Predicting Justice Behavior of Supreme Court of the United States

Kshitij Yogesh Gupta
Department of Computer Science, University of Toronto
kshgupta@cs.toronto.edu

Manasa Bharadwaj
Department of Computer Science, University of Toronto
bhaman@cs.toronto.edu

Abstract

In a dominantly democratic world, the judicial system is indispensable. It is the single choke point for law and order. With the advent of technology and perennial increase of cases any automation of decision making is a boon to the nation. In this context, we attempted to model the behavior of the Supreme Court of the United States using Recurrent Neural Networks - LSTM. We aim to design generic, robust and fully predictive model, which uses only the data that is available before the term. While experimenting with different architectures, we finalized on a model that gives 82.07% accuracy in predicting votes cast by a justice. As far as our knowledge goes, our model has the highest accuracy for similar predictions and beats the state-of-art performance by a margin of 10%. We used SCDB as our data source which contains records of cases spanning over two centuries.

Keywords
SCDB (Supreme Court Database), LSTM, Affirm, Reverse

1 Introduction

A majority of cases that are handled by the Supreme Court make their way from the lower courts. The Supreme Court of the United State agrees to a hear a case by granting a petition. This is followed by each party submitting supporting written materials and oral arguments.The Court needs to vote unanimously to give a judgment.The Supreme Court will either Affirm or Reverse the decision given by the lower court. Our motive is to predict voting patterns of a particular justice. Based on our literary analysis of [1, 2, 3, 4, 5], we understood that, this question has piqued the interest of reporters, pundits, academic journalists, law reviewers and machine learning enthusiasts.

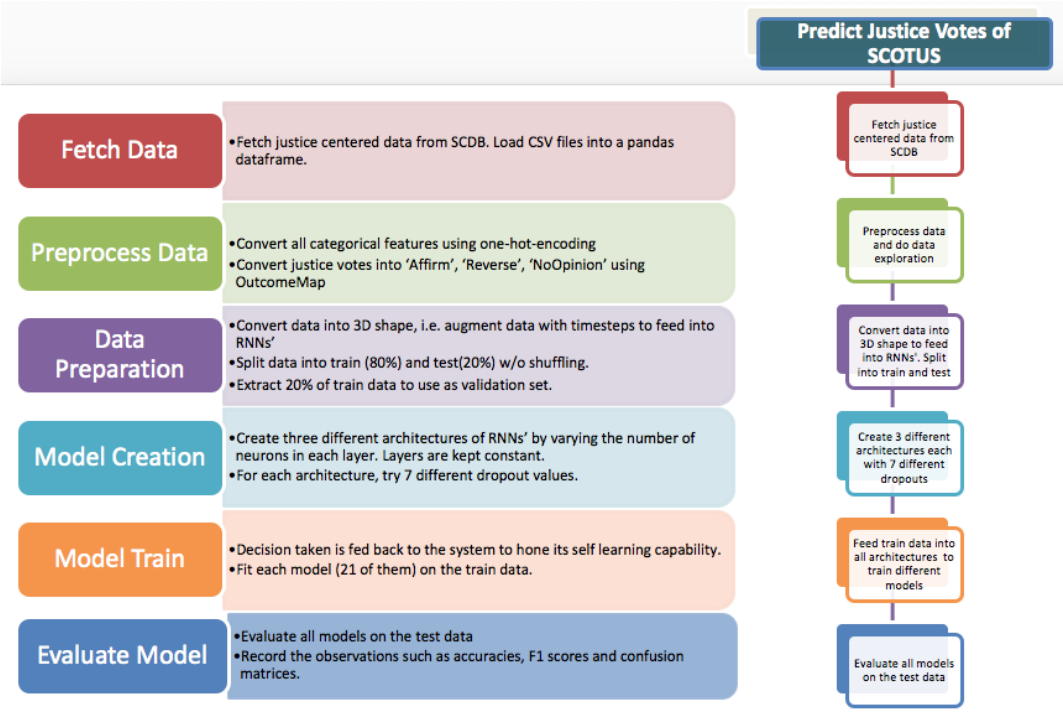
A judgment in a term can be made only with wisdom garnered from previous terms. This accentuates the importance of preserving the sequential relationship in the data. Hence the usage of randomized raw dataset would allow a model to learn correlations that violate the essence of time. There is an inherent temporal aspect to the data wherein a decision made in a particular case in a particular year may affect all subsequent decisions on future similar cases. Using a simple feed forward network on the raw features would neglect previous knowledge, and thus the temporal aspect. However RNNs' have proven to be effective at dynamic temporal behavior. LSTMs' provides us the ability to learn more complicated correlations than a feed forward NN architecture. We have used Dropout technique to perform regularization and reduce over fitting. We concluded the project by experimenting with the effects of number of activation units per layer on the model complexity and prediction accuracy.

