Government Document Classification

Stanford CS224N Custom Project

Andrew Tang
Department of Computer Science
Stanford University
andrewht@stanford.edu

March 17, 2021

Abstract

This project was proposed by USAFacts, an organization founded by former Microsoft CEO Steve Ballmer dedicated to leveraging data and quantitative methods to provide insight into the US government. The organization believes that having labelled government document data is important towards its mission. But, currently, employees have been classifying laws manually. The process is thus very time-consuming and slow. This project attempts to use the large corpus of federal legislation and automate the classification process with a trained NLP model. Other potential applications could include tracking how the amount of legislation by topic area has shifted over time and between Democratic and Republican controlled governments.

In this paper, I explore various methods of inputting long text documents into multiple models, and subsequently discuss trade-offs between performance and processing time. I also discuss the modification of various model features and their impact on model performance. The highest accuracy achieved was a 84% score with a fastText model.

1 Key Information

- Mentor: Tyler Mallon (USAFacts), Markus Pelger (MS&E108), Andrew Wang (CS224N)
- External Collaborators: Marlies Michielssen (marliesmstanford.edu), Brooke Tran (btran1stanford.edu), Kasha Akrami (kakramistanford.edu)
- Sharing project: MS&E108

2 Introduction

USAFacts is an organization dedicated to providing transparency into the workings of the US government. They continuously publish research and statistics aimed at explaining what the government is doing. For the past two years, the team has been manually classifying government documents into specific topic areas. These efforts are very laborious and slow, and the team has only been able to classify around 500 documents total (while more than 14,000 bills were introduced in the 116th Congress). The organization is very interested in automating the classification process with a trained NLP model.

After initial research, I discovered a huge amount of model types and began applying many of them to this problem. Most of the existing classification literature revolves around applying models to the same large document datasets, namely reviews taken from websites such as Yelp (Tang et al., 2015), IMDB, and Amazon, rather than long, specifically-written and formatted federal legislation. Additionally, during the process of creating a new, custom labelled dataset, I found that government legislation falls into a very large number of categories: 32, as opposed to reviews, which either range from 1-5 or 1-10. I thus had to discover which models would work best with a large amount of categories and long federal documents.

3 Related Work

Papers that inspired my approach are DocBERT (Adhikari et al., 2019), in which the authors describe a variety of document classification methods that they compare their DocBERT model to and HAN (Yang et al., 2016). I was able to have a baseline comparison across many different datasets (including the Yelp and IMDB datasets mentioned above), to see how the models performed on other datasets. I was also able to get a sense of document processing of the input text: for example, the DocBERT paper describes truncating text after a certain length, while other papers discuss breaking up the text into overlapping chunks. Overall, it was beneficial to have a sense of how models in these various papers performed, as well as descriptions of the respective architectures. Overall, most of these papers explore different approaches and model architectures revolving around the same problem of document classification.

4 Approach

After reading through a number of papers and discussing with the organization, I decided on creating a baseline logistic regression model using tf-idf scores as input. This was suggested by my external mentors. I then replicated a Hierarchical Attention Network (Adhikari et al., 2019 and Yang et al., 2016), a convolutional neural network (Adhikari et al., 2019 and Kim 2014), and a fastText model Joulin 2016). I was able to use the University of Waterloo's

Hedwig database (Hedwig) and use/modify their hierarchical attention network and convolutional neural network, adding features as discussed in subsequent parts of the paper. The LR and fastText models, along with the dataset creation scripts and additional features of the HAN and CNN were all written by myself. The HAN and CNN all ran using PyTorch, and all other code was written in Python using various libraries such as Pandas.

