DOI: 10.15439/2018F227
ISSN 2300-5963 ACSIS, Vol. 15

A Comparative Study of Classifying Legal Documents with Neural Networks

Samir Undavia, Adam Meyers, John E. Ortega
New York University
60 5th Avenue
New York, New York 10011, USA
Email: {su478,meyers,jortega}@cs.nyu.edu

***Abstract*—In recent years, deep learning has shown promising results when used in the field of natural language processing (NLP). Neural networks (NNs) such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs) have been used for various NLP tasks including sentiment analysis, information retrieval, and document classification. In this paper, we the present the Supreme Court Classifier (SCC), a system that applies these methods to the problem of document classification of legal court opinions. We compare methods using traditional machine learning with recent NN-based methods. We also present a CNN used with pre-trained word vectors which shows improvements over the state-of-the-art applied to our dataset. We train and evaluate our system using the Washington University School of Law Supreme Court Database (SCDB). Our best system (word2vec + CNN) achieves 72.4% accuracy when classifying the court decisions into 15 broad SCDB categories and 31.9% accuracy when classifying among 279 finer-grained SCDB categories.**

I. Introduction

Legal court opinions are lawful statements written by a judge providing the justification and legal reasoning for a court ruling. This paper describes an automated document classification model implemented as our Supreme Court Classifier (SCC) system. In theory, SCC could make obsolete many time-consuming manual tasks requiring legal experts. A legal expert would need to read hundreds or thousands of documents in order to place opinions into subject categories, whereas an automatic system like SCC could do this with little or no human effort.[1]

Some document classification efforts, particularly, those using unsupervised approaches, evaluate output based on human evaluation of automatically derived categories. However, when automatic document classification is based on human-defined categories, the results are, arguably, more "natural." Evaluation tends to be more straightforward with human-annotated classifications because it is usually easy for a human being to tell whether or not a document belongs to a human-defined category. In contrast, this determination is harder to make with purely unsupervised methods (e.g., topic modeling [1]), unless a manual component is added. For example, aligning automatically defined categories with some set of human categories will produce clearer results. Human-defined categories have names and notional definitions such as *Criminal Procedures*, *Civil Rights*, and *Federal Taxation* [2].

In contrast, automatically classified categories are usually defined as sets of keywords or other more oblique definitions using words in the corpus. For instance, the case *Roe v. Wade*, 410 U.S. 113 (1973)[2], may fit into a class labeled by a set of keywords like *abortion, reproduction, medical, ...* Although these words describe the case, they do not correctly encapsulate the legal significance of the case. Roe v. Wade would be classified under the *Privacy* legal issue and the *abortion: including contraceptives* sub-issue [2]. Unfortunately, it may be difficult for a human to decide what the boundaries of the classes are defined by these keyword sets. On the other hand, it may be possible to align the output of unsupervised classification with a manual set of categories. In fact, we do this for our baseline system that uses latent Dirichlet allocation (LDA) to model documents and then a logistic regression (LR) [3] model to align the results with the manual categories (see Section IV-A).

While categories can be extracted in many ways, we believe that some sort of human validation is preferable. Moreover, the legal domain often requires documents to be aligned with human-defined legal categories for a majority of legal tasks. For this reason, we have implemented a system that uses pre-defined document categories (from human evaluators) to train a model for classifying legal texts. Specifically, SCC automatically classifies legal opinions from cases seen by the Supreme Court of the United States (SCOTUS), manually classified into topics as part of the Supreme Court Database (SCDB) by Washington University School of Law [2]. Henceforth, we will refer to these as the SCDB categories. The SCDB categories are defined in Section III-A.

Our work systematically tests the application of a variety of machine learning (ML) techniques to automate semantic classification of SCOTUS legal opinions. Our most successful systems are based on neural networks (NNs). NNs are used to solve a variety of natural language processing (NLP) tasks because of their ability to extract relevant information from

[1]SCC can be downloaded from https://github.com/samir1/web_of_law_scotus_classification/ under an Apache 2.0 license (https://www.apache.org/licenses/LICENSE-2.0). In addition to computer code, the repository includes our training/development/test split of the SCDB data, ensuring that our results are both reproducible and comparable to future work.

[2]Roe v. Wade is a supreme court case that is famous in the United States for causing abortion to become legal.

text without having to specify features for any particular domain [4]. We examine two common NN architectures: the convolutional neural network (CNN) [5] and the recurrent neural network (RNN) [6], much like the medical text classification experiments using CNNs conducted by Hughes et al. [7]. Initially used for classifying images, variations of the original CNN architecture are used for NLP tasks [8]. Two main variations of the RNN, long short-term memory (LSTM) [9] and gated recurrent unit (GRU) [10], have recent successful results in their application to sequence modeling [11], [12].

We compare a series of different CNN and RNN architectures to documents represented by word embeddings. We show that our CNN performs better with the legal corpus than the other models implemented. SCC uses neural networks to select one of the SCDB categories for each supreme court case in our validation corpus.

