CIS 4190/5190: Applied Machine Learning | Spring 2025

News Source Classification

Team Members: Dhruv Verma, Kruthi Muralidhara Jety | *Project: News Source*

1 Summary

In this project, we tackled the task of classifying political news headlines by source — either *Fox News* or *NBC News*. This short-text binary classification problem has broader implications in **media bias detection**, **source attribution**, and **political analysis**. Our dataset contained over **3,800 headlines**, each labeled by origin.

We began by evaluating which **normalization technique** (cleaning, stemming, or lemmatization) performed best for both **TF-IDF-based** and **RoBERTa-based** approaches. For TF-IDF, **stemming** yielded the best results, while for RoBERTa, **lemmatization** was superior.

Next, we compared **contextual embeddings from RoBERTa** against TF-IDF vectors. Across all classical models, **RoBERTa embeddings outperformed TF-IDF**, showing improved representation quality.

We then experimented with multiple **deep learning architectures** — LSTMs, Bi-GRUs, CNNs, and ANNs — including hybrid combinations. However, the best results were achieved with the standalone **Bi-LSTM** model.

Finally, we fine-tuned the **entire RoBERTa + Bi-LSTM pipeline** end-to-end. This model achieved our best performance: **82.26% accuracy** and a **macro F1-score of 0.82**, significantly improving over the **TF-IDF + Logistic Regression baseline of 70–71%**. Thus, it was chosen as our **final model for deployment**.

2 Core Components

2.1: Data Collection and Preprocessing

We worked with a dataset of **3,804 news headlines**, each labeled as either "Fox News" or "NBC News." Given the short, high-context nature of headlines, careful preprocessing was essential. Our pipeline began with **web scraping** using `BeautifulSoup` and `requests` to collect headlines from Fox and NBC websites while removing HTML tags and noise. **Cleaning** involved lowercasing text, stripping punctuation, special characters, and extra whitespace using regex. We then applied **stopword removal** using NLTK's standard list to retain meaningful tokens. For **normalization**, we created three versions of the dataset: **Cleaned** (basic cleaning), **Stemmed** (using Porter Stemmer), and **Lemmatized** (using WordNet lemmatizer). Additionally, we performed **feature engineering** by adding a word count column to capture sentence length. Finally, we performed a stratified **train-test split** (80–20) to maintain class balance. This preprocessing pipeline enabled us to systematically compare the impact of different normalization strategies across various model types.

2.2: TF-IDF + Logistic Regression Results (Baseline Model)

To establish a strong baseline, we used **TF-IDF vectorization** on each of the three normalized versions of the headlines and trained **Logistic Regression** models. These models served as our starting point for evaluating the effects of normalization and provided a benchmark for performance comparison with more advanced approaches.

Below are the results of these experiments:

Text Version	**Accuracy**	**Precision**	**Recall**	**F1 Score**
Cleaned	71.1%	0.71	0.71	0.71
Stemmed	**71.2%**	**0.71**	**0.71**	**0.71**
Lemmatized	70.7%	0.70	0.70	0.70

Table 1: Performance of Logistic Regression using TF-IDF features across different normalization techniques

Conclusion:
All three versions performed similarly, with **stemming slightly outperforming** the others across all key metrics. Lemmatization showed the lowest performance, but only marginally. These results form our **baseline**, and we use this 70–71% performance as a reference point for evaluating contextual models and deep learning architectures.

2.3: Contextual RoBERTa Embeddings

We used **RoBERTa-base**, a transformer model well-suited for short-text tasks, to generate contextual embeddings from lemmatized headlines. Headlines were tokenized with padding and truncation (max length 128), passed through RoBERTa, and we extracted the `[CLS]` token (position `[0]`) from the final hidden state — yielding a 768-dimensional vector per headline. Using `[CLS]` vectors allowed us to capture semantic meaning efficiently and consistently outperformed TF-IDF, enabling robust performance across both simple and complex architectures.

2.4 Planned Deep Learning Models

We used the CLS token output from RoBERTa (lemmatized) as input to a variety of deep learning models, aiming to improve performance over classical baselines.

