Predicting Supreme Court Votes Through Conversational Dynamic Features

Tara Balakrishnan TARAGB@CS.STANFORD.EDU
Stanford University

Paula Kusumaputri PAULAKSP@CS.STANFORD.EDU
Stanford University

Luda Zhao LUDAZHAO@CS.STANFORD.EDU
Stanford University

Abstract

Data analysis of the political ecosystem often gives great clarity to opaque aspects of the political process. In this paper, we built a statistical machine learning model to predict the vote that a specific Supreme Court justice will cast solely given the oral argument transcript for a case. We extract a wide collection of features, including word features such as tf-idf scores, linguistic features such as interruption rates, and sentiment features such as positive/negative polarity scores in order to capture general conversational dynamics as well as more specific linguistic signals of agreement/disagreement. Using these features, we built a classifier that achieved results that exceed in some situations and come close to in others the state-of-the-art results in court decision predictions.

1. Introduction

Data analysis of the political ecosystem has been a trending topic in recent months, because it is incredibly valuable to citizens to understand the makeup and workings of the governing body. In part due to the negative rhetoric on the polarized political climate, we thought it would be interesting to analyze government operations from a non-partisan standpoint. As the premiere body in determining the Law of the Land, the Supreme Court of the U.S. is an influential but often opaque institution, which serves as a good target for our analysis. In particular, we wanted to understand if we can predict the vote that a specific Supreme Court justice will cast given solely their participation in the oral argument for a case. We will attempt to do this without access to data regarding how liberal or conservative a justice is regarded by pundits and the media, which is usually used in such predictions. Instead, we will be using a set of annotated transcripts of Justice conversations from past court cases, which we will be extracting useful word, linguistic, sentiment markers from in order to train our models. We have two main reasons for doing such: First, as a intriguing foray into general conversational analysis, this gives generalizable results for further work in other contexts where understanding conversations is critical. Secondly, we hope that our analysis provide a basis for understanding the inner workings of political discourse free of prior assumptions and biases.

Proceedings of the 33^{rd} International Conference on Machine Learning, New York, NY, USA, 2016. JMLR: W&CP volume 48.

2. Related Work & Literature Review

"Computational Analysis of the Conversational Dynamics of the United States Supreme Court." [Timothy W. Hawes][3]
In a 2009 paper, Hawes et.al. explored the dynamics of turn-taking in the United States Supreme Court. As one of the first papers on this topic, Hawes examined several computational tasks, among them the automatic labeling of turns with speakers, the follow-up behavior of justice on other justices questioning, and the tagging of linguistic markers in order to build predictive models. It is the last task that our project hopes to build upon, as Hawes explored various ways features can be extracted from conversational utterances that convey meaningful clues to whether or not the justice supports the case.

How To Read The Mind Of A Supreme Court Justice. [Oliver Roeder][6]

In an article on the popular data science blog FiveThirtyEight(www.fivethirtyeight.com), Oliver Roeder provides an accessible overview of the state-of-the-art in court case prediction models. First, this article provides key contex-

tual details regarding the general workings of the Supreme Court, which serves as great background information for our team to catch up on. More importantly, the article provides a detailed historical account of the different method researchers employed and discusses pros and cons of different methods. In particular, he discusses the work of two teams: Chris Nasrallah from UC Berkeley, who created CourtCast, a statistical model using pauses and interruptions to predict Supreme Court decisions, and the Marshall+ team, which built a model based on solely past contextual data. These two prior works inspired significant parts of our project.

"Echoes of power: Language Effects and Power Differences in Social Interaction" [Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang and Jon Kleinberg][2]
This paper explores how a linguistic analysis can determine power differentials between participants in a conversation. This paper describes how in this supreme court data set, it was possible to determine individuals in a position of power. This was accomplished by viewing the change in linguistic style of non-judge participants to match those of the judges. We hope to take some of the linguistic interpretations that this paper explores and turn them into features for our machine learning and natural language processing model.

