Code Report

CS109B Data Science 2: Advanced Topics in Data Science

Let's Replace Lawyers with Machines!

by Group 51, Module C
Authors: Brian Hall, Narsing Rao Dyavani, Manish Babel, Mohamed Elsalmi

Harvard University
Spring 2020
Instructors: Pavlos Protopapas, Mark Glickman and Chris Tanner

---

I. Introduction

The goal of our project is to create a model that will automatically generate summaries of U.S. legal cases. Commercial third parties (Westlaw, LexisNexis, etc.) currently maintain human-generated headnotes that are brief case summary statements but are typically available for a fee and subject to usage restrictions. An open-source model that can summarize legal cases could be a useful alternative to such fee-based summary services.

By using texts of legal cases, we aim to assess both an unsupervised and supervised model's ability to generate summaries of a legal case that can be used as an alternative to these pay services. The main question we seek to ask is: can a computer model create headnotes that capture the nuances of case law?

II. Body

Below is a section-by-section overview of our coding process, from data preparation and analysis, model selection and building, and a display of each model's predicted results. This Code Report references to appendices that contain the code, more details on the process and comments in the code itself. Appendices were used because the code was too long and resource intensive to include in this single notebook (some taking more than a day to run).

A. Data Preparation

Unsupervised Model Dataset

The data for our unsupervised model comprised of legal judgments from Arkansas, Illinois, North Carolina and New Mexico from the Caselaw Access Project. The judgements came as json files, which we loaded into pandas (see Appendix A - Section II: Loading the Raw Data). The casebody feature is of interest in that it also contains other tags within the text that can be used to parse and separate casebody into more features. For example there are many tags in the casebody that would repeat data already found above (first page, last page, docket number, decision date, etc.) but there are also new tags such as the names of the attorneys, the opinion type (majority or minority) and the author of the opinion (judge name) that can be used to generate potentially interesting data (see Appendix A - Section III: Parsing Features from the Case Text). Additionally, cases with missing features (null) were dropped (see Appendix A - Section III: Parsing Features from the Case Text). All of these extracted features were then exported and saved using parquet to be used later in EDA and modeling (see Appendix A - Section IV: Review of Unsupervised Model Dataset).

Supervised Model Dataset

With respect to our supervisd model, only one state North Carolina, had labeled outcome variable (headnotes), so a separate data extraction process was run on a North Carolina dataset that had this labeled data. The steps above were also repeated on this North Carolina dataset (see Appendix B - Section II: Loading the Raw Data; Section III: Parsing Features from the Case Text; Section III: Parsing Features from the Case Text). The labeled data (headnotes) were extracted using BeautifulSoup to find all text tagged as 'headnotes' in the case (see Appendix B - Section IV: Extracting the Outcome Variable). Not every case had headnotes, so those without headnotes were dropped from the Supervised Model Dataset. All of the extracted features and labeled outcome data were then exported and saved using parquet to be used later in EDA and modeling (see Appendix B - Section IV: Review of Supervised Model Dataset).

B. EDA

After combining the cases of the four states into one dataset, there are over 350,000 cases (see Appendix C - Section III: Data Visualization). This combined dataset has cases from the late 1700s to the present era, with many more cases per year occurring after the 1950s but more recently dipping, a potential reflection of the rise and fall of the regulatory activity of the state (see Appendix C - Section III: Data Visualization). Juding from the names of the courts, generally all of the cases are appellate and majority opinions (see Appendix C - Section III: Data Visualization). The lengths of the judicial opinions by judge revealed that there are certainly outliers, and to reduce compute cost it would make sense to exclude longer cases (see Appendix C - Section III: Data Visualization). Also, we noticed that the number of headnotes can vary by case, making the prediction length (number of headnotes) a more difficult moving target.

The length of the case may impact the effectiveness of a summary of the case. For example, a shorter case may prove on average easier to summarize then a long case. The thinking behind this may the longer the case, the more opportunity nuances could be lost in the text. As a result, splitting up the data into cases of different size ranges may be an interesting way to model the data to examine if the quality of the summaries change based on case length.