2 Motivation

After an exhaustive search for an exciting machine learning application domain, we realized that we wanted to do a project which will have an impact. We wanted to build something that might positively impact developing countries such as ours, India. Our project selection drew inspiration from a past work. The number of cases registered in India grows every year and exceeds more than what the courts are equipped to handle. The trivial solution is to employ more justices and construct more court rooms. This is a luxury that a third world country cannot afford.

The other practical solution is to take advantage of growing cheap technical ways to connect the world, i.e. create virtual courts. This should be coupled with preprocessing of cases to eliminate frivolous ones. We wanted to automate decision making by borrowing from machine learning ideas. There are two major challenges hindering our efforts towards designing this solution. Firstly, there are not enough resources to consolidate data regarding cases and decisions of Indian courts. This leads to a major problem plaguing machine learning researchers since time immemorial - namely lack of data. Secondly, we do not posses enough legal background to make meaningful causation and correlation of features to decision making. Solving the first problem in a developing country would require immense manpower and budget. However, the second problem can be solved with the help of just a laptop. We can train a network which would learn all the causations and correlations within the legal system on its own without our input, i.e. engineered features. We chose to work with the U.S. Supreme Court Database, since it has a well curated record which spans over two centuries.

3 Figure - Overall Flow



4 Feature and Dataset Analysis

We obtained the justice centered legacy dataset from SCDB[6]. We tried to analyze the feature to class relations before training the model using raw features.We generated multiple graphs to see any patterns which could explain the extensive feature engineering done in [1, 2] and discover our own feature relations.

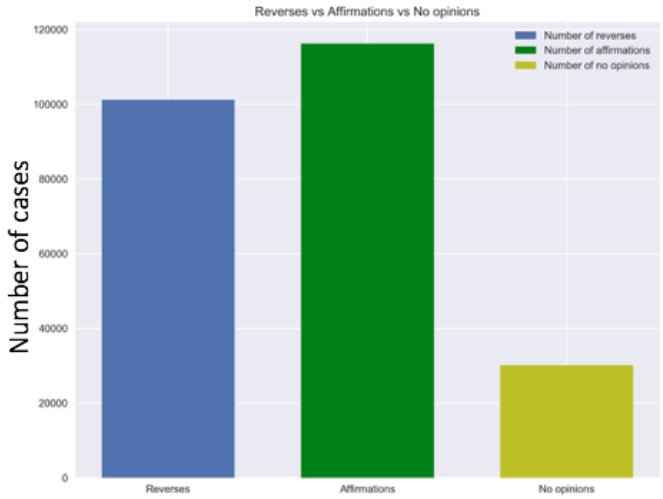


Figure 1: The obvious starting point was to check whether the data is balanced. This figure shows the numbers of each type of decisions in our dataset. In order to balance the difference in the amount of reversals and affirmations we have given a slight more weight to the reverse classes in the keras implementation.

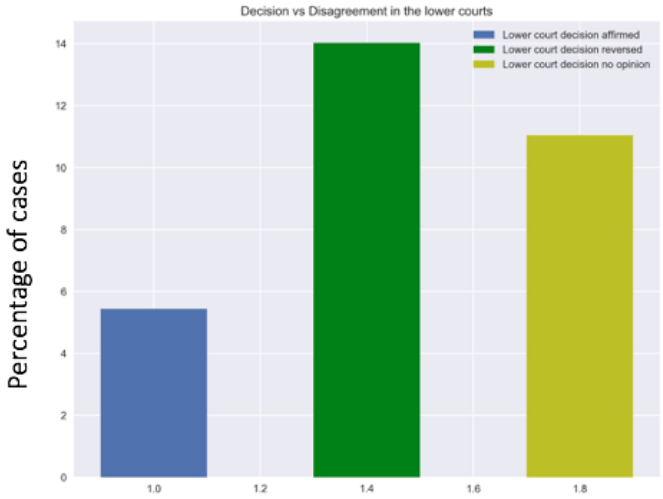


Figure 2: Our next area of interest was whether more disagreements among the lower courts would cause any influence on the decision given by the judge. Supreme Court has reversed more cases when there were disagreements among the lower courts.

