## **Political Stance Classification of U.S. Congressional Bills**

**Imaad Khan** July 2025  
 [GitHub Repo](https://github.com/ImaadKhan17/PoliticalModel)

### **Abstract**

Understanding the ideological stance of legislation is crucial for analyzing political behavior and polarization. This project develops a model that classifies U.S. Congressional bills by both topic and political stance, using a multi-task Longformer-based architecture. Bill summaries were obtained from the U.S. Congress API and matched with roll call vote data from Voteview to derive Nominate dimension 1 scores, which act as a proxy for ideological stance. The model simultaneously performs topic classification (199 labels) and stance regression, trained on over 25,000 labeled examples. Results show high topic classification accuracy (~80%) and reasonably strong stance predictions using Nominate-based labels. This project highlights the potential of NLP and deep learning to make complex legislative text computationally tractable, opening the door to automated political analysis, lawmaker ideology mapping, and policy trend monitoring.

### 1. Introduction

Legislation in the United States Congress is dense, nuanced, and ideologically driven — but often hard for the public and researchers to interpret at scale. While voting records provide insight into lawmaker positions, few tools exist to directly analyze the political stance of the **bills themselves**. This project aims to fill that gap using natural language processing.

My goal was to build a model that could, from the **text of a bill summary**, infer both:

* **What the bill is about** (topic classification)
* **Where it sits politically** (economic left vs. right stance)

This was not just a coding challenge, but a full pipeline of research, engineering, and experimentation. From raw bill titles to full summaries, data scraping, API limitations, label construction, and model training. Ultimately, I wanted to demonstrate how text models could be applied to understand congressional ideology and how lawmakers align on different policy topics.

### 2. Dataset and Labeling

This project leveraged two main data sources to create a labeled dataset for training and evaluation:

#### *Congressional Bills Project Dataset*

I started with the Congressional Bills Project dataset, which includes metadata and topic labels for over 20,000 U.S. Congressional bills. The dataset provides initial topic classifications based on a standardized code system with 199 distinct categories. However, the dataset only includes bill titles and metadata(titles), which are often too short to capture detailed content.

#### *U.S. Congress API Summaries*

To provide richer text data, I used the U.S. Congress API to retrieve official bill summaries. This involved writing a looped script to fetch summaries for each bill in the dataset, respecting API rate limits (5,000 requests per hour). These summaries offer much more detailed descriptions, enabling the model to better learn nuanced language patterns.

#### *Voteview Nominate Scores for Stance*

To label each bill with a political stance, I utilized the Voteview dataset, which includes Nominate dimension 1 scores. These scores reflect ideological positions of lawmakers based on roll call votes — ranging from economic liberal (-1) to conservative (+1). By mapping each bill’s associated roll call votes to these scores and averaging across legislators, I created a continuous stance label for the dataset.

This process combined topic classification labels (discrete categories) with stance regression labels (continuous values between -1 and 1) for multi-task learning.

### 3. Model Architecture

To handle the long and complex text of bill summaries, I selected the **Longformer** model, a transformer architecture designed for long documents by employing efficient attention mechanisms.

#### *Multi-Task Setup*

The model is designed for **multi-task learning**, with two output heads trained simultaneously:

* **Topic Classification Head:** A linear layer that outputs logits for 199 topic classes.
* **Stance Regression Head:** A linear layer that outputs a single continuous value representing the political stance.

#### *Input Processing*

Each bill summary is tokenized and padded/truncated to a maximum length of 1028 tokens using the Longformer tokenizer, balancing context retention with computational efficiency.

#### *Feature Extraction*

The Longformer base model produces contextualized token embeddings. To create a fixed-size document representation, the model computes a masked average of token embeddings weighted by the attention mask.

#### *Loss Functions*

* For topic classification, **cross-entropy loss** is used to optimize discrete class predictions.
* For stance regression, **mean absolute error (MAE)** loss is applied to capture continuous ideological scores.

This joint training enables the model to leverage shared representations, improving overall understanding of bill text and enhancing both classification and regression tasks.

### 4. Training and Evaluation

#### *Training Setup*

The model was trained on a Google Colab environment using PyTorch and Huggingface Transformers libraries. The dataset was split into training and validation sets to monitor performance and prevent overfitting.

#### *Loss and Optimization*

* The **topic classification** task used **cross-entropy loss** to optimize prediction accuracy over 199 classes.
* The **stance regression** task used **mean absolute error (MAE)** loss to capture ideological position on a continuous scale.
* The two losses were combined with equal weighting during backpropagation to jointly train the model.

