong learner, and stacking combines these strong learners to create a stronger meta learner. Figure 1 shows the high-level procedure of constructing this meta learner (Laan, Polley, and Hubbard 2007). The general idea is to train individual base learners, then train a model, a regularized logistic regression, to assign weights to predictions produced by each base learner. Nay (2017) adopts a very similar stacking model in predicting bill passage rate, and we test whether the architecture can be effective in predicting roll call votes.

5. Experiments/Results/Discussion

The main objective is to determine whether the proposed methods can better predict legislators' roll call behavior than existing alternatives. Before delving into the results, we describe the experimental design, briefly discuss the hyperparameter tuning, then define the evaluation metric.

5.1. Experiment Setup

For all experiments, our models are trained and tested on each Congress separately. The 70-20-10 split is applied to the incumbent members. For the newly-elected members' votes, we randomly select half of them and set them aside as a separate test set for the purpose of evaluating how well our models predict those members' voting behavior. We adopt the following set of baselines for the systematic assessment of our models:

1. All Yea: most legislators tend to vote "yea" on bills that make it to the chamber floor. This serves as the common lower-bound baseline that existing studies have adopted.

2. Ballard (2021) (Overall): though the time periods of datasets do not coincide, their work, to our knowledge, has achieved the highest accuracy on the prediction task.

3. Patil et al. (2019) (New Members): the time periods do not coincide, but their work specifically considers the newly-elected/unobserved members, rendering it a more suitable baseline for the models' performance on newly-elected members.[4]

5.2. Hyperparameter Tuning

There are many hyperparameters to tune given the variety of models considered. For ensemble methods – Random Forest, XGBoost, and Ensemble Stacking models – we tuned each model using three-fold cross-validation, which sufficed given the large number of observations. More specifically, we implement the randomized grid search cross-validation method to identify the most suitable hyperparameters for each method and given Congress.[5]

[4]It is not clear whether Patil et al. (2019) specifically considers newly-elected members in their analysis, as their writing suggests excluding a randomly selected group of legislators from the training set. Nevertheless, given the alignment in purpose, we take their work as a baseline.

[5]The detailed list of hyperparameters considered for each method is relegated to the Appendix.

5.3. Evaluation Metric: Test Set Accuracy

We assess the performance of all models by computing the prediction accuracy.[6] For all overall results, reported accuracy reflects averages weighted by the sample size of each Congress.

5.4. Comparison of Baseline Models

Table 2 reports the performance of baseline models, and there are at least two notable results. First, all baseline models perform very well, and each yields a higher accuracy compared to the existing state-of-the-art. XGBoost seems to slightly outperform both the regularized logistic regression and random forest, but the difference appears negligible. Second, word-embeddings do not significantly improve the performance of each model, which suggests that bill and legislator-specific features we included in the "base" version of the model suffices in delivering information relevant in predicting voting behavior.[7] Lastly, figures 2 and 4 (Appendix) show that the majority of errors result from the false positives – predicting yeas for actual nays. The better result achieved by XGBoost might be attributable to its assigning greater weights to mislabeled examples during the training, and it shows lower false positive rates.

Table 2. Baseline Results by Models

Model	Overall (Weighted Avg.)
Baseline (All Yea)	**0.824**
Ballard (2021)	**0.948**
Logistic Reg	
Base	0.8997
DM	0.9442
DBOW	0.9442
DBOW-DM	**0.9450**
Logistic Reg w/ L1 Regularization	
DBOW-DM	0.9428
Decision Tree Methods	
Random Forest	
Base	0.9510
DM	**0.9513**
DBOW	0.9513
DBOW-DM	0.9509
XGBoost	
Base	0.9523
DM	**0.9524**
DBOW	0.9523
DBOW-DM	0.9521

Notes: Ballard (2021)'s results are based on the 104th-114th congress. DM and DBOW embedding models each include 300 features, while DBOW-DM model includes 600 features.

[6]Specifically, we apply the following formula: $Accuracy = 1 - \left(\frac{1}{N}\sum_{i=1}^{N}(|y_i - \hat{y}_i|)\right)$.

[7]Analyzing the most predictive features in the random forest reveals the party affiliation of a given voter (legislator), the share of opposite party members in co-sponsors of a given bill, and state indicators (especially California) to be important.