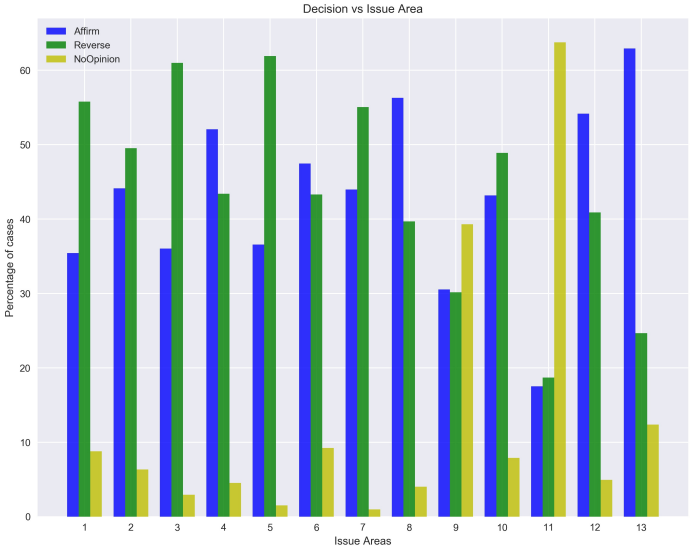


Figure 3: We have also tried to see how the decisions vary across Issue areas. Table [1] provides names for each issue. We have noticed that issues related to "Interstate Relations" had minimal number of affirmations or reversals. On the other hand, "Privacy" related cases had maximum number of reversals. "Economic Activity" cases (Other than Misc.) had maximum number of affirmations.

Table 1: Issue Areas

Issue Number	Issue Name
1	Criminal Procedure
2	Civil Rights
3	First Amendment
4	Due Process
5	Privacy
6	Attorneys
7	Unions
8	Economic Activity
9	Judicial Power
10	Federalism
11	Interstate Relations
12	Federal Taxation
13	Miscellaneous
14	Private Action

Table 2: Law Type

Law Type Number	Law Type Name
1	Constitution
2	Constitutional Amendment
3	Federal Statute
4	Court Rules
5	Other
6	Infrequently litigated statutes
8	State or local law or regulation
9	No Legal Provision

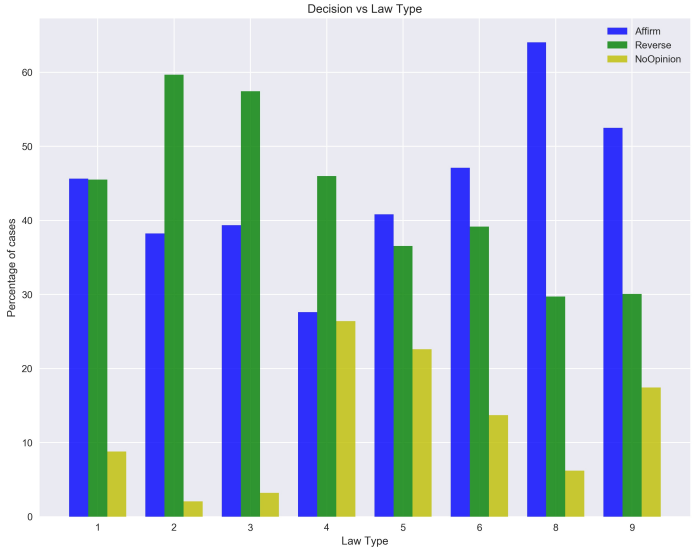


Figure 4: We try to see how decisions vary across each Law Type. Table [2] provides names for law types. We noticed that maximum number of reversals happened in cases of type 'Constitutional Amendments'. The maximum number of affirmations happened for 'State or Local Law or Regulation'. 'Court Rules' related cases had the highest percentage of No Opinions.

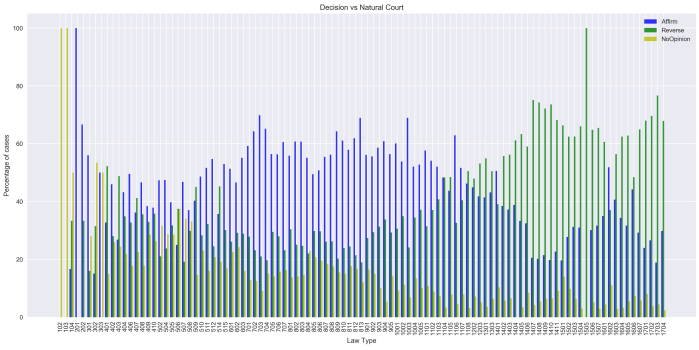


Figure 5: We next try to see how decisions vary across natural courts. The data was haphazard to make proper correlations. This has made us realize that we should make sure there has to be a change in natural court in the test set to make testing set generic.

