Evaluating Classification Systems on Texas State Legal Text

Abby Krishnan
abbykrishnan@utexas.edu

Mahathi Chillara
mahathi.chillara@utexas.edu

Abstract

This paper conducts a comparative study on the performance of various machine learning approaches for classifying text from the 86th Texas State House Legislature into topics, specifically, committee areas. Using a data set of 4,763 Texas State House bills, we looked into how different modern NLP methods compare against traditional machine learning models when applied to a legal corpus with lengthy documents. We tested a few major approaches such as a baseline perceptron, a heuristically improved perceptrons, a convolutional neural network, as well as a BERT-based neural network. While we achieved fairly good performance from these models, we determined that there is more work to be done to optimize NLP models for the legal domain.

1 Introduction

Every bill introduced to the Texas State House Legislature is assigned to a committee for further deliberation on the basis of topic. These topic areas are important, as the members of a given committee are often specialized to that content area. Though we specifically looked at committees, creating an generalized efficient legal topic classifier would allow legislation to be provided to the right experts and for previous, similar legislation to be efficiently retrieved. The system can also predict topics for all new text that is passed through it.

Despite its ability to greatly simplify legal text and jargon for the common man, NLP on legal text remains relatively unexplored. There exist few libraries and out-of-the-box models for legal analysis, in comparison to more popular NLP fields like chatbots and question-answering. This provided the motivation for us to explore how we could apply different models to Texas legal text. This would provide the ability for people to more easily learn about local politics in their specific content areas without having to spend large amounts of time parsing through legal jargon.

This paper uses a dataset of 4,763 Texas State House bills and their associated committee. Our specific research question was inspired by Soh et al. (2019), which asked a similar question to the one we asked: how do state-of-the-art NLP models compare against traditional machine learning models in legal text classification?

Our answer was similar to the one in the previously mentioned paper: traditional machine learning models did as well in our experiment, but only when adapted to use many legal heuristics. There are still a lot of challenges in legal text classification. The state-of-the-art models seem to be fit specifically to shorter length documents and we believe a more conversational type of language, which lends itself very well to the word embeddings. However, just as the Soh paper postulated, because the simple machine learning model performs so well just with extra text extraction methods, we believe that there is a lot of quality information in legal texts that can be extracted and dealt with in a smarter way to make deep learning more effective.

2 Problem Description

Classifying legal documents is a difficult task as it includes language that does not always sound "natural." There is an overuse of legal jargon that can often muddle the understanding of the text. We wanted to see how the application of different classification techniques, specifically, perceptron, perceptron enhanced with basic pattern-recognition tactics, convolutional neural networks, and BERT would fare against each other.

In order to make this a multi-class classification problem, we took each bill and its respective committee. We considered using the given "topic" by

the State of Texas, but determined this was far too varied to be a worthy task. While committees are traditionally determined by humans, we think that this experiment still helped us uncover knowledge that is relevant for legal NLP.

3 Data

The corpus includes all bills introduced to the Texas House of Representatives in the 86th Legislative Session (2019)[1]. It contains the full text of $4,763$ bills, along with the committees they were assigned to by the Speaker of the House. While each bill had a list of subjects assigned to it as well, the committee of the bill was designated as the label for this paper. This was done in order to decrease the amount of variance in the labels and limit the problem to one of multi-class classification rather than multi-label classification.

The legislative session had 43 distinct committees. However, the data-set was limited to 30 different labels as some committees were either subcommittees or did not have any bills assigned to them during the session.

4 Challenges

There were many challenges we dealt with when cleaning and classifying this data.

4.1 Class Imbalance

Although there were 38 committees (not including subcommittees), only 30 had bills assigned to them. Initially, this created artificially low F1 values for us. Additionally, because there was not an even distribution of bills across each committee, it was difficult for each class to get a sufficient amount of training. As a result, it became virtually impossible to get those right during testing. Given this, we ran every model through a full $\frac{1}{5}$ cross-validation split to deal with instances in which one split may not have specific labels in it, and chose the best result.