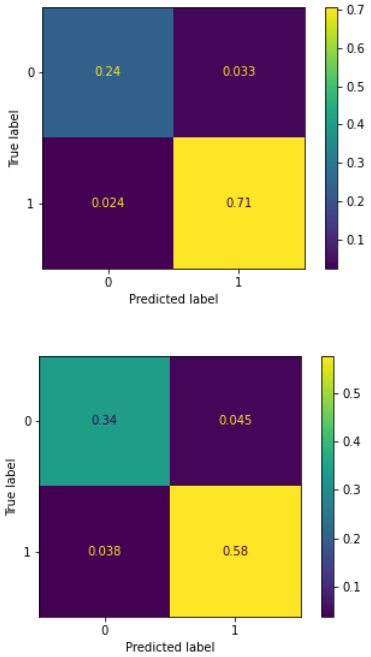

Figure 2. XGBoost Confusion Matrix (117th Congress)

Based on these results, we conduct an error analysis of comparing training and test set errors to determine whether the error results from bias or variance. Figure 3 plots the train (solid) and test (dash) error rates of logistic regression (red), random forest (green), and XGBoost (blue) models over Congress. For logistic regression, except the ongoing 117th Congress, for which we have fewer data, train errors are not significantly greater than test errors, suggesting little or no overfitting. This likely explains why regularization does not improve the performance of the Logistic regression. A similar pattern holds for tree-based models, and training error is generally lower than test error, which suggests overfitting, hence high variance.

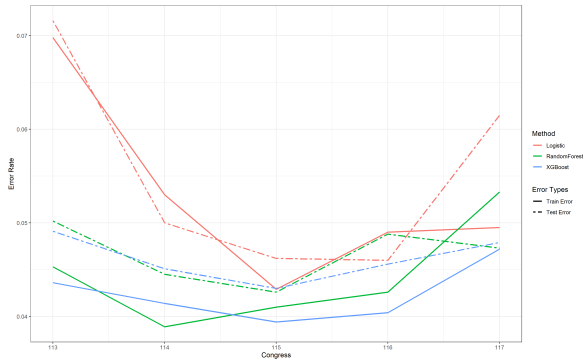

Figure 3. Error Analysis - Baseline Models

5.5. Comparison of XGBoost and Ensemble Stacking

We now implement the main model, the ensemble stacking model that takes the three baseline models as base learners. In the analysis, we also include XGBoost for the comparison, as it performed the best of the three. Table 3 reports the results for the incumbents and the new members. Although the stacking model does not yield a noticeably higher prediction accuracy, both baseline and bill-text augmented versions of the model outperform XGBoost counterparts. Figure 5 in the Appendix shows the results for the 117th Congress, and it remains to show a higher false-positive than false-negative rates.

Table 3. Main Results by Congress

Model - Congress	113	114	115	116	117	Overall
Incumbents						
Baseline (All Yea)	0.814	0.811	0.838	0.797	0.746	0.813
Ensemble Stacking						
Base	0.95	0.956	0.957	0.954	0.954	0.9545
DM	0.952	0.956	0.958	0.952	0.953	**0.9548**
XGBoost						
Base	0.951	0.955	0.957	0.954	0.952	0.954
DM	0.950	0.957	0.958	0.952	0.951	0.954
Newly Elected Members						
Baseline (All Yea)	0.809	0.890	0.856	0.827	0.615	0.836
Patil et al. (2019)	-	-	-	-	-	0.861
Ensemble Stacking						
Base	0.938	0.962	0.950	0.959	0.909	**0.9541**
DM	0.937	0.961	0.950	0.958	0.908	0.954
XGBoost						
Base	0.937	0.962	0.948	0.961	0.918	0.950
DM	0.937	0.963	0.949	0.960	0.910	0.950

6. Conclusion

We developed methods for predicting roll-call votes based on bill texts and some simple congress member features. Our models generally outperform existing methods, with the largest improvement seen in predicting votes by newly-elected members. Across almost all metrics, ensemble stacking achieves the best performance, as it combines predictions generated by all base learners. Logistic regression tends to perform the worst, presumably because it fails to account for interactions among features (*e.g.* members from the Democratic Party may be more likely to vote "yea" on bills related to infrastructure, hence an interaction between member partisanship and bill content).

The remaining errors may reflect factors not accounted for by our input, such as the district each member represents and sources of their campaign finance. Incorporating campaign contribution data as in Bonica (2018) into our models would be an interesting direction for future extensions.

Contributions

- Tom (Hyeon Seok) Yu
  - Collecting and processing bill-level data.
  - Implementing `Doc2Vec`
  - Implementing tree-based and ensemble stacking methods
  - Majority of the writing
- Floyd Jiuyun Zhang
  - Collecting and processing roll-call and member data; linking roll-call to bills
  - Implementing Logistic regressions
  - Minority of the writing
  - Poster