In this paper, we use CNNs and RNNs to classify legal documents with minimal pre-processing, in contrast to other machine learning approaches (e.g., support vector machines) that require manually specifying features for classification or manually determining key words [13]. Any additional pre-processing that could potentially improve performance requires manual editing since each document contains slightly different formatting resulting from OCR errors from scans of printed documents. We measure the efficacy of NN classification techniques applied to our corpus and show that our CNN outperforms RNNs for legal text classification based on SCDB categories (see Table I in Section V). In order to apply our classification models to text, we first represent each word in our corpus as a word embedding (vector representations of words were generated using an unsupervised learning method from Mikolov et al. [14]). We use the publicly available pre-trained word vectors trained on about 100 billion words from part of the Google News dataset.[3] We use 300-dimensional word2vec vectors trained using the skip-gram architecture with negative sampling by Mikolov et al. [14]. We map each word to a word vector and use neural network classifiers on the dataset. Additionally, we present results from using two other word embedding models, fastText [15] and GloVe [16], with our CNN (our best system). We describe the neural neural network models in Section IV and results in Table I.

II. Related Work

We have found a relatively small body of previous work about automatic text classification of legal documents. For example, support vector machines (SVMs) have been used to classify legal documents like legal docket entries [17] and to classify non-English legal texts [4]. Although our work also examines the application of machine learning to a corpus written in the legal context, we focus on classifying SCOTUS legal opinions without significant pre-processing. For example, the Nallapati and Manning [17] system includes several steps of pre-processing before using an SVM to classify documents with human-selected features and labels. We explore more recent automated document classification techniques that do not rely on significant pre-processing and human interaction. Moreover, we present a comparison of different machine learning techniques in order to determine methods with the highest performance for our task.

Our work on SCC is similar to the approach of Wood et al. [1] for classifying medical summaries in that we model our corpus using LDA and classify with pre-defined labels (see Section IV-A). In that study, an initial topic model was derived from some training documents. Then the topic model was modified with pre-labeled data in order to classify the new data. Likewise, we use a combination of LDA and pre-labeled legal opinions to create our baseline classification system. We compare the results of our NN-based classification results to our application of LDA and an LR classification.

Domain-specific automated document classification has been applied to several fields, including electronic medical records. Hughes et al. [7] use convolutional neural networks to detect features for sentence-level classification of medical texts, resulting in a much smaller input compared to our experiments. Unlike SCC, in which we feed an entire document into a neural network, the Hughes et al. [7] model classifies texts by first transposing each document into a matrix of sentences with fixed lengths. Their model also differs from ours in that their model is essentially two sets of two stacked convolutional layers followed by a pooling layer, whereas our model does not have any consecutive convolutional layers. Additionally, one of our experiments (following Hughes et al. [7]) tests the effectiveness of using doc2vec embeddings with an LR model as the classifier. Similarly, Weng et al. [18] use medical texts as the subject of their classification task, although they use a different neural network architecture to classify health documents. Weng et al. [18] apply a complex combination of CNN and RNN architectures to clinical note text classification; their model is summarized by three convolutional and pooling layers followed by a bidirectional LSTM [18]. Moreover, Yin et al. [19] present a comparison of different neural network architectures used to complete a variety of NLP tasks. Such tasks include sentiment analysis, document classification, and part-of-speech tagging. In their text classification experiment, Yin et al. [19] use a pre-labeled set of 10,717 sentences evenly distributed over 19 labels, compared to our unevenly distributed dataset, in which a third of the categories have under one hundred examples and four classes have over 1,000 documents. In contrast, our experiments aim to solve the specific problem of document classification applied to legal texts. Moreover, Kim [20] describes a general CNN used to classify sentences with word2vec word embeddings. Similar to the model we propose, Kim's model also includes three convolutional and pooling layers. We optimize hyperparameters and experiment with a combination of different convolutional, pooling, and dropout layers. We compare applications of the Kim [20] and Hughes et al. [7] models to our legal corpus. We ultimately obtain improved results by using the customized CNN on the SCOTUS legal opinion corpus.

