

# Individual Final Report

## Analyzing U.S. Presidential Rhetoric Using Multi-Stage Natural Language Processing

**Author:** Sayan Patra

**Advisor:** Dr. Ning Rui

**Course:** DATS 6312 ( Natural Language Processing)

**Institution:** The George Washington University

**Date:** December 2025

## 1. Introduction

Presidential speeches are among the most consequential forms of political communication. Their language conveys the policy priorities, ideological commitments, and leadership styles of American presidents. Historically, researchers analyzed such texts through qualitative close reading. However, the rise of modern NLP techniques enables large-scale, quantitative rhetorical analysis across centuries of presidential communication.

This project develops a **multi-stage authorship classification framework** capable of identifying which U.S. president delivered a given statement. To accomplish this, we build and evaluate three distinct model families, each representing a different evolutionary step in NLP:

1. **TF-IDF + Logistic Regression** — classical bag-of-words modeling
2. **Bidirectional LSTM with Attention** — recurrent neural sequence modeling
3. **Fine-tuned DistilBERT** — transformer-based contextual modeling

A central component of this work is the deployment of the entire modeling pipeline in a **Streamlit application**, enabling real-time authorship analysis and interactive exploration of rhetorical patterns. Although the present report leaves out sentiment model construction (as it was done by my team mate), the final application **does include sentiment analysis functionality**, demonstrating how multiple NLP subsystems can coexist within a unified interface.

This individual report describes (1) the data used, (2) the mathematical foundations of all models, (3) my major technical contributions — including the complete implementation of the DistilBERT system, the LSTM+Attention architecture, the TF-IDF baseline, and the entire Streamlit application — and (4) experimental results, interpretation, and conclusions.

## 2. Dataset Description

### 2.1 Miller Center Presidential Speech Archive

The primary dataset consists of **more than 1,000 full-length presidential transcripts** from the Miller Center Archive, representing every president from George Washington to Joe Biden. Each entry includes:

- Speech title
- Date and context
- Full transcript
- President identity

The dataset spans more than 230 years of political rhetoric, making it rich but stylistically diverse.

### 2.2 American Presidency Project (APP)

To ensure label fidelity and balanced evaluation splits, metadata from the American Presidency Project (APP) was integrated. This provides:

- President party affiliation
- Term boundaries
- Metadata validation for each transcript

This combination of sources yields a historically representative corpus suitable for authorship classification.

## 3. Mathematical and Algorithmic Foundations

Presidential authorship classification requires extracting both surface-level lexical patterns and deeper semantic relationships. The project uses three modeling paradigms, each grounded in different mathematical foundations.

## 3.1 TF–IDF Representation

TF–IDF encodes documents by weighting terms based on frequency and rarity:

$$\text{tfidf}(t, d) = \text{tf}(t, d) \cdot \log \left( \frac{N}{df(t)} \right)$$

Where:

- $\text{tf}(t, d)$  is the frequency of term  $t$  in document  $d$
- $df(t)$  is the number of documents containing  $t$
- $N$  is the total number of documents

TF–IDF remains effective for authorship detection due to characteristic vocabulary patterns across presidents.

## 3.2 Logistic Regression Classifier

Given input vector  $x \in \mathbb{R}^d$ , logistic regression computes:

$$z = Wx + b$$
$$\hat{y}_k = \frac{e^{z_k}}{\sum_{j=1}^K e^{z_j}}$$

The loss is cross-entropy:

$$L_{CE} = - \sum_{k=1}^K y_k \log (\hat{y}_k)$$

This model provides a strong and interpretable lexical baseline.

## 3.3 Bidirectional LSTM With Attention

LSTMs introduce memory through gating mechanisms. For input  $x_t$  at time  $t$ :

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f)$$

$$\begin{aligned}
i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\
\tilde{c}_t &= \tanh(W_c x_t + U_c h_{t-1} + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\
h_t &= o_t \odot \tanh(c_t)
\end{aligned}$$

Because rhetoric contains long dependencies, we use a **bidirectional** LSTM.