3. Dataset

From a research group based in Cornell, we were able to obtain a collection of 204 court cases, over a 5 year period, totaling 51498 utterances making 50389 conversational exchanges, from 204 cases involving 11 Justices and 311 other participants, mostly lawyers arguing for the respondent and petitioner sides[5]. Case outcome and voting results for the 9 Justices for each case are included with each case. The text is mostly given in their raw forms, although dashes are included for when a person is cut off by someone else. Our task is to predict the vote that a specific Supreme Court justice will cast given their participation in the oral argument for a case.

Since there are significant differences between each particular justice in their pattern and frequency of speaking, we decided to train a separate model for each justice. As such, we partitioned our dataset into utterances spoken per justice. Additionally, for Justice Thomas, Justice Alito, and (Former) Justice Reinquist, we found that they either did not appear often enough in our collection of cases (or in the case of Justice Thomas, spoke too little), which mean our dataset did not contain enough utterances from them. Therefore we were unable to build a significant model regarding their conduct in oral arguments. Thus, we excluded these three Justices from the remainder of our analysis.

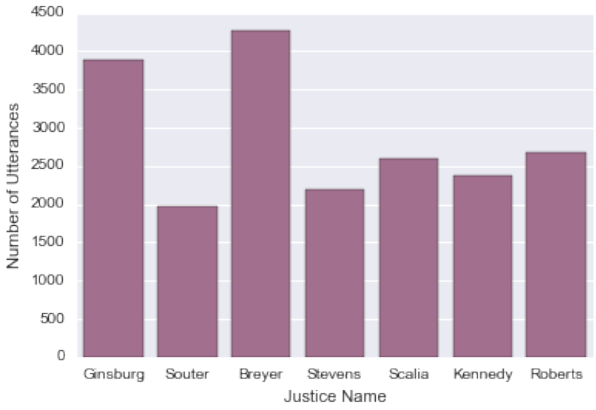

Figure 1. Number of Utterances per judge included

4. Feature Extraction

Our ability to predict votes is based on features within the oral argument alone. As mentioned above, we will not rely upon prior knowledge of a judge's political leanings of their past voting history in our model. Thus, it was essential that we extract features that are relevant in understanding how much a judge is leaning toward one side or the other.

Throughout the course of the project, we conducted an extensive search over many possible types of features we can extract. We discuss below in further detail the features that we extracted and the ones that were most predictive in our final model. The feature can be divided into four broad categories: basic linguistic markers, sentiment analysis, ideological bias, and textual similarity. Since we hope to detect which side a judge in leaning toward, we extracted all of our features for both sides of the case, once for all the utterances a judge made during the Respondent side of the argument, and once for utterances made during the Petitioner side of the argument.

4.1. Linguistic Markers

As a first step, inspired by prior work mentioned in the Related Work section, we wanted to extract basic linguistic and syntactic features in the utterances.

We started by including very simple metrics such as the average length of a justices utterances, average length of the utterances a justice responds to, and the average length of the response to the justice. Multiple prior work mentions the frequency of interruptions as an important proxy for whether or not a judge will ultimately vote for each side, and we decided to extract these features as well(both the percentage and the raw count). Interruptions are labeled in the dataset with dashes at the beginning or end of sentences. Again, we included counts of when a Judge interrupts the petitioner/respondent lawyer, and a lawyer inter-

rupts a judge, as the ratio of the two can indicate possible leanings toward one side versus the other(See Figure 2).

We also decided to tag the utterances from Stanford NLTK's POS tagger, which gives a part-of-speech tag to each words in a utterance. We then summed up the bigram of the POS tags and added it to our feature vector. Although it is not entirely clear what the direct relationship of such counts is to our prediction task, we feel the information it adds is useful for our purposes, as they convey particular linguistic styles that could discriminate between the two sides.

