New York Times Front Page Article Detection

Shruti Gupta, Alec Naidoo, Sarah Julius,
Philip Monaco

Motivation

- Front page articles both shape and reflect public opinion
- Which articles make it to the NYT front page?
  - Can we use headlines and other metadata to distinguish front page articles from regular page articles in the New York Times?
  - What features are most important in determining if an article will make it to the front page?

Data Overview

- Source: NYT API
- January 2014 to June 2024 = 635,736 total entries
  - Front page articles: 21,917
  - Non-front page articles: 392,846
- Main features:
  - Keywords
  - Word count
  - Headline
  - Date-related features
  - Byline
  - Snippet
  - Lead paragraph

Data Overview – General Statistics

- Distribution of Nulls
    - Most variables had similar percentage of nulls (<1%)
    - Variables that had larger differences included:
        - Snippet
        - News Desk
        - Print Section/Print Subsection
- Average Word Count for Front Page Articles: 1602 words*
- Average Word Count for Rest of Paper: 839 words*

*$p<0.005$

Approach

- Multiple different ML approaches
  - Baseline Model - offered high accuracy
  - Cross-Validation
  - Logistic Regression
  - Recurrent Neural Networks
  - XGBoost
- Metrics of success: F1-score, precision, recall
- Improvements over baseline seen in all models

Feature Engineering

Data Structure Conversion - JSON parsing

Filtering - NaN, stop-word, and special character removal.

Special Handling:

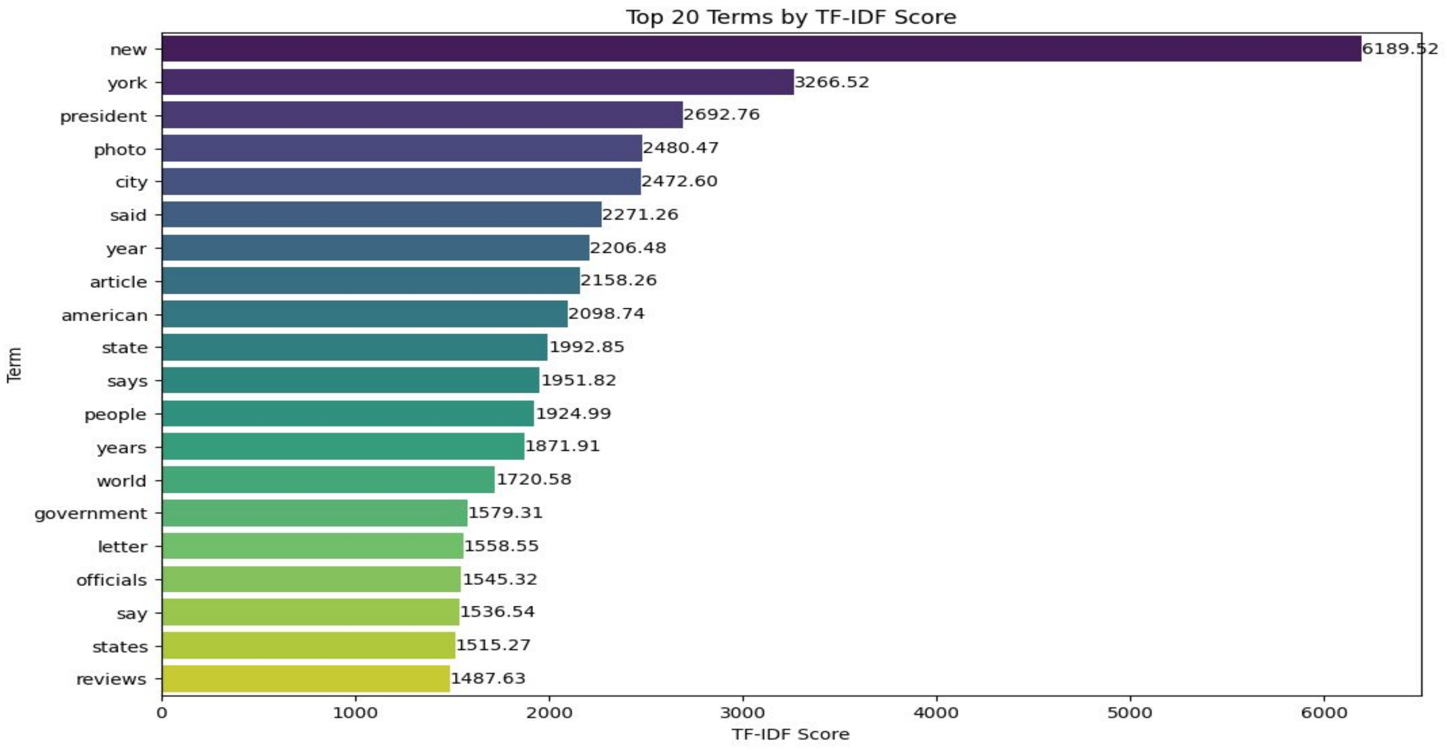
- Categorical - One-Hot Encoding
- Numeric - Non-numerics coerced to NaN
- Text - TF-IDF Vectorization, Count Vectorization (BoW), Ebeddings