We decided to initially load into each model the entire text of a case, to generate the predicted headnotes (see Appendices D and E). If we had more time, we could attempt to parse (A) only the sentence that preceded a citation of another case as those sentences are typically legal opinions of the judge used to generate a headnote or (B) feed the model only the first 100 and last 100 words of a case. Those are two interesting paths worth pursuing in the future, and we have begun the code related to this work (see the last section of Appendix D).

Intelligent Data Selection

Various Approach of Data Selection

All Data (A) Feed in all Data. File Text can be long. Longer Texts lead to less accurate the summaries

Location Based (B) Feed in Part of the text based on Location

Before Citation (C) Feed in text of *Judicial Statements Made Before Citation*

....<p id="b492-5">From a perusal of these cases, and the authorities cited therein, it will clearly appear: (1) That, on the facts presented, the court had full power to order a sale for reinvestment under the statute; (2) that the same can be effected by private negotiation, subject to the approval of the court, when it is properly made to appear that the best interest of all the parties so requires. This was the course pursued and directly approved in Dawson1s case, supra. (3) That ordinarily, and on the facts of this record, the purchaser is not charged with duty of looking after the proper disposition of the purchase money, but, when he has paid his-bid into court, or to the parties authorized to receive it by the court's decree, he is "quit of further obligation concerning it." Dawson v. Wood, supra; Pendleton v. Williams, supra, ...

C. Models

When selecting a model to answer the problem statement, it is important to consider the strengths and weaknesses of a model. An Unsupervised Model would not need the copyrighted labeled outcome variable (headnotes), while a Supervised model would require labeled data. The compute cost is much lower for the Unsupervised model, as it does not require training. The technology is not as cutting edge with an NLTK similarity matrix based approach compared to a Supervised deep learning model. Also, the interpretability of the results of an Unsupervised model may be difficult without an attorney generated outcome (headnote) to compare these results to. Given the trade-offs betwen the Supervised and the Unsupervised approach, we opted to try both approaches.

Unsupervised Model

We opted to use an extractive summarization approach, which uses a subset of words that retain the most important points of the text. This model takes the reads the case opinion and tokenizes the text, generates a similarity matrix across sentences, weighs the sentences (see Appendix D - Section II: Generating Summaries of Text), and then selects the sentences with higher rank to be the predicted output text (see Appendix D - Section III: Creating Predictions from Case Data). Extractive summarizes tend to give better results compared to abstractive summaries, which can stumble with semantic representation problems.