[3]https://code.google.com/archive/p/word2vec/

III. Experimental Setup

A. Dataset

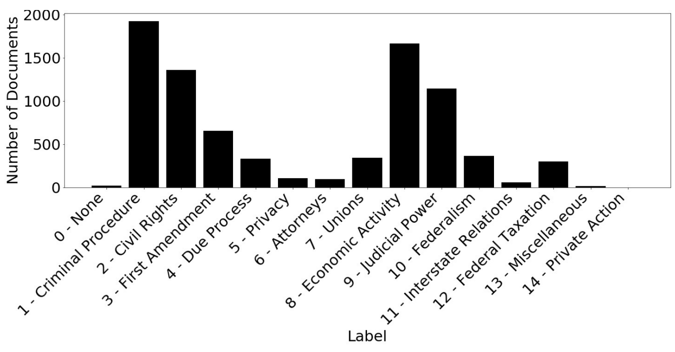

Fig. 1: 15 legal issues

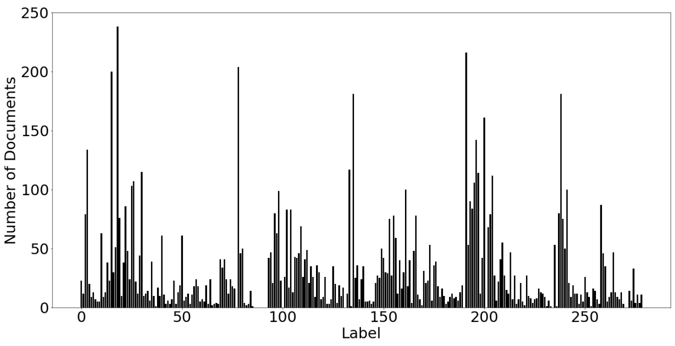

Fig. 2: 279 subtopics

We train and test our system on the manually-categorized SCOTUS legal opinion (Supreme Court Database or SCDB) corpus, from Washington University School of Law [2]. The dataset consists of 8419 US Supreme Court court opinions from "modern" cases (1946-2016), organized into 15 legal categories (Figure 1), which are further divided into 279 subtopics (Figure 2). We chose the modern dataset because both court opinions and pre-defined labels were available through the textacy Python package.[4] Textacy provides a formatted version of modern cases from FindLaw's US Supreme Court legal opinions database. The 8419 documents were randomly divided into training, validation, and test sets with a 80%/10%/10% split. Although the SCDB labels also covered "legacy" cases (1791-1945), FindLaw's database only reliably provided case text from the "modern" era of US law.

B. Initial Processing

Our system first removes (as noise) special characters that refer to footnotes. We also removed a number of characters used in formatting the original printed document. Next, each word in the corpus was mapped to a word2vec vector before being fed into a neural network for classification.

[4]http://textacy.readthedocs.io/en/latest/_modules/textacy/datasets/supreme_court.html

IV. Methods

A. LDA + Logistic Regression

Before the widespread use of neural networks for NLP tasks, probabilistic methods like LDA [21], [22], were used as a standard for a variety of NLP tasks including text classification. We use LDA as a baseline to compare the results of the NN-based classification models. Our process involves using LDA to represent each of the heavily pre-processed legal documents as a series of latent feature vectors. LDA is most commonly used to generate a collection of latent topics for a corpus, and then calculate the probability of a document belonging to a topic. We use the Gensim[5] library to create and train the LDA model. We classify vectors from the implementation of this model using LR.

The LDA Bayesian probabilistic model is an unsupervised machine learning method used to organize documents through topic modeling. In this model, each document is represented as a probability distribution over latent topics. These topics are derived from the assumption that the document's words themselves, modeled as a term frequency-inverse document frequency (tf-idf) matrix, with words represented using the bag-of-words (BoW) model, are distributed over latent topics as defined by the distribution of words in the corpus.[6]

After latent feature vectors are generated to describe each of the documents, we apply an LR to classify the documents into 15 legal issues and 279 legal subtopics. As in Wood et al. [1], we use a combination of LDA and pre-defined labels with corresponding documents to create our baseline classification system.

B. Doc2vec + Logistic Regression

Our first method of document classification using deep learning involves a higher-level application of word2vec. As described in [23], paragraph vectors, often referred to as doc2vec or document vectors, can be used to map semantic meaning from a variable-length document to a fixed-size vector. As in the word2vec learning method, a word is predicted by its neighboring words. The significant difference between the Distributed Memory Model of Paragraph Vectors and other similar learning techniques is that an additional paragraph token (treated like an additional word in the document) is used when learning the paragraph vector. Next, we classify the documents into both 15 and 279 classes using a logistic regression.

C. Bag-of-Words + Support Vector Machine

As another baseline, we represent documents using the BoW model and apply an SVM using Scikit-learn's SVM package[7] with default parameters. We chose SVM as a baseline because it is one of the highest performing traditional ML methods,

[5]https://radimrehurek.com/gensim/

[6]In order to find the ideal number of topics for the LDA-based classification, we conducted the experiment with different numbers of topics ranging from 100 to 600 in steps of 100 and and chose 300 topics because there was not a noticeable improvement in performance with more than 300 topics.

[7]http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

used for lots of different tasks. Similarly, Kim [20] also uses an SVM benchmark.

D. Word2vec + CNN

For our neural network classification approach, we designed a multi-layer model similar to the one described by Kim [20], but with additional layers and modified hyperparameters.

Our model first creates an embedding layer using pre-trained word2vec word embeddings, and then creates a matrix of documents represented by 300-dimensional word embeddings. We include three sets of the following: a dropout of 0.25, a convolution layer of 128 filters with a filter size of 5, and max pooling layer with a pooling size of 5. We also add a dense layer consisting of 128 units between two dropouts of 0.5 to prevent overfitting. The last layer is a dense layer with size equal to the number of labels for the test (15 or 279).

E. Other Embeddings

In addition to using word2vec embeddings with the CNN, we conduct the same experiments with two other pre-trained word embeddings: fastText vectors[8] from Facebook AI Research (FAIR) [15] and GloVe vectors[9] from Pennington et al. [16]. The pre-trained 300-dimensional fastText vectors are trained on Wikipedia using the skip-gram model described in Bojanowski et al. [15]. The pre-trained 300-dimensional GloVe vectors are trained on Wikipedia and the Gigaword 5 dataset using GloVe model [16].

We use the publicly available pre-trained word vectors trained on about 100 billion words from part of the Google News dataset.[10] We use 300-dimensional word2vec vectors trained using the skip-gram architecture with negative sampling by Mikolov et al. [14].

