Ideology of SCOTUS speakers based on audio and phonetics data

Methods to build relevant datasets and methods to classify speakers

Yassine Kadiri
New York University
Center for Data Science
yk1859@nyu.edu

Zsolt Pajor-Gyulai
New York University
Courant Institute of Mathematical Sciences
pgyzs1@gmail.com

Thomas Leble
New York University
Courant Institute of Mathematical Sciences
tl78@nyu.edu

ABSTRACT

Can we guess the political ideology of someone based on the way this person speaks? We investigate this question in the context of the hearings of the Supreme Court. The speakers are judges and attorneys, arguing over different cases. The conclusion is that basic phonetics data together with linear learning models will yield the right prediction with an accuracy of approximately 60%, thus beating random guessing. On the other hand, a deep learning approach fed with the entire audio data, turned out to be difficult to train to do better than random guessing.

KEYWORDS

Outliers, Null, Data Cleaning, Clustering, Numeric, Strings

ACM Reference Format:
Yassine Kadiri, Zsolt Pajor-Gyulai, and Thomas Leble. 2018. Ideology of SCOTUS speakers based on audio and phonetics data: Methods to build relevant datasets and methods to classify speakers. In *Proceedings of Machine Learning (DS-GA 1003).* ACM, New York, NY, USA, Article 4, 4 pages. https://doi.org/10.475/123_4

1 DATA SETS

We describe the data sets used for our project.

- The Oyez dataset[1] contains audio recordings of Supreme Court hearings between 1998 and 2013, together with a textual transcription of the arguments. The audio data has been parsed by a phonetics software (Praat) to extract the most relevant phonetics data. The final result is:
  1. Audio files containing the hearings.
  2. TextGrid files containing the breakdown of the hearing into speakers, sentences, words, syllables, and for each syllable the relevant phonetics data (called the **formants**) as to how this vowel was pronounced. For a given case `2006_04_1350` we have:
     (a) A text-grid file `2006_04_1350.TextGrid`, found in `/data/Dropbox/Data/Supreme_Court_Audio/Oyez_vowels/FAVE/oyez_full/2006`. This is the breakdown by sentences/paragraphs. It contains a number N of Âń items Âż which is indicated at the beginning as the size parameter (here, $N = 11$). Each item corresponds to one person speaking.
     (b) Corresponding files in `/data/Dropbox/Data/Supreme_Court_Audio/Oyez_vowels/FAVE/FAVE-extract/` For any given hearing, there are $4M$ files, where M is less than or equal to N - the number of items (or size) as above. For `2006_04_1350` for example, $N = 11$ but $M = 4$. File K corresponds to speaker K. The relevant files for us are those named `2006_04_1350_sK_norm.txt` where K is the index of the speaker. They contained the breakdown of speaker K speech by word/vowels, together with the relevant phonetics information for each vowel (the duration, the two first formants, and more formant data).
- The DIME dataset[2] collects "over 130 million political contributions made by individuals and organizations to local, state, and federal elections spanning a period from 1979 to 2014." An abridged version of this dataset, containing only the donators who have declared to be working in the judicial world (see below) can be found in `/data/WorkData/ideology_from_audio/RESULTS/contriblawyers.csv`.
- The normalized Martin-Quinn score[3] is an indicator of ideology for the justices (judges) at the Supreme Court. We imported it on the server `/data/WorkData/ideology_from_audio/RESULTS/justices.csv`.

2 DATA PREPARATION

2.1 Phonetics data

2.1.1 First steps. We will detail here the first data preparation steps. The corresponding code and this description can be fond in the notebook [4].

1. Gathering the phonetics data
   This is done with the TextGrid files described before and which breakdown into two files. From the first file (the .TextGrid), we find the name of speaker K, and we then gather the phonetics data associated to this speaker by going through the corresponding `_norm.txt` file.
2. Listing words and names
   In an exploratory phase, we figure out the words most pronounced by our speakers, as well as a list of all the speakers that appear. The number of speakers is of order 1000.

