NLP on Supreme Court Opinions: Authorship prediction, multiauthorship detection, and document similarity

Introduction

For this project, I took a look at U.S. Supreme Court (SCOTUS) opinions. Specifically, I was interested in *majority* opinions, the opinion expressing the final holding of the Court. These documents can have powerful, society-wide effects that can last centuries--or sometimes only a generation. Additionally, the language found in Supreme Court opinions is highly structured, both syntactically (as to introductory and concluding phrases, citations, quotes, arcane Latin, etc.) but also semantically, as they express arguments that have a specific, rigorous form. This rich, multi-level structure made me eager to analyze these texts from a data science perspective.

Much of this project was motivated by work found in "Detecting multiple authorship of United States Supreme Court legal decisions using function words," Rosenthal Yoon 2011, available here.

All modeling can be found in the main notebook. All work is done in python; I used sklearn's implementation for tf-idf vectorizing and the Naive Bayes classifier, and gensim's implementation of doc2vec.

Problem statements

1. Predict the author of Supreme Court opinion given its text
2. Characterize which Justices rely more on their law clerks for opinion text
3. Represent opinions as vectors and compute similarities

Data Acquisition

To retrieve the data files, I used software written by Rosenthal (available here) with some modifications to work on Mac OS's Terminal application. The files with these edits are available in the scraping folder.

I included volumes 479 to present, which covers roughly 30 years and ~2400 opinions. For preprocessing I used regex to remove ellipses, sklearn's standard tokenizer for tf-idf, and gensim's preprocessor for doc2vec.