Lastly, we included the presence of "hedge" words in our feature vector. Hedge words, such as "apparently", "on the whole", and "to my knowledge",are mitigating words that lessens the harshness of an utterance. They are interesting linguistic markers for our purposes, given that they often convey doubt and uncertainty in the current stream of conversation by the judges. We obtained a list of 201 hedge words from Hylands (2005) list of hedges and extract features corresponding to the frequencies of such words in a judges utterances[8].

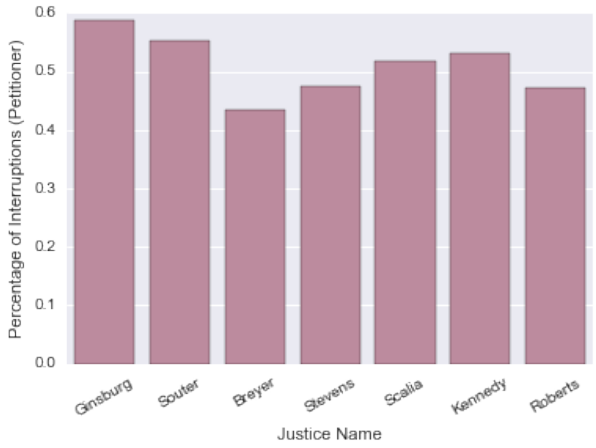

Figure 2. Percentage of interruption that were to the petitioner, per judge

4.2. Sentiment Analysis

While attempting to understand a judges leanings in a case by looking at utterances in an oral argument, it seems reasonable that identification of sentiment could provide a key insight into the judges viewpoint. While sentiment classification of a judges utterances appears to be extremely useful, it is also a very difficult problem to solve by itself.

Rather than building a sophisticated sentiment classifier by ourselves, which would be unfeasible given the time frame of this project, we used the VADER Sentiments classifier[9] to classify each utterance in a case with a positive, negative, or neutral sentiment score between 0 and 1, inclusive. The classifier is pre-trained on social media, which could lead to potential problems as the vernacular inside a courtroom is different from that on social media. Yet, we believed that the model contained enough relevant information to use as a feature extractor, which then could help predict a judges vote. Similar to the previous sections, we extract these scores for both the actual utterances of the judges, the utterance before, and the intermediate response utterance.

4.3. Ideological Basis

Ideological bias is very difficult to detect, even for humans - the task relies not only on political knowledge but also on the observers ability to pick up on subtle elements of language use. Although Supreme Court justice are nonpartisan, it is likely that their choice of words are informative, which we hope to detect through our raw text features.

We began with the traditional bag of words model and iterated upon that. This model retains multiplicity but disregards sentence structure, grammar, etc. To improve on the bag of words model, we built an n-gram extractor that stores some spatial information in addition to returning the frequency of the n-gram unit. We looked at a sequence values of n from n=1 (the bag of words features) up until n=3 (which encodes more information about sentence structure).

However, in particular, as n increases, we typically see an exponential increase in the number of n-grams present in the document. Additionally, it is definitely true that certain words or phrases are stronger predictors of ideological bias (or also sentiment) than others. Therefore, we need to reduce the number of n-gram features that we input into our classifier. To accomplish this, we used a two-step process. First we used the Stanford NLTK Package to stem the raw text in order to reduce the number of unique n-grams. Secondly, we ran recursive feature elimination on the features, which returns the words and phrases that are most predictive of a judges vote. The recursive feature elimination is a greedy algorithm which is similar to sequential backward selection. We were able to reduce bi-gram features from +100,000 unique bi-grams to roughly 1000 of the most informative bi-grams.

Expanding on word features with tf-idf scoring
We then experimented with tf-idf scoring on our bigram data. We used the stemmed bigrams in order to ensure that words were not unfairly given extra weight or that words were not decreased in weight because they appear in numerous forms (ie. discuss, discussed, discussing) and therefore a count of exact word matches wouldn't recognize that these are functionally the same word.