[1]https://www.oyez.org/about

[2]https://data.stanford.edu/dime
[3]http://mqscores.lsa.umich.edu/measures.php
[4]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/Data_gathering_summary.ipynb

(3) Searching one's ideology
To do so, we use the DIME dataset developed by Adam Bonica (link in footnote). This dataset lists the financial contributions to political campaigns in the USA. Some basic difficulties appear:
- The database is very large, with several dozens of millions of entries, and a naive thorough search takes too much time if it has to be done for around 1000 speakers.
- Some common names may appear several times.
- How to identify a speaker in the donation list? We also encounter the less obvious issue:
- It is fairly common that a given person will give money to both parties! Inspired by previous work on the subject, we chose the following approach:

(a) We go through the list of donation exactly once, and only keep the entry if the "job/activity" contains a word relevant for us: lawyer, justice, attorney... This effectively reduces the size of the list by a factor 10–100.
(b) Then for each speaker, we go through the reduced list and try to match first/last name with the first/last name entry of the donation. Here we make the following bet: there is only one person with a given first and last name who happens to work as a lawyer. A more careful procedure would involve checking the middle name (not always present in both tables) or the adresses and listing the possible collisions.
(c) If a speaker is not present, his/her ideology is set to "Undefined". Otherwise, we take the average amount

$$\textbf{Ideology}(\text{speaker}) = \frac{\text{Donation of speaker for Republican candidates}}{\text{Donation of the given speaker}}$$

Hence an ideology of 0 corresponds to a pure Democrat and ideology of 1 corresponds to pure Republican.

We run this on two datasets: the 2008 political campaign (including State and Local elections) and the combined Presidential campaigns (all Presidential elections since the mid-70's) and we take the average (which is admittedly not a proper barycenter).

This way, we obtain an ideology label for around 500 speakers over approximately 900.

2.1.2 Creating intermediary datasets.

- We use the TextGrid files to extract, for each speaker, a list of words they pronounced along with the formants' data for each word. The resulting file is `/data/WorkData/ideology_from_audio/RESULTS/wordssyllablesformants.txt`, whose entries are of the type

word, speaker, list of the syllables, formants data for each syllable

- We use the DIME dataset to find the ideology for each speaker.
- We also gather the judges âĂIJideologyâĂİ from their Martin-Quinn score[5].

[5]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/JudgeIdeo.ipynb

2.1.3 Creating training data. The second part of the preparation steps, with corresponding code, is described on the notebook[6] The main steps are the following:

(1) First, we reformat the data produced by the gathering scripts. We form a dictionary whose keys are tuples (speaker, word, vowel, position_in_the_word[7]) and values are the phonetics data, namely a list

$$[\text{count}, \text{frequence } 1, \text{frequence } 2, \text{duration}, \text{formant1}, \text{formant } 2]$$

where both formant1 and formant2 are themselves lists containing 3 frequences. The script also populates a list of all the speakers for whom we have phonetics data. This will help us compare and see if for those speakers we have an ideology data.
(2) We average the formants' data over all occurrences in the dictionary. (This is why we introduced count in our dictionary). The aim here is to avoid beforehand the suitation in which the models would overfit to one single speaker. As some speakers speak way more than others, this would skew our models towards some speakers if those happen to pronounce a word more than others. Also, this will help create datasets with distinct speakers which will provide independent training-testing sets when we'll split.
(3) Getting the ideology for the speaker and matching both dictionaries. We remove entries for which we don't have ideologies in order to avoid missing values. (If needed, a simple change will allow to keep speakers for which the ideology is missing)
(4) We find the most prounounced words and make some choice of 20 words which seem "ideologically relevant" (that we call *charged words*) and 15 words which seem neutral (that we call *not charged words*).
(5) For each syllable of each of this word, we create a file `word_syllable_position` whose entries are of the type

Ideology, phonetics data

They are in the folders `/data/WorkData/ideology_from_audio/RESULTS/non_charged` and `/data/WorkData/ideology_from_audio/RESULTS/charged`.

2.2 Raw Audio

To study the predictiveness of raw waveforms:

- We select some "ideologically relevant" keywords: and extract all the utterances of these words in the audio recordings into separate .wav files. This is described in the notebook[8].
- We extract the raw waveform file corresponding to each utterances for these words and store them on `data/WorkData/ideology_from_audio/RESULTS/WordAudio`. Within this directory every subdirectory corresponds to utterances of a particular keyword.