Author prediction

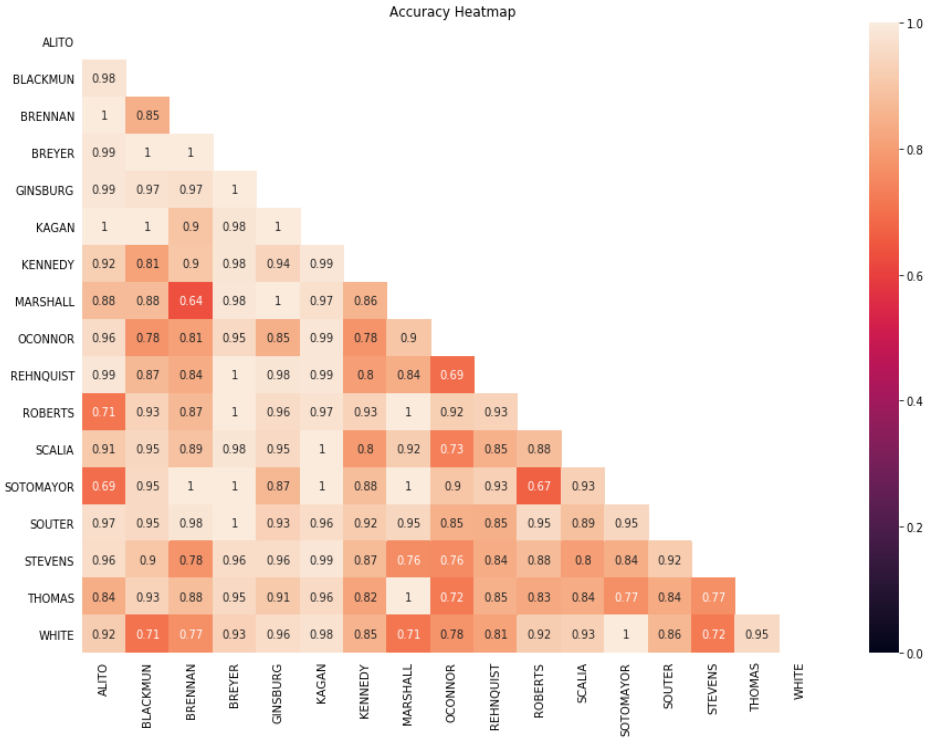
Analysis

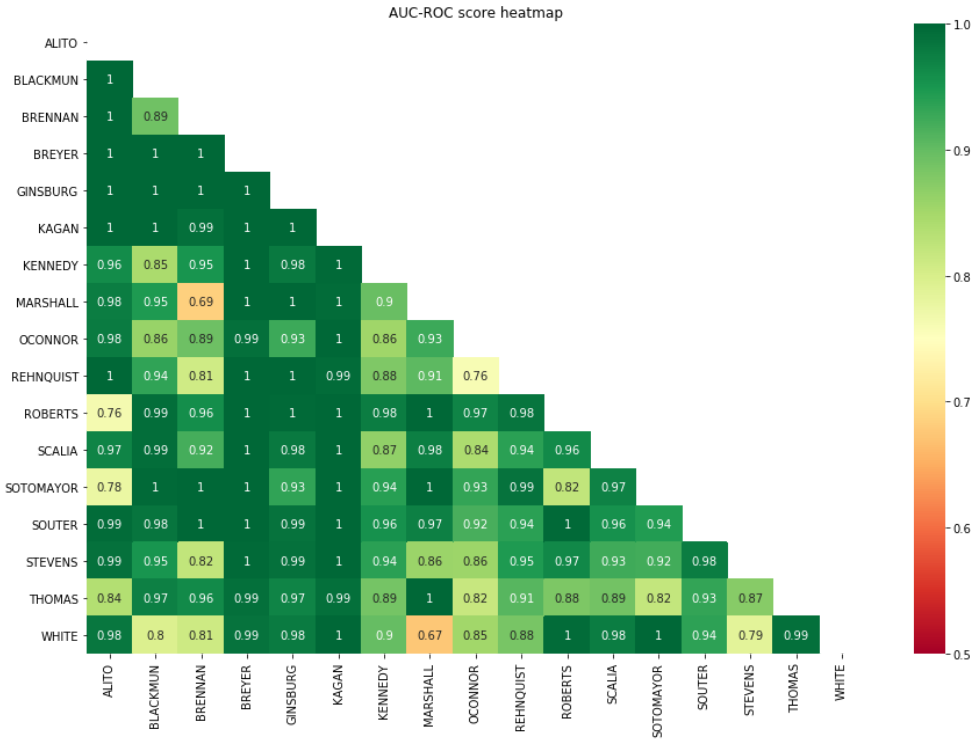
I intentionally structured this analysis after Rosenthal's approach (in [1]). Whereas Rosenthal used a narrow feature set of 63 specific features (function words, so-called as their frequency is hypothesized to be topic-invariant for a given author), I use 500 tf-idf features taken from a search over 1-grams, 2-grams, and 3-grams. Initially I selected a wider feature set, 2200 or so features, but I worried that on a data set this small (number of opinions tops out at around 200 or so) such a large feature set would be extremely overfit. I also initially included ngram ranges, but this led to a lot of "double dipping"--important features would include "second", "amendment", "second amendment", "the second amendment", and so forth. To cut back on this, I decided to search over single ngram values. Terms with high document-frequency were occasionally helpful, so I widened the parameter space for max_df--and, for similar reasons, stop_words. Stop words can be especially important at the phrase level.

An important note is that authorship predictions are done on a reduced dataset containing only the opinions of the two justices being compared. This simplifies the classification task at the cost of lacking generalizability. It would be nice, for instance, to know the features of Justice Ginsburg that are sufficient to distinguish her writing from any other judge.

Results

The resulting auc-roc and accuracy scores compare favorably to Rosenthal's results:





Takeaways

The best models used either 2-grams or 3-grams, and generally performed better than function word models (as reported by Rosenthal), which suggests that the best linguistic features to predict authorship occur at the phrase level rather than the word level. Function word models have the benefit of being more easily calculable (minutes rather than hours) and generalize more easily (identical features for each author), but they ignore context. Despite being a bag-of-words model, by including n-grams there's at least some role that phrase-level differences can enter the model.

Looking Ahead

It would be interesting to compare the performance of other models (logistic, support vectors) and other features (doc2vec vectors). Also, these predictions were on whittled-down datasets containing only the opinions of the two judges being compared; a more difficult problem would be to include *every* judge's opinion and perform a multinomial prediction.