The tf-idf weight of each word or word-pair is a better statistical measure than simply word-count alone. This weight is then used to evaluate how important a word is to a single supreme court case in our entire database. For each bigram, we calculated a basic tf-idf score. We then iterated upon this baseline to include laplace smoothing and a logistic normalization in order to account for extremes in word counts and appearances in a single case. The equation we used is as follows:

$tf_{weight}(t,d) = 1 + log(f_{t,d})$ where $f_{t,d}$ is the number of times that term t occurs in document d when a particular justice says it. and $idf_{weight}(t,D) = log(1 + \frac{N}{n_t})$, where N is the number of cases and n_t is the total number of documents that contained that term with that particular justice speaking it. tf-idf is then the tf score divided by the idf score.

However, the next problem was that we had over 20,000 bigrams per justice which let to overfitting. Because we had so many thousands of features per justice, we needed to figure out a method of using feature elimination to only choose the most salient words. This way our model wouldn't perform as poorly due to the noise of having so many correlated and simultaneously so many unimportant features.

We built a Naive Bayes classifier which we used to understand the relative weight that a particular bigram's tf-idf score has towards both the respondent and the petitioner sides. We then took a fraction of the most predictive bigrams for both sides and used those features in our final feature vector.

The tf-idf score and feature elimination was done on a per-justice basis since we needed to understand how each justice operates on an individual level.

4.4. Texual Similarity

Accessing and understanding the richer linguistic context can significantly improve on our n-gram approach and give us a better understanding of speaker sentiment. We attempted to build a model of a judge's spoken similarity to the petitioner or respondent, but ultimately the model wasn't robust enough to include in the final results.

In our attempt to accomplish this task, we used the word2vec model that was created by Google researchers. Word2vec takes in a corpus of text from which it produces word embeddings; the corpus defines a multi-dimensional vector space from which each word is assigned a vector. The vectors are positioned such that words that share common contexts are located in close proximity to each other. Once we created the vectors, we then used the cosine similarity distance formula to get related words given an input word.

As a concrete example: from our dataset, the word revenues was frequently brought up in a case on the petitioner side. When we input that word into our classifier to get similar words from the oral argument, it would return a list which includes words such as expenditures, transportation, and so-called. Whats interesting about these results is that some of the similarity inferences are more obvious than others. We can clearly see that the word expenditures is related to the word revenues, by virtue of their definition. On the other hand, it is likely that a human may not see the connection between the word so-called and revenues. This indicates that the word2vec results can be useful as they may provide more insight into non-obvious relationships between words or phrases. For future work, we would hope to extend this by also computing phrase2vec from each utterance, since similarities between phrases may end up conveying more information than similarities between single words.

5. Models and Evaluation

5.1. Baseline

As a baseline, we built a simple naive Bayes classifier with unigram features on each Justice's utterances for each case to predict binary labels on whether or not that Justice voted for/against the case. We divided our current dataset into a training and a validation set with a 80/20 split with 5-fold cross validation.

See the first row of Table 1-3 for our baseline results.

5.2. Pipeline

To recap, here is the complete list of features that we extracted and put into our final model: word features:

1. word features:
   - Raw n-gram features ($1 <= n <= 3$), current utterance + utterance before/after
   - stemmed ngram features
   - Tf-idf transformed ngram features
2. sentiment features
   - average positive/negative/neutral scores, current utterance + utterance before/after
3. linguistic features (all features extracted for current utterance + utterance before/after):
   - average length
   - Presence of interruptions, as percentage
   - Raw count of interruptions
   - hedge word features
   - Raw count of part-of-Speech bigrams

These features are extracted twice (once for each side of the argument), per judge, per case, and converted into SciPys sparse CSR matrix format for easier computation. We then used the open source sklearn package to build and test our models[10].