[6]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/Getting_the_Ideology_Formants_dataset_2.ipynb
[7]This last key position_in_the_word was introduced as for some words, speakers pronounce them differently which ends up in the formants data attributiing differents "heard" vowels for the same vowel
[8]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/Raw_Waveforms.ipynb

- To keep records, this dataframe also outputs a dataframe `data/WorkData/ideology_from_audio/RESULTS/raw_wave_word.csv` containing entries for each utterance in the form: SPEAKER - WORD - YEAR - .wav FILENAME.
- Finally, we incorporate the ideologies of the speakers.

3 MODELING

For the phonetics data, we worked with linear models. The main models we used are Lasso and Ridge Regression, along with Elastic Net Regression. This choice comes from the size of our datasets. Given how we built the files for each word, we find ourselves with files with at most 150 instances. Therefore, we're dealing with small data and we need to be careful with overfitting so this is why we chose regularized linear models. In order to choose the regularization strength, the idea is to test several values for λ for each regressor and choose the value for which the error is minimized or equivalently the value for which the score is maximized. Then, we try to predict the ideology of speakers. We first took this problem as a regression task but shifted to a classification task given the size of the dataset which wouldn't allow a precise enough prediction. We set the threshold to 0.5 i.e. if the regressor predicts an ideology greater than 0.5 we set the prediction to 1 and else to 0. The validation metric we then chose was the accuracy of our prediction as this is what matters after all (as opposed to recall or other metrics). The idea here is to have a regressor ready for each word and each vowel for that word so that when given a new speaker, we can predict his ideology based on how he pronounces the words we selected. Given how we built our dataset, we can not overfit to a single speaker as each dataset contains only distinct speakers. Also, in order to avoid overfitting and make the most of the small datasets we have, we decided to make a cross-validation for each of the regressors. The script implementing and plotting the results can be found on GitHub[9].

To analyze the raw audio files, we have built a neural network model the architecture of which we now describe. We first feed the raw waveform files of length T (varying from file to file), which are sequences of amplitudes into a convolutional layer with receptive field size F, stride S, and C output channels to achieve the preliminary extraction of some learned features. This is followed by a max pooling layer with unit size and pooling field size PF. The output of this is then fed into a Recursive Neural Network (RNN) with Long Short Term Memory (LSTM)-cells with hidden state dimension H. The initial hidden and cell states of the LSTM network is set to zero. The final output of this network is then mapped into probability scores over the two classes by the combination of a linear and a softmax layer. We train this network using the Adam algorithm through backpropagation using the cross-entropy loss function. The script implementing this network can be found on GitHub[10]

We train such a model for each keyword (except for SCIENCE and TAXATION) optimizing the hyperparameters T, F, S, C, PF, H over the corresponding validation set. Once these are fixed, each model will be retrained on the full corresponding data. The final predictor is going to be given by the weighted vote of the individual models,

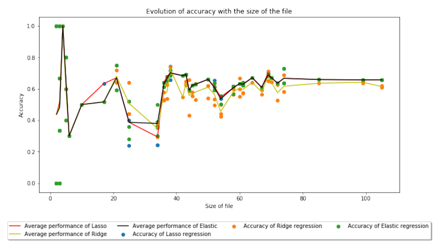

Figure 1: Results for ideologically charged words

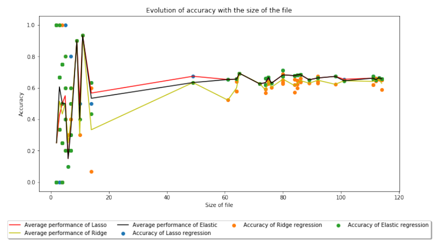

Figure 2: Results for ideologically uncharged words

where the weights are optimized with respect to the validation sets of SCIENCE and TAXATION.

We also performed a similar exercise this time using a WaveNet architecture model[11] adapted for classification.

We are going to simply use accuracy as a measure of success for these models.

4 ANALYSIS

For the phonetics data, the scores were really poor and we were able to increase them by removing some features (such as 'Count' which the number of times a speaker had spoken and that we thought would be predictive). Once this was done, we got better scores, around 0.6 and could adequately choose our regularization strength for each regressor and for each file. For each file (`word_vowel.txt`) we therefore choose the appropriate λ for each regressor and for each of the 5 folds of our cross-validation, predict the ideology of the 20% of speakers from that fold. Then we compute the accuracy for each of the 5 predictions and average it, for each regressor. We decided to plot the results as a function of the size of the file for which we made the predictions. The idea here is, given the small data we have, to be able to estimate how our predictions perform 'asymptotically' when the data becomes bigger.