Supervised Model

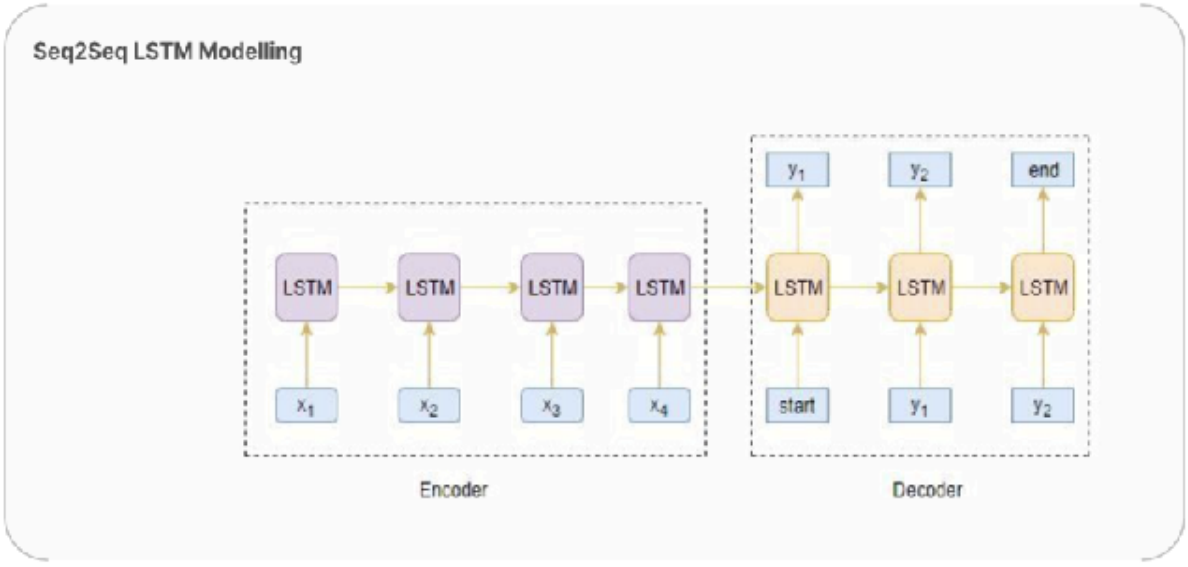
An LSTM deep learning model was selected for the supervised approach to predicting headnotes. Tensorflow's tokenizer and pad_sequences are imported ito tokenize the words in the cases and headnotes and perform a rare word and coverage analysis (see Appendix E - Section III: Tokenizing Data). Tensorflow Kera's was used to create the LSTM model (see Appendix E - Section IV: LSTM Deep Learning Model). The max text length is applied and dropouts are used for regularization in both the encoder and decoder as set out below.

Used North Carolina Data - Model Parameters

	Headnote	Case
Data	<=500 words	<=2,000 words
Vocabulary	8,000 words	19,000 words

Model Architecture

Seq2Seq LSTM Modelling



D. Results

Unsupervised Model

Below is a sample output from a case using the Unsupervised model, though we created predictions for every case across all four states and exported these results in a pandas-ready format using parquet (see Appendix D - Section III: Creating Predictions from Case Data). As you can see, the output below actually turned out to sound coherent, though it appears this case had several legal points, not only the ones contained in the predicted output.

Supervised Model

The results of the Supervised model were not as promising as the Unsupervised model (see Appendix E - Section IV: LSTM Deep Learning Model). Below is a sample case, it's human generated headnote (the labeled outcome data) and the headnote predicted from the LSTM model. The LSTM's results do not seem to line up with the text of the case itself. Some takeaways from our Supervised model results were: (A) tweaking some hyper parameters slightly improved the results and (B) with more time (it took a day to run each attempt), we could experiment using different layers, using a larger training set (only 60% of cases were used due to time constraints) and running more epochs in attempt to improve the results.

Hyper Param for LSTM layer

1. Return Sequences = True: When the return sequences parameter is set to True, LSTM produces the hidden state and cell state for every timestep
2. Return State = True: When return state = True, LSTM produces the hidden state and cell state of the last timestep only
3. dropout=0.4,
4. recurrent_dropout=0.2

Supervised Approach - Example

Example Text

> “ campbell judge the defendant assigns as error that he did not receive sentence the same as the sentence imposed upon one of the co conspirators there is no merit in this assignment of error state garris 265 n c 711 144 s e 2d 901 1965 we have reviewed the record in this case and find no prejudicial error no error judges britt and hedrick concur

Human Generated Headnote

> “ criminal law 138— sentence of defendant different than sentence of co conspirator the fact that defendant did not receive the same sentence as that received by one of his co conspirators is not ground for legal objection

Predicted Headnote

> “ criminal law 155 5— failure to docket record on appeal in apt time appeal is dismissed for failure to docket the record on appeal within the time allowed by the court of appeals rule 5

F. Performance Evaluation

After generating summaries using a number of different methods, one thing that is crucial to any Machine Learning task is performance evaluation, i.e. generating a numerical score that is indicative of how well the model is doing at it's given task. Several methods exist to evaluate text output. However, there is no 'standard' when it comes to a performance test.

The need for a meaningful score is essential for any machine learning system for many reasons, including:

- Compare different models
- Evaluate the increase/decrease in perfromance when a change is implemented in our system.

