SCOTUS: Recurrent Supreme Court Decision Prediction

DiMare-Baits, Christian
dimare.c@northeastern.edu

Elhabr, Nicholas
elhabr.n@northeastern.edu

Franz, Vaughn
franz.v@northeastern.edu

1 Objective

To study RNNs versus CNNs and the rate of convergence of a Justice to their opinion given oral argument utterances from Supreme Court cases.

2 Introduction

CNNs encoding context by pooling micro averages using convolution kernel oveer context windows around each sequence item.

RNNs, alternatively, encode context recurrently by accumulating updates to hidden parameters as they see new items in a sequence.

The former roughly translates to predicting an outcome, like a Justice vote, given the entire case, and weighing all the words, grams, or utterances simultaneously against each other in a micro-batched way.

The latter roughly translates to training for the changes in opinions of a Justice over individual words, grams, or utterances, where recent information is weighed against an accumulated context as new arguments arise.

This leads to the question, how do we model a Justice's opinion?

2.1 Hypothesis

Let a Justice be a decision-maker following the argument utterances of a new case modeled as an RNN with 2 forward hidden inputs:

1. A RCNN recurrently encoding the opinion of the Justice on previous cases and returning an historical experience embedding $H_{self}^{(c)}$, updated on a case-by-case basis.
2. A Standard hidden context layer, $C_{(u)}$, encoding the opinion of the Justice on this particular case, as it changes on an utterance-by-utterance basis.

Lastly, we train the RNN on new utterance by concatenating new utterance embedding $U_{(i)}$ by with $H_{self}^{(c-1)}$, $C_{(i-1)}$ to form our total input $x^{(i,c)} = \left[U_{(i)}; C_{(i-1)}; E_{self}^{(c-1)}\right]$, and measure the number of utterances until convergence on a Justice's vote as an emitted feature.

The idea behind this architecture is that Justices make decisions based on hisotircal precedence encoded in their previous decision-making, without access to future case decisions. By encapsulating this training recurrently, we can avoid learning from future cases and utterances.

3 Methodology

'convokit' is a Python library provided by Cornell University with various conversational corpuses structured around Utterances and Speakers. In particular, we chose the downsample the "supreme-corpus" to the longest sequence of cases decided by the same Justices so we can minimize variance.

We embed utterances using a dimensionally reduced count-based embedding within a case, at an utterance level, and avoid bias by targeting the result of the case instead of the result of an individual Justice (most likely using sklearn).

We embed cases similary, using a pretrained CNN architecture.

Using pretrained CNNs and RNNs (ideally in PyTorch) for benchmarking court decision-making, we focus our energy on downsizing their architectures into an RCNN hidden input and focusing on building a good training pipeline for each of these models.

4 Analysis

Our objective is to measure how quickly a Justice makes a decision on a case given changings arguments based on their own (and potentiall others') previous voting history.

We can visualize this via a histogram with horizontal axis counting the number of utterances until convergence, and vertical axis counting the number of cases which converged in that number of utterances grouped by Justice.

The distribution of cases that are decided very quickly versus very slowly can potentially demonstrate polarization and bias. In the event that all Justices decide very quickly, we would see left-heavy distributions. In the event that all Justices decide quickly, combined with information about the vote on political alignment, we construct a numerical argument that the court is strongly biased.

5 Timeline

Week 8:	Proposal Presentation Using github repo.
Week 9-10:	Data Preprocessing stage. Minimal sub-sample training pipeline testing.
Week 11-12:	Train data at scale. Parameter tuning.
Week 13:	Finalizing result sets and with visualizations.
Week 14:	Report submission.