5.3. Models

After running the classifier on our original unigram features, we continued to look at Naive Bayes in order to get a sense of how well we could classify judge's votes given all of the features we have extracted from the text thus far. We choose to start off with Naive Bayes as our initial model, since it performs surprisingly well "out of the box". The downside is that Naive Bayes cannot fit the data any better than the initial results, and the only way to improve on the model is to potentially increase the number of features that we input into it. Additionally, given the variety of the features we used for our analysis, it is reasonable to expect high degrees of possible co-correlation and collinearity between features. As Naives Bayes is based on applying Bayes' theorem with strong independence assumptions between the features, this model may not be the best to use in this situation.

We decided to try both Logistic Regression and SVM (with a linear kernel) as models for our prediction task. We performed parameter tuning on both models in order to get the most optimal results given our input feature vectors. Both are generalizable, robust models for binary classification, although there are trade-offs between the two. On one hand, logistic regression would seem to be better equipped to handle the very high level of noise. On the other hand, SVM with a soft-margin is adept at finding a reasonable separable hyperplane given large numbers of potentially co-correlated predictors. Irregardless, we applied both of them to our task and were able to obtain decent results(see Results section for more detail).

6. Evaluation

6.1. Oracle

Our oracle is a human - likely a law student or someone with some legal background who would be able to understand salient content. Additionally, since predicting justice votes also relies on detecting human sentiment, a human will be a good oracle to detect subtleties in the justice's tone. However, it was infeasible for the scope of this project to find a law expert to tag our entire dataset, and thus no explicit comparison could be performed. Citing prior literature, it is estimated that a knowledgeable expert could predict the outcome of a case around 75% - 80% [7]. However, the expert does have prior knowledge of the political leanings of the Supreme Court justices, which our model does not assume.

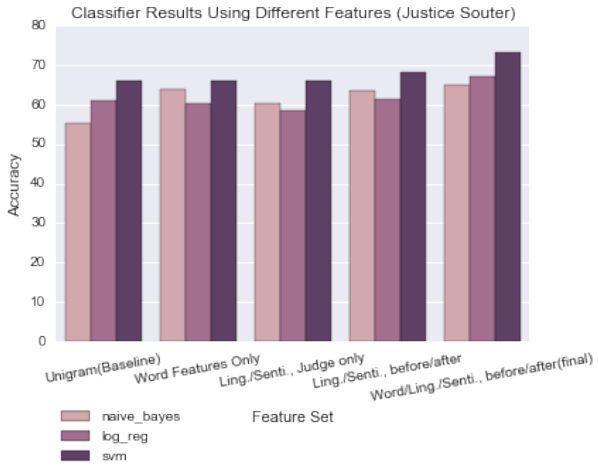

Figure 3. Accuracy for different feature types (Justice Souter

6.2. Metrics

When evaluating the success of our algorithm, we will look at the increase in accuracy from our baseline. The accuracies we obtain from our model will be an average of the cross-validation folds we perform for each model. Additionally, since even a human (our oracle) would be unable to get anywhere close to 100% accuracy, it would be interesting to compare against other published models which deals with predicting court cases.

7. Results and Discussion

7.1. Results

Since we have a collection of diverse features, we applied them stepwise to the problem to obtain intermediate results that allows us to investigate the relative impact of the different classes of features. First, we used only the word features(raw n-grams, tfidf scores)(Table 1). Secondly, we used only the linguistic + sentiment features for only the judges utterance(Table 2). Then, we added the linguistic + sentiment features from the utterance right before/after the judges utterance(Table 3). Finally, we combined all of our features(Table 4). At each step, we applied the models discussed in the previous section per judge to obtain our results.

In figure 3, we zoom in on the result of one justice, Justice Souter, and compare the results of different models applied on subsets of the feature we extracted. We see that although the general trend as we add more more features is an increase in prediction accuracy, there are significant variation between the different models.