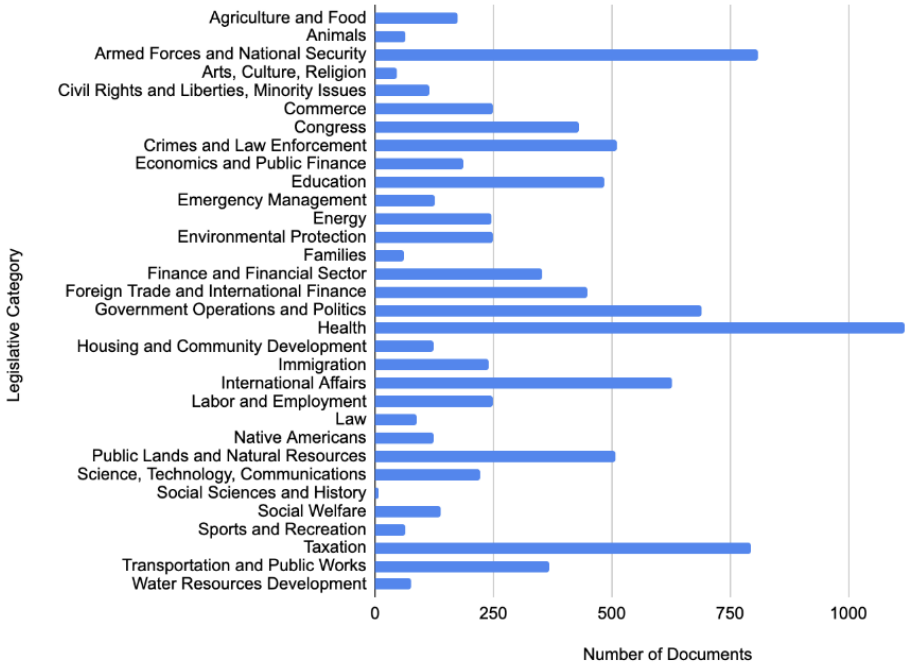
I made various changes to the existing HAN model which differed from Hedwig and the referenced paper: the original model was a bidirectional model using GRU and word2vec embeddings. I first ran the HAN with various different word embeddings: word2vec downloaded from GoogleNews in 300 dimensions, GloVe, and fastText subtext embeddings. Additionally, I changed the bidirectionality of the RNNs in the HAN, and tweaked the GRU to a LSTM at the very end. Overall, I had bidirectional GRU word2vec, GloVe, and fasttext models, a bidirectional LSTM word2vec model, and a non-bidirectional GRU word2vec model.

5 Experiments

This section contains the following.

5.1 Data

Although the USAFacts team was able to assemble a dataset of a few hundred documents, it soon became clear that this was nowhere near enough data to properly train and evaluate. The other related papers had tens of thousands of documents. I thus decided to not touch the USAFacts dataset and use it to test the models instead. I then decided that I had to create my own labelled dataset, with sufficient volume for creating the NLP models. I then wrote Python scripts to scrape bill text data from Congress.gov, as well as the policy areas for each of those bills. Currently, many federal legislative documents are manually labelled by Congressional Researchers and assigned a specific policy area, with 32 policy areas in total. I managed to put together a labelled dataset of 10,000 documents, with each category having proportional representation. To calculate the percentage for each category, I computed an average of the number of documents in each category across the past decade (116th, 115th, 114th, 113th, and 112th Congresses). The first figure below show the number of documents per category in the final scraped dataset. The second figure shows the categories and their relevant encodings.



{'Science, Technology, Communications': 0, 'Public Lands and Natural Resources': 1, 'Education': 2, 'Housing and Comm
unity Development': 3, 'Congress': 4, 'Foreign Trade and International Finance': 5, 'Commerce': 6, 'Emergency Managem
ent': 7, 'Arts, Culture, Religion': 8, 'Environmental Protection': 9, 'Finance and Financial Sector': 10, 'Crimes and
Law Enforcement': 11, 'Immigration': 12, 'Water Resources Development': 13, 'Taxation': 14, 'Sports and Recreation':
15, 'Energy': 16, 'Families': 17, 'Labor and Employment': 18, 'Transportation and Public Works': 19, 'Agriculture and
Food': 20, 'Government Operations and Politics': 21, 'Armed Forces and National Security': 22, 'Social Sciences and H
istory': 23, 'Social Welfare': 24, 'Health': 25, 'Economics and Public Finance': 26, 'Civil Rights and Liberties, Min
ority Issues': 27, 'International Affairs': 28, 'Animals': 29, 'Native Americans': 30, 'Law': 31}

Data cleaning was the next step, given that the scraped data was directly from a website and often contained meaningless characters and was sometimes strangely formatted given the HTML structure. This included removing non-alphanumeric characters, removing excess spaces, removing stop words, etc. The final structure of the data was one column for the cleaned text of each bill, and a label associating that text with one of 32 categories. The categories were sometimes encoded as integers, and are shown in the below graphic. The average word length was around 5,000 words.