4.2 Lack of High Quality Data

The data-set we used was put together by us by scraping the government web-pages. It was not a standardized detest designed for a classification project. We had fewer data examples than preferred (the original paper used about 6000), and they varied greatly in size.

[1]The dataset is available through https://capitol.texas.gov/

4.3 Document Length

Traditionally, state-of-the-art NLP models are designed to run classification tasks on shorter documents. The documents we were working with were very long. The longest document had 108,095 tokens, while the shortest had 48 tokens, making it difficult to set a reasonable limit on the number of tokens analyzed. The median length was 413.5, but the standard deviation in token length was an astonishing 2923.14. This created a very varied, and difficult dataset to work with, and the varying length created an even greater problem during deep learning.

4.4 Lack of Control in Deep Learning

Much of the performance upgrade we noticed on basic machine learning models like the perceptron was from incorporating basic natural language heuristics (TF-IDF weighting) and legal heuristics (LexNLP library). For the deep learning tasks, they were oriented around the internal word-embedding structure, which gave us a lack of granular control beyond hyperparameters to improve the models.

5 Models

In our project, we explored 3 different types of models - perceptron (basic and enhanced), a convolutional neural network and BERT.

5.1 Baseline Models

$perceptron_{base}$ is a relatively simple classifier that does not do anything clever to improve accuracy. It simply runs the perceptron algorithm on all the tokens in the text, with punctuation and stop-words removed. The initial weights were the count of each token in the document. This performs at about 73% accuracy, showing how we can achieve significantly better than guessing with a simple model that required very little preparation, cleaning or implementation time.

5.2 Improving Baseline with Heuristics

$perceptron_{legal}$ is also a perceptron model, but here, we do significantly more preprocessing than before.

First, we calculated term frequency-inverse document frequency and used these values as the initial weights for the model. This was very beneficial in reducing the importance of words like "Texas", "section, "law", or "person". This essentially helped us get rid of the "stop-words" of legal

text—words that are so common, they add little meaning.

Second, we also used pattern-based recognition for sections/chapters, descriptions, summaries to pull out words that we knew were important and contained important information about the text. This helped us manually adjust initial weights based on what we knew about the text, proving that there is a lot of surface level information in legal text that can be extracted without any deep learning.

Third, we used an open-sourced library called LexNLP to pull out features that were specific laws. For example, using this library, we could extract the term *"20 usc Section 6399"* and use it as a feature. This specific term refers to a specific part of US Code describing state educational agencies, which ended up being an important feature for public education committees. LexNLP is still in its development stages, and seems to be missing a lot of important features, like a meaningful and documented tokenizer to pull out all legal specific language.

We also tried many other things that did not help the performance of the enhanced perceptron. This included lemmatizing the words and pulling out bi-grams alongside the current features.

5.3 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are most commonly used for image classification and are made of convolutional layers that scan an image for features. However, CNNs can also be used for text classification if we view convolutional layers as sliding windows that look at n-grams within a text.

Like the legal classification paper our project is based on, we decided to use pre-trained word vectors in order to improve performance on CNNs. For this project, we used publicly available pre-trained GloVe vectors (Pennington et al., 2014) of size 300 and relativized the vectors to only include the vocabulary relevant to our data-set. The pre-processing for the data in this model was similar to the $base\_perceptron$ model in that all the text was converted into lowercase and stripped of punctuation.

The data-set was trained through 6 epochs with a learning rate of 4×10^{-3} and a batch size of 1. The batch size was set as 1 as any higher batch size created memory issues locally. The learning rate was modified to accommodate for the lower batch size. In addition, the number of epochs was decreased to 6 through trial and error. While the loss continued to decrease with 15 epochs, there seemed to be no significant improvement in accuracy beyond 6 epochs.