XGBoost

Motivation:

- Efficiency with sparse datasets
- High interpretability for feature engineering
- Good for imbalanced datasets

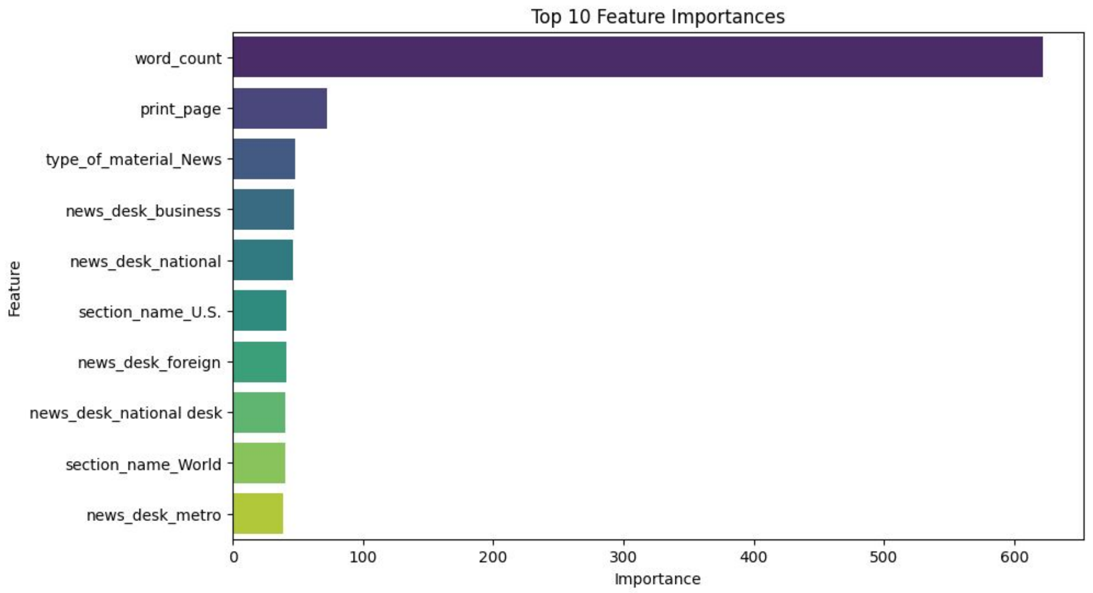
XGBoost – TF-IDF



XGBoost – Feature Importance

With numeric and categorical features:

