Supreme Court Verdict Prediction

CAPP 30255—Advanced Machine Learning

Eujene Yum
eujeneyum@uchicago.edu

Carolyn Liu
crliu@uchicago.edu

Maggie Chen
gchen22@uchicago.edu

May 22, 2023

Abstract

As the highest court in the U.S., the Supreme Court hears cases that impact the country's social, political and economic future. The outcomes of these decisions trickle down and impact our day to day lives as well. In this paper, we aim to predict the outcome of Supreme Court hearings solely using oral arguments of judges and advocates. We process a total of 667 cases from 2010 - 2019. We fine-tune and deploy a total of 4 models which include logistic regression, simple neural network, multiple tree-based models, and BERT. Through many iterations of model input, and loss functions, we find that the BERT model provides us with the highest accuracy of 69%. All relevant code is stored in our Github with a clear summary and table of contents in the README.

1 Introduction

The impact of US Supreme Court case outcomes reaches far and wide, significantly affecting the lives of Americans. In light of recent decisions, such as the reversal of Roe v. Wade and rulings on the availability of abortion pills at the state level, this topic holds particular relevance. The vast amounts of data available via transcribed oral arguments make this topic a great use case for NLP models since they can consume massive amounts of documents that humans cannot efficiently comprehend. Leveraging data sourced from Convokit (Cornell Conversational Analysis Toolkit), we focus our examination on a decade's worth of cases spanning from 2010 to 2019.

2 Literature Review

There has been significant work done by academics to determine how to best make use of oral arguments to garner useful insights. The following papers utilize different mechanisms and model inputs to answer questions around court cases.

2.1 Dickinson (2019)

Dickinson (2019) retrieved oral argument transcripts of nearly 1,400 cases argued and decided from 1998 to 2015 from the Supreme Court's website. The transcripts were paired with case outcomes from SCDB (Supreme Court Database) as labels. The research adopted four categories of features: question count features, question chronology features, question sentiment features and n-gram features. Stanford CoreNLP sentiment annotator was used to get the "question sentiment features".

An exploratory analysis of the features showed that almost all the Justices ask longer questions to the party they finally vote against, and their tone is less friendly and they ask more follow-up questions as well. Given each Justice has a different questioning style, the research created separate feature matrices for each Justice and a SVM model (Support Vector Machine) was built upon these feature matrices. Since every n-gram is represented as a column, the feature matrix is extremely wide and SVM was ideal due it its effectiveness in high dimensional feature spaces. The study uses a ten-fold cross validation throughout the model training process. The SVD model, with all four categories of features, correctly predicted individual Justice votes in 73% of the cases, with the highest accuracy of around 86.8% for Justice Rehnquist and the lowest of around 52.3% for Justice Sotomayor.

Dickinson also examined the performance of each category of features. Though the category of "n-gram features" accounted for the vast majority of features, each category generally added some incremental predictive accuracy beyond the n-grams alone. This research paper includes features such as "question count features" which are not typical in other papers, and provides us with alternative features that can be adopted in our project. One caveat of this paper is that the feature matrix may

turn out to have more columns than rows, given each n-gram is represented as a column. Since p¿¿n violates the common assumption in Machine Learning and potentially overfit, we would try to avoid it in our own project.

2.2 Aletras et al. (2016)

Aletras et al. (2016) used NLP techniques to predict whether specific articles of the European Convention on Human Rights have been violated as decided by the European Court of Human Rights, given text extracted from the case. The Court judgments have a distinctive structure: procedure, the facts (case circumstance, relevant laws), etc.

The authors obtained data from HUDOC, parsed for English cases, then achieved a balance between classes. Text was extracted using regular expressions and excluded any information that directly pertain to the outcome of the case.

For each section of the case, the authors derived n-grams features and topics from the text extracted from each section. For each set of cases, the top-2000 most frequent N-grams where $N \in 1, 2, 3, 4$ were computed. Each feature represents the normalized frequency of a particular n-gram in a case or a section of a case. They then "extracted n-gram features for the Procedure (Procedure), Circumstances (Circumstances), Facts (Facts), Relevant Law (Relevant Law), Law (Law) and the Full case (Full) respectively (note that the representations of the Facts is obtained by taking the mean vector of Circumstances and Relevant Law)". Topics for each article were then created by clustering together n-grams that are semantically similar by leveraging the distributional hypothesis suggesting that similar words appear in similar contexts.

