Document Classification with DocBERT, et. Al.

Dale Angus dangus@stanford.edu

NOT FOR GRADING.
JUST WANT TO LET YOU KNOW THAT I DID THIS!

Abstract

In recent years, automated text classification has become a very important tool in the legal industry. (Jackson & Moulinier, 2002) write: "There is no question concerning the commercial value of being able to classify documents automatically by content. There are myriad potential applications of such a capability for corporate Intranets, government departments, and Internet publishers." Obviously, the ability to automatically classify legal documents contributes to the increased productivity and efficiency of lawyers and could potentially reduce the millions of dollars spent on discovery costs and legal support staff. In this paper, I present the results of my investigation of the different NLP models with the goal of finding the best model that is suited for the task of classifying legal documents, particularly long-length ones, in this case, the United States Supreme Court decisions.

1. Introduction

There have been several deep neural network NLP models that have attempted to classify documents and achieved state of the art results during their time. In this study, I went through some of them to see how each performs in the task of classifying long-length documents, in this case, United States Supreme Court decisions. Every decision document is labeled with one specific "issue area", a higher-level categorization of the issue, e.g., "Civil Rights". The models that have been tested here are the same models used by the authors of DocBERT (Adhikari, Ram, Tang, & Lin, DocBERT: BERT for Document Classification, 2019) in their study. Their code is publicly available in GitHub and is the same codebase this study used with some modifications to allow the code to work with this particular dataset and some additional code for capturing into files the various epochal metrics such as loss and accuracy values. The authors claim that their model is the first application of BERT to document classification. BERT stands for Bidirectional Encoder Representation from Transformers, a language model published by researchers at Google AI Language (Devlin, Chang, Lee, & Toutanova, 2018).

2. Related Work

I have cited several works related to text and/or document classification in the Models section below under Approach. In addition, another study which deals with "long-length legal documents" (and which I found after I have finalized all the experiments), is the work of (Wan, Papageorgiou, Seddon, & Bernardoni, 2019) where they addressed the limitation that current models impose on the length of the input text and where they show that dividing the text into segments improved results. I have performed the "chunking" of text in three different ways (four, including "as-is") as described in the Data Preparation section under Experiments.

3. Approach

I initially just wanted to study how Hierarchical Attention Network as stated in the initial project proposal but while researching, I came across DocBERT. Fortunately, the author has provided their implementation and more. More meaning that they also provided several other models that they used to benchmark DocBERT. I studied the code and

saw that with some modifications to their codebase, Hedwig (University of Waterloo, n.d.), I could run all the models to classify the dataset for this study, the US Supreme Court decisions. All code is based on Pytorch 0.4.1.

In addition to the class-sponsored Azure compute environment, I used my personal computer that is well-equipped to train some of the less compute-intensive models and another compute environment using the free $100-credit for students from Azure. I ran the HAN, Reg-LSTM, DocBERT in the Azure environment. The Kim CNN, CharCNN, XML CNN, LR, fastText were ran using my personal computer. See description of these models below.

For baseline purposes, I use the Logistic Regression[1] and fastText as co-baseline models since both provide competitive baselines. The models are evaluated by their accuracy score.

3.1. Models

Listed below are the models that are included in this study, according to the publish date.

Convolutional Neural Networks for Sentence Classification (Kim, 2014). The author developed a CNN model trained on top of pre-trained word vectors (word2vec) for sentence classification. Referred to as **KIM CNN** in the tables and charts below.[2]

Character-level Convolutional Networks for Text Classification (Zhang, Zhao, & LeCun, 2015). The authors offer an empirical exploration on the use of character-level convolutional networks (ConvNets) for text classification. Referred here as **CharCNN**.[3]

Bag of Tricks for Efficient Text Classification (Armand Joulin, 2016). The authors wrote that their experiments show that their fast text classifier fastText is often on par with deep learning classifiers in terms of accuracy, and many orders of magnitude faster for training and evaluation. Referred here as **fastText**.[4]

Hierarchical Attention Networks for Document Classification (Yang, et al., 2016). The authors describe the model as a model that progressively builds a document vector by aggregating important words into sentence vectors and then aggregating important sentences vectors to document vectors. Referred here as **HAN**.[5]

Deep Learning for Extreme Multi-label Text Classification (Liu, Chang, Wu, & Yang, 2017). The authors were the first to attempt at applying deep learning to XMTC with a family of new Convolutional Neural Network (CNN) models which are tailored for multi-label classification in particular. Extreme multi-label text classification (XMTC) refers to the problem of assigning to each document its most relevant subset of class labels from an extremely large label collection where the number of labels could reach hundreds of thousands or millions. Referred here as **XML CNN**.[6]