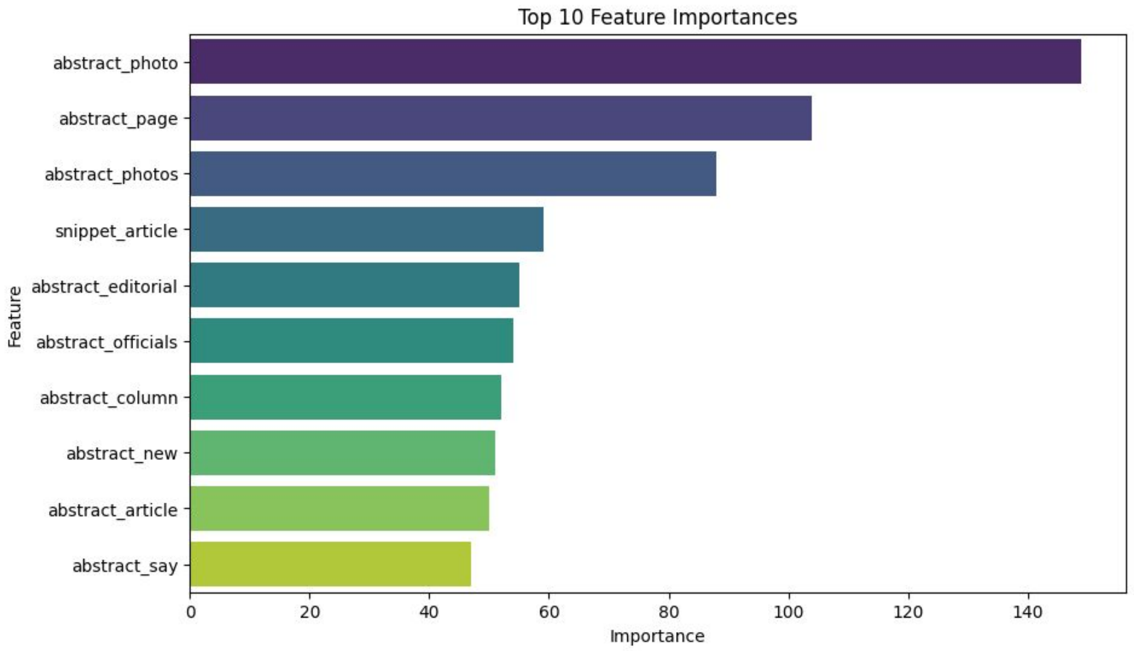
- High accuracy
- Too much reliance on word_count
- Categorical features might be problematic



XGBoost – Feature Importance

With article text only:

- Top words could be placeholders
- Word frequency or embeddings?



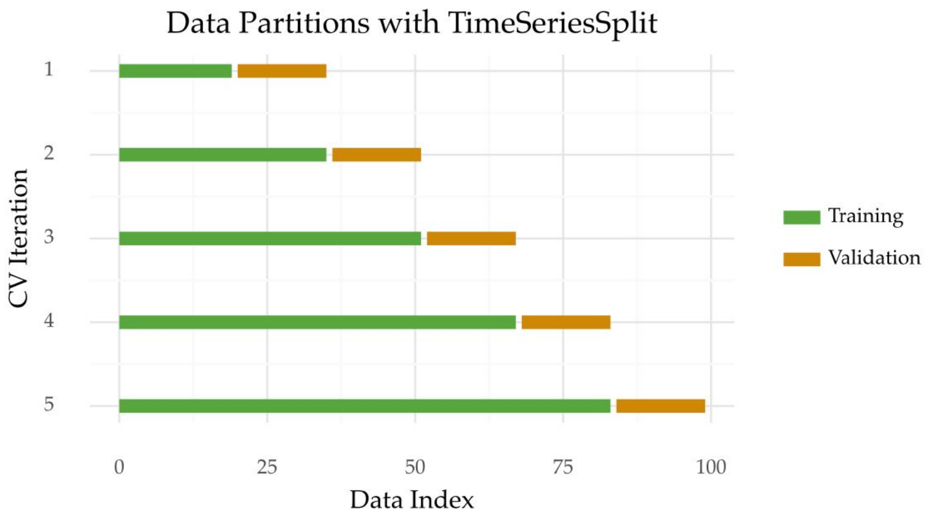
Cross-Validation Approach

Time Series Cross-Validation

- Implemented TimeSeriesSplit using Scikit-learn for more accurate temporal predictions.
  - Training: 2014 - 2023 (incrementally increased training size to reflect real-world scenarios).
  - Test: 2024 (held-out test set for final model evaluation).
- **F1-Score**, Precision, and Recall