We obtain ROGUE scores for our two approaches and attempt to implement a rather novel solution to evaluate the generated summaries using sentiment analysis.

ROUGE

One of the most commonly used evaluation measures is *ROUGE* (Recall-Oriented Understudy for Gisting Evaluation) is commongly used for evaluating summarization tasks. Basially ROGE captures the n-gram overlap between system and human composed summaries. This attempts to see how much *coverage* is present in the generated text. This is a sensible approach when attempting to determine the accuracy when trying to summarize text with a large portion of factual information. There are several variations of ROUGE:

- Rouge-1: Overlap of unigram between the system and reference summaries

- Rouge-2: Overlap of bigram between the system and reference summaries
- Rouge-l: Longest Common Subsequences

Although succesful at capturing the aforementioned overlap, ROUGE has several limitations. First, it is not a good assement of meaning, since it only captures overlaps, it does poorly with synonyms. This is not a particularly good feature, specifically with abstractive summarizations where we are aiming to generate summarized text often with the use of synonyms -- which ROGUE does not take into account. Moreover, it does not take into account the topics that are important in the reference text. For example, suppose we have a document with 30% of its content deemed essential, if the generated summary only captures that other 70%, our ROGUE score would be misleading. Since we achieve of score = 0.7, which is fairly high, but fails to capture the actual important parts. To summarize the drawbacks with ROGUE can be defined as:

- It does not consider meaning
- It doesn't map well to human Judgements.

For our implmentation we utilize Rouge, a python library for the ROUGE Metric, available here: https://pypi.org/project/rouge/ (https://pypi.org/project/rouge/)

Sentiment Analysis

Taking into account the drawbacks that exist for ROGUE, we decided to explore using sentiment analysis as an evaluation metric. Though not a widely used evalation metric, it seems intuitive that a generated text should be a *true* representation of the original text in terms of essence, i.e. polarity and subjectivity. In other words, the gist of the original text should carry over in the generated summary. Our intuition has been further developed by recent developments in text generation, spefically a paper published by OpenAi, through generating text using an LSTM model, they found that it could be possible that sentiment could have some form of predictive capability for language modelling. This approach is only intended as a proof-of-conecpt, namely to illustrate the possibility of using sentiment as an evaluation metric for NLP tasks.

Obviously there are significant limitations to this approach as well. First, this approach assumes that the process of extracting sentiment from within a text is accurate. Though there has been major improvements in that field it is not reliable enough to adopt as an evaluation metric thus far. Another significant limitation that is present in this approach is that scores can be fairly misleading. For example, it is easy to see how a generated summary can have a similiar sentiment score to a completely irrelevant piece of text, even though they are not corelated in any meaningful way. On the flip side however, this score can be a good discriminator against poor summarizations. For example, suppose a summary has polarity score = 1 but the original text has a polarity score = -1 It would be a fair to say that the generated summary is not a very good representation of the actual text.

Several libraries exist that are capable of performing senetence analysis, for our task we opted to go for TextBlob's sentiment feature. Which gives a score of (-1, 1) for polarity and (0, 1) for subjectivity. After generating a score for both the original text and generated summary we then calculated the euclidean distance between the two observed scores to obtain the MSE.

Results

ROGUE Scores

Supervised:

- Rouge-1: {'f': 0.2526647980025532, 'p': 0.7992448496414776, 'r': 0.16294280503992503},
- Rouge-2: {'f': 0.13892572610457266, 'p': 0.45700418802087966, 'r': 0.08872223816078555},
- Rouge-l': {'f': 0.2391257235110693, 'p': 0.596278611832916, 'r': 0.1582649962597007}

Unsupervised:

- Rouge-1: {'f': 0.3714546751362649, 'p': 1.0, 'r': 0.28751559407671573},
- Rouge-2': {'f': 0.3645307049243793, 'p': 0.9832523052762513, 'r': 0.2815903639572497},
- Rouge-l': {'f': 0.46168703611867346, 'p': 1.0, 'r': 0.350442028853022 4}}