The authors used SVM with a linear kernel in order to identify important features that are indicative of each class by looking at the weight learned for each feature to predict the Court's decisions. Additionally, they tuned the regularization parameter C using grid search. To evaluate model performance, the authors computed the mean accuracy obtained by 10-fold cross-validation. The author's achieved ¡ 70% classification accuracy for most different feature types across articles, with 'Circumstances' n-grams giving the highest accuracy for Articles 6 & 8. We can potentially adapt the authors' use of n-gram features for each section of the case to our project as well as using grid search to tune our models.

2.3 DM (2017)

DM (2017) constructed a model designed to predict the behavior of the Supreme Court of the United States in a generalized, out-of-sample context. They developed a time-evolving random forest classifier that leverages unique feature engineering to predict more than 240,000 justice votes and 28,000 cases outcomes over nearly two centuries (1816-2015). They achieved 70.2% accuracy at the

case outcome level and 71.9% at the justice vote level. Over the past century, they outperformed an in-sample optimized null model by nearly 5%. Their model is distinctive because it can be applied out-of-sample to the entire past and future of the Court, rather than a single term.

They use data from the Supreme Court Database, and each case contains as many as 240 variables, many of which are categorical. The target variable can have 3 values – reverse, affirm, or other. The paper only includes cases that reverse or affirm. They also convert categorical variables into multiple binary variables.

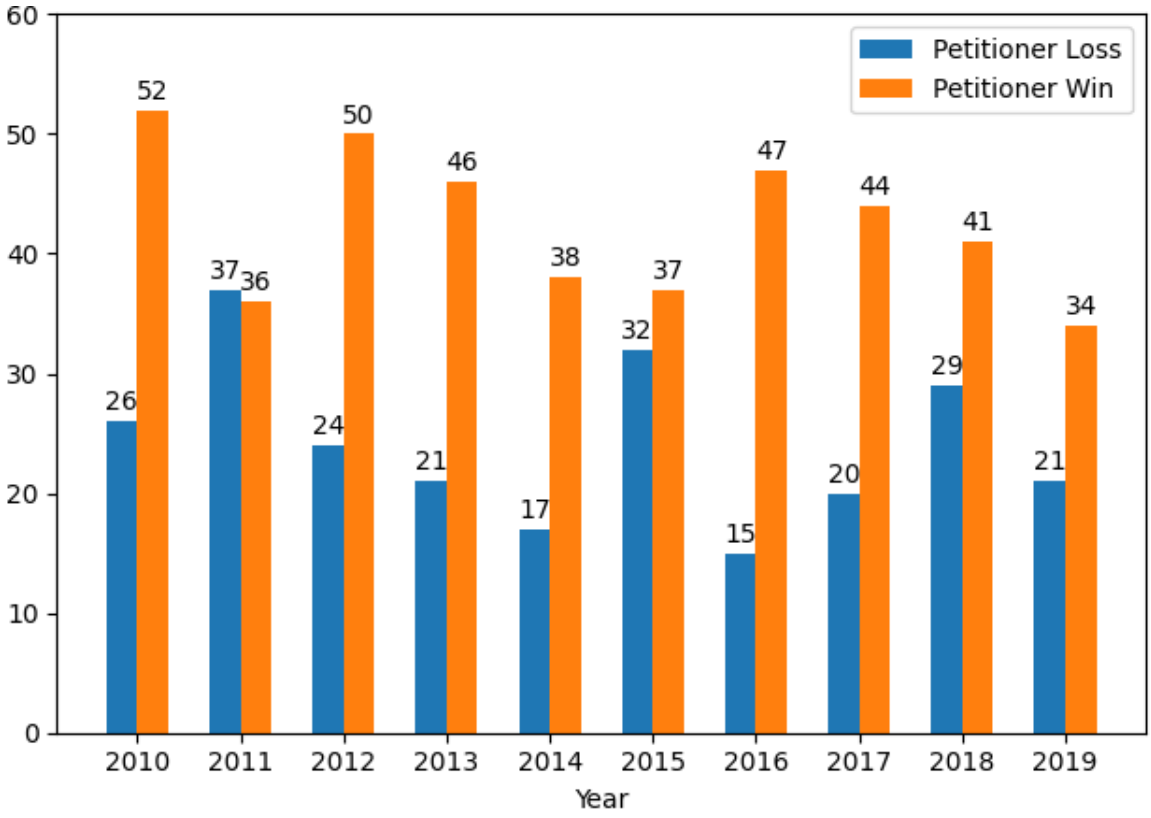
They focused on building a general model—one that could stand the test of time across many justices and many distinct social, political and economic periods. They constructed their model using a random forest classifier, which does not generally require pre-processing. They mention that one drawback of the scikit-learn implementation of random forests relative to alternatives like xgboost, is that it handles missing data by mapping missing values to a separate "missing" indicator column during encoding, but a historical mean imputation is used sometimes. The paper states that random forests are very effective and have proven to outperform support vector machines and multi-layer perceptron models. They also experiment with "growing" forests where they add onto the existing forest with new terms and a "fresh" forest model where new forests are created each term with a number of trees selected by cross-validated hyperparameter search. The "growing" approach allowed for faster simulation time and stable prediction, and cross-validation and hyperparameter search did not have a noticeable impact on accuracy over "default" random forest configurations. A random forest model may be worth implementing to observe whether it achieves reasonable accuracy for the ten years of court cases we are analyzing.