The results are given in figures 1 and 2 (separate plots for each classifier can be found in the following notebook [12])

It seems that the Lasso, Ridge and Elastic 'classifiers' have similar results and asymptotically reach an accuracy of 60%. We notice that for small datasets, the results have really high variance, as expected, but as the size grows, the accuracy increases and converges to a

[9]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/gathered_data/Regressions.ipynb
[10]https://github.com/yasskad/Ideology-based-on-audio/blob/master/RNN_classifier/conv_lstm_classifier.py

[11]https://arxiv.org/pdf/1609.03499.pdf
[12]https://github.com/yasskad/Ideology-based-on-audio/blob/master/data_gathering_scripts/gathered_data/Regressions.ipynb

Table 1: Convergence analysis for charged words

Classifier	n	Accuracy (in %)
Lasso	0	57.12
Lasso	20	60.97
Lasso	50	63.15
Lasso	100	65.71
Ridge	0	54.63
Ridge	20	57.91
Ridge	50	59.20
Ridge	100	61.43
Elastic Net	0	56.59
Elastic Net	20	61.28
Elastic Net	50	62.92
Elastic Net	100	65.71

Table 2: Convergence analysis for non charged words

Classifier	n	Accuracy
Lasso	0	53.49
Lasso	20	66.50
Lasso	50	66.48
Lasso	100	65.79
Ridge	0	51.42
Ridge	20	63.20
Ridge	50	63.19
Ridge	100	64.09
Elastic Net	0	54.87
Elastic Net	20	66.42
Elastic Net	50	66.49
Elastic Net	100	65.70

final value. In tables 1 and 2 we sum up the evolution of the average accuracy with the files' sizes (i.e. the number of speakers) with an emphasis on accuracy for files with size greater than *n* for several values of *n*. (The accuracy is rounded to the 2nd decimal)

We therefore see that the 'limit' is around 65% for both Lasso and Elastic 'classifiers' on both types of words and around 61% for Ridge on charged words and 64% on uncharged words. The convergence is faster for non charged words as the classifiers only need 20 words to reach the 'limit' while for charged words, the sequence is strictly increasing and only reaches its final value after 100 words. However, the datasets containing charged words had a size of 105 instances at most, so we can expect the classifiers to do even better perhaps with words that appear among more speakers given the fact that the sequence is strictly increasing. The results are satisfying and a possible topic to analyze would be to understand what do the classifiers discriminate.

Unfortunately, we are unable to report similar success based on the raw audio. Our neural network models turned out to be difficult to train. We speculate that this is because in order to capture, for example, the wowel level data explicitely extracted for our other methods, one need to have a receptive field that is very long compared to the sampling period. An architecture with such a receptive field necessarily needs to be too deep and consequently hard to train. However, we still find this a promising possibility worth exploring further.

5 CONCLUSION AND LESSON LEARNED

We found that simple learning models worked fairly well given the small size of our final data sets and the errors that may have been introduced along the way (e.g. the possibility that the ideology of a speaker had been computed by taking into account the donations of an homonymous attorney, the fact that we did not take gender or region of origin into account), showing some predictive power. It is yet unclear whether this really "detects" any form of *ideology* rather than more basic characteristics (e.g. masculinity, low tone of voice, geographic accent) that would be correlated with ideology.

The more sophisticated method of *deep learning* would in principle be able to detect fine variations in the way people speak. However, the training time on large datasets (the raw audio files) is prohibitive - it could perhaps be more successful by first selecting a smaller portion of the audio that one considers to be ideologically relevant and train the network on this.

On a simple, pragmatical side, here are some lessons that we have learned:

- It is important to think the dataframe through at first, to define a format and to stick to it. We have spent a lot of time converting files and entries from one format to another and to ensure compatibility. A structure convenient for programming might be inconvenient for data analysis.
- The basic learning algorithms are fairly easy to implement using libraries. It could thus be clever to rapidly get some data set, even if partial and still messy, for a preliminary analysis of the size and the predictive power of the data.