Features./Justice	Ginsburg	Souter	Breyer	Stevens	Scalia	O'Connor	Kennedy	Roberts
Unigram (Baseline)	0.54357	0.60910	0.62711	0.53769	0.61537	0.72361	0.59222	0.65313
Word features Only	0.58819	0.60372	0.66711	0.56926	0.59598	0.73160	0.64686	0.68449
Ling./Senti., Judge only	0.61282	0.58333	0.53579	0.54794	0.56195	0.70261	0.59710	0.62177
Ling./Senti., before/after	0.66769	0.61462	0.68711	0.62927	0.65061	0.72208	0.67100	0.61581
Word/Ling./Senti., before/after (Final)	0.69149	0.67000	0.68158	0.57926	0.65024	0.72313	0.67188	0.70111

Table 1. Comparison of model accuracy using different features, Naives Bayes

Features./Justice	Ginsburg	Souter	Breyer	Stevens	Scalia	O'Connor	Kennedy	Roberts
Unigram (Baseline)	0.54357	0.60910	0.62711	0.53769	0.61537	0.72361	0.59222	0.65313
Word features Only	0.58819	0.60372	0.66711	0.56926	0.59598	0.73160	0.64686	0.68449
Ling./Senti., Judge only	0.61282	0.58333	0.53579	0.54794	0.56195	0.70261	0.59710	0.62177
Ling./Senti., before/after	0.66769	0.61462	0.68711	0.62927	0.65061	0.72208	0.67100	0.61581
Word/Ling./Senti., before/after (Final)	0.69149	0.67000	0.68158	0.57926	0.65024	0.72313	0.67188	0.70111

Table 2. Comparison of model accuracy using different features, Logistic Regression

Features./Justice	Ginsburg	Souter	Breyer	Stevens	Scalia	O'Connor	Kennedy	Roberts
Unigram (Baseline)	0.61771	0.66000	0.70211	0.62439	0.66512	0.76266	0.68163	0.71564
Word features Only	0.61295	0.66000	0.70211	0.58913	0.66512	0.76266	0.68163	0.73102
Ling./Senti., Judge only	0.61771	0.66000	0.70211	0.58874	0.64561	0.76266	0.68163	0.71564
Ling./Senti., before/after	0.61963	0.68153	0.70886	0.64968	0.65432	0.76250	0.66875	0.69231
Word/Ling./Senti., before/after (Final)	0.66871	0.73248	0.70886	0.66242	0.65432	0.76266	0.66875	0.69231

Table 3. Comparison of model accuracy using different features, SVM

7.2. Discussion

First thing we note is that the task of predicting votes for Supreme Court justices is a challenging task. Indeed, our baseline method of applying a unigram model results in around 60% accuracy for most of the judges, which is better than a 50% random guess baseline but not ideal.

As shown by the charts above, we were able to have success by applying sets of features incrementally to our models. First, by expanding the unigram model to include n-grams, doing feature reduction using RFE, and applying the tf-idf transformation on the raw counts, we were able to raise the accuracy of our model for almost all justices. For example, for Justice Ginsburg, the baseline model with Logistic Regression results in a 54% accuracy, not much more than random chance. The word features were able to raise that to 58% accuracy, which is significant.

Our linguistic and sentiment features did not work well initially when we extracted these features for only the judges utterances(see charts above), hovering around the 55% - 60% range. However, there is a marked improvement in prediction accuracy once we decided to include these features for utterances surrounding the judges utterance. For example, for Justice Breyer, extracting features only from his sentences results in a dismal 53% percentage accuracy(using Logistic Regression). Once the features from utterances directly before and after his utterances are added, however, the accuracy jumps to over 68% . This is reasonable when the natures of these features is considered: features such as interruptions and sentiment words are much more informative if we examine not only the judges reactions, but also what he/she responded to(sentence before) and how his/her response is received(sentence after). For Justice Breyer, it is likely that his responses did not have enough data to indicate strong like/dislike toward one side vs. the other, but the responses to his spoken words did.

