DATA SCIENCE PRACTICUM: STATE LEGISLATURE BILL SUMMARY TEXT CLASSIFICATION

Katrina Greene

DATA 793
American University
College of Arts and Sciences

ABSTRACT

For my Data Science Practicum, I assisted in a research project that aims to analyze the minority party's influence in bills that pass through state legislatures. A requirement for this analysis is a data set of bills from each state that have become law as well as a numeric label that maps to the bill's topic. In order to produce this data set, I gathered and cleaned publicly available bill summaries for a subset of states and fine-tuned a selection of pre-trained Hugging Face transformer models for text classification. I found that the BERT model produced the highest F1 scores and performed better on the test set than DistilBERT and Longformer models.

1. INTRODUCTION

This semester, I worked with Professor Ballard on an extension of his research project *Tracing Policy Through Congress*. Professor Ballard's prior work focused on Congressional legislation, and he has gathered full bill text and is developing a classification model. The labels are defined in the U.S. Policy Agendas Project Code Book, which defines 20 separate labels for the major bill topics. Professor Ballard will use this data to see the substance of policy-making outcomes and whether there are patterns in them. An exhaustive data set of labeled state legislature bill is required to extend this project to include state legislatures. In order to reduce the amount of manual labor required to produce this data set, the goal of this practicum is to develop a classification model that accurately labels the data.

2. OBJECTIVES

My main objective to support my practicum client was to start the process of creating a labeled data set of all state legislation that has become law. In order to provide results within the scope of this semester, my work focused on two main goals:

1. Data collection and cleaning
2. Proof of concept classification model(s) for a state or group of states

3. DATA

3.1. Collection

The end goal of this project is to locate the full text of each state legislature bills. After unsuccessfully trying to reach out to vendors whom we knew provided access to this data, we decided to use publicly available data on bill summaries in order to keep the scope of this work within this semester. As such, I collected CSV files for the states Arizona, California, Colorado, Florida, Louisiana, Maine, Michigan, Missouri, Nevada, New York, Ohio, South Dakota, and Texas from the replication files of the paper *How Do Electoral Incentives Affect Legislator Behavior? Evidence from U.S. State Legislatures*. I also used the R package WashEx to query the Washington State Legislative API to get bill summaries for Washington. The resulting collection of CSV files contains common data for BillId, Summary, and Session.

3.2. Cleaning

In order to train a model to predict the correct topic label, I needed a data set of labeled data. Instead of hand coding a data set, I utilized the format that the Louisiana data set came with: the summary text included a key word in the beginning of the string. In a python script, I separated out this keyword and created a series of if/else statements that mapped a code based on whether the label matched the keyword. As a result, I produced a data set of 14,114 labeled observations of bill summaries from Louisiana's legislature. The classes of labels are not evenly distributed; for example, there is a much larger amount of bills about crime (4,539) than about agriculture (173). immigration and foreign trade were not present at all; this makes sense given that this is a state-level application, so Professor Ballard and I agreed the model would be trained on 18 classes instead of trying to find state examples of immigration or foreign trade legislation. Below are is an example observation from the training data set from class label 1 (crime):

- Summary: Creates crimes of sales, purchase, and use of air brush propellant involving minors

4. METHODS

Given that the eventual goal of this project is to apply a model to a large amount of input text (full bill text), I focused on pre-trained transformer models for my model development. I chose to use the transformer architecture for my model development because transformer models handle large text inputs for natural language processing very well. Prior to transformer models, most NLP models utilized Recurrent Neural Networks (RNNs) that process text sequentially in a loop-like approach. The limitations with this are that as the model progresses, the relevance of previously processed words diminish. Transformer deep learning models were introduced to address this with the concept of self-attention, which allows the model to process the entire input all at once. The paper *Attention is All You Need* proposed the transformer model, which includes an encoder and a decoder section, each made up of "attention" layers with feed-forward neural networks. The encoder creates an embedding of the input text by converting the text to numerical representations with weights to how each word relates to one another. The attention layers take this input and use the neural networks to update and improve the embedding. The decoder then translates that embedding back to text. A text classification model only uses the encoder with a softmax output layer that outputs the probability that the input text belongs to a certain class.

Since I am using transformer models, I utilized the Hugging Face library for pre-trained models since it provides a range of transformer models for natural language processing. Pre-trained models are a powerful tool because they been trained on a large amount of data, and the weights calculated from this generic data can be used for your own application. From this library, I selected a handful of models to test with my data:

1. BERT
    - Bidirectional transformer pretrained with a large language corpus (including Wikipedia) using masked language modeling and next sentence prediction
2. DistilBERT
    - A smaller, cheaper, and faster transformer model trained by distilling BERT base
3. Longformer
    - A model that utilizes an attention mechanism that scales linearly with sequence length instead of quadratically by specifying pairs of input locations attending to one another. This addresses transformers' computational limitation with processing long sequences

5. MODEL DEVELOPMENT

Using these model architectures with my labeled training data, I followed these steps for each separate model:

5.1. Data Pre-Processing

In order to pass text into a model for text classification, you need to map the input text to numeric values using a tokenizer. Hugging Face provides a tokenizer function based on which model you are using, and it splits the text into tokens based off of that model's pre-training. These tokens are converted into numbers and then tensors, which become the model inputs.