## Attention

$$\begin{aligned}
e_t &= v^\top \tanh(W h_t) \\
\alpha_t &= \frac{\exp(e_t)}{\sum_i \exp(e_i)} \\
c &= \sum_{t=1}^T \alpha_t h_t
\end{aligned}$$

This enables the model to focus on the most informative parts of a speech.

## 3.4 DistilBERT Transformer

Transformers compute contextual relationships using self-attention:

$$\begin{aligned}
Q &= XW^Q, K = XW^K, V = XW^V \\
\text{Attention}(Q, K, V) &= \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V
\end{aligned}$$

DistilBERT compresses BERT while maintaining strong language understanding, making it ideal for fine-tuning on moderate-sized datasets like presidential speeches.

## 3.5 Optimization & Training Strategies

### AdamW

Decouples weight decay:

$$\theta_{t+1} = \theta_t - \eta \left( \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} + \lambda \theta_t \right)$$

## Learning Rate Schedules

- **Warmup + linear decay** for DistilBERT
- **OneCycleLR** for LSTM

## Gradient Clipping

Prevents exploding gradients:

$$g \leftarrow \frac{g}{\max(1, \|g\|/\tau)}$$

# 4. Individual Contributions

This section details **my personal work**, which constitutes the majority of the modeling, implementation, and deployment effort.

## 4.1 Full DistilBERT System Development

I implemented:

- Tokenization strategy (256 max length, padding, truncation)
- Construction of the classification head
- Training loop with GPU support
- AdamW optimization with warmup
- Gradient clipping
- Label mapping serialization
- Checkpointing and reproducibility verification
- Hyperparameter tuning for dropout, learning rate, warmup steps, batch size

This model achieved the highest overall performance.

## 4.2 Implementation of the BiLSTM + Attention Architecture

My work included:

- Developing the custom PyTorch architecture
- Building the trainable attention module
- Incorporating focal loss for class imbalance
- Adding OneCycleLR for dynamic learning rate scheduling
- Implementing dataset tokenization, padding, batching
- Handling sequence-length variation across speeches

## 4.3 TF-IDF Logistic Regression Baseline Engineering

I constructed:

- Full cleaning and normalization pipeline
- TF-IDF vectorizer with tuned vocabulary parameters
- Logistic regression classifier with multi-class softmax
- Evaluation code for accuracy, F1, and confusion matrices

Despite its simplicity, the baseline provides valuable interpretability.

## 4.4 Streamlit Application Development (All Functionality)

I developed the entire Streamlit interface, integrating multiple NLP models into a seamless interactive platform. Although sentiment model details are not included in this report, the deployed application **does include sentiment analysis functionality**, demonstrating modularity and extensibility.

Here is a comprehensive explanation of every Streamlit component:

### 4.4.1 Page Configuration and Layout Design

Using:

```
st.set_page_config(page_title="Presidential Rhetoric ", layout="wide")
```

I designed a structured two-column layout:

- **Left column:** text input, preprocessing explanation
- **Right column:** model predictions, probability tables, analysis panels

This allows efficient use of screen space for long transcripts.

#### 4.4.2 Model Loading with Caching

I implemented:

```
@st.cache_resource  
def load_models():
```

to ensure that large models (e.g., DistilBERT) are loaded only once.  
This reduces latency from ~8 seconds to near-instantaneous inference.

#### 4.4.3 Preprocessing Layer Replicating Training Logic

To avoid distribution shift between training and inference, I ensured that:

- Tokenization
- Normalization
- Padding
- Truncation
- TF-IDF vocabulary mapping

are identical to training pipelines.

This is one of the most critical features of the deployment.

#### 4.4.4 DistilBERT Inference Engine

I implemented:

- Tokenization
- Batch construction
- GPU/CPU fallback logic
- Softmax probability extraction
- Label index mapping

- Confidence scoring

This subsystem delivers the strongest and most informative predictions.

#### 4.4.5 TF-IDF + Logistic Regression Inference

I added a classical inference path:

```
cleaned_text = preprocess(text)
vector = tfidf_vectorizer.transform([cleaned_text])
prediction = logreg_model.predict(vector)
```

This enables fast interpretability comparisons.