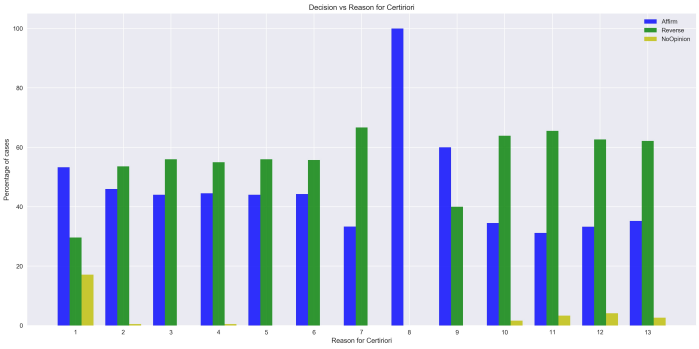


Figure 6: This figure plots the relation between the reason SC agrees to hear the case and justice votes. Table 3 lists the different reasons. In majority of the cases of type 'State Court Confusion', Supreme Court has stayed with lower court's decision. On the other hand, the 'Federal Court Confusion' cases has the highest number of reversals.

Table 3: Reason for Granting Petition

1	case did not arise on cert or cert not granted
2	federal court conflict
3	federal court conflict and to resolve important or significant question
4	putative conflict
5	conflict between federal court and state court
6	state court conflict
7	federal court confusion or uncertainty
8	state court confusion or uncertainty
9	federal court and state court confusion or uncertainty
10	to resolve important or significant question
11	to resolve question presented
12	no reason given
13	other reason

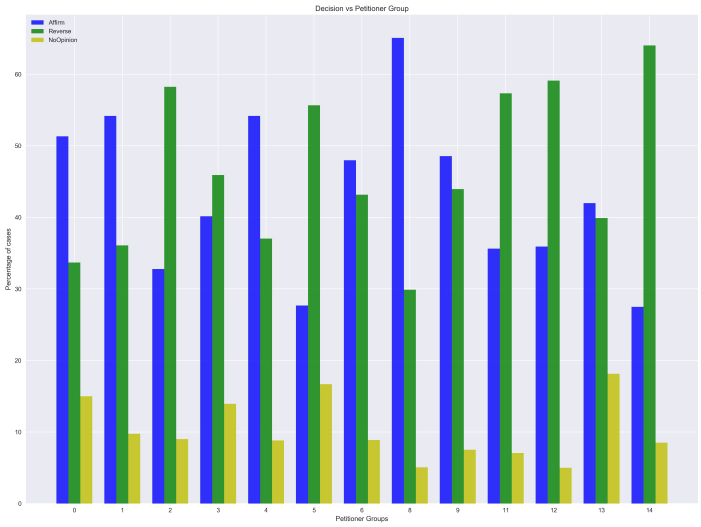


Figure 7: We try to see how decisions vary across different Petitioner types.

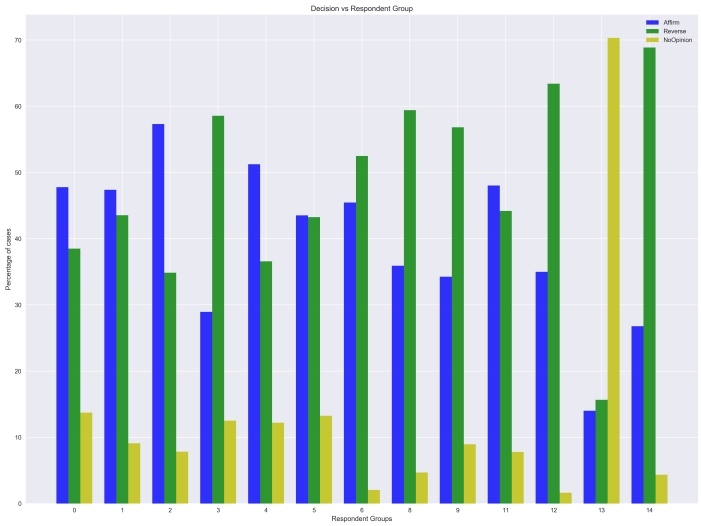
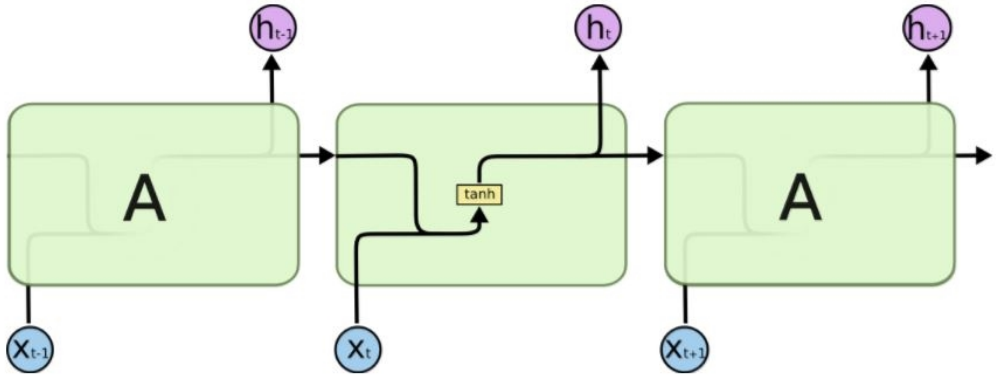


Figure 8: We then try to see how decisions vary across different Respondent types.