3 Data

We use data from Convokit which contains approximately 1,700,000 utterances over 8,000 oral arguments transcripts from 7,700 cases. We limit our analysis to 10 years of data ranging from 2010 - 2019, with a total of 667 cases.

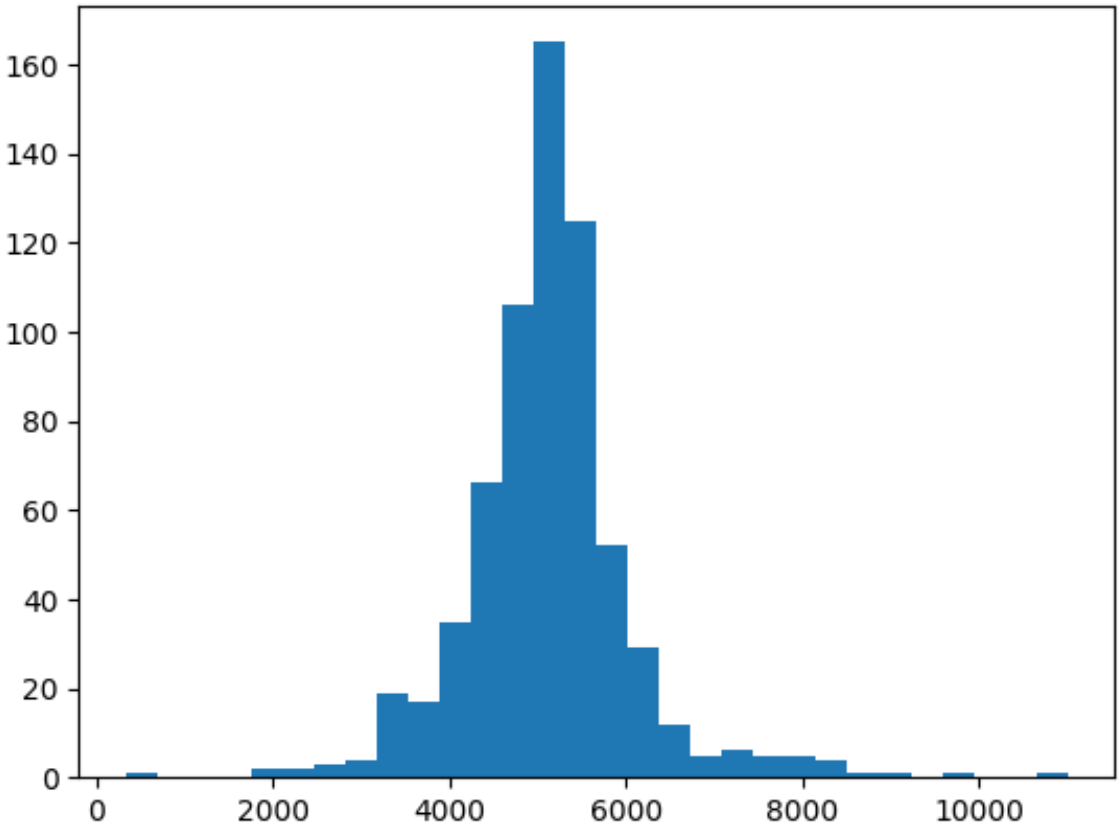
As revealed in Figure 1, there are more 1's (petitioner win) than 0's (petitioner loss) in our case outcomes. The imbalanced nature of our dataset means that our model will see more training data with 1 as the outcome and will tend to predict 1. We balance our training data by oversampling the minority class.

