How a Bill Becomes a Law - Predicting Votes from Legislation Text

David Goldblatt DTG@STANFORD.EDU
Tyler O'Neil TONEIL@STANFORD.EDU

Abstract

Thanks to the efforts of organizations like GovTrack (Tauberer, 2012), a tremendous amount of roll-call and bill-text data has become available over the past few years. At the same time, most analyses are based on ideal-point models that use only the votes themselves in modelling voting decisions, and ignore bill text. We analyze a variety of models for predicting the outcomes of roll-call votes given vote text. We arrive at a neural network classifier which takes counts of words in bill texts as input and achieves state-of-the-art accuracy in predicting vote outcomes.

1. Introduction

Academic interest and economic pressures have created an entire field of quantitative political science that seeks to understand how governments behave. These researchers frequently use the rich dataset of legislative roll calls, which are a history of how congress members vote on legislation. In previous work, this data has been mined to find underlying structure like partisan affiliation, evidence of polarization, and even predict future voting outcomes (Clinton et al., 2004; Gerrish & Blei, 2011). These approaches typically involve complex models, such as *ideal point modeling* that explicitly map legislators to a point along a political line.

Given a large and accessible dataset, and a field that is historically reliant on complex modelling, we decided to use simple bag-of-words models, a topic-modelling based classifier, and a neural network to address the problem of predicting roll calls from the text of bills. These approaches have the benefits that they are easier to implement and understand, and make fewer complex modelling decisions that are open for questioning.

2. Baselines

To begin, we trained models for each voter separately. We took seven members of congress and trained six different models on two-thirds of their voting record, for a total of forty-two models. The training set and testing set were randomly shuffled and split. Our input features were the top 4000 most common words, with a binary feature of whether or not they existed. We then tested on the remaining third of the dataset. We chose senators by being active and recognizable, like the senate and house leaders and former presidential candidates. We used 3 Republicans and 4 Democrats. All of the classifers except the neural net were taken from Matlab libraries. (MATLAB, 2010)

Binary Classifier	Accuracy
SVM	0.7661
Random Forest	0.8343
Naive Bayes	0.7457
Regression NN	0.8222
Logistic NN	0.8169
Always 'Yea'	0.7659

Table 1. Classification Test Accuracies. Classifiers were trained on seven senators, and we report above the weighted average of their accuracies. The SVM was cross validated and tuned to use a polynomial decision boundary of degree 3. Both Neural Nets used a logistic function for their hidden units, but the regression used a linear output layer, whereas the logistic used a logistic output layer.

The biggest challenge faced by classifiers is handling the fact that there is very little data with high dimensional features. Some senators have as few as six-hundred bills, and each bill has 4000 features. This creates a challenge for algorithms like SVMs that cannot find a reasonably good decision boundary in this high dimensional space.

3. Data Cleaning

One problem we identified was the low-quality of tokens we trained on. Many of the most common words in the corpus were low-content ones which did not impart significant insight into the meaning of a document. We adopted a number of strategies to increase the quality of our input features. These improvements to the source text increased the accuracy of our final classifiers by a few percent.

As a first cleaning pass, we eliminated as many of the formatting effects as we could. The bill texts contain line and page numbers, section headings, non-ascii characters, and numerous other characteristics that undermine the effectiveness of our classifiers. We therefore strip out any digits or non-ascii characters from our token stream, and then try to join words which have been split across lines or pages. This converts sequences of tokens like "this appears at a page boun- 8 9 §6.3 dary" into "this appears at a page boundary".

We began using n-grams rather than single word tokens in computing features for a given document. This let us capture phrases such as "homeland security", which have some semantic meaning that cannot be inferred from either