The models included:

- **ANNs:** Fully connected networks with 3 and 4 hidden layers (each with 512 units), serving as non-sequential baselines.
- **LSTMs:** Recurrent models tested in two configurations — one with `hidden_dim=128, layers=2` and another more complex version with `hidden_dim=256, layers=3`.
- **Bi-GRUs:** Efficient recurrent models tested with increasing depth and capacity — from `hidden_dim=128, layers=2` to `hidden_dim=512, layers=4`, all bidirectional.
- **CNNs:** Implemented with 2 to 4 convolutional blocks. Each block includes 1D convolution (`kernel_size=3, padding=1`), ReLU activation, MaxPooling (`kernel_size=2`) These were designed to capture local sequential patterns from RoBERTa embeddings.

All models were evaluated on the same test set, and their exact configurations match those reported in Table 3. Combined models and performance comparisons are discussed in detail in the exploratory questions section.

2.5 Iterative Design Process

Our project was guided by four core exploratory questions. We first tested **which normalization method works best**: **stemming performed best with TF-IDF**, while **lemmatization worked better with RoBERTa**. Next, we confirmed that **RoBERTa outperforms TF-IDF** in all classical ML models due to its **contextual embeddings**. We then explored deep learning models like **CNNs, LSTMs, and GRUs**, finding that the best standalone model — a **Bi-LSTM** — outperformed all others, including combined architectures. Finally, we **fine-tuned the full RoBERTa–BiLSTM network**, achieving our best performance: **82.26% accuracy** and **0.82 F1 score**, a significant gain over baseline models (around 71%). These **insights iteratively guided** both our preprocessing and model design choices.

2.6 Evaluation Protocols and Results

We used `BCEWithLogitsLoss` as our loss function, which is well-suited for binary classification. Models were trained and evaluated on accuracy, precision, recall, and F1-score using test predictions.**Final Model Performance (RoBERTa–BiLSTM): Accuracy: 82.26%**, **Precision: 0.83**, **Recall: 0.82**, **F1 Score: 0.82.** This result significantly outperforms our classical TF-IDF + LR baseline (∼71%), highlighting the effectiveness of contextual embeddings, normalization, and iterative tuning.

3 Exploratory Questions

1. Which Normalization Method Works Best?

In this section, we aim to identify which **text normalization method** — cleaned, stemmed, or lemmatized — is most effective for our classification task. This experiment was conducted in two parallel settings:

- Using **TF-IDF** vectorization followed by **Logistic Regression**
- Using **RoBERTa embeddings** (CLS token) followed by **Logistic Regression**

This allowed us to evaluate how different normalization strategies interact with sparse statistical features (TF-IDF) versus contextual dense features (RoBERTa).

Evaluation Results: TF-IDF vs. RoBERTa (Logistic Regression)

Method	Normalization	Accuracy	Precision	Recall	F1 Score
TF-IDF + LR	Cleaned	71.10%	0.71	0.71	0.71
TF-IDF + LR	**Stemmed**	**71.20%**	**0.71**	**0.71**	**0.71**
TF-IDF + LR	Lemmatized	70.70%	0.70	0.70	0.70
RoBERTa + LR	Cleaned	73.33%	0.73	0.73	0.73
RoBERTa + LR	Stemmed	72.70%	0.72	0.72	0.72
RoBERTa + LR	**Lemmatized**	**74.51%**	**0.74**	**0.74**	**0.74**

Table 2: Comparison of normalization techniques using Logistic Regression with TF-IDF and RoBERTa embeddings

Conclusion: Small preprocessing choices can meaningfully impact performance. **Stemming** improved TF-IDF results by reducing vocabulary size, while **lemmatization** worked best with RoBERTa due to its alignment with grammatically structured pretraining. This emphasizes the importance of tailoring preprocessing to the chosen embedding method.