Rethinking Complex Neural Network Architectures for Document Classification. (Adhikari, Ram, Tang, & Lin, 2019) The authors agree with other members of the NLP community that neural networks have grown increasingly complex in recent years, making training and deployment more difficult. They present a simple BiLSTM architecture with appropriate regularization. Referred here as **Reg-LSTM**.[7]

DocBERT: BERT for Document Classification (Adhikari, Ram, Tang, & Lin, 2019). The authors present the very first application of BERT to document classification and show that a straightforward classification model using BERT was able to achieve state of the art across four popular datasets. The author acknowledges that their code is inspired from (huggingface)'s implementation. Referred here as **DocBERT**.[8]

---

[1] Used sklearn.feature_extraction.text.TfidVectorizer (scikit-learn, 2011)
[2] Used embedding word2vec/GoogleNews-vectors-negative300.bin (Mikolov, Chen, Corrado, & Dean, 2013)
[3] Ibid.
[4] Ibid.
[5] Used embedding glove.6B.300d (Pennington, Socher, & Manning, 2014)
[6] Used embedding word2vec/GoogleNews-vectors-negative300.bin (Mikolov, Chen, Corrado, & Dean, 2013)
[7] Ibid.
[8] Used model bert_pretrained/bert-based-uncased (Google Research, 2018)

It should be interesting to the reader that all models except DocBert have a custom implementation of torch.nn.Model subclass. DocBERT uses huggingface's BertForSequenceClassification, a regular PyTorch module, from the transformers Python package.

4. Experiments

4.1. Data

The dataset is the US Supreme Court decisions which can be downloaded from Textacy (Textacy, 2017). Textacy collected the data from the FindLaw (FindLaw.com, 1995) website and added various metadata[9]. It contains exactly 8419 decisions with various labels.

The label used in this study is **issue_area** with the codes and histogram distribution below. Each record is classified with only one issue area.

-1: None
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

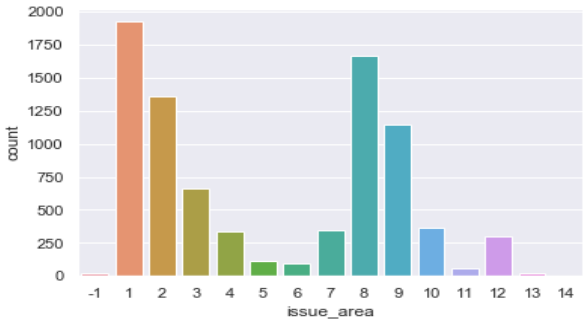

Figure 1 Histogram showing the counts per issue area.

The average number of words per decision is 6,962 with a maximum of 87,246 words and minimum of 11.

4.2. Data Preparation

The **issue_area** codes were converted to one-hot format while the **text** was prepared four different ways (see Table 1). First preparation, **Data Prep 1**, the dataset was setup with the class in one-hot format and the text, as-is, regardless of the length. Second, **Data Prep 2**, the text was separated in 50,000-character chunks. All derived chunks are added as a sample of the same class. Third, **Data Prep 3**, the text was divided in 20,000-character chunks with the subsequent chunk having a 5,000-character overlap from the previous chunk. And the fourth data preparation, **Data Prep 4**, is specifically for the DocBERT model. Because of the 250-limit sequence length required by BERT, similar to the 2nd and 3rd preparation method, the document was chunked into 250 words with an overlap of 50 words. The dataset for the 4th data preparation ballooned to 389,886 rows with an average of 199.74 words per row with a maximum of 200 and a minimum of 11.

[9] https://chartbeat-labs.github.io/textacy/build/html/api_reference/datasets.html?highlight=supreme%20court#textacy.datasets.supreme_court.SupremeCourt

The dataset was split 60-15-25 to allow more test samples for measuring the accuracy score. Since this is a classification task, accuracy is the best measure for the performance of the models.

Table 1- The same dataset with different preparations. Total samples and the 60-15-25 split count.

Dataset	Samples	Train	Dev	Test
Data Prep 1	8,914	5,348	1,337	2,229
Data Prep 2	10,931	6,559	1,639	2,733
Data Prep 3	23,713	14,228	3,557	5,928
Data Prep 4	389,886	233,932	58,482	97,472

4.3. Results

Running the models using the first data preparation proved to be very problematic. Multiple models ran into one or a combination of problems such as out-of-memory issues very slow training and poor performance. The problem may be due to the sparsity of the encoded text. As noted, the average is 6,962 words with a single row having 87,246 words. This data preparation had to be abandoned as there were no meaningful results from it. The second and third data preparation did not have problems and completed without problems. This proves that dividing documents into chunks before inputting them into the models resulted in higher accuracy scores.