F. Word2vec + LSTM

One of the RNN-based networks we used to classify our legal corpus is the LSTM, which is defined by these equations:

$$\mathbf{i}_t = \sigma(\mathbf{x}_t\mathbf{U}^i + \mathbf{h}_{t-1}\mathbf{W}^i + \mathbf{b}_i) \quad (1)$$

$$\mathbf{f}_t = \sigma(\mathbf{x}_t\mathbf{U}^f + \mathbf{h}_{t-1}\mathbf{W}^f + \mathbf{b}_f) \quad (2)$$

$$\mathbf{o}_t = \sigma(\mathbf{x}_t\mathbf{U}^o + \mathbf{h}_{t-1}\mathbf{W}^o + \mathbf{b}_o) \quad (3)$$

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(\mathbf{x}_t\mathbf{U}^c + \mathbf{h}_{t-1}\mathbf{W}^c + \mathbf{b}_c) \quad (4)$$

$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \quad (5)$$

In this model, $\mathbf{x}_t$ represents an input x at time t. The three gates of the LSTM are the input gate $\mathbf{i}_t$, forget gate $\mathbf{f}_t$ and output gate $\mathbf{o}_t$. $\mathbf{c}_t$ is the memory cell state. $\mathbf{h}_t$ is the hidden state. The input weights are defined by $\mathbf{W}$, recurrent weights by $\mathbf{U}$, and bias by $\mathbf{b}$.

Our implementation of the LSTM consisted of the embedding layer formed with pre-trained word2vec vectors, an LSTM layer consisting of 128 units, and a dropout of 0.5 to prevent overfitting. Lastly, we had a dense layer representing the number of labels for the experiment.

G. Word2vec + GRU

Our last experiment involves the memory-enhanced GRU [10], a variation of the RNN. The GRU is described by the following equations:

$$\mathbf{z} = \sigma(\mathbf{x}_t\mathbf{U}^z + \mathbf{h}_{t-1}\mathbf{W}^z) \quad (6)$$

$$\mathbf{r} = \sigma(\mathbf{x}_t\mathbf{U}^r + \mathbf{h}_{t-1}\mathbf{W}^r) \quad (7)$$

$$\mathbf{s}_t = tanh(\mathbf{x}_t\mathbf{U}^s + (\mathbf{h}_{t-1} \circ r)\mathbf{W}^s) \quad (8)$$

$$\mathbf{h}_t = (1 - \mathbf{z}) \circ \mathbf{s}_t + \mathbf{z} \circ \mathbf{h}_{t-1} \quad (9)$$

where $\mathbf{x}_t$ represents an input vector x at time t, $\mathbf{r}$ is the reset gate, and $\mathbf{z}$ is the update gate. The input weights are defined by $\mathbf{W}$ and recurrent weights by $\mathbf{U}$.

As with our CNN and LSTM models, our application of the GRU begins with a word2vec embedding layer. Next, we include a GRU layer of size 128 and a dropout of 0.5 before the final dense layer for classification.

H. Hyperparameters and Regularization

In our experiments, we tested our model with a series of different hyperparameters and found that our best NN systems use 128 units for the RNNs and 128 filters for each of the convolutional layers in the CNN. For both these settings, we tried values of 32, 64, 128 and 256 and 128 gave the best results. Basically, the 128 gave better results than the lower settings and it turned out that the 256 setting could not be run effectively when training with an Nvidia GPU. It seems that additional GPU memory would be required (or a more efficient algorithm) to use 256 units. It is probable that 128 is simply the largest (power of 2) setting that is practical to use given the available equipment. This seems to be supported by the fact that many other NN systems (e.g. Kim [20]) use a value around 100.

Additionally, each of the models are regularized with a dropout [24], which works by "dropping out" a proportion p of hidden units during training. We found that a dropout of 0.5 before the final dense layer and batch size of 32 worked best for the LSTM, GRU, and CNN. We also found that the Adam optimizer [25] worked best for both the for CNN and RNN networks.

V. Results

Our goal is to determine the best method for applying automated document classification to legal texts with the hopes of facilitating legal experts in their classification of court documents. Our experiments look not only at comparing existing networks, but also at developing our own superior network. As shown in Figure I, our CNN model achieves the

[8]https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

[9]https://nlp.stanford.edu/projects/glove/

[10]https://code.google.com/archive/p/word2vec/

highest accuracy, both for the 15-label and 279-label tasks (72.4% and 31.9% accuracies, respectively). We present an analysis of our results.

TABLE I: Classification Accuracy Results on the Test Set

Model	15 labels	279 labels
Word2vec + CNN	**72.4**	**31.9**
fastText + CNN	67.3	25.1
GloVe + CNN	67.1	17.7
Word2vec + GRU	68.6	14.5
Word2vec + LSTM	43.8	6.5
Word2vec + CNN (Kim [20])	65.9	14.7
Word2vec + CNN (Hughes et al. [7])	54.7	19.8
LDA + LogR [21]	40.3	13.4
Doc2vec + LogR [23]	54.1	28.6
BoW + SVM	64.0	30.5

The accuracy results of correctly classifying documents with our CNN, LSTM, and GRU models compared to other classification methods. Our CNN best overcomes the problem of an uneven distribution of classes, as shown in Figures 1 and 2.

