Political Speech Classification

Group Report

Group 3*

December 8, 2025

Abstract

This report examines the effectiveness of classical and Transformer-based NLP methods for classifying the political party of U.S. Congressional speech authors. Using a filtered subset of 1981–1989 speeches designed to improve text quality, the analysis compares a TF-IDF + Logistic Regression baseline with a fine-tuned RoBERTa model. The results show that data filtering substantially improves performance for both approaches, and that RoBERTa achieves the highest accuracy and F1 scores, indicating the value of contextualized embeddings in political text classification.

*Group 3 consists of only one member: Minwoo Yoo

1 Introduction

This study compares classical and Transformer-based NLP methods in predicting the political party of U.S. Congressional speakers. The full dataset contains more than eleven million speeches, but I restrict attention to the 1981–1989 period. This restriction reduces computational requirements and keeps the linguistic environment stable across all models.

The classification task is binary: identify whether each speech was delivered by a Democrat or a Republican. I evaluate two modeling strategies. The first uses a TF-IDF representation with Logistic Regression as a baseline. The second fine-tunes the RoBERTa Transformer model, which provides contextual embeddings and captures richer linguistic patterns.

The analysis focuses on how data filtering, preprocessing, and model choice affect predictive accuracy. I apply progressively stricter filters to remove short or low-information speeches and examine how these filters influence model performance. The results show how modern Transformer architectures perform relative to traditional bag-of-words approaches when both operate on high-quality political text.

2 Data Description

2.1 Source and Labels

The dataset consists of speeches from the U.S. Congressional Record. Each entry includes the speech text and metadata identifying the speaker. After merging member information with the speech corpus, each speech receives a binary party label (D or R). This labeling provides the supervised structure needed for classification.

2.2 Data Reduction

I apply a series of filters to remove short or low-information speeches and to reduce the computational burden of model training. Table 1 summarizes how these filters narrow the

dataset.

Table 1: Sample Reduction Summary

Subset	Period	Filter	Size (M)
A	1981–Present	$\geq$ 30 chars	2.27
B	1981–1989	$\geq$ 30 chars	0.81
C (Final)	1981–1989	$\geq$ 2 sents, $\geq$ 200 chars	0.39

The final subset contains 0.39 million speeches and provides the basis for both the TF-IDF baseline and the RoBERTa fine-tuning exercises. This filtering step raises average speech quality and improves downstream model stability.

3 Model Background

3.1 TF-IDF

The TF-IDF representation maps each document into a weighted bag-of-words vector. The term-frequency component,

$$TF(t,d) = \frac{\text{count}(t,d)}{\text{total terms in } d},$$

captures how strongly a document uses a specific term. The inverse-document-frequency component,

$$IDF(t) = \log\left(\frac{N}{1+n_t}\right),$$

down-weights terms that appear in many documents across the corpus. The final weight is

$$TFIDF(t,d) = TF(t,d) \times IDF(t),$$

which produces a high value only when a term is frequent within a document but relatively rare across the dataset.

This encoding yields a high-dimensional but sparse vector. I train a Logistic Regression classifier on these vectors. Because TF-IDF treats each term independently, it cannot represent context or word meaning beyond frequency patterns, but it provides a strong and interpretable baseline.

3.2 RoBERTa

RoBERTa adopts the Transformer encoder architecture, which uses self-attention to create contextual representations of every token. Unlike TF-IDF, RoBERTa generates embeddings whose meaning depends on the surrounding words. This feature allows the model to capture rhetorical framing, topic emphasis, and subtle linguistic cues that often correlate with political identity.

RoBERTa modifies the original BERT training strategy in several ways:

- It trains on a substantially larger corpus, improving generalization.
- It applies dynamic masking, exposing the model to different masked tokens during training.
- It removes the next-sentence prediction task, allocating more capacity to language modeling.

These changes improve stability during fine-tuning and typically deliver stronger downstream performance. When fine-tuned on Congressional speeches, RoBERTa adapts its contextual embeddings to distinguish partisan language patterns more effectively than classical bag-of-words models.

4 Experimental Setup

4.1 Train/Validation/Test Split

I divide the data into training (80 percent), validation (10 percent), and test (10 percent) sets. The training set determines model parameters, the validation set guides early stopping and hyperparameter choices, and the test set provides final out-of-sample evaluation. All splits are stratified by party label to maintain a stable class ratio across subsets.

