CIS 4190/5190: Applied Machine Learning | Spring 2025

Headline Classifier Model

Team Members: Wesley Dalton, Allen Liu, Ryan Xia | *Project: News Source*

1 Summary

This project develops a news headline classifier to distinguish between Fox News and NBC sources, exploring various machine learning approaches from traditional to deep learning models. We collected **3,735 headlines** by scraping the provided URLs and applied preprocessing including lemmatization and stopword removal. Our baseline **TF-IDF + Logistic Regression** model achieved **68.94% accuracy**, providing strong interpretability through feature weights that revealed outlet-specific vocabulary patterns (e.g., "cnn" and "history" for Fox News vs. "gaza" and "netanyahu" for NBC).

We then implemented a **Deep Averaging Network (DAN)** that learns dense embeddings and achieved **76% accuracy** with automatic embeddings, demonstrating a **7% macro F1 improvement** over the baseline. An alternative DAN configuration using learning rate scheduling performed similarly with an accuracy of 76%. Our best model, **XGBoost with ensemble features** (word TF-IDF, character n-grams, handcrafted features), achieved **78.31% accuracy**. Surprisingly, character n-grams dominated feature importance (93.07%), significantly outweighing word features (6.05%), indicating that news outlets have distinct textual "fingerprints" at the character level rather than purely semantic differences.

Through ablation studies, *character n-grams alone* reach 73.63% accuracy—nearly matching the full ensemble—contradicting our expectation that semantic features would dominate. This suggests formatting conventions and stylistic patterns are more informative than topic or vocabulary choice. Overall, even brief headlines carry subtle stylistic cues that models can exploit to distinguish news sources.

2 Core Components

Data Collection:

- Data was collected by scraping the provided URLs for news headlines. We used Python's `requests` library to fetch each page and Beautiful Soup to parse the HTML. The data was cleaned by converting all headlines to lowercase, removing special characters and numbers, stripping extra whitespace, removing stopwords using NLTK's English stopword list, and lemmatizing via spaCy. We then split the cleaned dataset randomly into an 80% training set and a 20% validation set.

Model Design:

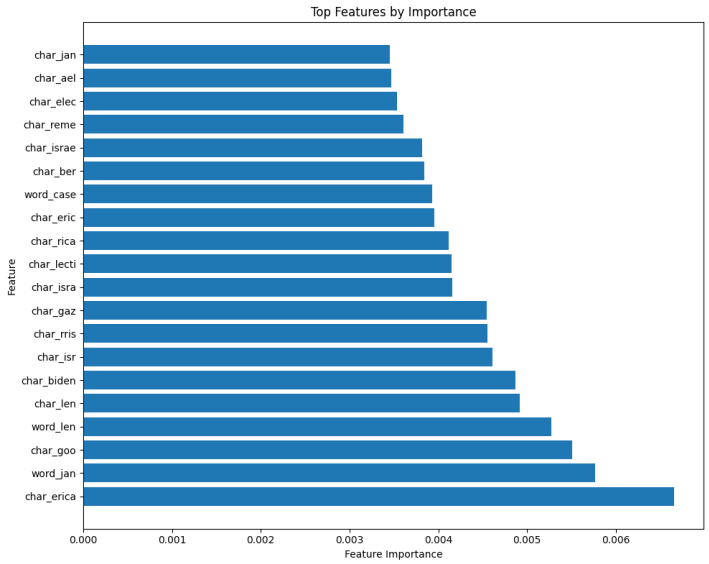
- **TF–IDF + Logistic Regression (Baseline).** We vectorized headlines using TF–IDF (English stopwords removed, max features = 100), then trained a logistic regression classifier with L_2 regularization ($C = 1.0$) for up to 100 iterations.
- **Deep Averaging Network (DAN).** We implemented a DAN that (1) embeds each token (embedding dim = 300), (2) averages all token embeddings to form a sentence vector, and (3) passes this through two dense layers (256, 128 units) with dropout = 0.3. We trained with Adam (learning rate = 1e–3) and binary cross-entropy loss, and experimented with learning rate scheduling, the dropout rate and other optimisers.
- **XGBoost with TF–IDF, character-grams, and handcrafted features.** We vectorized headlines using both word n-grams (1–2, max_features=10 000, min_df=3) and character n-grams (3–5, max_features=10 000, min_df=3). We concatenated the sparse TF–IDF matrices with the dense features and trained an XGBClassifier with 300 trees, max_depth=4, learning_rate=0.08, subsample=0.8, colsample_bytree=0.8, and log-loss eval metric.