5.1.1. Fine-Tuning

Once the data is pre-processed, I selected which Hugging Face model I needed and specified the number of classes that my application needs. This is called fine-tuning the model.

5.1.2. Hyperparameters

Most of my work in model development was involved in finding the best hyperparameters to increase the model's performance. Hyperparameters refer to the different training arguments that you can pass to the model; the ones that I focused on related to the optimizer. The optimizer is the algorithm that the model uses to find the global minimum of the loss function used to calculate the best weights. For manual hyperparameter tuning, I tried different values for the learning rate, the warmpup steps, and the weight decay. The learning rate is the size of the steps the optimizer takes to converge at the global minimum. The warmup steps argument adds steps between 0 and the learning rate step size to allow an adaptive optimizer to tune the attention layers before using the learning rate to converge. Finally, weight decay is a type of regularization that prevents overfitting by keeping the weights from getting too large by adding a penalty term to the weight calculations. I also experimented with utilizing Hugging Face's hyperparameter search offering. Hugging Face provides an integration for hyperparameter searching with RayTune, which is a Python library that provides out of the box algorithms for hyperparameter selection. I provided a search space with ranges of values for the hyperparameters listed above, and the hyperparameter search runs through the space and selects the best combination. This is very computationally expensive, however, and my client may not need this level of tuning, especially if the training data size grows in the future. I learned a lot with setting up the pipeline though and have provided it in case it is helpful in the future work.

5.1.3. Evaluation Metrics

For evaluation metrics, I focused on utilizing the f1 score to see how the model preformed. I used the f1 score instead of accuracy because of the imbalance of the data set. Using just the accuracy, which is the amount of correct predictions out of total observations, is less informative than the balance between precision (amount of true positives over true positives and false positives) and recall (amount of true positives over true positives and false positives).

6. RESULTS

After training the 3 different models with different combinations of hyperparameters, epochs, and training/test sizes, I found that the BERT model performed the best with an f1 score of 0.846 with a learning rate of 0.001, weight decay 0.001, and warmup steps at 50. The DistilBERT model was the second best but not by much over the Longformer with an f1 score at 0.759 with learning rate 0.001, weight decay at 0.01, and 50 warmup steps. The Longformer performed the worst with an f1 score of 0.735 with the default hyperparameters of learning rate 0.001 and weight decay at 0 and 0 warmup steps. Given the size of the input for my training data, I think it makes sense that the BERT model performed the best. I am inputting a smaller string of text (1 to 2 sentences), so the the Longformer's sliding attention window was probably too complex and narrow. The full attention from the BERT model performed better, and it did not need to be reduced with the DistilBERT model.

7. FUTURE WORK

This work will be continued by Professor Ballard, his co-authors, and incoming undergraduate RAs at the University of Virginia. The goal of this practicum was to get a start on this data collection work and provide a proof-of-concept model for classification that can be further developed and refined as new data is gathered. I have provided my client with all of the code that I have developed this semester and have had knowledge transfer meetings with the new RA's. My suggestion for continuing model development is to try training this BERT model with hand-coded training data from another state and see if it performs similarly to the Louisiana training data. I would continue with the BERT model if the training data continues to consist of bill summaries; if full bill text data is collected, I would suggest testing the Longformer model again and see if it performs better on larger input text.

8. REFERENCES

BALLARD, ANDREW O., and JAMES M. CURRY. "Minority Party Capacity in Congress." American Political Science Review, vol. 115, no. 4, 2021, pp. 1388–1405., doi:10.1017/S0003055421000381.

Fouirnaies, Alexander; Hall, Andrew B., 2021, "Replication Data for: How Do Electoral Incentives Affect Legislator Behavior? Evidence from U.S. State Legislatures", https://doi.org/10.7910/DVN/LHTRWM, Harvard Dataverse, V1, UNF:6:XGr/wbiEPZkoRpVSVv+zIg== [fileUNF]

Beltagy, Iz, et al. Longformer: The Long-Document Transformer. 2 Dec. 2020. "How Do Transformers Work? - Hugging Face NLP Course." Huggingface.co, huggingface.co/learn/nlp-course/chapter1/4?fw=ptgeneral-architecture.

Vaswani, Ashish, et al. Attention Is All You Need. 6 Dec. 2017.

Dorfer, Thomas A. "How to Choose the Best Evaluation Metric for Classification Problems." Medium, 17 Apr. 2023, towardsdatascience.com/how-to-choose-the-best-evaluation-metric-for-classification-problems-638e845da334.