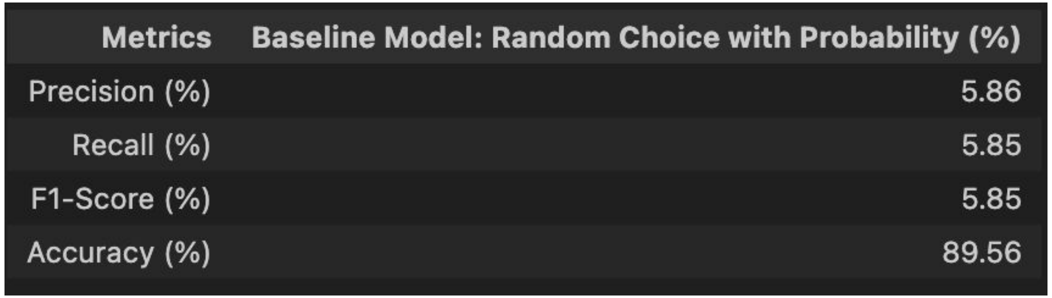
Explored BlockedTimeSeriesSplit

- Initially attempted BlockedTimeSeriesSplit but encountered insufficient data in each fold.
- Adjusted approach to ensure more reliable cross-validation results by using TimeSeriesSplit.



Baseline Model

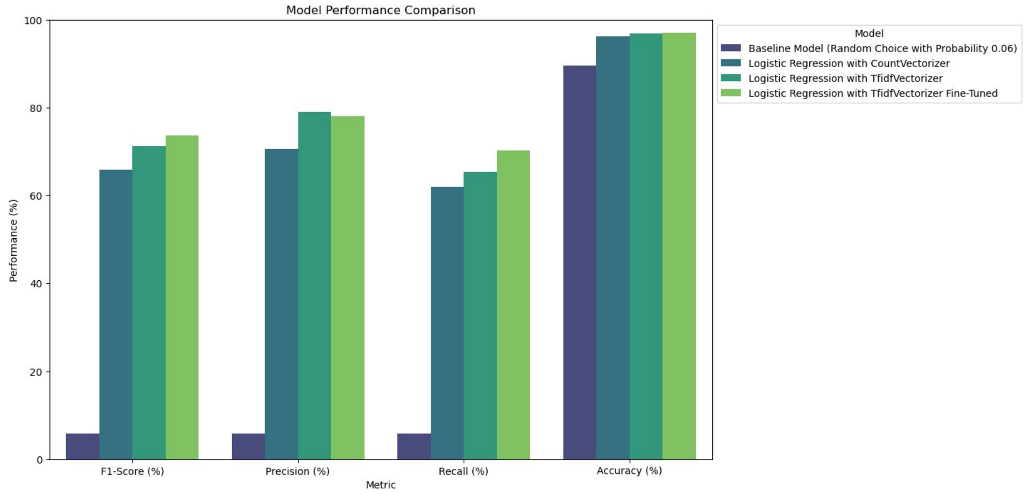
- Focus on F1-Score
- Baseline Model predicts class “1” with a probability that reflects class distribution in the dataset (6%).
- Low F1-Score of 5.85% → ineffectiveness of random guessing model

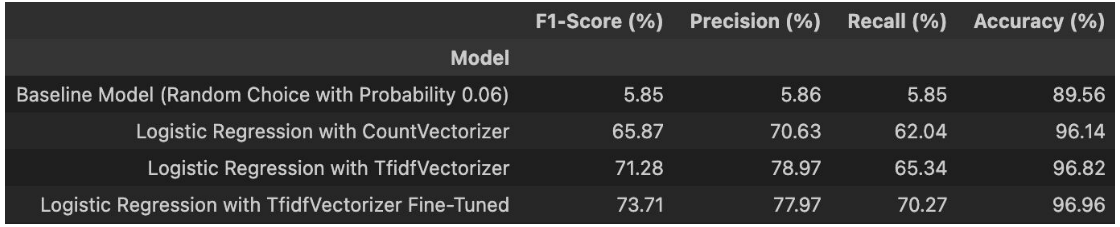


Metrics	Baseline Model: Random Choice with Probability (%)
Precision (%)	5.86
Recall (%)	5.85
F1-Score (%)	5.85
Accuracy (%)	89.56