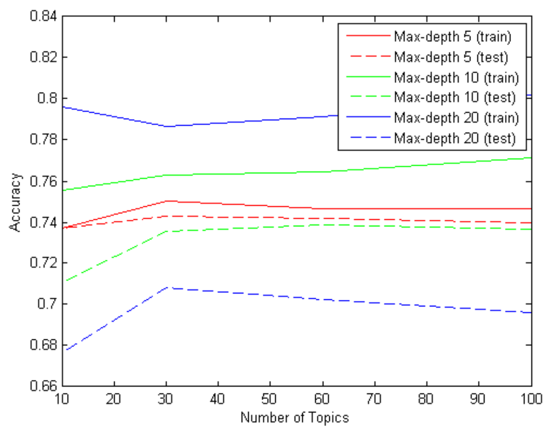

Figure 1. Random Forest performance

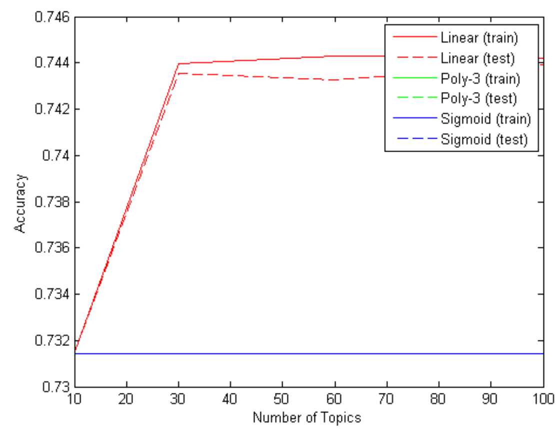

Figure 2. SVM performance. The poly-3 line matches the sigmoid line closely, and is covered up by it

word in isolation. We additionally switched to using the term frequency-inverse document frequency (tf-idf) scoring (Jurafsky & Martin, 2000). This increases the score of an n-gram that appears frequently in a document, but decreases it the more documents it appears in, so that words receive higher scores for being distinctive to a document. We removed short words (words of length 3 or less), which tend to be low content, and which furthermore hurt some of the significance of our n-grams (consider "Department of Education" in a bigram model; "department education" captures more meaning than "department of"). We excluded stop words and lemmatized tokens to make the features independent of tense or declension.

4. Topic Modelling

We still had two problems. First, despite the cleanup discussed above, our features tended to be rather low quality. Second, our classifiers had the ability to overfit to the data available. The low-quality of our features stems from the lack of semantic information available from simple n-gram-counts models. For example, some of the top 100 n-grams we choose (if picking n-grams of length in between 1 and 3) include "house", "described", "project", "within", "entity", and "authorized". These do not give us any information that might be useful in determining whether or not a given representative will vote for a given bill. Is the named project being expanded or reduced? What changes in the US code are being described? These facts dramatically change the nature and meaning of legislation, and thus the likelihood that representatives will vote for them.

The problem of overfitting is even more severe. For us to be able to get even a gist of what most pieces of legislation concern, we need to pick a few thousand representative n-grams. However, in the 10-session span we consider, most members of congress will vote on at most a few thousand pieces of legislation (and many will vote on far fewer - 400 voters out of the 1,207 in our corpus vote on less than one thousand pieces of legislation. 4 vote on fewer than 5). This sort of data makes it very difficult to fit models with low test error.

To address these deficiencies, we tried to adopt strategies to convert our large number of weak features into a smaller number of stronger ones. To achieve this, we assume that the documents are generated from some fixed number of topics, and attempt to infer these topics from the documents. To be specific, we ran an online Latent Dirichlet Allocation algorithm (Řehůřek & Sojka, 2010) on the tf-idf scores of cleaned 1-grams (theoretically, we should have used raw word counts rather than td-idf scores, but in practice this approach gave better results), making 20 passes over the entire corpus (at which point the topics seemed to have converged). We then used the resulting output topic distributions as the inputs to our classifiers. We ran experiments with 10, 30, 60, and 100 topics, using both Random Forests and Support Vector Machines (implemented in (Pedregosa et al., 2011) as our classifiers. In the Random Forest classifiers, we tried maximum tree depths of 5, 10, 20, and 50, and in the SVM classifiers, we tried linear, degree-3 polynomial, and sigmoid kernels. Because of the computational cost of training SVMs on the 100-topic model, we only trained on 100 randomly selected representatives, rather than on every member of Congress. The test accuracy is reported via 3-fold cross-validation performance. As a baseline measurement, the classifier that assumes representatives will always vote "yea" has an accuracy of 0.731 on the corpus of all votes.

We see in figure 4 that the Random Forest classifiers tend to overfit as the forest depth is allowed to grow. This makes sense - even with relatively few features, the expressive power of the forest quickly grows to be able to exactly match any test set data. Limiting forest depth therefore functions as regularization of the model. We suspect that with more data, this problem could be eliminated. Another promising avenue would be to adopt some sort of feature selection strategy such as mutual information between topics and the output vote variable.

With SVM-approaches, we see in figure 4 an opposite effect; the SVM training and test accuracies match almost exactly, and improve as more features are added. The problem is that despite this, we have relatively poor results. This points to a need for more or better features. Unfortunately, we seem to have levelled out at 100 topics; our experiments with higher topic counts did not improve the test or training error of any of our SVM classifiers.