Figure 1: Verdict Count per Year



The length of the oral arguments in the 667 cases follows a normal distribution, with a mean of 5000 characters, as shown in Figure 2. This works well for our baseline models (logistic regression, simple neural network, and tree-based models), but requires modification for BERT, as the maximum number of input tokens that it can take is 512. We will expand upon this process in the model section.

Figure 2: Distribution of Oral Argument Length



Before diving into modeling, we preprocessed our dataset. First, we only kept utterances that were spoken by judges and advocates. We then removed stop words, and extracted only the stems of words using the NLTK package. Our input to the model is the preprocessed text and our target variable *win_side*, where 1 signifies petitioner win and 0 signifies petitioner loss. We create an 80-20 split to create training and testing data.

4 Models and Results

In total, we fit four different models in an attempt to predict Supreme Court verdicts based on case oral arguments. Our baseline approaches incorporate logistic regression, simple neural network, and a tree-based model. We also adopted a pre-trained BERT model to further explore ways to improve our accuracy via bidirectional encoding. In the Model Selection Process section, we discuss the various attempts made to improve model outcome and the Final Models section expands upon the final four models.

4.1 Model Selection Process

4.1.1 Model Inputs Selection

We tried two different methods to extract features from the text. The first was using a count vectorizer, which took the text from each conversation and turned them into unigram token counts. The second was using trigrams instead. For each court 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 was a 320 * 100 matrix, where 320 corresponds to the number of cases (we were using only five years of data from 2014-2019 for midterm checkpoint), 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 (1 if yes, 0 otherwise). We decided to use the unigram token counts as our model inputs going forward since they gave us better results than the one-hot encoded vector of top 100 trigrams.

Additionally, we originally separated utterances from the same case by speaker type - justice or advocate. Advocates were further separated by the role they played (e.g., argued for the petitioner and vice versa). Thus for each case, our dataframe had a total of three observations per case, separated by speaker types. This resulted in an inflated and inaccurate test accuracy when evaluating our model. To remedy this problem, we tried three variations: (1) using only justice utterances; (2) concatenating all advocate utterances regardless of the role they played and keeping only advocate utterances; and (3) concatenating all utterances for a case regardless of speaker type. The results from the first two variations were lower than having all utterances from a case, suggesting that including all utterances plays an important role in determining case outcome.

4.1.2 Model Selection

Due to the unbalanced nature of our data, we were advised to try a support vector machine (SVM) model since SVMs might handle the imbalance more elegantly. We used the support vector classifier from scikit-learn and performed hyperparameter turning via grid search on the regularization term. Despite the fact that SVMs are supposed to handle unbalanced datasets better, the model predicted the majority class most of the time.

In the mid-quarter report, we used a random forest model with a one-hot encoded vector of the top 100 trigrams (see Section 4.1.1) as the inputs. After switching to a matrix of unigram tokens, balancing our data, and performing hyperparameter tuning to find the best model, random forest predicted the majority class for all but two test cases. This led us to try a gradient boosted model (XGBoost or LightGBM). Both XGBoost and LightGBM are acceptable models, but XGBoost is comparatively slower than LightGBM, leading us to choose LightGBM as our tree-based model below.