Our model combining all features was clearly the best model we considered, achieving between 65% - 80% ac-

curacy for all judges. This is quite remarkable, given that our dataset is much smaller than most of the datasets used in other works, and given the inherent noisy nature of the prediction problem. One of the works we drew inspiration on was the FiveThirtyEight analysis called CourtCast, which were able to obtain over 70%(exact figures unknown) accuracy[6]. Based on these results, our model seems to perform at a comparable level to state-of-the-art results.

Comparing between different models, it is interesting to note that SVM performed much better than Logistic Regression when we used our lower quality features sets, whereas Logistic Regression quickly improved once we extracted better features from our dataset. This could be due to the forgiving nature of a soft-margin SVM, as well as the fact that only a few points are used to determine the boundary of a SVM, and thus the model is more useful even though many of the data points are not optimally distributed. Naive Bayes performed consistently lower than Logistic Regression and SVM; this is reasonable, given that the naive independence assumption made in a Naive Bayes model is very unlikely to be accurate in our context(many of our covariates have correlations with each other).

8. Conclusion

By using a large collection of semantic, syntactic, and advanced sentiment features, we were able to build a good models that predicts the vote of a Supreme Court justice with only the oral transcripts of proceedings. We were able to investigate various machine learning models and evaluate their effectiveness on different subset of our features. We were able to compare our results with both our baseline and various state-of-the-art results, including human performance, and we find that our model achieve results that are quite close to the latest results.

There are a few techniques that we did not have the time to try, but future researchers may find useful. First, since we used an off-the-shelf sentiment classifier, applying a model trained on a more relevant data could improve these features. Secondly, using word/phrase embeddings such as word2vec (which we played with but ultimately discarded for our final model) for detecting potential political leaning would further augment our work in word features. The key challenge would be to add the word embeddings to a feature vector in such a way that it doesn't take influence away from the other features. There has been exciting work done in this field, and we would expect our analysis with these features used properly would be significantly more robust.

As political processes in the U.S. becomes more polarized, it is critical that everyday citizens stay informed. Any work that allows greater transparency into the workings of institutions such as the Supreme Court provide value. We foresee that our models could be useful for future researcher and reporters to examine, without systematic biases, the workings of the Supreme Court and gain better understanding of the court process.

References

[1] http://www.nltk.org/modules/nltk/tag/stanford.html

[2] "Computational Analysis of the Conversational Dynamics of the United States Supreme Court." [Timothy W. Hawes]

[3] "Echoes of power: Language Effects and Power Differences in Social Interaction" [Cristian Danescu-Niculescu-Mizil, Lillian Lee, Bo Pang and Jon Kleinberg]

[4] "Elements of a computational model for multi-party discourse: The turn-taking behavior of Supreme Court justices." [Timothy Hawes, Jimmy Lin, and Philip Resnik]

[5] https://confluence.cornell.edu/display/llresearch/Supreme+Court+Dialogs+Corpus

[6] "How To Read The Mind Of A Supreme Court Justice" [Oliver Roeder] (http://fivethirtyeight.com/features/how-to-read-the-mind-of-a-supreme-court-justice/)

[7] "Uh, Robots Arent Better Than Humans At Predicting Supreme Court Decisions" [Oliver Roeder] (http://fivethirtyeight.com/datalab/uh-robots-arent-better-than-humans-at-predicting-supreme-court-decisions/)

[8] Hyland, Ken. "Hedges, boosters and lexical invisibility: Noticing modifiers in academic texts." Language Awareness 9.4 (2000): 179-197.

[9] Hutto, Clayton J., and Eric Gilbert. "Vader: A parsimonious rule-based model for sentiment analysis of social media text." Eighth International AAAI Conference on Weblogs and Social Media. 2014.

[10] https://docs.scipy.org/doc/scipy-0.18.1/reference/generated//Tscipy.sparse.csr_matrix.html