5. Neural Network Model

Given the failure of topic modelling approaches, models that more directly use vote outcomes in distilling counts seemed like a promising approach. Our goal is to map a feature vector for a document to a vector of length N with the predicted votes for voters $i \in 1 \ldots N$. We use a neural network with layers $j \in 1 \ldots J$, each with a response of C_j composed of A^j neurons. That is, at each layer j, we have responses $C_j^a \in 1 \ldots A^j$. To calculate the responses, the neural net uses the standard non-linear function: $C_j^a = \sigma(W_j^a \dot{C}_{j-1} + b_j)$ We use the sigmoid function, $\sigma(x) = \frac{1}{1+e^{-x}}$. We did some experiments with $\sigma(x) = tanh(x)$ but found no significant difference.

The highest level of the network is composed of a linear regression layer, that only differs from the previous layers in that there is no sigmoid function. We also tried an additional logistic layer but with little improvement, as in the baseline experiments. The output layer uses tied weights, in that we are predicting all the congress members' votes in the same model. During training, we ignore 'Not Voting', 'Abstain' and the case where the voter was not in that house of congress at that time by not back-propagating any error. This has the effect of 'turning off' that output node.

Whereas our baseline models required us to train different models for each voter, this model combines all those models into one with shared parameters for the hidden layers. This has two major advantages. First, the shared parameters mean that training a model was much quicker: training 1,207 different models is a daunting task. Second, the shared parameters help prevent overfitting by learning structure common to all voters.

6. Experiments

We have 6,280 bills spread over nearly 20 years, and split them into a training set of size 5,000, a test set of size 800, and a validation set made of the rest. We shuffled the bills, so they did not appear in any particular order. Then we trained a number of tied-weight neural networks on the data using different features and parameters.

6.1. Interaction with Features

In addition to the tf-idf features we explored in previous sections we also tried a number of other features. First of all, we tried just using the raw binary-word features (1 if the word exists in the document, 0 otherwise) and raw counts. In addition, we tried more complex features like Latent Semantic Analysis which is a form of dimensionality reduction of the tf-idf matrix (Dumais, 2005). Since logistic classifiers are not very effective with sparse data in practice, we thought dimensionality reduction may help. Furthermore, we tried hashing the counts of all our words into a feature vector. That is, for every word in the document, we hash it to find its position in a feature vector and increment that feature. Since the number of occurrences of our words have a long tail, and many of these tail words are likely good differentiators (like 'Kodiak' may indicate an environmental bill), we hoped hashing would save the long-tail words in a way that would not overfit and keep our feature dimensions small enough for training.

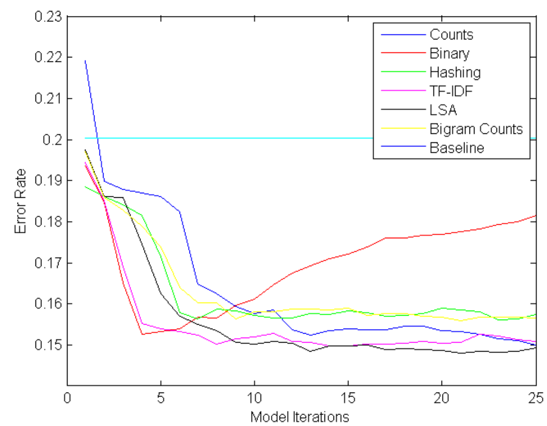

Figure 3. Features. We ran the same single layer network with 32 hidden units on several different document representations. 'Counts' is a bag-of-words of raw counts. 'Binary' is a bag-of-words with only '1' for present and '0' otherwise. 'Baseline' is the result of always guessing 'Yea'.

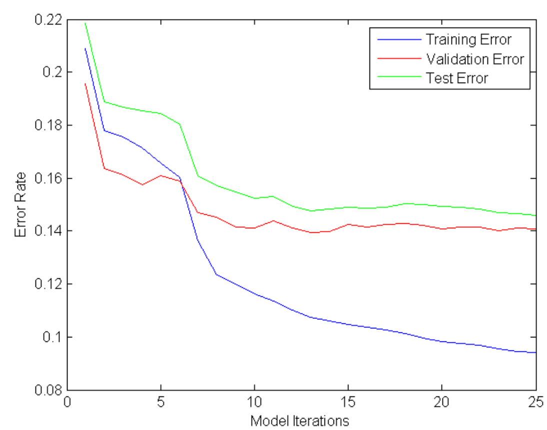

Figure 4. Train/Test Error over 250 iterations of l-bfgs. We ran the same single layer network with 32 hidden units on the 'Counts' bag-of-words different document representations.