TABLE II: CNN Results on the Development Set by Category

Label	Precision	Recall	F-Score	# of Docs
0 - None	0.33	0.33	0.33	3
1 - Criminal Procedure	0.82	0.91	0.86	182
2 - Civil Rights	0.72	0.70	0.71	138
3 - First Amendment	0.87	0.79	0.82	84
4 - Due Process	0.45	0.33	0.38	30
5 - Privacy	0.56	0.62	0.59	8
6 - Attorneys	0.33	0.40	0.36	5
7 - Unions	0.73	0.76	0.75	29
8 - Economic Activity	0.72	0.72	0.72	172
9 - Judicial Power	0.57	0.56	0.56	116
10 - Federalism	0.41	0.41	0.41	34
11 - Interstate Relations	0.67	0.67	0.67	6
12 - Federal Taxation	0.90	0.79	0.84	33
13 - Miscellaneous	0.50	0.50	0.50	2
14 - Private Action	0.00	0.00	0.00	0
Avg/Total	0.71	0.72	0.71	842

The relation between frequency and f-measure for the development set.

TABLE III: CNN Results on the Test Set by Category

Label	Precision	Recall	F-Score	# of Docs
0 - None	0.20	1.00	0.33	1
1 - Criminal Procedure	0.81	0.85	0.83	183
2 - Civil Rights	0.77	0.81	0.79	121
3 - First Amendment	0.79	0.88	0.83	56
4 - Due Process	0.48	0.30	0.37	33
5 - Privacy	0.57	0.44	0.50	9
6 - Attorneys	0.80	0.73	0.76	11
7 - Unions	0.77	0.70	0.73	33
8 - Economic Activity	0.72	0.74	0.73	145
9 - Judicial Power	0.56	0.54	0.55	102
10 - Federalism	0.48	0.33	0.39	33
11 - Interstate Relations	0.50	0.40	0.44	5
12 - Federal Taxation	0.86	0.96	0.91	25
13 - Miscellaneous	0.00	0.00	0.00	1
14 - Private Action	0.00	0.00	0.00	0
Avg/Total	0.72	0.72	0.72	758

The relation between frequency and f-measure for the test set.

Tables II and III show the CNN's (our best system) performance on individual classes. Although the details are slightly different (e.g., a different number of documents for each category), the relative scores are about the same. We now do a more detailed analysis on the development set results, rather than the test set because we do not want to examine the test results too closely and bias our future work. It is clear that the model's accuracy tends to be higher for the most frequent categories. Categories 1, 2, 3, 8 and 9, all of which are labels on more than 50 documents, mostly have f-measures of over 70%, with one outlier at 56%. It is difficult to generalize about the least frequent categories (frequency < 10), including labels 0, 5, 6, 11, 13 and 14, as there is too little data to analyze. Some of these have f-measures of 0 or near 0, and on average, they do much worse than the high-frequency categories. This is expected since the high-frequency categories have more training data and thus provide more evidence for the model to build on. Thus, as expected, cases with correct labels of 1, 2, 3, 8 and 9 tend to be classified correctly and the categories with little to no training data (0, 5, 6, 11, 13, 14) are most often misclassified.

On the other hand, category labels 4, 7, 10, and 12 each have a similar moderate number of test documents (around 30 documents), but have very different results: the model achieves relatively high results for labels 7 and 12 and relatively poor results for 4 and 10. Thus it would seem that the results for labels 4, 7, 10 and 12 cannot be explained purely on the basis of frequency. Figure 3 is a confusion matrix for our CNN results on the development/validation set. We observe some patterns which may help us understand these results. For labels 7 and 12, where the model performs well, the correct category clearly dominates–no other category is marked for more than 4 documents. However, the poorly performing categories, each have a second (or third) dominant category in addition to the correct one. Label 4 (*Due Process*) is applied to 10 true *Due Process* cases and incorrectly classifies 7 as *Civil Rights* cases, 6 as *Economic Activity* cases and 4 as *Criminal Procedure* cases and another 3 miscellaneous erroneous labels. To the extent that a case may be given multiple classifications (*Due Process* and *Civil Rights*) or (*Due Process* and *Economic Activity*), these errors are understandable and may even reflect a defect in the experiment–perhaps cases should have multiple classifications and the one classification per case assumption is unrealistic. Similarly, label 10 (*Federalism*) is applied 14 times correctly to *Federalism* cases and 11 times incorrectly to *Economic Activity* cases (and rarely to other categories). It is expected that some federalist issues (issues about the power of the federal government) will overlap with economic issues. So these may also be understandable errors.

We now attempt to better understand these errors, focusing our error analysis on *Federalism* classification. We examine three samples from our development set, each sample consisted of four cases. We look at 4 cases that are correctly classified by our CNN as *Federalism* cases (True_Fed); 4 cases that were correctly classified as *Economic Activity* cases (True_Eco); and 4 *Federalism* cases that our system misclassified as *Economic Activity* cases (False_Fed). We compare the False_Fed cases to both True_Fed and True_Eco. Our goal is to understand the sort of factors that might cause a human or a machine learning algorithm to mis-classify the False_Fed documents.