4.2 Final Models and Results

4.2.1 Logistic Regression

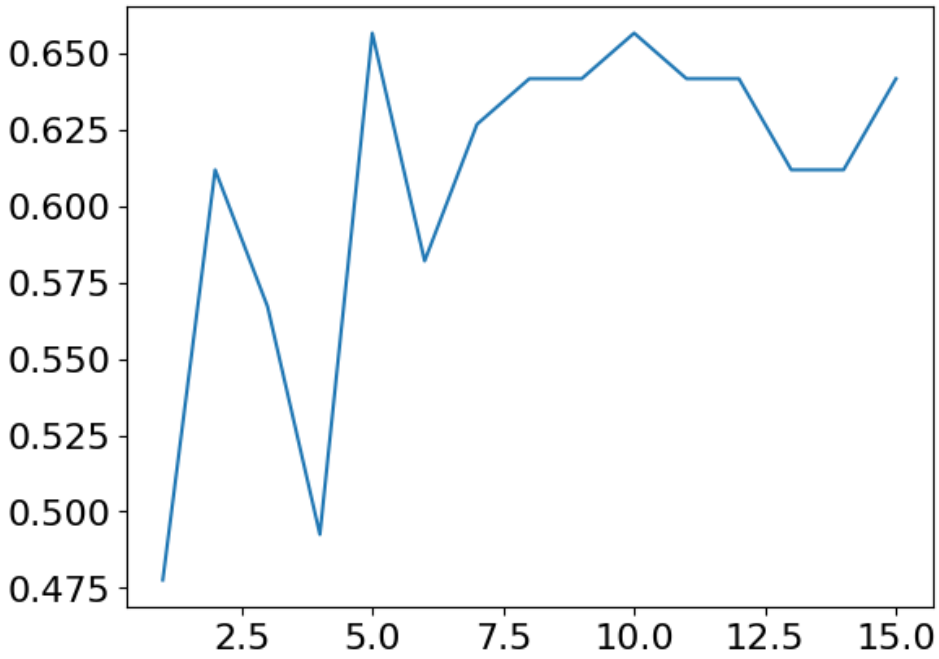
We used a matrix of unigram token counts (via CountVectorizer) in order to perform logistic regression. We included utterances spoken by both the judge and the advocates. We split the training and testing data in an 80-20 split and oversampled from the data with 0 labels so that we could obtain a balanced training set. We then performed hyperparameter tuning for the regularization parameter before applying the model on our test data to ensure that we have the best possible logistic regression model. When we evaluated our model on the test data, we got an accuracy of 61.9% (see Figure 4).

4.2.2 Simple Neural Network

We used the same unigrams as above as our model input and run a simple neural network with two hidden layers. We adopted a simple architecture for our baseline Neural Network model. We incorporated two hidden layers and used Tanh() and ReLU() as our activation functions. Softmax is used to acquire our final probability prediction.

Consistent with the Logistic Regression model, we also used 80 percent of the data as our training dataset. We observed the model's accuracy on the test set after each of the 15 epochs, as shown in Figure 3.

Figure 3: Accuracy of Neural Net over Epochs



We can see that the accuracy is fluctuating in the beginning, which means that some propor-

tion of examples is classified randomly, as the number of correct random guesses always fluctuate. However, fluctuation of the accuracy seems to plateau around about 8 epochs. As shown in the classification report in Figure 4, testing accuracy is 60.0%, which indicates that there is still much space of improvement for this simple neural network model.

4.2.3 Tree-Based Model

We chose a gradient boosting model, LightGBM, for its general accuracy, efficiency and interpretability. As with the models described in the previous sections, we used a unigram count vectorizer matrix for each conversation as the input. To train the model, we split the training and testing data in an 80-20 split and balanced our training set to prevent the model from being biased towards the majority class. To fine-tune our model, we performed hyperparameter tuning via randomized search cross validation for the regularization term, number of leaves, maximum depth, learning rate, and boosting type. Our test accuracy is 57.0%.

Figure 4: Baseline Model Results

	Precision	Recall	F1
Logistic Regression			
Respondent (0)	0.67	0.76	0.71
Petitioner (1)	0.5	0.39	0.44
Accuracy			***0.62***
Neural Network			
Respondent (0)	0.61	0.35	0.45
Petitioner (1)	0.59	0.81	0.68
Accuracy			***0.60***
LightGBM			
Respondent (0)	0.63	0.75	0.68
Petitioner (1)	0.40	0.27	0.33
Accuracy			***0.57***

4.2.4 BERT

Apart from the baseline models, we also used advanced models such as the Huggingface pre-trained BERT model. BERT can better capture bidirectional textual understanding across the text, which may improve model performance. The specific pre-trained model we adopted is ”bert-base-uncased”, which is a pre-trained model on English language using a masked language modeling (MLM) objective.