Figure 3 shows the results of different features. Our first experiment with different features showed us that our document representation did not have a major impact. We tried different numbers of features between 2,000 and 4,000 in our bag-of-word models, but found little difference and

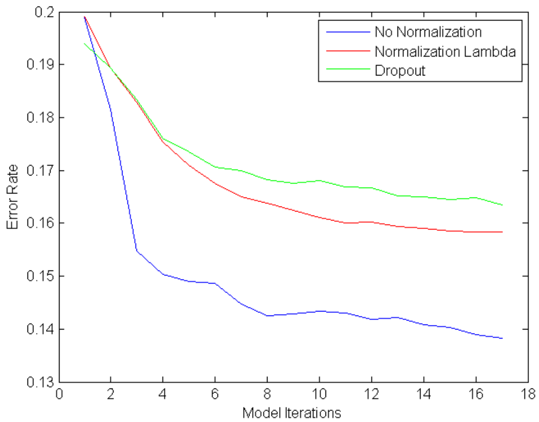

Figure 5. Word-Count Features. Test error over 250 iterations of l-bfgs or stochastic gradient descent. Single layer network with 100 hidden units and the word-count features. $\lambda = 0.1$.

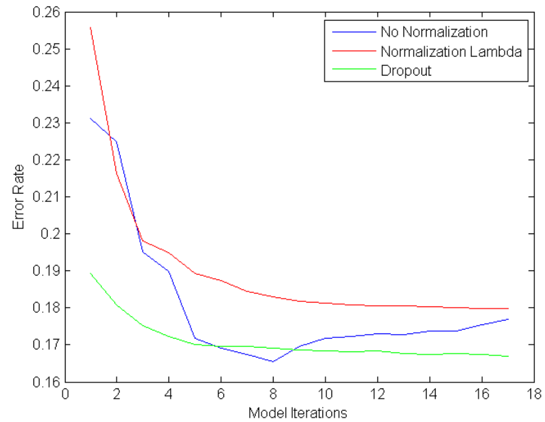

Figure 6. Binary-Word Features. Test error over 170 iterations of l-bfgs or stochastic gradient descent. We ran the same single single layer network with 100 hidden units on the binary bag-of-words document representation. We used a weight decay term, λ *of 0.1.*

eventually settled on 4,000. As for representation, using counts was approximately as effective as more complex features like LSA and bigram counts. Using raw counts turned out to give us the best results on our validation set so we continued our development on those.

6.2. Model Structure

Next, we tried different neural network models by tuning the number of hidden layers and the number of hidden units in each layer. Once again, we found that simpler did better. Multi-layer neural nets depressed results below the naive baseline of always voting 'Yea' and any a single layer network with above 10 units did about equally as well. We were limited by machine memory, and had trouble extending beyond 400 hidden units as a consequence of our high feature dimension, time to train, and computer memory.

Figure 4 has the results of the best system we tuned. It uses 32 hidden units using 'Counts' bag-of-words and performs 85.12% accuracy on the 800 held-out test set bills. This is 5% below the baseline of 20% error rate of always voting 'Yea'. Additionally, it does not show signs of enormous overfitting or underfitting. Without tied weights, the training error dropped below 5% and the testing error spiked well above the naive baseline, so we concluded that this behavior was a positive side effect of using tied weights.

We experimented with regularization using both dropout (Hinton et al., 2012) and weight decay (adding a quadratic penalty to weights appearing in the model). As Figures 5 and 6 show, we get much better performance on binary features, which we were overfitting.

6.3. Comparison with Gerrish & Blei

To compare with Gerrish and Blei's paper, we re-created an experiment where they partitioned six 2-year congresses into 6 folds each for a total of 36 folds. They then performed k-fold cross validation on each congress separately. With this technique, their baseline accuracy of always voting 'Yea' had an accuracy of 85%, and their model had an accuracy of 90%. (Gerrish & Blei, 2011)

When re-creating this experiment, we found that the baseline was not 85%, but 83.3%. From that baseline, using count-features and parameters tuned on our validation set, we found we had 90.6% accuracy. Using a neural net to learn the structure that Gerrish & Blei explicitly modelled was at least equally as effective.

7. Network Analysis

Having achieved performance competitive with traditional approaches, we set out to see if the neural network learned structure like party lines and topics that Gerrish & Blei explicitly modelled. All the following experiments were done on a network that used word-count features with a single layer of 16 hidden units.