The True_Fed cases include *Testa et al. v. Katt*; *Bethlehem Steel Co. et al. v. New York State Labor Relations Board*; *Rice et al. v. Santa Fe Elevator Corp. et al.*; and *Rice et al. v. Board of Trade of the City of Chicago*. These all involved the interaction of state and federal authorities and laws, including questions of whether a state authority should be compelled to enforce a federal law or whether a state agency/law takes precedence over a federal agency/law.

The 4 True_Eco cases included *Halliburton Oil Well Cementing Co. v. Walker et al.*; *Champlin Refining Co. v. United States et al.*; *United States v. Howard P. Foley Co., Inc.*; and *Richfield Oil Corp. v. State Board of Equalization*. The Halliburton case is a patent dispute. The Foley case is about the government's liability in a contract dispute. The Champlin case examines whether the Interstate Commerce Commission, a federal entity, could require information from an oil refining company operating across several states. While similar to the True_Fed cases in some ways, there is no conflict between a state and a federal authority. The Richfield case is a dispute about whether a state sales tax applies to a sale to a foreign government. This seems similar to Federalist concerns, but there is no conflict between a state and federal authority. Rather the concern is whether or not a state sales tax effects a possibly-foreign transaction.

The False_Fed cases include *Phillips Chemical Co. v. Dumas Independent School District*; *Panhandle Eastern Pipe Line Co. v. Michigan Public Service Commission et al.*; *Wyeth v. Diana Levine*; and *North Dakota v. United States*. These cases all involve financial transactions, state authorities and federal authorities. The topics covered includes: the legitimacy of state taxes on federal land leased to a company; state regulation of alcohol and other goods procured for sale on a federal military base; liability of a drug company (in state civil court) for damages from harm by their drug, even though the FDA, a federal agency, granted them clearance for the drug; and whether the sale of natural gas was subject to state regulation, in spite of a federal law licensing the sale. While some of these issues seem to include federal/state authority conflicts, those conflicts are not as clearly articulated as in the True_Fed cases. So it is clear how experienced annotators may be able to consistently distinguish the *Federalism* and *Economic Activity* classes. However, we would imagine that inexperienced annotators may have trouble.[11] Similarly, machine learning may require more evidence (more documents) to correctly classify these cases.

The sparsity and imbalanced classes of the dataset presented itself as the most challenging obstacle for training the neural networks. For instance, nearly three fourths of all documents fell under 4 of the 15 legal area categories (*Criminal Procedure*, *Civil Rights*, *Economic Activity*, and *Judicial Power*). Not only was there not an even distribution of documents over each of the labels, many of the classes had little to no training data. Furthermore, our input sequence length was several

[11]We would expect lower inter-annotator agreement on these sort of cases.

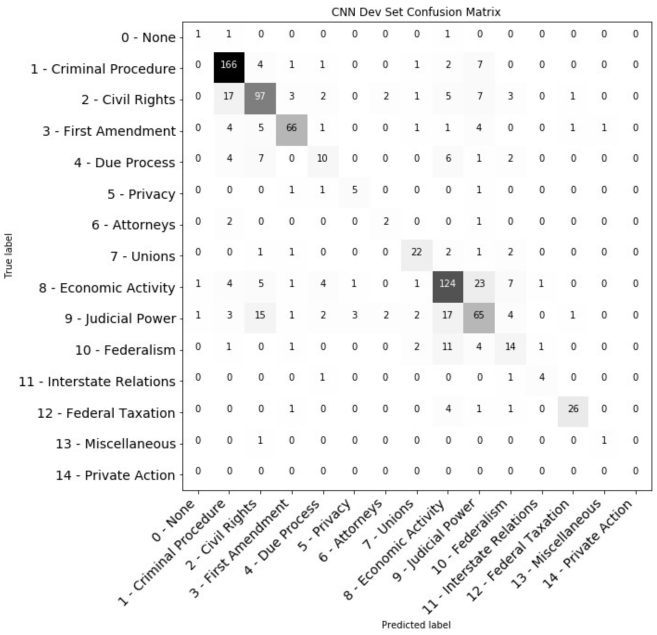

Fig. 3: Confusion Matrix for CNN Development Corpus

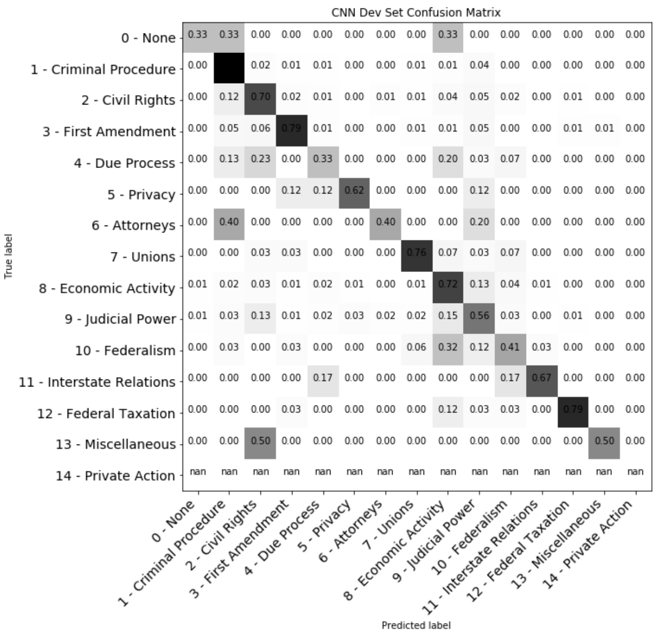

Fig. 4: Normalized Confusion Matrix for CNN Development Corpus

orders of magnitude larger than the inputs in experiments conducted by Kim [20] and Hughes et al. [7].