BERT can take a maximum of 512 input tokens. However, we learned during our exploratory data analysis that for each court case, the average length of oral arguments is about 5000 words. This is much longer than what BERT is able to handle. In order to tackle this issue, we split the text for each case into different chunks, with each chunk containing less than 500 words. We then feed the data into the bert-base-uncased pre-trained model and fine-tune the model on our training data.

We observed some overfitting issues in our first attempt, which is due to the fact that bert-base-uncased has 110 million parameters and the number of parameters far outweighs the number of tokens we feed into the model. To address this issue, we added a dropout layer and also changed the dropout rate from 0.1 to 0.5 in the pre-trained model configurations. We also experimented with different loss functions. BCE loss was used because we have a binary classification problem. We also attempted Hinge loss so that the model can learn more discriminative representations and improve classification performance. However, we chose BCE loss in our final BERT model as BCE loss gave us better model performance than Hinge loss. Our final testing accuracy came out to 69%, which is higher than all of our baseline models and pretty close to the testing accuracy we saw in past literature.

5 Conclusion and Next Steps

In this project, we run a total of 3 baseline models (logistic regression, tree based model, neural network model) and many iterations of BERT to predict Supreme Court verdicts. Out of the four models, BERT gives the best testing accuracy and quite close to what is observed in past literature. This is most likely due to the fact that BERT takes care of bidirectional contextual understanding across the text while other models do not.

A big portion of our time was spent making sure that the dataset used as the model input was preprocessed properly. In order to deal with the imbalanced nature of our dataset, we used Random Oversampler to oversample from the minority class. Oversampling proved useful with our baseline models but we see a drop in testing accuracy when oversampled data is fed into BERT. This may signal that pre-trained BERT models may already be good enough to handle imbalanced datasets.

Potential next steps for this project would include bringing in additional features such as divisiveness of topic, measurement of public attention, and political leanings of the justices. This may allow for a more contextualized and nuanced way to predict verdict outcomes. Additionally, we could build our own embeddings based on the the Supreme Court corpus, which could better represent the text data from oral arguments and achieve higher model accuracy.

References

Aletras, Nikolaos, Dimitrios Tsarapatsanis, Daniel Preoţiuc-Pietro, and Vasileios Lampos. "Predicting judicial decisions of the European Court of Human Rights: A natural language processing perspective." PeerJ Computer Science 2 (2016): e93.

Dickinson, Gregory M., A Computational Analysis of Oral Argument in the Supreme Court (July 3, 2019). 28 Cornell J.L. & Pub. Pol'y 449 (2019), Available at SSRN: https://ssrn.com/abstract=3198401

Katz DM, Bommarito MJ 2nd, Blackman J. A general approach for predicting the behavior of the Supreme Court of the United States. PLoS One. 2017 Apr 12;12(4):e0174698. doi: 10.1371/journal.pone.0174698. PMID: 28403140; PMCID: PMC5389610.

Appendix: Reflections

NLP was a new topic for all of us, so this project was on the steeper side in terms of learning curve. We had to learn different ways to manipulate text in order to properly tokenize the text to create matrices as model inputs. We already had experience with scikit-learn from previous machine learning courses and knowledge of traditional machine learning models such as logistic regression and tree-based models, but have not used raw text as sole features of a model. We spent a decent time sifting through past literature to see what other academics have done with Supreme Court oral arguments in order to gain insight into which model might work for best for us. However, the biggest portion of our time was spent on preprocessing the data in different ways (unigrams vs. top 100 trigrams) and fine-tuning our BERT model. We consulted the internet, our TA, and Professor Chaudhury to improve upon our BERT models and tried changing the loss functions, the input data structure, and model configurations. We divided ownership of the baseline models (Eujene - logistic regression, Maggie - Simple NN, Carolyn - tree-based models), but most of our work was done in person together.