The time to completion for the experiments vary. The fastText model, on all the data preparations, was the quickest to complete. The claim made by the authors of fastText that it is in the order of magnitudes faster than the deep learning models proved to be true. The CNN models on Data Prep 2 and 3 finished in less than 20 minutes. Each CNN models were given 30 epochs to complete but the models stopped making performance gains and triggered an early-stop where patience=5. The Logistic Regression took longer to finish because it completed the 50 epochs given to it making continuous gain at every epoch. The Reg-LSTM and HAN models took hours to complete on Data Prep 2 and 3. The DocBERT took ten (10) days to complete training with Data Prep 4.

Non-BERT models versus co-baselines LR and fastText

Tables 2 and 3 below show the accuracy scores of each model for Data Prep 2 and 3, respectively. Except for CharCNN, all models improved their respective accuracy score from Data Prep 2 to Data Prep 3. Figures 2 and 3 show where the respective model's accuracy in predicting by class. The results from the two data preparations show that Logistic Regression and fastText produce better accuracy scores than the deep learning models.

Table 2 *– Model Accuracy Scores using dataset Data Prep 2. Each document is split into 50,000-character chunks.*

Model	Accuracy
Logistic Regression TF-IDF	**0.8042 (2nd)**
Hierarchical Attention Network	0.6926
KIM CNN	0.6352
XML CNN	0.5935
Regularized LSTM	0.5540
CharCNN	0.2704
fastText	**0.8138 (1st)**

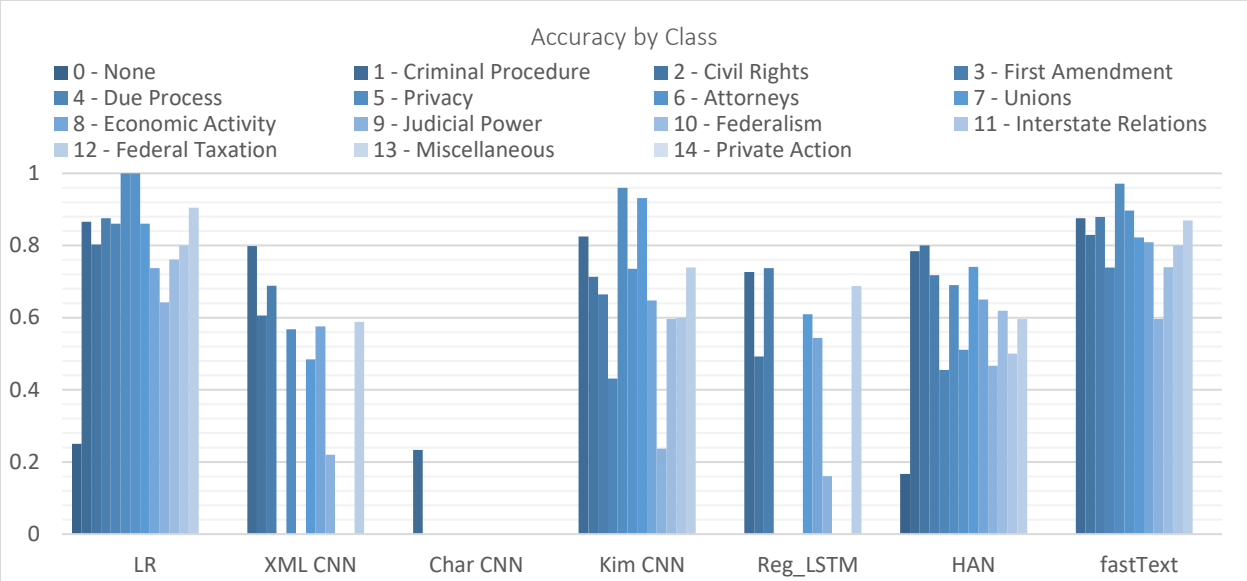

Figure 2 Accuracy by class of each model using Data Prep 2.