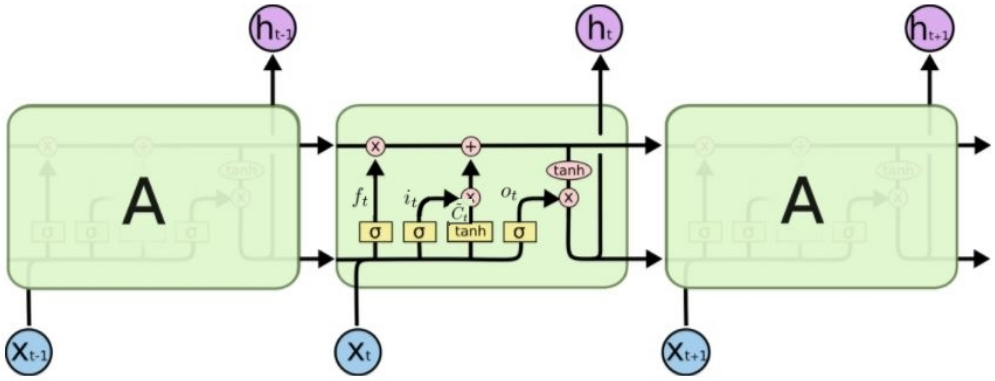
Data was preprocessed before feeding into the LSTM architecture. All the categorical variables were one-hot encoded. A timestep size of one was used to create 3-D input shape needed for the LSTM. We have used 80% of the data for training (with 20% of that going to the validation set) and 20% for testing.

5 Formal Description



The repeating module in a standard RNN contains a single layer.

Figure 9: Basic RNN[7]



The repeating module in an LSTM contains four interacting layers.

Figure 10: Basic LSTM[7]

Recurrent Neural Networks enable storage of information. They have directed loops among the activation units which allows information to be passed from one step of the network to the next. This makes them an ideal choice for processing temporal inputs. LSTM is the most popular variant of RNN. As shown in Figure 10 they offer additional functionality and flexibility in combining current input and previous output. Figure 11 shows the equations used for training LSTM networks. f_t is related to "forget gate" which when set retains information or forgets otherwise. i_t is the "input gate layer" that decides which values are updated. This is combined with temporary cell state values to generate updates for cell state information C_t. Finally the output o_t from last sigmoid is combined with new cell state value and custom output values are generated. In this way LSTM architecture allows us to selectively store past information to be used for current processing, and also mask unnecessary parts of the output. This output is the extra knowledge passed to next activation unit as the prior knowledge.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

Figure 11: Basic LSTM[7]

As discussed before, the usage of RNNs' is inherently different from Random Forests used by the baseline [2] since RNNs' can capture temporal information present in the data. We have tried three variations in our base LSTM architecture with increasing model complexity. Refer to Table 4 for basic architecture diagram. There are two LSTM layers with dropouts, followed by a dense layer with ReLU activation. The output is given to the final dense softmax layer with three neurons for the multi-class classification. The three architectures differ in the number of activation units in each layer. Each architecture was trained on different dropout rates to get the best one. The number of activation units were picked by hand.

Table 4: Best Architectures and Layers

Layer	Name	Activation Units		
		Arch1	Arch2	Arch3
1	LSTM	300	600	1000
2	Dropout	0.15	0.2	0.35
3	LSTM	300	400	800
4	Dropout	0.15	0.2	0.35
5	Dense (ReLU)	100	150	150
6	Dense (Softmax)	3	3	3

6 Results and Comparisons

The baseline model [2] exhibits accuracy of 71.9% at the Justice vote predictions for the same dataset. Our best architecture provides 82.07% accuracy.Table [5] and Table [6] shows comparison of baseline model to our three best models. We have also included results for a dummy model which just predicts the most frequent class in the test data set.