Evaluation and Model Performance:

- For each model, we trained on the 80% split and computed accuracy, precision, recall, and F_1 on the 20% hold-out set.
- **Baseline (TF–IDF + LR):** Accuracy = 0.6894, Macro F_1 = 0.6800.
- **DAN (automatic embeddings):** Accuracy = 0.7600, Macro $F_1 \approx 0.7550$.
- **XGBoost (TF–IDF + n-grams):** Accuracy = 0.7831, Macro $F_1 \approx 0.7831$.
- We also inspected the features (LR coefficients, XGBoost feature importances) to ensure each model was capturing meaningful signal rather than noise.

FoxNews		NBC	
cnn	2.4615	jan	-3.5409
history	2.0659	accord	-2.8686
reveal	1.9730	good	-2.4534
school	1.8369	gaza	-2.3424
fox	1.8236	netanyahu	-2.2275
illegal	1.7377	capitol	-1.9826
family	1.5491	hostage	-1.8300
time	1.4907	israel	-1.6483
american	1.3661	israeli	-1.4520
warn	1.3287	iran	-1.4153

Table 1: Top indicative words by TF–IDF weight for FoxNews vs. NBC



Take-away: Even with minimal text length, the baseline TF–IDF + logistic regression reaches nearly 69% accuracy, while richer representations (DAN, XGBoost) achieve 76–79% accuracy. This shows the value of learned embeddings and non-linear feature interactions in modeling differences between news sources.

3 Exploratory Questions

Question 1: Does the Deep Averaging Network's use of learned embeddings improve the macro F_1-score and per-class F_1 compared to the TF–IDF + Logistic Regression?

Motivation: The TF–IDF + Logistic Regression baseline uses sparse bag-of-words features capped at 100 tokens, which may limit its ability to capture semantic relationships between words. By contrast, the DAN learns dense embeddings for every token and aggregates them through hidden layers, potentially enabling richer feature representations. Given the provided metrics, we want to assess whether the embedding-based DAN outperforms the linear TF–IDF pipeline.

Prior Work or Course Material: Course lectures on TF–IDF discuss how sparse representations can struggle when vocabulary is large and features are limited. Deep Averaging Networks (Iyyer et al., 2015) and widely used embedding techniques demonstrate that dense vector representations can capture distributional semantics and boost classification performance. These insights suggest that a DAN should surpass a linear logistic model on text classification tasks.

Methods and Baseline:

- *Baseline:* TF–IDF vectorizer followed by Logistic Regression ($c = 1.0$, L_2, max_iter=100).
- *DAN:* Embedding dimension = 300; two hidden layers (256, 128); dropout = 0.3; trained with Adam optimizer and BCELoss.
- *Evaluation:* Precision, recall, and F_1 computed on the existing 20% test split for the baseline and 10% validation split for the DAN.

Model	Class	Precision	Recall	F1-score
TF–IDF with Logistic Regression	NBC (label = 0)	0.70	0.55	0.62
	Fox News (label = 1)	0.68	0.80	0.73
	Macro average	—	—	0.67
Deep Averaging Network (epoch 4)	NBC (label = 0)	0.71	0.79	0.75
	Fox News (label = 1)	0.78	0.69	0.73
	Macro average	—	—	0.74

Table 2: Precision, recall, and F1-score for each class and macro-average by model.

Interpretation: The DAN shows an absolute improvement of +0.07 in macro F_1 over the baseline, with gains in both classes and particularly improved F_1 for the NBC class. This suggests that dense embeddings and the deeper network structure enable better generalization than sparse TF–IDF features alone.

Limitations:

- The baseline and DAN were evaluated on slightly different splits (20% test vs. 10% validation), which may introduce sampling variance.
- No further hyperparameter tuning was performed; both models used default settings aside from those specified.
- Vocabulary in the DAN is limited to tokens seen in training; truly unseen words are mapped to `<UNK>`.

Question 2: Does pre-trained word embedding improve DAN performance over randomly initialized embeddings?

Motivation: This question investigates whether using pre-trained word embeddings (GloVe) in a Deep Averaging Network improves the binary classification of news sources compared to randomly initialized embeddings. The motivation stems from the hypothesis that pre-trained embeddings, trained on vast amounts of data, encode rich semantic information that can enhance downstream performance, especially when the training dataset is limited or noisy.

Prior Work or Course Material: Research shows that initializing models with pre-trained embeddings often boosts performance in text classification tasks (Mikolov et al., 2013; Pennington et al., 2014). In the DAN paper (Iyyer et al., 2015), pre-trained embeddings were shown to be highly effective when combined with a shallow architecture. My expectation was that using GloVe would lead to higher accuracy and

F1-score compared to training the embeddings from scratch.

Methods for investigation and Baseline: We implemented two variants of DAN:

Baseline: DAN with randomly initialized word embeddings (learned during training).