4.2 TF-IDF Baseline

The TF-IDF baseline serves as a classical benchmark against which I compare RoBERTa. I construct a vocabulary of 50,000 features consisting of unigrams and bigrams. To reduce noise, I drop rare terms using `min_df = 5`. I fit a Logistic Regression classifier with the SAGA solver, which is well suited for large sparse matrices. Because Democrats and Republicans appear at different frequencies in the filtered dataset, I use balanced class weights to correct for label imbalance.

4.3 RoBERTa Fine-Tuning

For the Transformer model, I fine-tune `roberta-base` on the same filtered speech dataset. All text is tokenized using the fast RoBERTa tokenizer with a maximum sequence length of `256`. The training procedure uses a batch size of 32, learning rate of 1×10^{-5}, and the AdamW optimizer with weight decay 0.01. I train models for 1 to 3 epochs, following standard practice for moderate-sized fine-tuning tasks.

During training, I monitor accuracy and F1 on the validation set after each epoch. This step helps detect overfitting and guides the selection of the final checkpoint. All reported metrics in the Results section use the held-out test set, which remains untouched during model development.

In addition, the fine-tuning code employs gradient clipping and mixed-precision training (AMP) when running on GPU hardware, which improves numerical stability and speeds up training without affecting results.

5 Results

5.1 TF-IDF Performance

Table 2 reports the performance of the TF-IDF baseline models. Accuracy remains near 0.65 when using minimal preprocessing, but the stricter content filter increases accuracy and F1 to roughly 0.69.

Table 2: TF-IDF Model Performance

Model	Period	Filter	Acc	F1
TF1	1981–2011	min	0.650	0.649
TF2	1981–1989	none	0.649	0.648
TF3	filtered	yes	0.694	0.692

5.2 RoBERTa Performance

RoBERTa improves performance relative to TF-IDF. Models trained on the filtered subset outperform those trained on unfiltered text. Accuracy rises from 0.662 (M2) to 0.729 (M5) as the model trains for additional epochs on the cleaned dataset.

Table 3: RoBERTa Model Comparison

Model	Period	Filter	Epoch	Acc	F1
M1	1981–2011	none	1	0.658	0.634
M2	1981–1989	none	1	0.662	0.659
M3	filtered	yes	1	0.692	0.679
M4	filtered	yes	2	0.715	0.699
M5	filtered	yes	3	**0.729**	**0.713**

5.3 Confusion Matrix

The best model (M5) shows higher recall for Democratic speeches than for Republican speeches:

$$Recall_D = \frac{15507}{15507 + 4261} \approx 0.78, \qquad Recall_R = \frac{13235}{13235 + 6396} \approx 0.67.$$

The model tends to classify some Republican speeches as Democratic, producing an asymmetric error pattern.

6 Conclusion

6.1 Main Findings

- Data filtering (2 sentences, 200 chars) materially improves model accuracy.
- Fine-tuned RoBERTa substantially outperforms the TF-IDF baseline.

6.2 Limitations

- Limited hyperparameter exploration.
- Choices such as `max_length`, filtering variations, and preprocessing alternatives were not systematically tuned.

7 Future Work

Future extensions include integrating these models into economic analyses that rely on bigram-based partisanship measures, evaluating robustness across architectures, and conducting a larger hyperparameter sweep.

References

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of NAACL-HLT.

[2] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., ... & Stoyanov, V. (2019). *RoBERTa: A Robustly Optimized BERT Pretraining Approach*. arXiv preprint arXiv:1907.11692.

[3] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). *Attention Is All You Need*. Advances in Neural Information Processing Systems.

[4] Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). *Transformers: State-of-the-Art Natural Language Processing*. Proceedings of the 2020 EMNLP: System Demonstrations. (HuggingFace library used for model fine-tuning.)

[5] Apache Arrow. (2023). *Apache Arrow: A Cross-Language Development Platform for In-Memory Data*. Used for streaming Parquet batch processing to prevent OOM on EC2.

[6] Kingma, D. P., & Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. arXiv preprint arXiv:1412.6980. (Optimizer used: AdamW variant implemented in PyTorch.)

[7] Loshchilov, I., & Hutter, F. (2017). *Decoupled Weight Decay Regularization*. arXiv preprint arXiv:1711.05101. (Weight decay regularization as implemented in AdamW.)

[8] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... & Duchesnay, É. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research. (Used for Accuracy and F1 score evaluation.)

[9] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Chintala, S. (2019). *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Advances in Neural Information Processing Systems.