Since [2] differed in how they tested their model (essentially they developed an online learning model of random forests which grew in number after each term), we also trained a Random Forest classifier on the same train test split that was used for our architectures. This model had all the hyper-parameters set to the ones used in the baseline model. It gave an accuracy of only 60.60%

Table 5: Accuracy Comparison with Baseline

Model	Accuracy in %
Baseline Paper	71.90
Baseline Git	65.86
Dummy Model	48.48
Random Forest	60.60
Arch1	79.22
Arch2	82.07
Arch3	81.76

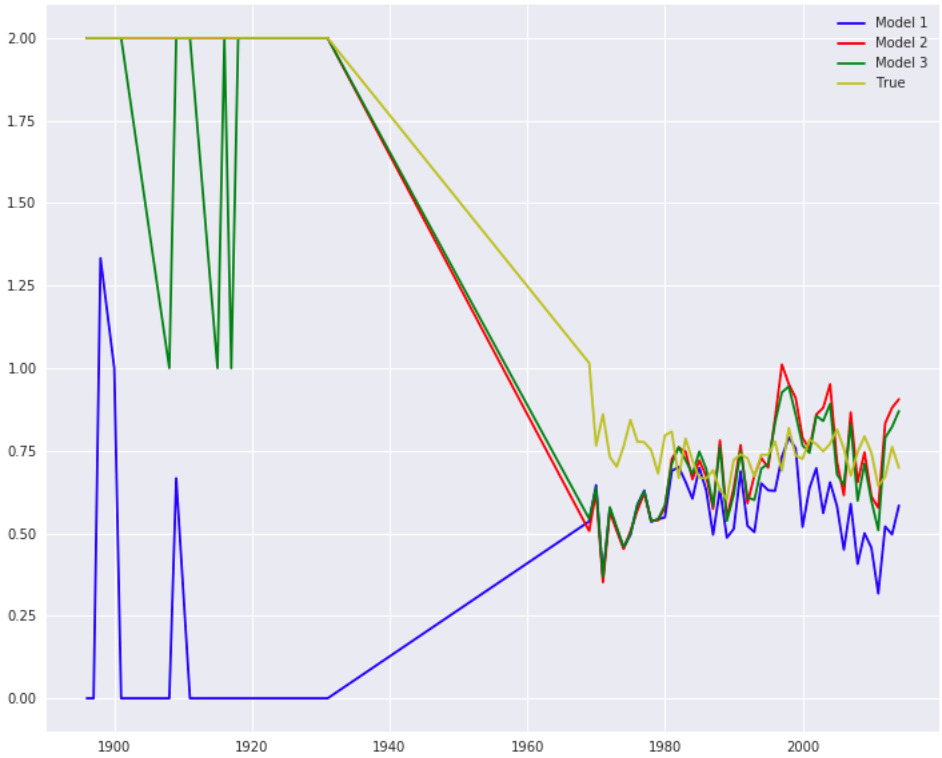


Figure 12: Displaying the mean of predictions over the test years of different models

Table 6: Performance Metrics Comparison with Baseline

Class		Precision	Recall	F1-Score
Affirm	Baseline	0.61	0.79	0.69
	Arch1	0.90	0.76	0.83
	Arch2	0.89	0.83	0.86
	Arch3	0.89	0.82	0.85
	Dummy	1.00	0.48	0.65
Reverse	Baseline	0.64	0.48	0.55
	Arch1	0.81	0.81	0.81
	Arch2	0.82	0.78	0.80
	Arch3	0.82	0.78	0.80
	Dummy	0.00	0.00	0.00
Other	Baseline	0.84	0.59	0.69
	Arch1	0.54	0.87	0.67
	Arch2	0.69	0.86	0.76
	Arch3	0.68	0.87	0.76
	Dummy	0.00	0.00	0.00
Mean/Total	Baseline	0.66	0.64	0.64
	Arch1	0.82	0.79	0.80
	Arch2	0.83	0.82	0.82
	Arch3	0.83	0.82	0.82
	Dummy	1.00	0.48	0.65

Without resorting to extreme feature engineering or a very deep architecture we could generate an increase of 10% in the accuracy.