Before inputting the text into the model, I pre-processed the data in several different ways: I attempted truncating after 30,000 characters, as well as splitting longer documents into 25,000 chunks with 5,000 character overlap between chunks. However, I did not get much of a difference in accuracy, and my reported results are on the 25,000 length with 5,000 overlap dataset (which initially had the best results on the logistic regression and fastText models on the dev set).

However, in the code I wrote, there is a very easy way to change these parameters. Nevertheless, the final dataset consisted of 13865 rows, and I did a 60/20/20 ratio for train validation test splits.

5.2 Evaluation method

Given the large number of categories and dataset size, I simply replied on accuracy as my metric. I thought of false positives and false negatives as equal, and thus avoided precision and recall. Thus, for the test set, I simply looked at the number of correct predictions over the number of total documents.

5.3 Experimental details

The logistic regression baseline model took under 5 minutes to run, and ran for 500 iterations. The fastText model took under 2 minutes to train and ran with word embedding dimensions of 20, 30 epochs, and a lr of 0.95. It ran on word bigrams, although there is an option to change the size of n-grams.

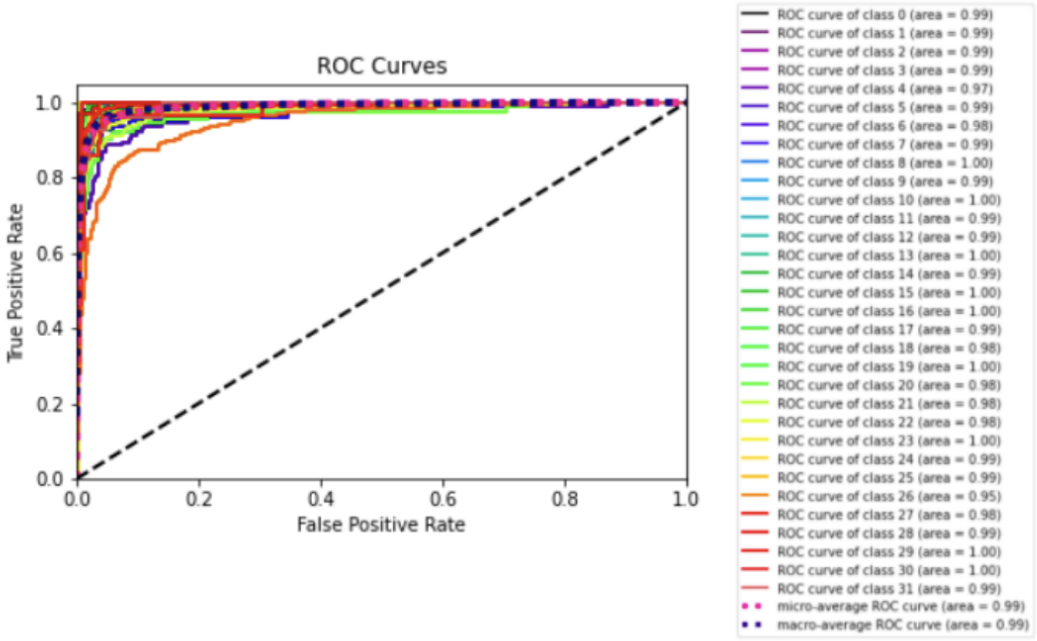
The CNN was run with batch size 32, learning rate 0.01, and ran for 15 epochs. It ran for about twenty to thirty minutes.

Finally, each of the HAN models (word2vec, GloVe, fastText subtext, LSTM, and non-bidirectional) ran for around two hours. I ran with batch size 32 and learning rate 0.01. The non-bidirectional was slightly faster. The fastText subtext model ran for 30 epochs since it kept learning at each epoch, whereas the other HAN models stopped early and I thus ran them for only 15 epochs.

All models were run on Microsoft Azure, with the exception of the logistic regression baseline and fastText models, which I ran on my local machine.

5.4 Results

The baseline model ROC curves are below.



