CS505 Final Project– Predicting Congressional Voting Based on Transcripts of Floor Deliberation

Timothy Evdokimov, Ezra Newman

December 2023

1 Abstract

Congress' full proceedings, including debate on the bills before them, is published daily in the Congressional Register. In theory, this includes spirited legislative debate on every bill that eventually passes or fails. We were interested to see if we could predict which votes would pass or fail just based on this data. Using word-vectorization and a support-vector machine classifier model, we were able to predict which votes would pass or fail with 90.3% accuracy. We also attempted to use a few other models, but the SVM was the one that yielded the best results. We were not able to predict the exact number of votes with any meaningful accuracy; our best technique's root-mean-square deviation was 52 votes, only slightly lower than the 56.7 vote standard deviation of yea votes in the dataset – in other words, about as good as guesswork.

2 Results

2.1 Problem Statement

We sought to analyze the transcripts of congressional floor debate in order to make predictions about (a) the outcome of the bill (passed or failed), and the number of votes it received in congress. While the outcome of congressional legislation is obviously influenced by considerably more factors than honest debate on the legislature floor, it is nonetheless interesting to see if it is possible to predict the outcome of a bill based solely on what congresspeople say about it during official debates. We were also interested in making some other predictions, like the size of a majority of a passed bill based on debate. Can one tell how controversial a bill is based on the congressional transcript?

2.2 Data

Our data comes from two sources. Firstly, a complete record of every congressional roll call vote is available here, from the VoteView dataset.

We obtain our transcripts from the Congressional Register, which allows you to (very slowly) download a full transcript from their API. We register 59 applications and use 59 API keys to speed up the download process. Even so, we had to run our scraper for several days to collect the full dataset.

Cleaning the data and merging the dataset of votes to bill transcripts also proved somewhat challenging. We fist attempted to filter based only on those votes that were for the final passage of a bill, i.e. filtering out all those votes on subcommittees, bill amendments, etc. However, this proved impractical because all but a few votes that passed this filter were passing, which makes an interesting observation – almost all bills that make it to a final floor vote are going to pass, those bills that fail almost universally don't make as far a floor vote. Note that modelling the passage-only data proved trivial, as the naive strategy of guessing that every bill passes proved 99% accurate, and totally useless. Only one bill ultimately failed to pass at all (the other eventually passed). It was H.R. 2842, designating the name of a post office building in Petaluma, CA. It needed a 2/3rds majority (290 votes) but only got 245.

Our system for merging the roll call votes dataset with the transcript one was to utilize the fact that the most transcript state, out loud, which bill they are discussing. We used the regex

```
re.compile(r"(H\.R\. \d+|H\. Res\. \d+|S\. \d+|S\. Res\. \d+)")
```

To isolate bills mentioned in the transcript, and transformed them to the format used in the VoteView dataset by removing spaces and punctuation, using the regex

```
re.sub(r'\W+', '', item['bill_number'])
```

To match each transcript to its entry in the VoteView dataset. You can find the merged dataset, consisting of all votes and transcripts from the 117^{th} congress (2021-2023) in the file `annotated_transcirpts_117C_v3.csv` It's worth noting that while the complete dataset of bills and transcript was less unbalanced than the dataset of only final passage votes, it was still fairly unbalanced, with about 3.35 times more passing votes than failing votes.

We also cleaned the contents of each transcript *after* the dataset merge, by removing punctuation, text enclosed in brackets (text that is information about the transcript, and not actually things said in congress, is in brackets in the "transcript" and not in a separate column in the dataset), and normalizing the whole thing to lowercase. We could potentially avoid a few additional normalization errors by parsing each transcript with an HTML parser instead of or in addition to a series of regexes, but this would have been prohibitively slow for

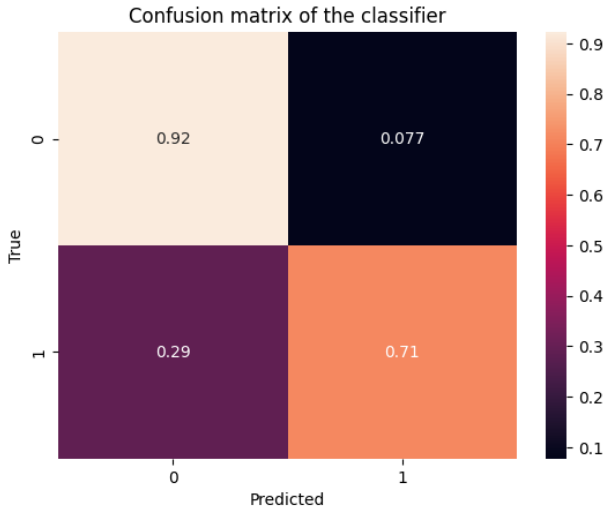
the thousands of records. The regex solution works well enough and only takes a couple of minutes.