2. RoBERTa + ML Models: Is RoBERTa Better Than TF-IDF?

Building on our findings from the previous section, we identified that **lemmatized text paired with RoBERTa embeddings** produced the strongest performance across normalization techniques. Using this insight, we designed a focused experiment to evaluate whether **RoBERTa embeddings consistently outperform TF-IDF vectors**, regardless of the machine learning classifier. We tested four different ML models — Logistic Regression (LR), Support Vector Classifier (SVC), Random Forest (RFC), and XGBoost (XGB) — with both feature types: TF-IDF vectors trained on **stemmed text** (best for TF-IDF). RoBERTa CLS embeddings trained on **lemmatized text** (best for RoBERTa)

Evaluation Results: RoBERTa vs. TF-IDF Across Classifiers

Conclusion:

This experiment validated our hypothesis: **contextual embeddings from RoBERTa outperform TF-IDF across the board**. Based on these results, we selected **lemmatized RoBERTa embeddings** as the

Model	TF-IDF Accuracy	RoBERTa Accuracy	TF-IDF F1 Score	RoBERTa F1 Score
Logistic Regression	71.48%	**74.51%**	0.71	**0.74**
SVM (SVC)	70.57%	**75.69%**	0.70	**0.75**
Random Forest	68.33%	**72.40%**	0.68	**0.72**
XGBoost	69.91%	**75.30%**	0.70	**0.75**

Table 3: Performance comparison of TF-IDF and RoBERTa embeddings across ML classifiers

feature base for all subsequent deep learning models explored in the next sections.

3. Can combining Deep Learning Models improve performance?

We implemented and evaluated a total of **19 deep learning models** — both standalone and combined — using contextual embeddings from RoBERTa (lemmatized headlines). For standalone models, we designed configurations that tested a variety of architectural choices such as layer depth, hidden size, and number of convolutional blocks.

3.0.1 Individual Model Results

Below are the best variants from each category, with full configuration details and their corresponding evaluation metrics. All models were trained on the same embedding inputs and tested on the same dataset to ensure fair comparison.

Model	Architecture Details	Precision	Recall	Accuracy	F1 Score
LSTM	hidden_dim=128, layers=2	0.7364	0.7507	75.43%	0.7435
LSTM	hidden_dim=256, layers=3	0.7604	0.7609	76.08%	0.7605
Bi-LSTM	hidden_dim=256, layers=3, bidirectional	0.7710	0.7575	**76.61%**	0.7642
Bi-GRU	hidden_dim=128, layers=2, bidirectional	0.7412	0.7618	76.08%	0.7514
Bi-GRU	hidden_dim=256, layers=3, bidirectional	0.7131	0.8450	73.98%	0.7735
Bi-GRU	hidden_dim=512, layers=4, bidirectional	0.7240	0.7701	75.30%	0.7463
CNN	2 conv blocks, kernel=3, padding=1, maxpool=2	0.7874	0.6565	75.30%	0.7160
CNN	3 conv blocks, kernel=3, padding=1, maxpool=2	0.7404	0.6399	72.27%	0.6865
CNN	4 conv blocks, kernel=3, padding=1, maxpool=2	0.7131	0.8450	74.90%	0.7735
ANN	3 hidden layers (512 units)	0.7778	0.7350	75.03%	0.7558
ANN	4 hidden layers (512 units)	0.6183	0.7673	66.49%	0.6848

Table 4: Performance of individual deep learning models using lemmatized RoBERTa embeddings

3.0.2 Combined Model Results

To test whether hybrid models outperform standalone ones, we combined the best-performing individual models in three ways — sequentially, in parallel, and interleaved. All models used the same RoBERTa + lemmatized embeddings.

Conclusion

- **Bi-LSTM** emerged as the best-performing model (Accuracy 76.61%), consistently outperforming both standalone and hybrid configurations. While combinations like Bi-GRU + Bi-LSTM showed potential in some parallel setups, none surpassed Bi-LSTM alone. These results suggest that **well-tuned single architectures are more effective** than complex hybrids, which often introduce instability or marginal performance gains.
- *Limitation:* We did not explore the full architectural space for each deep learning model.