The results (reported in terms of accuracy on the test set) are as follows. The first table compares the four types of models, while the second table shows the various HAN models against one another.

Model	Test Set Accuracy
Logistic Regression tf-idf baseline	78.54%
Hierarchical Attention Network (GRU, word2vec, bidirectional)	82.29%
Convolutional Neural Network	72.08%
fastText	84.31%

Model	Test Set Accuracy
GRU, word2vec, bidirectional	82.29%
GRU, GloVe, bidirectional	81.21%
GRU, fastText subtext, bidirectional	77.46%
LSTM, word2vec, bidirectional	83.38%
GRU, word2vec, non-bidirectional	81.82%

6 Analysis

Surprisingly, the baseline logistic regression model that only took into account tf-idf scores performed extremely well. This may suggest that a simple measure taking into account only the relative importance of a word to each document (discarding the effects of word context/meaning, order, etc.) is enough to accurately classify the majority of documents.

The most accurate model, the fastText model, seems to also leave behind much of the more complicated RNN architecture of the hierarchical attention network, relying on a simple shallow neural network with word vectors. Thus, this may suggest that many of the more advanced features of the HAN (such as the bidirectional RNNs taking into account word order and the attention mechanism) may not be as important for the task of categorizing long documents. Furthermore, various features of the fastText model seem to improve model performance with large numbers of labels, such as adding in a hierarchical softmax function, as well as averaging of word vectors with subword vectors. It seems that fastText is also a model that was specifically designed for the purposes of text classification, while the hierarchical attention network was adapted from existing RNN architectures and applied to the task. Overall, however, the improvement of the word vector based models over the tf-idf score baseline suggests that a representation of word meaning is a better input that a metric based off of word counts.

Looking at specific misclassified examples, it seems that the improvements over the baseline tf-idf model occur with regard to the interpretation of particular words. As one example, a document that mentions "Hawaii" many times gets incorrectly classified by the tf-idf model as "Armed Forces and National Security" most likely due to the fact that the word appears so often and most often appears in defense bills (due to the large presence of the military in the state) when the bill is actually in the "Law" category. However, word-vector and attention-based models deduce that the meaning of Hawaii is merely a place, and that other words in the document are more important. In terms of misclassified examples for the HAN and fastText models, the toughest category was "Economics and Public Finance." Large, omnibus spending bills with many provisions and thus many categories often go in this category, limiting the effectiveness of our predictive models in this particular area.

It seems also that the fastText model is superior in terms of speed as well, due to various adaptations such as hashing and not having a linear structure (like the RNN-based models). Finally, interestingly, removing the bidirectionality from the HAN model resulted in a slightly faster training time while having comparable accuracy. This suggests that bidirectionality in the HAN model may not be as important of a characteristic to have. I have thus shown that

both advanced, neural-network based models and simpler regression models all perform relatively well on the task of document classification: in existing papers that I referenced, the accuracy is comparable.

7 Conclusion

In summary, over the course of this project, I managed to develop a number of models that achieved high levels of accuracy in classifying legislative action documents to their respective topical categories. I believe that the project was successful because I assisted USAFacts with delivering a project true to their mission and research: developing and utilizing data science and technology to further understanding of governance.

These trained models can easily be applied to categorize other kinds of documents, such as executive actions, state legislation, and legislation from other countries. Furthermore, I demonstrated that many existing models will work when applied to a significantly different type of document, and that such NLP models can be useful for organizations operating in the social science space. In the future, large pre-trained attention-based models such as BERT could also be applied to this specific classification task, to see how performance compares to the models already considered in this paper.

8 References

Ashutosh Adhikari, Achyudh Ram, Raphael Tang, Jimmy Lin. 2019. DocBERT: BERT for Document Classification. arXiv.

Duyu Tang, Bing Qin, and Ting Liu. 2015. Document modeling with gated recurrent neural network for sentiment classification. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1422–1432.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of Tricks for Efficient Text Classification. arXiv.

Hedwig. GitHub Repository. https://github.com/castorini/hedwig/tree/master/models

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, pages 1746–1751.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In

Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 1480–1489.

A Appendix

Code Repository: https://github.com/brooketran/USAFacts