Appendix

Additional Figures and Tables

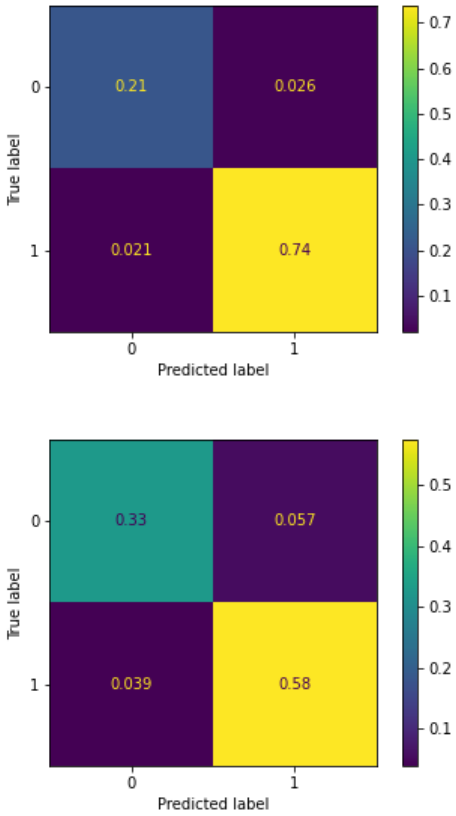

Figure 4. Random Forest Confusion Matrix (117th Congress)

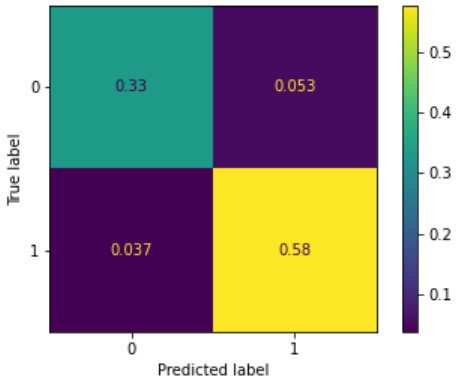

Figure 5. Ensemble Stacking Confusion Matrix (117th Congress)

Hyperparameter Tuning for each Method

Below lists all hyperparameters tuned using the `RandomizedSearchCV` package from `scikit` API. For each parameter, a range or a set of values is considered at each iteration. We limit the number of iterations to 10 due to time constraint. Detailed specifications for each Congress are omitted and available upon request.

- Logistic regression: {L1 regularization term λ}
- Random Forest: {number of trees, max. depth of each tree, max. number of features, min. number of samples required to split an internal node, min. number of samples required to be at a leaf node}.
- XGBoost: {number of trees, max. depth, learning rate, L1 regularization term (α) on weights, subsample ratio of training instances, subsample ratio of columns when constructing each tree, subsample ratio of columns at each level of depth}.
- Ensemble stacking: {regularization method (lasso, ridge, elastic net), the regularization term weight λ, the weight on each regularization term for elastic net}.

References

Ballard, Andrew O. (20, 2021). "Bill Text and Agenda Control in the U.S. Congress". *The Journal of Politics*.

Bonica, Adam (2018). "Inferring Roll-Call Scores from Campaign Contributions Using Supervised Machine Learning". *American Journal of Political Science* 62.4, pp. 830–848.

Breiman, Leo (1, 2001). "Random Forests". *Machine Learning* 45.1, pp. 5–32.

Chen, Tianqi and Carlos Guestrin (13, 2016). “XGBoost: A Scalable Tree Boosting System”. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785–794. arXiv: 1603.02754.

Cox, Gary W. and Mathew D. McCubbins (26, 2005). *Setting the Agenda: Responsible Party Government in the U.S. House of Representatives*. Cambridge University Press. 360 pp.

Gerrish, Sean M. and David M. Blei (28, 2011). “Predicting legislative roll calls from text”. *Proceedings of the 28th International Conference on International Conference on Machine Learning*. ICML’11. Madison, WI, USA: Omnipress, pp. 489–496.

James, Gareth et al. (25, 2013). *An Introduction to Statistical Learning: with Applications in R*. 1st ed. 2013, Corr. 7th printing 2017 edition. New York: Springer. 440 pp.

Kornilova, Anastassia, Daniel Argyle, and Vlad Eidelman (21, 2018). “Party Matters: Enhancing Legislative Embeddings with Author Attributes for Vote Prediction”. *arXiv:1805.08182 [cs]*.

Kraft, Peter, Hirsh Jain, and Alexander M. Rush (2016). “An Embedding Model for Predicting Roll-Call Votes”. *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. Austin, Texas: Association for Computational Linguistics, pp. 2066–2070.

Laan, Mark van der, Eric Polley, and Alan Hubbard (7, 2007). “Super Learner”. *U.C. Berkeley Division of Biostatistics Working Paper Series*.

Le, Quoc V. and Tomas Mikolov (22, 2014). “Distributed Representations of Sentences and Documents”. *arXiv:1405.4053 [cs]*. arXiv: 1405.4053.

Nay, John J. (10, 2017). “Predicting and understanding lawmaking with word vectors and an ensemble model”. *PLOS ONE* 12.5.

Patil, Pallavi et al. (2019). “Roll Call Vote Prediction with Knowledge Augmented Models”. *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. CoNLL 2019. Hong Kong, China: Association for Computational Linguistics, pp. 574–581.

Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). “Glove: Global Vectors for Word Representation”. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, pp. 1532–1543.