Multiple Authorship

Analysis

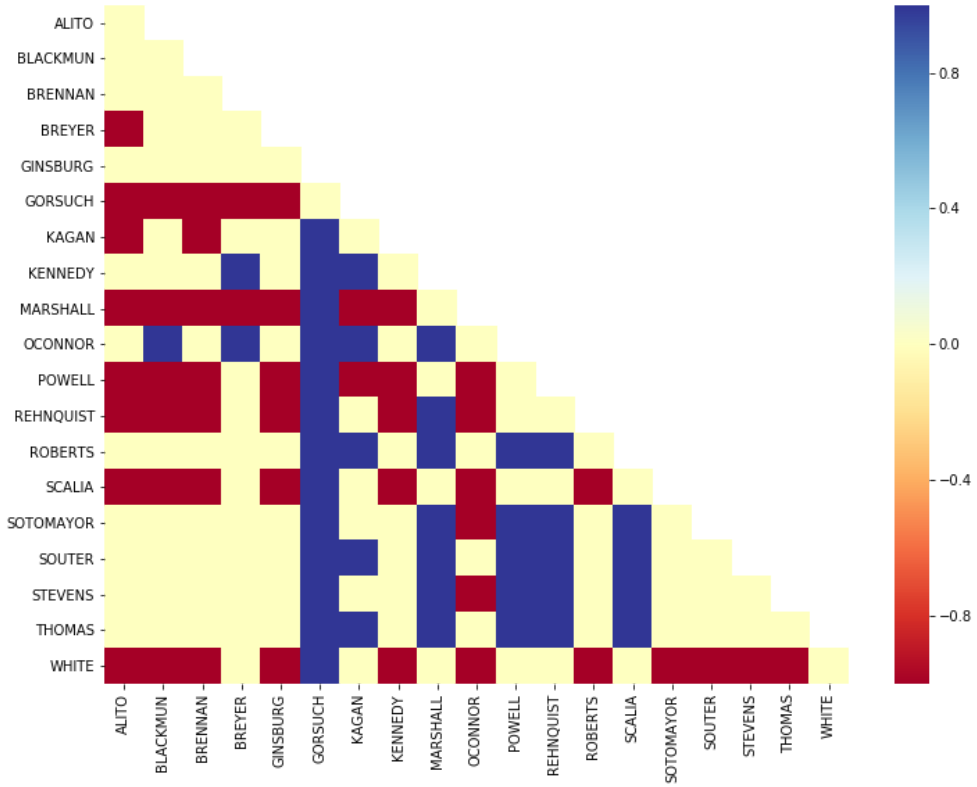
Following Rosenthal's approach (and in the tradition of the Federalist Papers function word analysis), I compute a χ^2 statistic based on bootstrapped function word counts in order to find a 95% confidence interval for a difference in function word frequency between two judges. This involved randomly selecting 100 opinions at random (with replacement) from a judge's corpus,

computing and storing the statistic for the batch, then repeating 1000 times, resulting in a variance list containing 1000 variance scores. Pairwise comparison was accomplished by approximating the distribution of the *differences* in variance by considering every possible pairing of variance scores (a list of 1,000 x 1,000 = 1,000,000 pairs), computing and storing their differences, and forming a 95% confidence interval by finding the 0.025 percentile and the 0.975 percentile. (For more detail, see Rosenthal's paper.)

The statistic computes the variance of the expected word counts given the independence assumptions. The general idea is that higher variance means more collaboration; the more authors who collaborate on a given text, the noisier the function word frequencies.

Results

Since this is an unsupervised task, the scores can't be explicitly verified. Some common wisdom results are borne out--Scalia fares well, for instance.



Takeaways

This is a very rough statistic without a clear way to validate its results. These results are only reliable insofar as the frequency invariance assumption is correct, and it's unclear what