Each bill example was sent in batches which were padded to the maximum bill length. Since the median document length varied significantly, we trimmed all documents to a maximum size of 1500 tokens. The batches were first sent through an embedding layer to convert tokens into word embeddings. Tokens that did not have a word embedding were initialized to a randomized word vector. None of the word embeddings were fine-tuned during training.

Our implementation of a convolutional neural network uses PyTorch and is based on Kim (2014)'s model. We passed each batch through a convolutional layer that used multiple filters with various window sizes to get feature maps. After using rectified linear units on the feature maps, we applied a max pooling operation over them to create a feature vector of the filter outputs. Finally, we performed dropout regularization on the feature vector, then passed it through a linear layer to classify the predictions into labels.

For the neural network, we used filters of sizes 3, 4 and 5, with 100 feature maps each and a dropout rate of 0.5, the hyperparameters stated in Kim's paper. We attempted to fine-tune these parameters to optimize performance but did not find any improvements after much trial and error. In fact, the legal classification paper used filters of 3, 3, 3 and 600 feature maps, but we found that this lowered our performance.

5.4 BERT

For our final model, we decided to test a pre-trained language model like BERT (Devlin et al., 2019) to evaluate its performance on legal data. This model required less cleaning and seemed more like an 'out of the box' model.

Before sending the bills into BERT's predefined tokenizer, we preprocessed the documents to remove punctuation. Since BERT only takes in up to 512 tokens, we also trimmed all the documents to the first 512 tokens after preprocessing them. We used HuggingFace's[2] implementation of $bert_{base}$

[2] https://github.com/huggingface/transformers/

Model	Accuracy	Precision	Recall	F1 Score
$perceptron_{base}$	74.0	70.7	67.7	69.2
$perceptron_{legal}$	80.2	76.1	73.3	74.7
CNN	79.3	78.0	71.7	74.7
$bert_{base}$	82.3	73.2	73.7	73.4

Table 1: Model performance

(12-layers and 110M parameters) in Pytorch and trained the examples over 4 epochs with a batch size of 16 and a learning rate of 3×10^{-5}.

6 Testing

To make models more robust and prevent overfitting, we used K-Fold cross validation to separate data points randomly into train and test sets. Then, we averaged the testing accuracies from each fold in a model and chose the model with the highest accuracy for analysis.

7 Results

As can be seen in Table 1, we evaluated the models on accuracy, precision, recall, and F1.

In our experiments, while the base perceptron lags behind all the others, the enhanced perceptron put up impressive results. It consistently outperformed CNN in accuracy and matched it in F1 scoring. Considering that it runs in a fraction of the time, it shows how deep learning lags behind in adapting itself to legal text.

Figure 1 demonstrates the confusion matrix for every model. For each model, the class with best predictive performance were 23, 24, and 29. These are also the classes with the largest amount of bills (439, 326, 308) associated with it, so our training was very effective, given a greater amount of data. Similarly, the worst performing classes were often 13 and 25, which had very few bills (18, 7) associated with it. This furthers the difficulty of prediction with such a large class imbalance.

We predict that both BERT and CNN do not provide significant performance gains for a few reasons.

(1) It lacks the adaptability to longer text. Even when increasing the maximum length of the CNN, with that extra information it was not able to make better guesses on the legal text.

(2) A word embeddings based approach may not be the best approach for legal text. The "nearest neighbors" approach of GLoVe may lag behind in legal text because the extra, unnecessary jargon that neighbors many of the tokens in legal text does not provide meaningful comprehension to the model. Legal text lacks a "natural" way of speaking, is repetitive, and hard to decipher, which does not lend itself well to a word vector representation like GLoVe.

Additionally, though it does require many hours of preprocessing, the ability to pull out the regulations, use tf-idf to de-rank words that carry little meaning in legal text was significant in giving insight into the most influential words for a specific class.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Jerrold Soh, How Khang Lim, and Ian Ernst Chai. 2019. Legal area classification: A comparative study of text classifiers on singapore supreme court judgments. In *Proceedings of the Natural Legal Language Processing Workshop 2019*, pages 67–77, Minneapolis, Minnesota. Association for Computational Linguistics.