***Table 3**- Model Accuracy Scores using dataset Data Prep 3. Each document is split into 20,000-character chunks with 5,000-character overlap from the previous chunk.*

Model	Accuracy
Logistic Regression TF-IDF	**0.8797 (1st)**
Hierarchical Attention Network	0.7686
KIM CNN	0.7351
XML CNN	0.6354
Regularized LSTM	0.5833
CharCNN	0.2314
fastText	**0.8765 (2nd)**

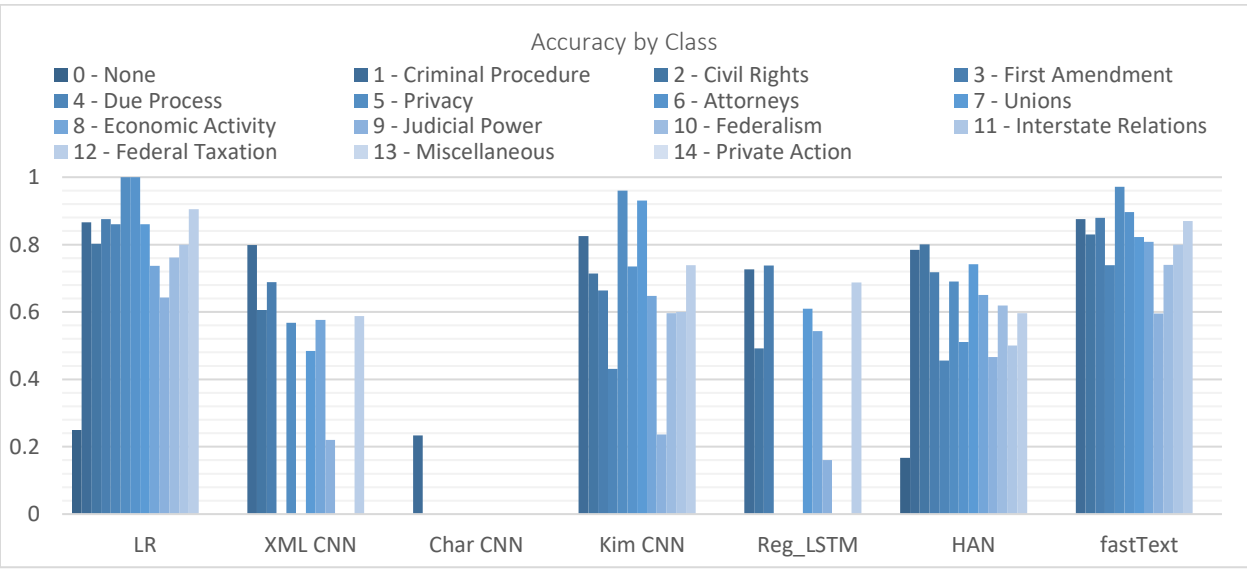

Figure 3 Accuracy by class of each model using Data Prep 3.

DocBERT model versus co-baselines LR and fastText

Table 4 shows the accuracy of DocBERT compared to the co-baseline LR and fastText and HAN, the 3rd best model from the results using Data Prep 2 and 3. Figure 4 shows that DocBERT has a higher accuracy on the four classes with the highest number of samples, classes 1, 2, 8 and 9 (see also Figure 1).

It took 10 days to train the DocBERT model to classify the US Supreme Court decision documents compared to the few hours to train the others, but it surpassed the benchmark accuracies significantly. The accuracy of DocBERT could have reached higher levels if it kept running further and not stopped after a pre-defined setting of thirty (30) epochs.

Table 4 *- Model Accuracy Scores using dataset Data Prep 4. Each document is split into 200-word chunks with 50-word overlap from the previous chunk.*

Model	Accuracy
Logistic Regression TF-IDF	0.8747
fastText	0.8599
Hierarchical Attention Network	0.6355
DocBERT	**0.9173**

