Mid Quarter Report

Predicting Verdict from Supreme Court Oral Arguments

Members: Maggie Chen, Carolyn Liu, Eujene Yum

I. Proposal Synopsis

We are interested in using the US Supreme Court oral arguments to predict case outcomes. We utilize data from Convokit (Cornell Conversational Analysis Toolkit) which contains approximately 1,700,000 utterances over 8,000 oral arguments transcripts from 7,700 cases. Our data is hierarchical in nature; each sentence is called an "utterance," and a group of unique "utterances" make up a "conversation." The number of conversations has an approximately 1 to 1 mapping with the number of cases. Our project goal is to make case-level predictions from 2015 - 2019.

II. Data Preprocessing

As summarized above, we are using 5 years of Supreme Court oral arguments from 2015 - 2019, which amounts to 320 cases to train and test on.

We dropped utterances that were spoken by impartial advocates and unknown speakers. We then removed stop words, and extracted only the stems of words using NLTK. The key features (excluding the actual conversation) that we look at include the following:

- *Speaker_ type* tells us whether the speaker is a judge or advocate
- *Side* informs us of the speaker's side
    - 0 (respondent), 1(petitioner), -1 (judge)
- *Cleaned_text* contains the pre-processed original text for each conversation ID, grouped by speaker type.
- *Win_side* is our target variable, which tells us whether the petitioning party won.
    - 0 (respondent wins), 1(petitioner wins)

III. Exploratory Data Analysis

We conducted Exploratory Data Analysis (EDA) before running our baseline models to better understand patterns within the corpus data. In this section we will show the result of our EDA in terms of case-level outcome distribution and top 100 trigrams.

As mentioned in the previous section, we have 320 cases in total across 5 years. Below is a graph displaying the distribution of cases, along with the win side over different years.

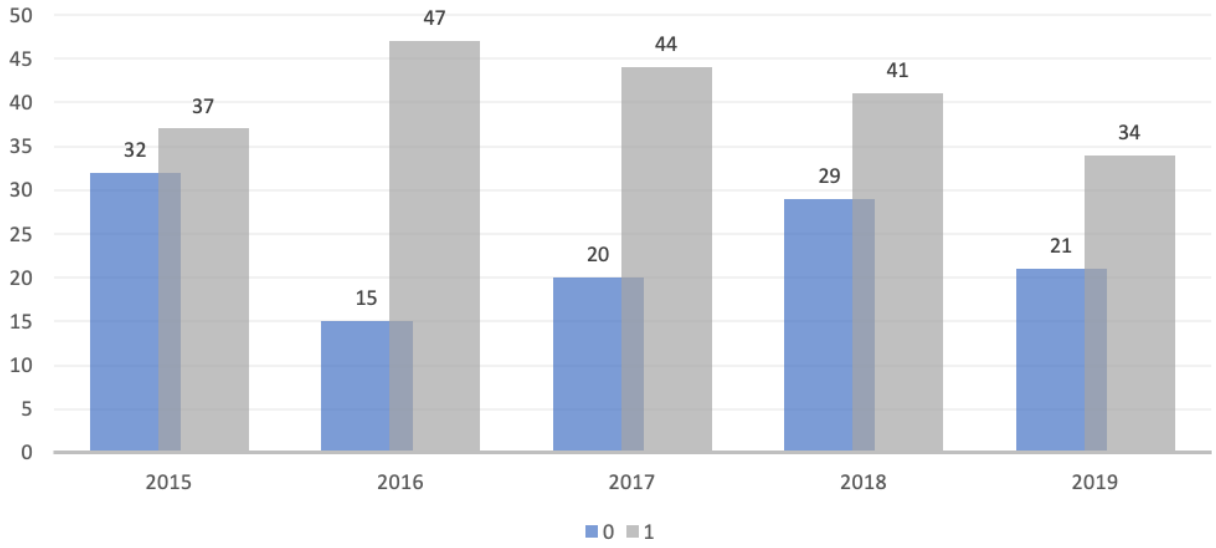

Figure 3.1 Distribution of cases from 2015-2019

Figure 3.1 shows that during 2015 - 2019, there are more 1's (petitioner wins) than 0's (respondent wins) as case outcomes. It indicates that our model will see more training data with 1 as outcome and might, consequently, tend to predict 1. This brings up the necessity of dealing with an imbalanced dataset when we are building the models.

Following the feedback from our project proposal, we also explored what the top 100 trigrams across all the 320 cases are. N-gram is a frequently used tool when it comes to natural language processing and it also facilitates building more advanced models like Convolutional Neural Networks. The word cloud below (Figure 3.2) provides a straightforward visual representation of word frequency for the texts of the 320 cases across 5 years. The accompanying table (Table 3.3)

gives a more detailed view of top trigrams extracted from the text, along with the occurrence counts.

Figure 3.2 Word Cloud

Trigram	Counts
'mr chief justice'	1286
‘chief justice pleas’	688
'justice pleas court'	685
'thank counsel mr'	426
'thank mr chief'	415
'll hear argument'	292
'case submitted ll'	289
'submitted ll hear'	289
'thank thank counsel'	254
'hear argument morn'	188

Table 3.3 Showcase of top trigrams

As expected, key words such as “chief”, justice”, “thank” have frequent occurrences and these words also appear in the top trigrams. However, simply through EDA and looking at the top trigrams presented, we cannot conclusively determine the extent to which the most frequently used trigrams impact the prediction of case-level outcomes. We decided to use a baseline simple Neural Network model to get a sense of model performance with N-grams as inputs. Extending the number of top trigrams or splitting the top trigrams between different speakers (i.e. respondent and petitioner) could also be an option for future work.