Looking Ahead

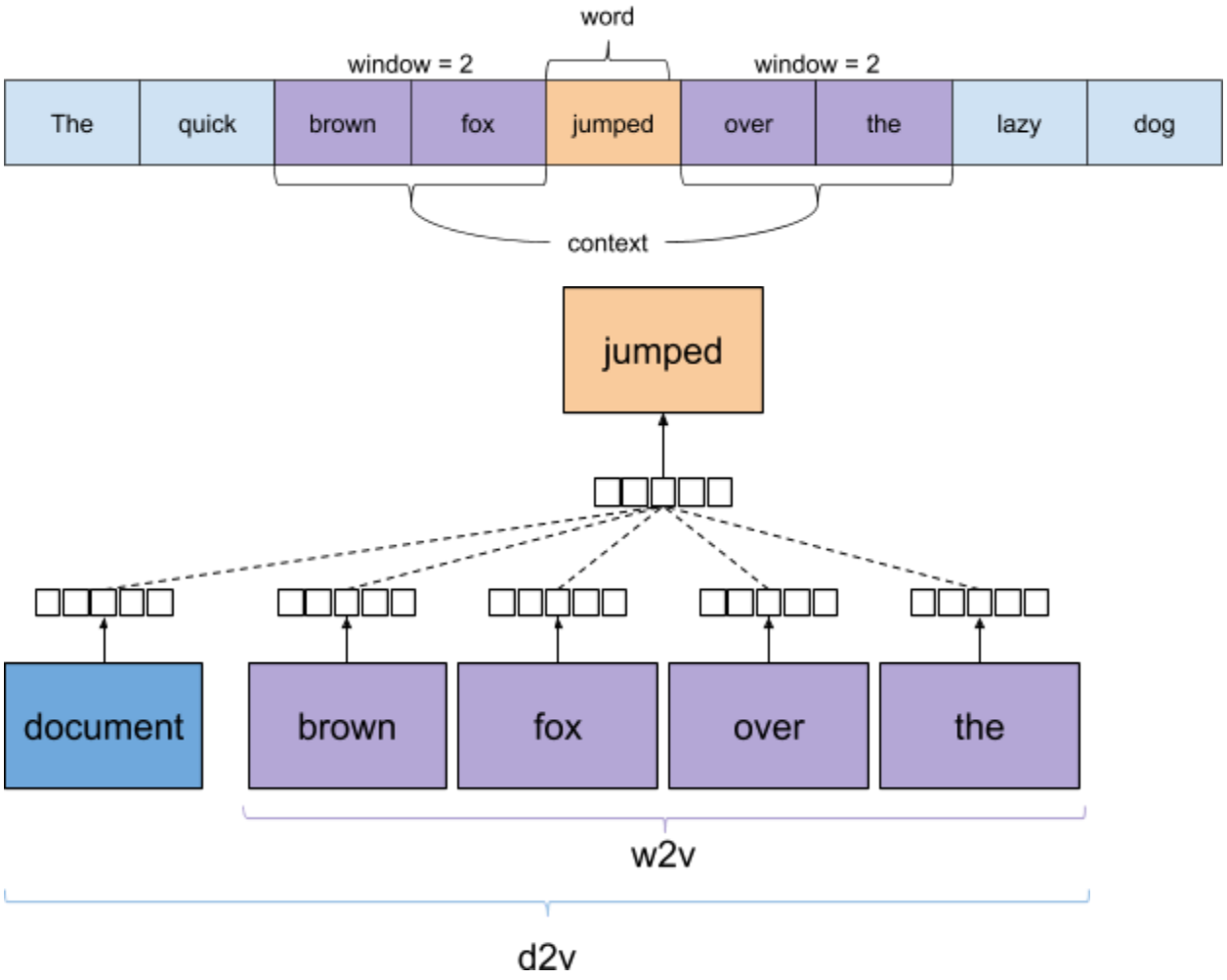
The next step should probably be to find a more robust statistical approach, or find whether frequency invariance can be empirically validated on at least legal text datasets.

The statistic in question is pretty slow to calculate, at least in my implementation. This may be due to the vicissitudes of python (Rosenthal's C-level scripts run much faster) but it would be an interesting challenge to try to optimize the calculation in python.

Document Similarity

Using gensim's doc2vec model, I computed document vectors for each opinion. Document similarities track subject matter and legal issues fairly well--partially due to citations and quotes linking opinions, but I take this to be a feature rather than a bug.

As a quick overview, doc2vec is an augmented version of word2vec. The word2vec model uses a "sliding window" to extract features, and then assigns vectors to words, tuning the coefficients to best predict either a word from its context, or the context from the word.



As seen above, the doc2vec algorithm does one better by including a document feature, included in every epoch with the word/context features found in that document.

In published work using doc2vec, frequently only 10-20 epochs were needed. I saw no great gain by increasing the epochs to 30. It was widely recommended to reduce the learning speed in each epoch, and to train epochs individually. My implementation follows this recommendations.

Results

Since this was an unsupervised task, there wasn't a great explicit loss function to compute. I decided to validate the results by looking at the similar cases of three famous Supreme Court cases. The top three most similar cases for Lawrence v. Texas, D.C. v. Heller, and Citizens United all bore striking similarities to the original, both narrowly by topic and more generally by legal issue.

Takeaways

Document similarities seem to work well at capturing both factual and legal topics.

Looking Ahead

I'd like to do more with the opinion vectors in order to get a more concrete measure of their performance. Possible applications include authorship attribution or topic modeling / labeling. An interesting extension of this approach would be to find vector representations of judges. One approach could be to use multiple tags on the opinion, a document tag and a judge tag. See e.g. "Vector Representations of Legal Belief", Elliott Ash and Daniel L. Chen 2018 (link).