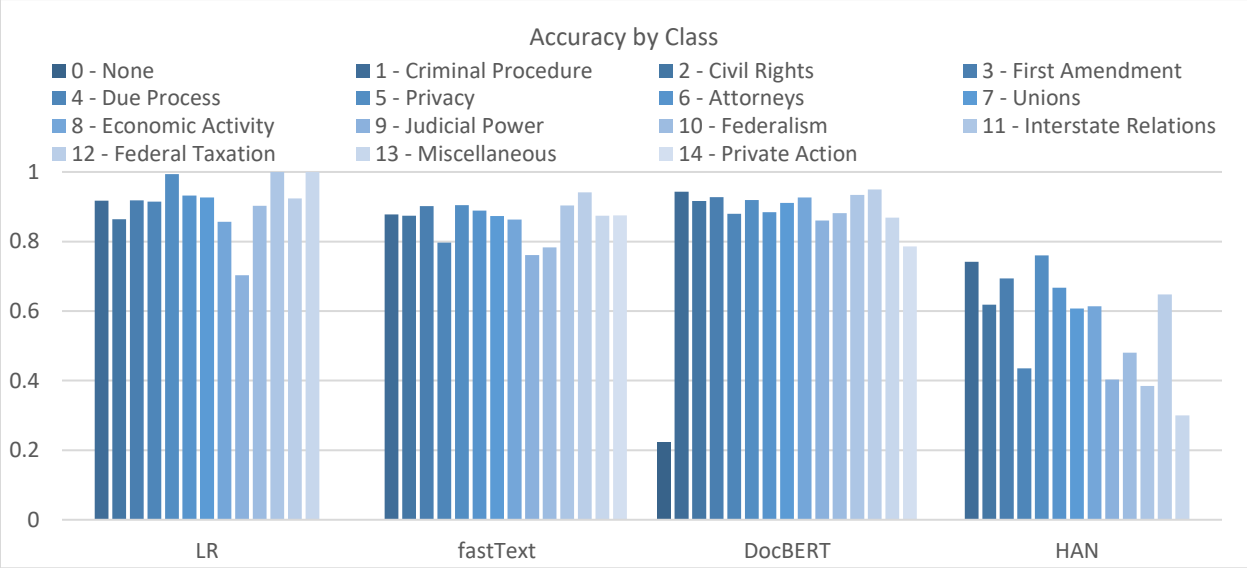

Figure 4 Accuracy by class of each model using Data Prep 4.

Analysis

TODO 😐

Conclusion and Future Work

In this study I have shown which models are effective in classifying long-length documents. I was able to replicate the success reported by the authors of DocBERT as far as surpassing the baseline models, Logistic Regression and fastText. TODO 😐

One possible work for the future is to study how RoBERTa (Liu, et al., 2019) can be applied to document classification. It should be easy to adapt RoBERTa for this study. A small change must be done in adding the tokens to fit the RoBERTa specification. Adding the tokens is slightly different between BERT and RoBERTa, for example, BERT uses [CLS] and [SEP] while RoBERTa uses <cls> and <sep>.

References

Adhikari, A., Ram, A., Tang, R., & Lin, J. (2019). DocBERT: BERT for Document Classification. *Arvix*.

Adhikari, A., Ram, A., Tang, R., & Lin, J. (2019). Rethinking Complex Neural Network Architectures for Document Classification. *ACL Anthology*, 4046-4051.

Armand Joulin, E. G. (2016). Bag of Tricks for Efficient Text Classification. *Arvix*.

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Arvix*.

FindLaw.com. (1995). Retrieved from FindLaw: http://caselaw.findlaw.com/court/us-supreme-court

Google Research. (2018). *google-research/bert*. Retrieved from http://github.com/google-research/bert

Jackson, P., & Moulinier, I. (2002). *Natural Language Processing for Online Applications: Text Retrieval, Extraction and Categorization.* John Benjamins.

Kim, Y. (2014). Convolutional Neural Networks for Sentence Classification. *CoRR*.

Liu, J., Chang, W.-C., Wu, Y., & Yang, Y. (2017). Deep Learning for Extreme Multi-label Text Classification. *SIGIR '17: Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, (pp. 115-124).

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., . . . Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach. *Arvix*.

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *word2vec*. Retrieved from Google Code Archive: https://code.google.com/archive/p/word2vec/

Pennington, J., Socher, R., & Manning, C. D. (2014). Retrieved from GloVe: Global Vectors for Word Representation: https://nlp.stanford.edu/projects/glove/

scikit-learn. (2011). *sklearn.feature_extraction.text.TfidfVectorizer*. Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Textacy. (2017). Retrieved from Textacy: https://chartbeat-labs.github.io/textacy/build/html/api_reference/datasets.html#api-reference-datasets

University of Waterloo. (n.d.). *Hedwig*. Retrieved from Github.com/castorini/hedwig.

Wan, L., Papageorgiou, G., Seddon, M., & Bernardoni, M. (2019). Long-length Legal Document Classification. *Arvix*.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Brew, J. (2019). HuggingFace's Transformers: State-of-the-art Natural Language Processing. *ArXiv*. Retrieved from Transformers: https://github.com/huggingface/transformers

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., & Hovy, E. (2016). Hierarchical Attention Networks for Document Classification. *Proceedings of the 2016 Conference of the North American Chapter of the Association for*

Computational Linguistics: Human Language Technologies (pp. 1480–1489). San Diego: Association for Computational Linguistics. Retrieved from www.cs.cmu.edu.

Zhang, X., Zhao, J., & LeCun, Y. (2015). Character-level Convolutional Networks for Text Classification. *Advances in Neural Information Processing Systems 28*, 649-657.