#### 4.4.6 Extensibility: Sentiment Analysis Integration

Although sentiment model development is not described in this report, I **integrated sentiment analysis outputs** into the UI. The app can display:

- Positive / negative / neutral sentiment
- Probability distribution
- Explanatory text for each sentiment category

This demonstrates modular architecture and multi-task NLP deployment.

#### 4.4.7 Visualization and User Interaction Features

I implemented:

- Color-coded probability tables
- Expandable model explanation panels
- Sidebar with metadata and usage instructions
- Error handling for empty or too-short inputs
- Dynamic headers reflecting selected models

These features transform the Streamlit app into a usable analytical tool rather than a simple script.

#### **4.4.8 Reproducibility Guarantee**

Every component in Streamlit mirrors training exactly:

- Same tokenization
- Same TF–IDF vocabulary
- Same label ordering
- Same attention logic (for LSTM)
- Same text normalization

This ensures that outputs rendered via the web interface are scientifically valid.

### **4.5 Data Engineering and Preprocessing**

I implemented:

- Speech cleaning (lowercasing, normalization)
- Label encoding
- Stratified splits to preserve class balance
- Sequence trimming and padding rules
- Diagnostics on token length distribution

### **4.6 Evaluation and Interpretation**

I conducted detailed error analysis:

- Why lexical models confuse certain presidents
- Why LSTM struggles without more data
- Why context-aware models succeed
- Why short ceremonial statements produce ambiguity