Due to adjustments we made for the dataset and configuration changes (see Section IV), our CNN performs better than other CNN models (see Table I). The adjusted parameters include dropouts to account for overfitting that occurs earlier in training and the addition of extra layers.

After generating a model for each of our NN-based tests, we use them for our entire corpus and analyze the documents that are misclassified in order to find patterns in the way each of NN architectures complete the classification task. Figure 5 shows a combination of the normalized confusion matrices resulting from classifying our entire corpus with our CNN and the simple RNN models, and each of the true label rows describe the fraction of documents classified per predicted label. As in Figure 5, the CNN performs the best for each of the categories, not just the top four labels (although a small number of errors made by the CNN were with documents from these classes). Unlike our CNN, the classifications from the GRU (our second best system) are more scattered. Figure 5 shows some of labels the GRU most frequently misclassifies. In contrast to the CNN and LSTM, the GRU tends to significantly mis-classify documents from both frequent and infrequent categories, as in the frequent category of *Criminal Procedure* and the uncommon category of *First Amendment*. The GRU also does not classify entire categories correctly, whereas the CNN had high classification accuracy for every category. For example, the GRU mis-classifies every legal opinion in the categories: *Interstate Relations*, *Miscellaneous*, and *Private Action*. Additionally, there was no single label L, such that our GRU system correctly classified more documents in class L than our CNN system.

It seems that the relatively low frequencies of some of the categories is more of a challenge to some of the learning algorithms than others. In particular, CNN and GRU appear to be somewhat more resilient to this effect, than LSTM, as evidence by the merged confusion matrices shown in Figure 5. The LSTM incorrectly classified a majority of the documents to one of the two most frequent labels (*Criminal Procedure* and *Economic Activity*), as shown in Figure 5. Despite categorizing almost all of the documents to only two legal issues, the LSTM did not have a higher accuracy for those two labels. In fact, the LSTM did worse than our LDA baseline system (see Table I). This was somewhat surprising considering that LSTMs often perform similarly to GRUs for a given task. We plan to investigate this further in future research, possibly trying additional models.

As in Chung et al. [11], we test the performance of LSTMs and GRUs. While we apply these models to categorizing legal text, their application was music transcription. Our results show that the structure of the simpler GRU leads to greater accuracy compared to the LSTM when classifying a relatively small number of documents over a large number of labels. The GRU performs better than the LSTM with document-length sequences. The LSTM tends to remember the wrong information needed for the classification because of the small size of the dataset.

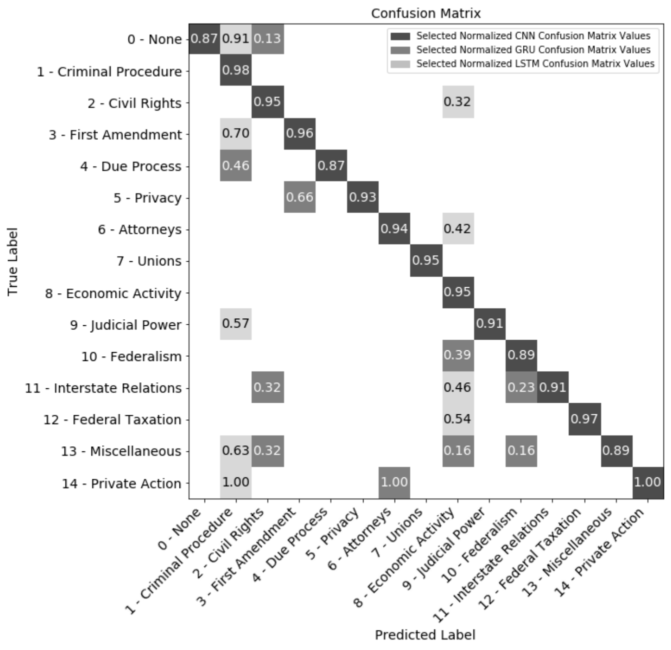

Fig. 5: Merged Confusion Matrix for CNN, GRU, and LSTM

Our results also show that the simple BoW+SVM model performs very well for both classification tasks. As Nallapati and Manning [17] mention, "the SVM assigns high weights to many spurious features owing to their strong correlation with the class" [17, p. 442]. In other words, even very infrequent words that have very high correlations with certain classes would help the SVM associate certain words with uncommon classes. This aspect of the SVM seems to explain why the SVM performed well with the 279-label classification task (nearly as well as the best system), in which only a few documents define each category.

In our results, the word2vec model out-performs the simpler bag-of-words model. With more statistical information, the classifiers find common features and patterns to describe categories with higher accuracy. The positive results from our experiment in which we apply an LR to paragraph vectors (doc2vec) show how well the embeddings capture the meaning of the documents (refer to Table I).