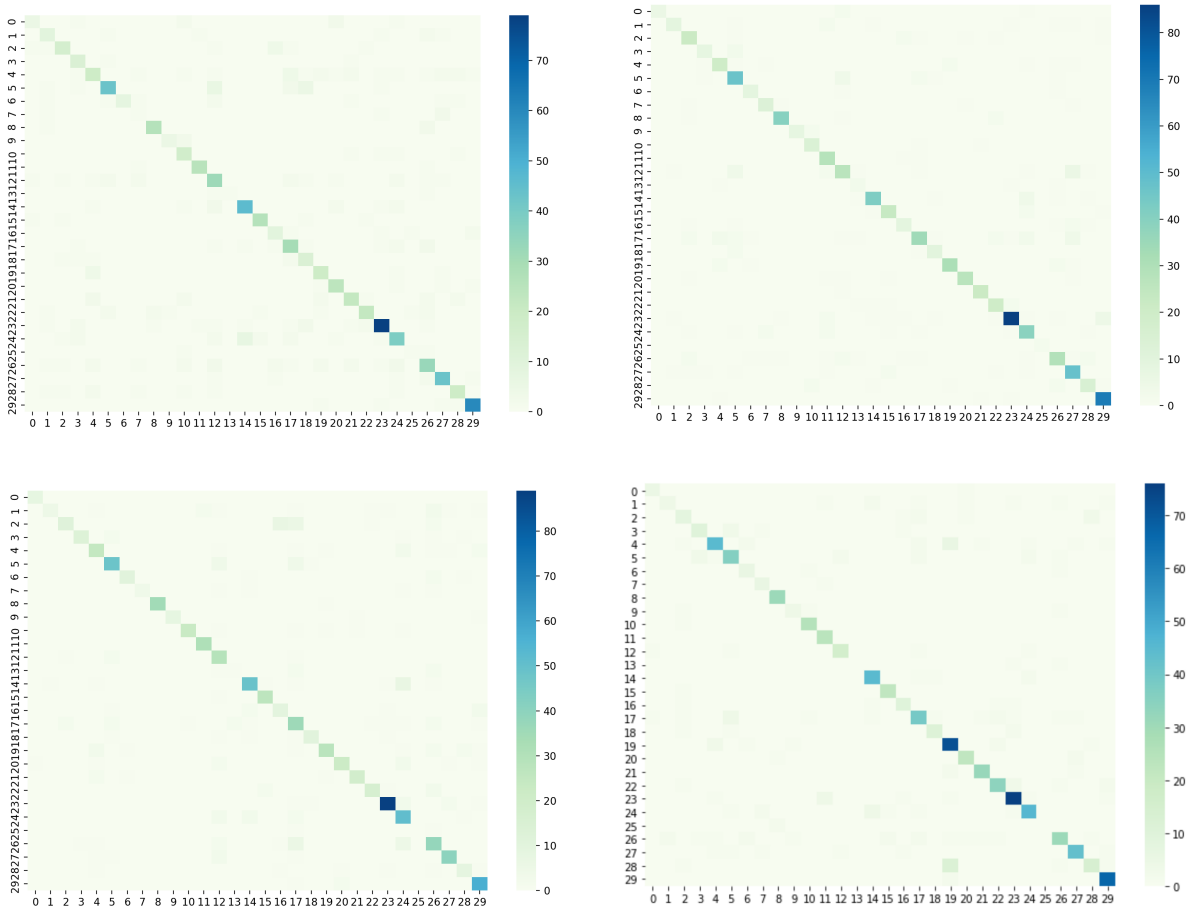

Figure 1: Confusion Matrix for Base Perceptron, Enhanced Perceptron, CNN, BERT (left to right)