2.3 Methodology

From there, we need to pair each transcript (that's about a specific piece of legislation) with a bill. We take a fairly naive approach to this, and only consider transcripts that explicitly mention a bill by number, eg. that contain phrases like H. R. 1234 or S. Res. 5678. This surely misses some number of transcripts, but is good enough– using only transcripts from 2022, we filter our dataset down to about 21 million words about 1852 bills.

From there, we calculate an embedding for each transcript. We split it into a list of words, and tried both the `GLoVE 100d` pretrained embeddings, and training our own, custom word2vec-based embedding model. The latter proved to have slightly better results, to the tune of a 3 percentage point accuracy improvement over the GLoVE model, but had considerably more computational overhead needed to get it working. It is worth noting that in both cases, simply averaging together the embeddings for the words in the transcript proved a very good model, and any attempts to improve upon this using PCA or SVD decomposition to try and extract the most meaningful features only did not work out – they invariably (a) took forever to run and (b) made the model considerably worse.

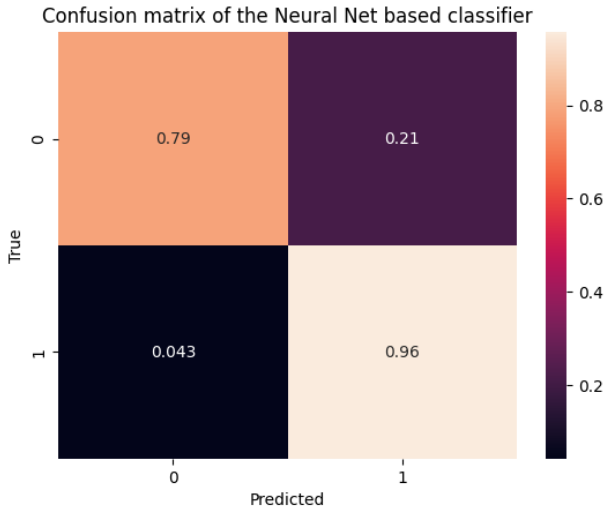
From there, we have a dataset of `(transcript_embedding, vote_result)` pairs and tested several models on them. The most effective ones proved to be a Support-Vector Machine Classifier and a Logistic Regression, which proved to be almost equally effective, with the SVM achieving a 90.3% accuracy and the Logistic Regression achieving 88.2%. Note the confusion matrix of the SVM classifier:



Which indicates that although the dataset was quite unbalanced to start with, the classifier can predict both passes and fails reasonably, which is a good sign considering that unbalanced accuracy is a common failure mode for classification models.

We also tried several other models for to classify passing vs failing bills, but they proved generally inaccurate. One initially promising candidate was the XGBoost model, which is generally quite powerful. However, it scored only 58% accurate in predicting passed bills, although it was 98% accurate in predicting failed ones. However, this imbalanced means it is ultimately an ineffective model. Stochastic gradient suffered a similar problem, scoring only 66% accurate in predicting passed bills, and again over 90% accurate in predicting failed ones.

We also attempted to use a simple neural net based model, trained on the same inputs. We tried a 5-layer dense ReLU model, and much like with the other models classification turned out to be much easier than vote prediction. Because failed votes are much more common than successfull ones, we scale the class weights proportionally. We achieved an 82% pass-fail prediction accuracy (89% without normalization) and a root mean squared error on vote prediction of 59.6.



We also attempted to predict the number of yea votes a bill would receive by treating the problem as a regression and evaluating our results using the root mean squared error. However, this proved unsuccessful. The best results we had– using an XGBoost regression model based on the word2vec embeddings– had an RMSE error of about 52, which is only slightly better than the standard deviation of the dataset, which is 56.7, so we do not consider this model to be successful.

2.4 Conclusion

Overall, we were successful in predicting results of a congressional bill based solely on the transcript of the congressional floor debate, as our best classifier had over 90% and balanced results. However, we were generally unsuccessful at predicting the exact number of votes a congressional bill gets, as our root mean squared error was so high as to be only a marginal improved over guessing.

2.5 Future Work

We were able to obtain data for 2010-2022, we limited our analysis to 2022 for practical reasons (the other data set is quite large and took a very long time to download, so we didn't have time to clean and process it). It would be interesting to apply the same techniques to the larger dataset.

We also considered trying to predict court decisions based on the arguments presented, but decided to focus on congress instead. This could be a good future project of a similar nature.

3 Code

Our code is available here: https://github.com/Timevdo/CS505-Final-Project

Our primary dataset of congressional transcripts takes between several hours to several days to scrape depending on scope. All our data and saved models can be accessed in this google drive folder

4 Contributions

Ezra Newman wrote the data collection code and cleaned up the datasets, including pairing transcripts with votes across our two datasets. He also built the Neural Network based modeling.

Timothy Evdokimov built the LinearSVC, SGDClassifier, and XGBRegressor models, and experimented with several others. He also found the VoteView dataset, and the wrote the word embedding code used to process the raw text data.

We both worked on the report.