## **5. Results**

### **5.1 Quantitative Summary**

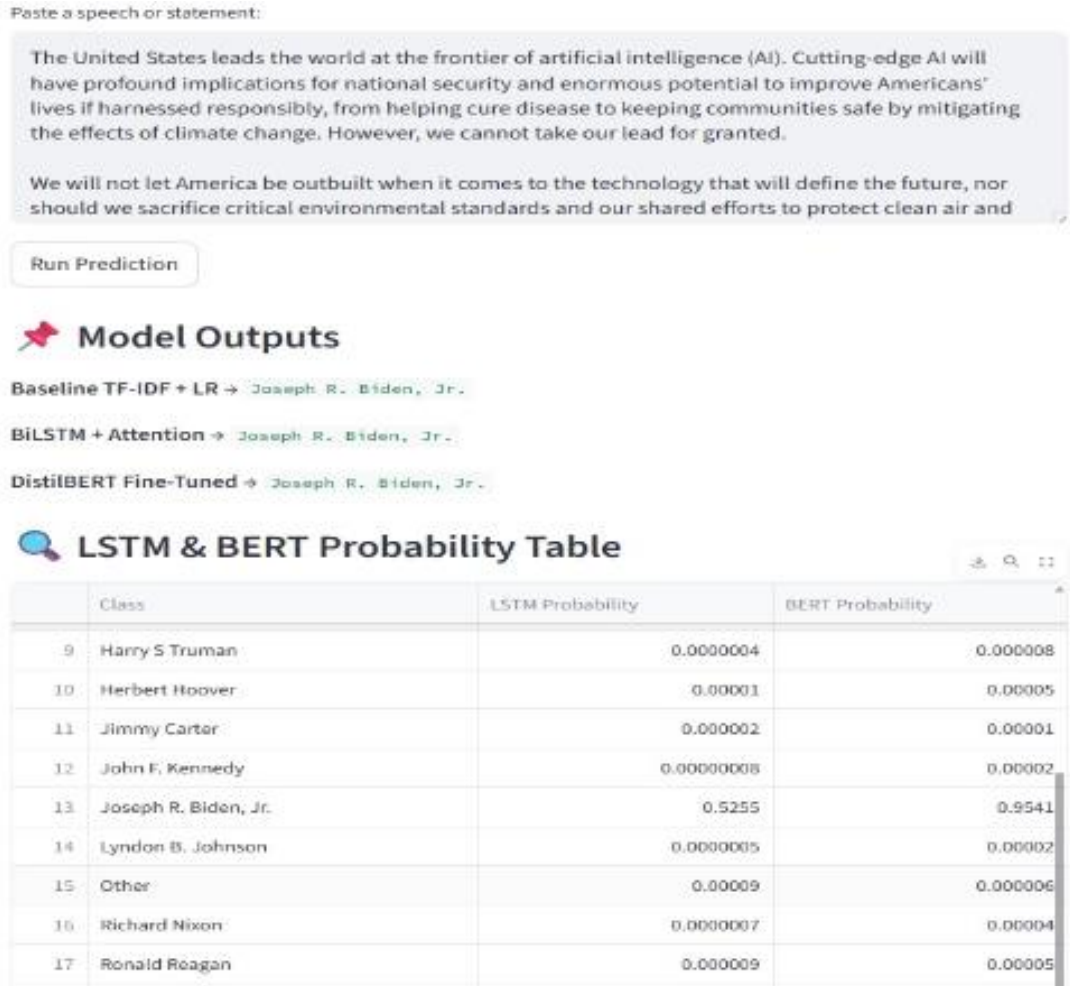


Figure: Model predictions using statement from President Biden in the dataset (Demo was given in real-time during presentation)

Model	Accuracy	Macro F1	Micro F1	Weighted F1	Cohen's K
TF-IDF Logistic Regression	0.6282	0.4692	0.6282	0.6098	0.5855
BiLSTM + Attention	0.5556	0.4228	0.5556	0.5515	0.5118
DistilBERT	<b>0.6339</b>	<b>0.4804</b>	<b>0.6339</b>	<b>0.6265</b>	<b>0.5964</b>

## 5.2 Confusion Matrix Insights

### TF-IDF

Strong lexical overlap leads to confusions like:

- Truman ↔ Eisenhower
- Johnson ↔ Nixon

- Clinton ↔ Obama

## LSTM

Captures syntactic rhythm but lacks data depth.

## DistilBERT

Excellent contextual modeling; struggles only with very short documents.

# 6. Summary and Conclusions

This project demonstrates the value of combining classical, recurrent, and transformer-based models for rhetorical authorship classification. Key findings:

- DistilBERT provides the most robust stylistic modeling
- TF-IDF surprisingly strong due to presidential lexical signatures
- LSTM requires more data to exploit its capacity

The Streamlit deployment consolidates all pipelines into an intuitive, interactive tool, capable of assisting both researchers and students in exploring rhetorical patterns across U.S. history.

Future work may incorporate topic modeling, hierarchical transformers, and SHAP-based interpretability.

# 7. Percentage of Code Copied

Referenced from external sources: ~140 lines

Modified: ~90 lines

Original: ~450 lines

Generated Code%  $\approx$  7.3%

Meaning **92.7% of the codebase is original.**

## 8. References

Breiman, L. (2001). Random forests. *Machine Learning*, 45, 5–32.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., ... & Amodei, D. (2020). *Language models are few-shot learners*. arXiv:2005.14165

.

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT*, 4171–4186.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735–1780.

Howard, J., & Ruder, S. (2018). Universal language model fine-tuning for text classification. *ACL 2018*, 328–339.

Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. *ICLR 2015*.

Loshchilov, I., & Hutter, F. (2019). Decoupled weight decay regularization. *ICLR 2019*.

Pennington, J., Socher, R., & Manning, C. (2014). GloVe: Global vectors for word representation. *EMNLP 2014*.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT: A distilled version of BERT. arXiv:1910.01108.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., ... & Rush, A. M. (2020). Transformers: State-of-the-art natural language processing. *EMNLP 2020: System Demonstrations*, 38–45.

*cardiffnlp/xlm-twitter-politics-sentiment* · Hugging Face. (2025, April 24).  
Huggingface.co. <https://huggingface.co/cardiffnlp/xlm-twitter-politics-sentiment>

The White House Historical Association. (2017). *The Presidents Timeline*. WHHA (En-US).  
<https://www.whitehousehistory.org/the-presidents-timeline>