IV. Baseline Model I: Logistic Regression

To run a logistic regression, we transformed the text documents into a matrix of token counts. We split the training and testing data in an 80-20 split and performed hyperparameter tuning for the regularization term.

Our classification report below (Table 4.1) shows that the logistic regression model gives quite satisfactory results in terms of accuracy, precision, recall and f1-score. The final testing accuracy is around 86%. Noticeably, the model is performing slightly better on predicting 1 (petitioner wins) versus predicting 0 (respondent wins). This aligns with the imbalanced dataset issue as discussed in Section III during exploratory data analysis.

	precision	recall	f1-score
Petitioner (1)	0.87	0.92	0.89
Respondent (0)	0.86	0.76	0.81
accuracy			0.86

Table 4.1 Classification report for Logistic Regression model

V. Baseline Model II: Simple Neural Network

We use the top 100 trigrams as inputs and run a simple neural network with one hidden layer. For each case, we obtained all the trigrams associated with utterances under that case. We then observed whether any of the top 100 trigrams exists within the obtained trigrams of that case. The result is a 320 * 100 matrix, where 320 corresponds to the number of cases, and 100 corresponds to the number of top trigrams. Within this matrix, each cell indicates if a specific top trigram exists within a specific case (coded 1 if yes, 0 otherwise).

We adopted a simple architecture for our baseline Neural Network model. We only incorporated one hidden layer and used Tanh() as the activation function. Softmax is used to acquire our final probability prediction.

Consistent with the Logistic Regression model, we also split 80% of data into a training dataset. Additionally, 10% of data is used as a validation set. We observed the model's accuracy on validation set after each of the 20 epochs, as shown in Figure 5.1. We can see that the accuracy is fluctuating, which means that some proportion of examples is classified randomly, as the number of correct random guesses always fluctuate. As shown in the classification report in Table 5.2, testing accuracy is 47%, which is not satisfactory and indicates that the model is performing random guesses rather than learning from the training data and making good predictions.

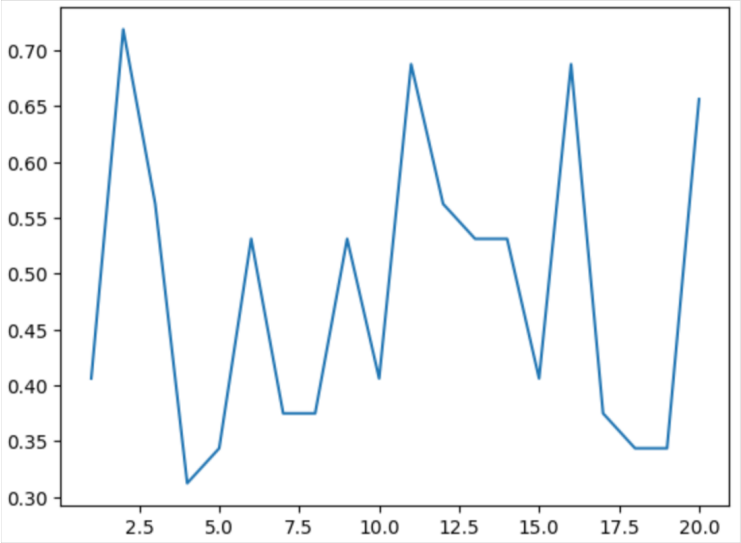

Figure 5.1 Validation accuracy of simple Neural Network model

	precision	recall	f1-score
Petitioner (1)	0.50	0.53	0.51
Respondent (0)	0.43	0.40	0.41
accuracy			0.47

Table 5.2 Classification report for simple Neural Network model

VI. Baseline Model III: Random Forest

As with the Simple Neural Network model in the previous section, we used a one-hot encoded vector of the top 100 trigrams for each conversation as the features. We chose a random forest model for its accuracy, efficiency and interpretability. To train the model, we split the training and testing data in an 80-20 split and performed hyperparameter tuning for the number of estimators, maximum depth, minimum number of samples required to split a node/at each leaf node and whether bootstrap samples are used. Our test accuracy came out to 61%, leaving room for improvement.

The classification report in Table 6.1 shows that our random forest model suffers even more from the imbalanced dataset. Specifically, the recall (0.12) and f1-score (0.19) are significantly low for predicting 0 (when respondent wins).

	precision	recall	f1-score
Petitioner (1)	0.63	0.90	0.74
Respondent (0)	0.43	0.12	0.19
accuracy			0.61

Table 6.1 Classification report for Random Forest model

Next Steps

As for the next step, we would mainly focus on how to improve the current models we have and how to include more advanced models. We plan to improve the tree-based model and simple neural network model which had worse performance than the logistic regression model. Since both models have top 100 trigrams as the input, we would want to explore if including more trigrams will help and discuss with TAs and the professor to see if using trigrams as inputs is indeed a good enough approach for our project. Apart from the N-grams, we will also deal with the imbalanced dataset issues, discussed in Sections II, III and VI. We observe that especially for the tree-based model, the imbalanced dataset is significantly impacting the model performance on predicting 0 (when respondent wins). Additionally, we would also like to encode additional features like *speaker_type* and *side* rather than just concatenating texts from different speaker types and sides to make our models more robust.

Apart from our existing models, we received advice to use some advanced models, specifically transformer-based models, BERT, CNN, and LSTM. We will confirm with TAs and the professor on which model would work best before moving forward with more advanced models. Our current plan is to attempt LSTM as our next model because unlike models such as CNN, which focus on local patterns, LSTM focuses more on longer dependencies. Supreme Court discourse is also expected to have longer dependencies, which is why we suspect that LSTM may work better for our project.