The simple GRU network has promising results compared to the CNNs because the GRU is designed to handle long sequences. While a word may carry a large weight in most contexts, the GRU allows for a word's weight to diminish based on specific examples. Yin et al. [19] shows that the accuracy of the CNN decreases as sequence length increases and eventually falls under the accuracy of the GRU. Since our experiments involve inputting entire documents instead of sentences, sequence lengths are orders of magnitude larger than than those used in the experiments conducted by Yin et al. [19], Kim [20], and Hughes et al. [7].

VI. Conclusion and Future Work

In this paper, we find the best method for automated legal document classification is the SCC system that uses a CNN (72.4% accuracy for 15 general categories and 31.9% accuracy for the 279 more specific categories). On the other hand, the GRU architecture shows promising results compared to our tuned CNN (nearly as high for the 15 category task). We believe that a tuned GRU-based network can potentially complete the task with higher accuracy.

The SCC system uses word embeddings from a general domain (Google News). It is possible that embeddings from the legal domain would improve results. Accordingly, we plan to compile a much larger corpus of US legal opinions from appellate and local courts in order to generate domain-specific word embeddings for our model. We will conduct experiments using these embeddings instead of the Google News embeddings used for the results reported here, or possibly in addition.

In future work, we plan to investigate the reasons behind the substantial difference between the performance of the LSTM and GRU. Moreover, we believe that an application of transfer learning as shown in [26] could be used to train classifiers for more specific topics and different subdomains of the legal field.

References

[1] J. Wood, "Source-lda: Enhancing probabilistic topic models using prior knowledge sources," *CoRR*, vol. abs/1606.00577, 2016. [Online]. Available: http://arxiv.org/abs/1606.00577

[2] H. J. Spaeth, L. Epstein, A. D. Martin, J. A. Segal, T. J. Ruger, and S. C. Benesh, "2017 supreme court database, version 2017 release 01," 2017. [Online]. Available: http://supremecourtdatabase.org

[3] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society. Series B (Methodological)*, pp. 215–242, 1958.

[4] O. Sulea, M. Zampieri, S. Malmasi, M. Vela, L. P. Dinu, and J. van Genabith, "Exploring the use of text classification in the legal domain," *CoRR*, vol. abs/1710.09306, 2017. [Online]. Available: http://arxiv.org/abs/1710.09306

[5] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.

[6] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[7] M. Hughes, I. Li, S. Kotoulas, and T. Suzumura, "Medical text classification using convolutional neural networks," *CoRR*, vol. abs/1704.06841, 2017. [Online]. Available: http://arxiv.org/abs/1704.06841

[8] A. Conneau, H. Schwenk, L. Barrault, and Y. LeCun, "Very deep convolutional networks for natural language processing," *CoRR*, vol. abs/1606.01781, 2016. [Online]. Available: http://arxiv.org/abs/1606.01781

[9] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, Nov. 1997. [Online]. Available: http://dx.doi.org/10.1162/neco.1997.9.8.1735

[10] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014. [Online]. Available: http://arxiv.org/abs/1409.1259

[11] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[12] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *CoRR*, vol. abs/1803.01271, 2018. [Online]. Available: http://arxiv.org/abs/1803.01271

[13] F. Sebastiani, "Machine learning in automated text categorization," *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, Mar. 2002. [Online]. Available: http://doi.acm.org/10.1145/505282.505283

[14] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *CoRR*, vol. abs/1310.4546, 2013. [Online]. Available: http://arxiv.org/abs/1310.4546

[15] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[16] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.

[17] R. Nallapati and C. D. Manning, "Legal docket-entry classification: Where machine learning stumbles," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08. Stroudsburg, PA, USA: Association for Computational Linguistics, 2008, pp. 438–446. [Online]. Available: http://dl.acm.org/citation.cfm?id=1613715.1613771

[18] W.-H. Weng, K. B. Wagholikar, A. T. McCray, P. Szolovits, and H. C. Chueh, "Medical subdomain classification of clinical notes using a machine learning-based natural language processing approach," in *BMC Medical Informatics and Decision Making*, 2017.

[19] W. Yin, K. Kann, M. Yu, and H. Schütze, "Comparative study of CNN and RNN for natural language processing," *CoRR*, vol. abs/1702.01923, 2017. [Online]. Available: http://arxiv.org/abs/1702.01923

[20] Y. Kim, "Convolutional neural networks for sentence classification," *CoRR*, vol. abs/1408.5882, 2014. [Online]. Available: http://arxiv.org/abs/1408.5882

[21] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944937

[22] D. M. Blei, "Probabilistic topic models," *Commun. ACM*, vol. 55, no. 4, pp. 77–84, Apr. 2012. [Online]. Available: http://doi.acm.org/10.1145/2133806.2133826

[23] Q. V. Le and T. Mikolov, "Distributed representations of sentences and documents," *CoRR*, vol. abs/1405.4053, 2014. [Online]. Available: http://arxiv.org/abs/1405.4053

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014. [Online]. Available: http://arxiv.org/abs/1412.6980

[26] C. B. Do and A. Y. Ng, "Transfer learning for text classification," in *Proceedings of the 18th International Conference on Neural Information Processing Systems*, ser. NIPS'05. Cambridge, MA, USA: MIT Press, 2005, pp. 299–306. [Online]. Available: http://dl.acm.org/citation.cfm?id=2976248.2976286