Variant: DAN with fixed 300-dimensional GloVe embeddings

We kept all other hyperparameters (optimizer, learning rate, etc.) constant to isolate the effect of pre-trained embeddings.

Results and updated beliefs:

Baseline

	Precision	Recall	F1-Score	Support
NBC	0.78	0.66	0.72	181
Fox	0.72	0.83	0.77	193
Accuracy			0.75	374
Macro avg	0.75	0.75	0.75	374
Weighted avg	0.75	0.75	0.75	374

With GloVe embeddings

	precision	recall	f1-score	support
NBC	0.76	0.78	0.77	181
Fox	0.79	0.77	0.78	193
accuracy			0.78	374
macro avg	0.78	0.78	0.78	374
weighted avg	0.78	0.78	0.78	374

Pre-trained embeddings performed better and more consistently across most metrics compared to randomly initialised embeddings, most notably in recall for NBC. This confirms our prior expectation that semantic richness from pre-trained vectors helps the classifier generalize better.

Limitations: One limitation was fixing the GloVe embeddings rather than fine-tuning them, which might have further improved performance. Additionally, embeddings trained on general text might miss domain-specific nuance in political news. Given more time, I would experiment with domain-adapted embeddings or contextualized models like BERT to compare their effectiveness with static embeddings in DAN.

Question 3: How do different feature types contribute to the XGBoost ensemble's performance, and what interactions between features does the model learn?

Motivation:

Our XGBoost model achieved accuracy of 78.3% by combining three types of characteristics: word n grams (TF-IDF 1–2 grams), character n grams (3–5 grams) and hand-crafted metrics (headline length, word count) which outperformed both the simple baseline (68.5%) and our DAN (76%). By examining feature importance we can understand which features drive performance, and more importantly, how they interact to reveal why this ensemble outperforms simpler models. Furthermore, investigating feature importance could reveal if certain feature types specialize in capturing different aspects of the Fox News vs. NBC distinction.

Prior Work or Course Material: Ensemble diversity is known to improve generalization by combining varied perspectives through features, algorithms, or data, as covered in Lecture 22. Unlike Random Forests, which randomize features per split, XGBoost sequentially fits trees to residuals $f_{t+1}(x_i) \approx y_i - F_t(x_i)$, allowing different feature subsets to specialize at each stage. We anticipate word n-grams will dominate overall importance by capturing semantic content, with character n-grams and handcrafted features filling

complementary roles in later iterations.

Methods and Baseline:

- *Baseline:* Full XGBoost model combining word n-grams (1-2, max_features=10,000), character n-grams (3-5, max_features=10,000), and handcrafted features (headline length, word count).
- *Ablation Models:* Three separate XGBoost models, each using only one feature type: (1) word n-grams only (2,498 features), (2) character n-grams only (10,000 features), and (3) handcrafted features only (2 features).
- *Configuration:* All models used identical hyperparameters (max_depth=4, learning_rate=0.08, subsample=0.8, colsample_bytree=0.8), with 400 trees for full/n-gram models and 100 trees for handcrafted features.
- *Evaluation:* Accuracy, precision, recall, and F_1 computed on the 20% test split. Feature importance analyzed using XGBoost's built-in importance scores, aggregated by feature type.

Model	Features	Accuracy	Precision	Recall	F1-score
Full Model	12,500	0.7416	0.7413	0.7414	0.7414
Word N-grams	2,498	0.7282	0.7298	0.7264	0.7265
Character N-grams	10,000	0.7363	0.7360	0.7358	0.7359
Handcrafted Features	2	0.6734	0.6755	0.6748	0.6732

Table 3: Performance comparison of XGBoost models using different feature sets

Results and updated beliefs: Our ablation study shows the full XGBoost ensemble achieves 74.16% accuracy and a 0.7414 F_1 score—only marginally better than character n-grams alone (73.63% accuracy, 0.7359 F_1). Word n-grams score 72.82% (0.7265 F_1), and handcrafted features lag at 67.34%. Character n-grams dominate feature importance (93.07% vs. 6.05% for word n-grams and 0.88% for handcrafted), with top features like "jan", "rris", and "meri" hinting at outlet-specific text patterns. The tiny performance boost from combining feature types suggests character-level signals are the primary driver of Fox News vs. NBC classification, challenging our expectation that semantic content (word n-grams) would be most important.

Limitations:

- Feature interactions were not explicitly measured; XGBoost may learn complex interactions that our ablation study cannot capture.
- The dataset may have temporal biases (e.g., certain topics or names appearing during collection) that inflate character n-gram importance.
- We used only XGBoost's default feature importance measure (gain); alternative metrics like SHAP values might reveal different patterns.
- The small dataset size (3,735 headlines) may limit the model's ability to learn robust semantic patterns from word n-grams.