Confusion Matrices

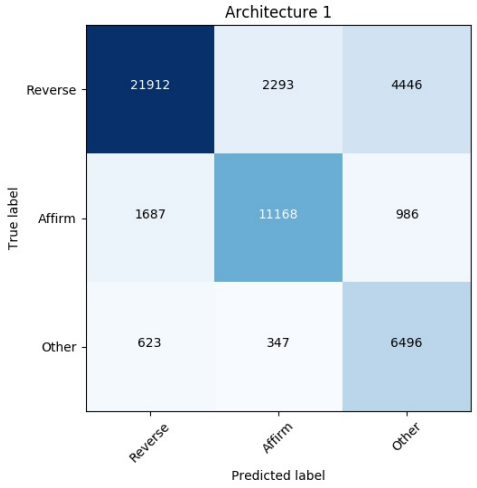


Figure 13: Confusion Matrix for Architecture 1

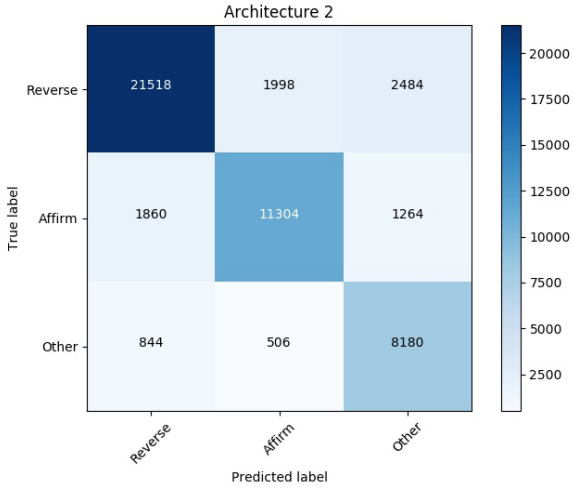


Figure 14: Confusion Matrix for Architecture 2

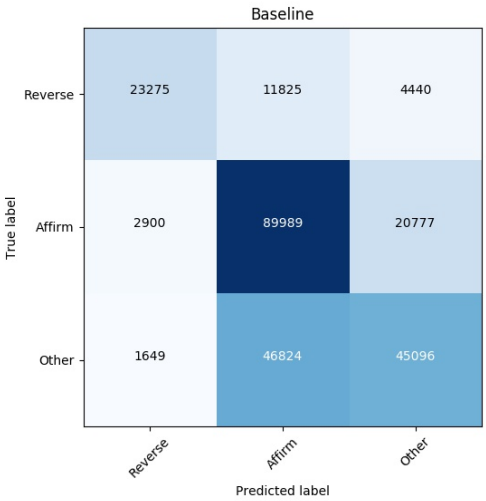


Figure 15: Confusion Matrix for Baseline

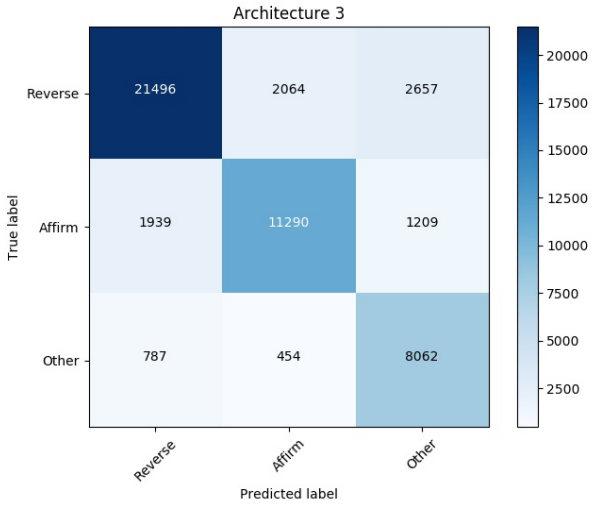


Figure 16: Confusion Matrix for Architecture 3

7 Discussion and Limitations

We have achieved an unprecedented margin in the performance improvement. We account this for LSTM's ability to maintain the context of the previous term's case predictions. One limitation to our approach could be the possibility of overfitting. [8, 9] talk about possibility of overfitting for the baseline model [1] and in this domain in general. However, we ruled out overfitting since our validation accuracy was higher than our train accuracy. If in case we encounter a clear cut case of our model overfitting the dataset, we can rely on higher dropout rates to counteract that.

The other limitation we are worried about is the *"Curse of Dimensionality"*. The usage of one-hot encoding of categorical input features increases the number of features in the dataset. This is because some of the features can take as many as 343 different values. Feature transformation for deploying some LSTM variants can cause further increase in dimensions with no increase in dataset. This could lead to decrease in performance and cause underfitting.

8 Related Work

We have used [1] [2] as the main reference papers. Both papers have thoroughly documented the feature engineering done on the SCDB. Since the authors are from a law background, we relied on their expertise to do proper engineering from a legal standpoint. They have used growing ensemble Random Forest architecture. [1] Has performed binary classification on sixty decades data, whereas [2] has performed multi-class (three) classification on two centuries of legacy data.

Our approach was initially built on the engineered features. The distinction in our project is the usage of RNN - LSTM architecture as opposed to growing Random Forest. We have taken advantage of inherent temporal dependence in the data. We obtained baseline results for the cases we tried, using their git hub source code. In order to fully exploit deep neural networks, we then reran all our architectures without engineered features. We wanted to rely on the RNN to learn all dependencies on its own. Since we achieved better accuracy with raw features we have presented these models alone. We have reused small parts of their code to create outcome map, party map, circuit map, and to do one hot encoding. While this creation of such maps might be considered feature engineering, it was done to simply avoid the explosion of features which would be generated by their one hot encoding. The encoding is an off the shelf code, and we didn't find any value addition in rewriting this functionality.