Sentiment Analysis MSE:

Supervised:

- Polarity MSE: 0.017046
- Subjectivity MSE: 0.024472

Unsupervised:

- Polarity MSE: 0.00652
- Subjectivity MSE: 0.009169

III: Conclusion

In conclusion, it does not appear easy to use computer generated models to generate headnotes for legal cases. We believe our results could be improved with more time and compute resources. Instead, we chose to devote time to experiment with both Unsupervised and Supervised approaches in order to get more experience with both techniques. Our results are not suprising given the amount of resources being invested into Legal AI Research and judging from past academic papers that have attempted to automate the analysis of legal documents. We look forward to tinkering with this code more in the future.

IV: Reference List

Nay, John, “Natural Language Processing and Machine Learning for Law and Policy Texts” (April 7, 2018). Available at : https://ssrn.com/abstract=3438276 (https://ssrn.com/abstract=3438276)

Pandya, Varun, “Automatic Text Summarization of Legal Cases: A Hybrid Approach” (Aug. 2019). Available at: https://arxiv.org/ftp/arxiv/papers/1908/1908.09119.pdf (https://arxiv.org/ftp/arxiv/papers/1908/1908.09119.pdf)

Polsley, Seth, Pooja Jhunjhunwala and Ruihong Huang. "CaseSummarizer: A System for Automated Summarization of Legal Texts." (Dec. 11, 2016). Available at: http://faculty.cse.tamu.edu/huangrh/papers/coling16_caseSummarizer.pdf (http://faculty.cse.tamu.edu/huangrh/papers/coling16_caseSummarizer.pdf)

Shifu Jain, How to Build a Legal Document Summarizer? Available at: https://www.affineanalytics.com/how-to-build-a-legal-document-summarizer/ (https://www.affineanalytics.com/how-to-build-a-legal-document-summarizer/)

University of Cincinnati Library, “Researching Case Law – Finding Cases by Headnote”. Available at: https://guides.libraries.uc.edu/c.php?g=222559&p=1472874 (https://guides.libraries.uc.edu/c.php?g=222559&p=1472874)

Georgetown Law Library, “Digest, Headnotes, and Key Numbers Research Guide”. Available at: https://guides.ll.georgetown.edu/digests (https://guides.ll.georgetown.edu/digests)

Radford, et al. “Learning to Generate Reviews and Discovering Sentiment.” ArXiv.org, 6 Apr. 2017, www.arxiv.org/abs/1704.01444.

Tatman, Rachael. “Evaluating Text Output in NLP: BLEU at Your Own Risk.” Medium, Towards Data Science, 16 July 2019, www.towardsdatascience.com/evaluating-text-output-in-nlp-bleu-at-your-own-risk-e8609665a213

“Simplified Text Processing.” TextBlob, www.textblob.readthedocs.io/en/dev/.

Sulem, Elior, et al. “BLEU Is Not Suitable for the Evaluation of Text Simplification.” ACL Anthology, 1 Jan. 1970, www.aclweb.org/anthology/D18-1081/.

Important notes to TA:

1. The Unsupervised notebook takes about 18 hours run.
2. The Supervised Model takes 1 day to train. So you will not be able to re-run it all.
3. We have model weights saved and results also saved. Link to the model weights https://drive.google.com/drive/folders/1kQulZ1PZ276p7Z1t7IJfxsyRxZukCdqy?usp=sharing (https://drive.google.com/drive/folders/1kQulZ1PZ276p7Z1t7IJfxsyRxZukCdqy?usp=sharing)
4. Link to saved Parqeut files which contains results and intermediate dataframes https://drive.google.com/drive/folders/1EeBU2w2sh6G08p8ERiOAj-_K9tkl72ih?usp=sharing (https://drive.google.com/drive/folders/1EeBU2w2sh6G08p8ERiOAj-_K9tkl72ih?usp=sharing)