Model Combination	Architecture Type	Accuracy	Precision	Recall	F1 Score
Bi-GRU + Bi-LSTM	Parallel	76.22%	0.7505	0.7512	0.7502
Bi-GRU + Bi-LSTM	Sequential	75.16%	0.7240	0.7701	0.7463
CNN + Bi-GRU	Parallel	75.03%	0.7232	0.7673	0.7446
CNN + Bi-GRU	Sequential	52.56%	0.7105	0.7975	0.7515
CNN + Bi-GRU	Interleaved	72.27%	0.7404	0.6399	0.6865
CNN + Bi-LSTM	Parallel	69.65%	0.7316	0.5725	0.6424
CNN + Bi-LSTM	Sequential	52.56%	0.7105	0.7975	0.7515
CNN + Bi-LSTM	Interleaved	66.49%	0.6183	0.7673	0.6848

Table 5: Performance of combined hybrid deep learning models

- *Future Work:* We plan to run a more extensive grid search over hyperparameters and layer combinations to further optimize performance.

4. Is Training the Full RoBERTa–BiLSTM Network Beneficial?

Yes. We fine-tuned **all layers** of the RoBERTa model in conjunction with our best performing Bi-LSTM classifier — allowing gradient updates to affect the transformer encoder as well.

Final Architecture Details:

RoBERTa-base tokenizer (max_length=128) and embeddings (768-dim) → **Bi-LSTM** (hidden_dim=256, layers=3, bidirectional=True) → **Classifier Head:** Linear(512→128) + ReLU, Linear(128→64) + ReLU, Linear(64→1) + Sigmoid → Output, **Loss:** BCEWithLogitsLoss, **Optimizer:** AdamW (lr=2e-5), **Epochs:** 10, **Batch size:** 16

Performance Summary:

Accuracy: 82.26%, **Precision:** 0.83, **Recall:** 0.82, **F1 Score:** 0.82

Conclusion:

- *Limitation:* Our experiments were limited to RoBERTa as the transformer backbone.
  Future Work: Future efforts can explore alternative architectures like BERT or DistilBERT to compare generalization and efficiency tradeoffs.

- Training the full network end-to-end — from RoBERTa embeddings through the Bi-LSTM sequence model and classifier head — produced our best model. It achieved an **accuracy of 82.26%**, with **precision of 0.83**, **recall of 0.82**, and an **F1-score of 0.82**, showing a **significant improvement over TF-IDF baselines** (average 70–71%).

- The model balances precision and recall well, generalizes effectively across both classes, and is **our chosen architecture for final deployment and submission**.

4 Team Contributions

- **Dhruv Verma:** Web Scraping, Computed RoBERTa embeddings, model design for exploratory questions 3 and 4, PPT

- **Kruthi Muralidhara Jety:** Data Cleaning, Normalization, model design for exploratory questions 1 and 2, report writing

5 Appendix

5.1 Code Files Summary:

- **1_Dhruv_Kruthi_AML_Web_Scraping_Final.ipynb** - Performed Web Scraping
- **2_New_Dhruv_Kruthi_Normalization.ipynb** - Cleaned and Normalized text to Stemmed, Lemmatized, and Cleaned
- **3_Dhruv_Kruthi_RoBERTa_Stem.ipynb** - Computed RoBERTa embeddings for Stemmed Text
- **4_Dhruv_Kruthi_RoBERTa_Lemma.ipynb** - Computed RoBERTa embeddings for Lemmatized Text
- **5_Dhruv_Kruthi_RoBERTa_Clean.ipynb** - Computed RoBERTa embeddings for Cleaned Text
- **6_Dhruv_Kruthi_explore1.ipynb** - Answered Exploratory Question 1
- **7_Dhruv_Kruthi_explore2.ipynb** - Answered Exploratory Question 2
- **8_Dhruv_Kruthi_Final_RoBERTa_lemma.ipynb** - Answered Exploratory Question 3
- **9_Dhruv_Kruthi_RoBERTa_lemma_tuning_e10.ipynb** - Answered Exploratory Question 4
- **10_Dhruv_Kruthi_Hidden_Test.ipynb** - Final Predictions on Hidden Test Data

5.2 Acknowledgement

We want to thank TA Joey and TA Haorui for their constant guidance and support throughout the project.