Our first experiment was visualizing the hidden unit activations for all the documents in a high dimensional space to see if partisan clusters are obvious. Figure 7 and 8 show our results with t-SNE for visualizing in two dimensions (Van der Maaten & Hinton, 2008). The votes are colored by Sander Levin and Roy Blunt's vote outcome on each bill to see party divides. We chose these two senators because they were to the far left and far right of Gerrish & Blei's ideal point model, so we believed we could see party separation clearly. We were pleased to see the two figures have a few clear clusters that change from green (voting 'Nay') to blue (voting 'Yea') between the senators. For good measure, we also tried a few other representatives who had long voting records and found that they colored the nodes about as predictably. Our model seems to be learning party separation clearly by projecting bills into a space where they are easily differentiated by partisan biases. There is also a larger cluster farther away that encompasses bills that are universally agreed upon. Also, we can see that 'Not Voting' bills are very evenly distributed amongst the clusters, showing that our objective function is ignoring them well.

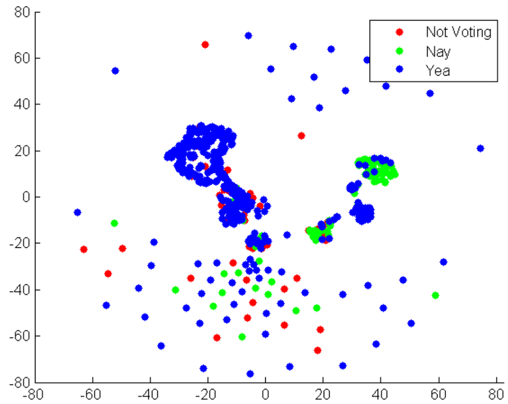

Figure 7. Sander Levin's Votes. We took the neural net response and colored them by how Sander Levin voted on them. We used t-SNE to visualize the high-dimensional features.

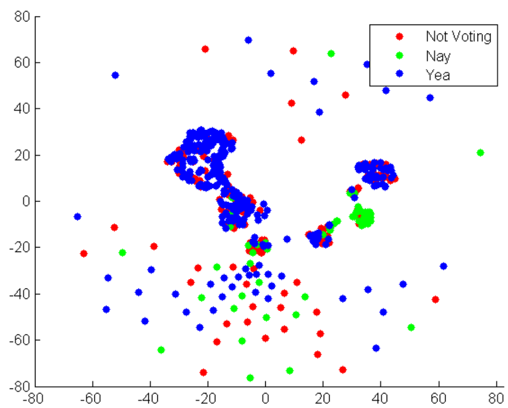

Figure 8. Roy Blunt's Votes. We took the neural net response and colored them by how Roy Blunt voted on them. We used t-SNE to visualize the high-dimensional features.

We tried to analyze the clusters from the titles of the bills in each cluster and had some trouble coming up with very obvious similarities between the bills. However, there were some patterns, like the large area of all blue seems to take many inconsequential bills (e.g. a bill to designate the square dance as the national folk dance).

8. Conclusion

We introduced a neural network architecture for predicting the outcomes of votes in the United States congress and analyzed the performance of different feature representations and regularization techniques. We also explored topic modelling, which turned out to be a less effective approach. Our results show that neural networks have similar performance to more complex models, while revealing structure about which topics affect voters in which ways.

9. Notes

This is a joint project with CS 224n. David is in both classes, but Tyler is only in 224n. Both sets of staff have confirmed this is fine. We used some neural network code written by Andrew Maas for a project Tyler is working on separately (we modified the output layers, dropout, and gradient descent). We also borrowed the ICML LaTeX style.

References

Clinton, J., Jackman, S., and Rivers, D. The statistical analysis of roll call data. *American Political Science Review*, 98(02):355–370, 2004.

Dumais, S.T. Latent semantic analysis. *Annual Review of Information Science and Technology*, 38(1):188–230, 2005.

Gerrish, S. and Blei, D.M. Predicting legislative roll calls from text. In *Proc. of ICML*, 2011.

Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R.R. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Jurafsky, Daniel and Martin, James H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000. ISBN 0130950696.

MATLAB. *version 7.10.0 (R2010a)*. The MathWorks Inc., Natick, Massachusetts, 2010.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Řehůřek, Radim and Sojka, Petr. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pp. 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

Tauberer, Joshua. Govtrack. `http://www.govtrack.us/data/us/`, 2012. Accessed: 11/16/2012.

Van der Maaten, L. and Hinton, G. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.