The Adam optimizer with weight decay was used to update model parameters, with a learning rate schedule to balance convergence speed and stability.

#### *Evaluation Metrics*

* **Topic classification accuracy:** Percentage of correctly predicted bill topics on the validation set. The model achieved approximately 80% accuracy, showing strong topical understanding.
* **Stance regression MAE:** The mean absolute error of predicted stance scores compared to true Nominate dimension 1 averages, showing reasonable performance despite noisy labels.

#### *Challenges*

* API rate limits required careful batching and delays during data fetching.
* Label noise and missing data in the Voteview dataset introduced some uncertainty in stance labels.
* Processing long texts necessitated using Longformer rather than standard BERT, increasing computational cost.

Despite these challenges, the model generalized well, balancing classification and regression tasks effectively.

### 5. Results

The trained model demonstrated strong performance on both tasks, showcasing its ability to understand and classify congressional bills by topic and political stance.

#### *Topic Classification*

The model achieved approximately **80% accuracy** on the validation set for the 199-topic classification task. This high accuracy indicates the model effectively learned the diverse range of legislative topics from textual summaries.

#### *Stance Regression*

For the continuous stance prediction task, the model produced a **mean absolute error (MAE)** of around **0.15** on the validation set. While this shows a reasonably close approximation of ideological position, some variance remains due to inherent noise in the roll call-based labels.

#### *Example Predictions*

* A bill summary discussing healthcare reform was correctly classified under the appropriate health-related topic, with a left-leaning stance prediction aligning with known political positions.
* A defense spending bill received a conservative stance score, consistent with typical partisan views.

#### *Insights and Limitations*

* Using summaries instead of titles significantly improved model understanding.
* Joint multi-task training helped improve performance on both tasks by leveraging shared representations.
* Some misclassifications occurred on bills with ambiguous or broad topics.
* Stance labels derived from voting averages inherently include noise, limiting regression precision.

Overall, these results validate the approach of combining advanced NLP architectures with congressional data to analyze legislative text and political ideology.

### 6. Conclusion & Future Work

This project developed a multi-task Longformer-based model capable of classifying U.S. Congressional bills by topic and predicting their ideological stance using legislative summaries and roll call data. Despite challenges like noisy labels and long-text processing, the model achieved strong topic classification accuracy and reasonable stance regression performance.

The approach demonstrates the potential of modern natural language processing to enhance political analysis by automating interpretation of complex legislative text. It offers a scalable method for tracking lawmaker behavior and ideological shifts over time.

**Future work** could improve this foundation by:

* Incorporating more granular political stance labels, possibly using expert annotations.
* Expanding input data with full bill texts beyond summaries.
* Exploring alternative architectures like Longformer-Encoder-Decoder models or attention-based graph networks to better capture legislative structure.
* Expanding to state level or even local bill data and classification
* Building a user-facing tool or dashboard for real-time stance analysis of new bills.

This project has laid the groundwork for computational political science applications, blending deep learning with public data to better understand governance and policy.

### 7. Appendix

#### Code Snippet: Model Definition

from transformers import AutoModel

from torch import nn

class PoliticalModel(nn.Module):

def \_\_init\_\_(self, model\_name, num\_classes):

super(PoliticalModel, self).\_\_init\_\_()

self.model = AutoModel.from\_pretrained(model\_name)

self.dropout = nn.Dropout(0.2)

self.topic\_head = nn.Linear(self.model.config.hidden\_size, num\_classes)

self.stance\_head = nn.Linear(self.model.config.hidden\_size, 1)

def forward(self, input\_ids, attention\_mask):

outputs = self.model(input\_ids=input\_ids, attention\_mask=attention\_mask)

last\_hidden\_state = outputs.last\_hidden\_state

mask = attention\_mask.unsqueeze(-1).expand(last\_hidden\_state.size()).float()

masked\_embeddings = last\_hidden\_state \* mask

x = masked\_embeddings.sum(dim=1) / mask.sum(dim=1)

x = self.dropout(x)

topic\_logits = self.topic\_head(x)

stance\_pred = self.stance\_head(x)

return topic\_logits, stance\_pred

#### *Repository*

All code, data processing scripts, and model files can be found at:  
<https://github.com/ImaadKhan17/PoliticalModel>

#### *Additional Resources*

* U.S. Congress API documentation: https://api.congress.gov
* Voteview Nominate data: https://voteview.com/data