There are multiple papers published predicting the behavior of Supreme Court, be it using the SCDB or recordings of oral processions in the court. We have selected the papers which compared their model's performance to the baseline performance of [2]. The authors of [9] have used AdaBoosted Decision Trees(ADT) to do the predictions. They have used data for one decade along with oral arguments transcripts. There is only a 2% increase in the performance with the inclusion of text data. It will be intractable to get or use text data for two centuries. There is no baseline to compare accuracy for text related predictions for dataset for long period of time. [10] has used emotional content based on voice pitch differences. They correlate higher pitch to a negative response and vice versa. Even though they have achieved slightly better accuracy than the baseline, the underlying assumption seems to be impractical. This method can't be deployed in a live court to make predictions as that would mean measuring the pitches during a court hearing. If a justice has never talked in that trial, we would be waging a less informed guess than a justice who interacted with the parties involved. Our model doesn't rely on live information, hence the confidence or accuracy doesn't depend on justice interactions in the current court session. [11] combines [2] with data about audio features of lawyers' introductory statements and facial attributes. This improves the performance by around 4%. Since we have achieved more than 10% increase without using extra data of different genre, our model will be easy to use and understand. [12] has worked with text, word embedding and CNN architecture to predict affirming or reversal of lower court decisions. They have attained 79% accuracy, but have used data only for fifteen thousand cases. Our model obtained higher accuracy for more amount of data. Finally [3] has used feed forward neural network architecture to do the prediction. They also relied only on raw features and ended up using only 15 of them. They attained accuracies of 70.2%.

9 Future Work

As part of extensions to the current code, we want to try other variations in the LSTM architecture. We want to use peephole variant which enables us to feed output or previous cell state to the next activation unit. We wanted to draw inspiration from the papers in related work section which have used a different genre of data to make predictions. We will try to build an ensemble model which makes predictions using case transcripts, sessions recordings data along with the existing features. We would also like to experiment with how crowd sentiments and analysis can be incorporated into the ensemble model[13]. We also want to explore various ways to encode categorical features, that will make the model robust to the *"Curse of Dimensionality"*.

Conclusion

Predicting the behavior of the Supreme Courts has been a challenge that has excited researchers for decades. We build a simple LSTM architecture trained on the data collected from SCDB. This model gives an accuracy of 82.07% which, not only outperforms the best known accuracy of 72.9%, but also outperforms legal and political experts.

This project shows how Deep Neural Networks can work in a variety of domains and their adaptability to different tasks.

References

[1] Daniel Martin Katz, Michael J Bommarito II, and Josh Blackman. Predicting the behavior of the supreme court of the united states: A general approach. 2014.

[2] Daniel Martin Katz, Michael J Bommarito II, and Josh Blackman. A general approach for predicting the behavior of the supreme court of the united states. 2016.

[3] Ranti Dev Sharma, Sudhanshu Mittal, Samarth Tripathi, and Shrinivas Acharya. Using modern neural networks to predict the decisions of supreme court of the united states with state-of-the-art accuracy. 2015.

[4] Andrew D. Martin, Kevin M. Quinn, Theodore W. Ruger, and Pauline T. Kim. Competing approaches to predicting supreme court decision making. *Perspectives on Politics*, 2(4):761–767, 2004.

[5] Guimerà R and Sales-Pardo M. Justice blocks and predictability of u.s. supreme court votes. 2011.

[6] *SCDB Dataset*. http://scdb.wustl.edu/data.php.

[7] *LSTM Basics*. http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[8] Nay JJ. Predicting and understanding law-making with word vectors and an ensemble model. 2017.

[9] Aaron Kaufman, Peter Kraft, and Maya Sen. Improving supreme court forecasting using boosted decision trees. 2017.

[10] Bryce J. Dietrich, Ryan D. Enos, and Maya Sen. Emotional arousal predicts voting on the u.s. supreme court. 2015.

[11] Daniel L. Chen, Manoj Kumar, Vishal Motwani, and Phil Yeres. Is justice really blind? and is it also deaf? 2016.

[12] Sharan Agrawal, Elliott Ash, Daniel Chen, Simranjyot Singh Gill, Amanpreet Singh, and Karthik Venkatesan. Affirm or reverse? using machine learning to help judges write opinions. 2017.

[13] Blackman J, Aft A, and Carpenter C. Fantasyscotus: Crowdsourcing a prediction market for the supreme court. 2012.