Model 1 – Logistic Regression

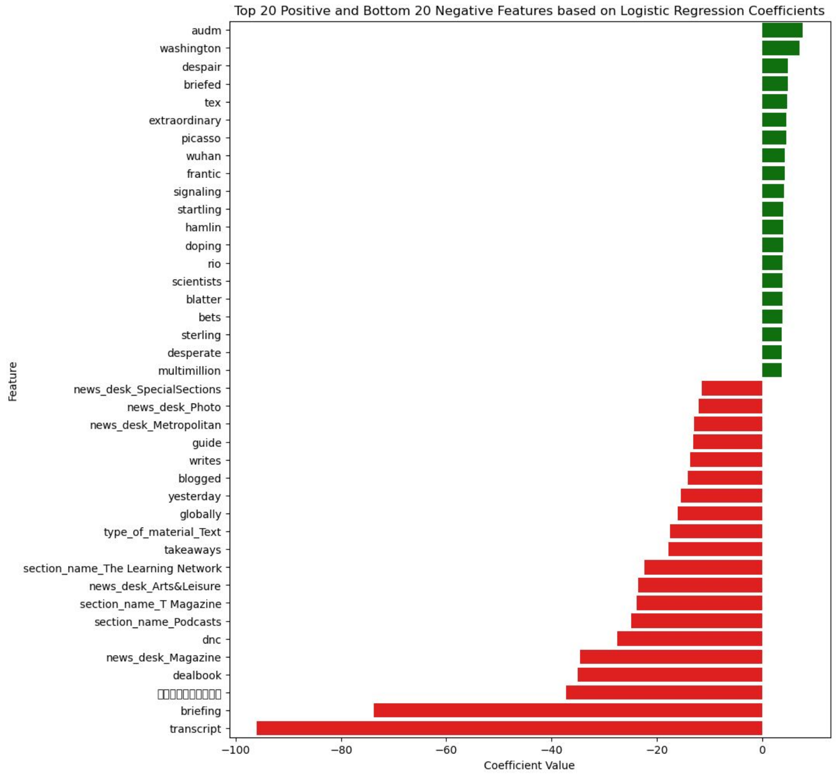
- Well-understood and interpretable model
- Experiments included:
  - Vectorization Techniques for Text Features
    - Count Vectorizer
    - TF-IDF Vectorizer
  - Including Features (The more the better)
- Grid Search Hyperparameter Tuning (5hr Build Time)
  - C=1
  - class_weight=None
  - max_iter=25
  - penalty='l1'
  - solver='liblinear'
  - tol=0.001
- Results are based on Validation Sets →



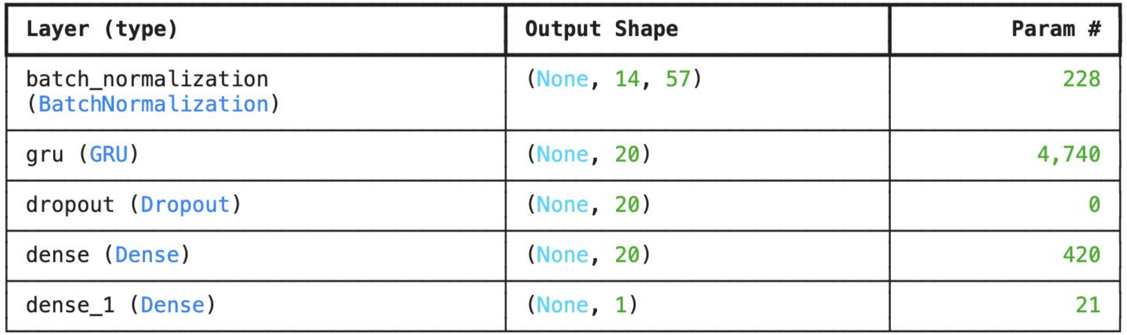


Model	F1-Score (%)	Precision (%)	Recall (%)	Accuracy (%)
Baseline Model (Random Choice with Probability 0.06)	5.85	5.86	5.85	89.56
Logistic Regression with CountVectorizer	65.87	70.63	62.04	96.14
Logistic Regression with TfidfVectorizer	71.28	78.97	65.34	96.82
Logistic Regression with TfidfVectorizer Fine-Tuned	73.71	77.97	70.27	96.96

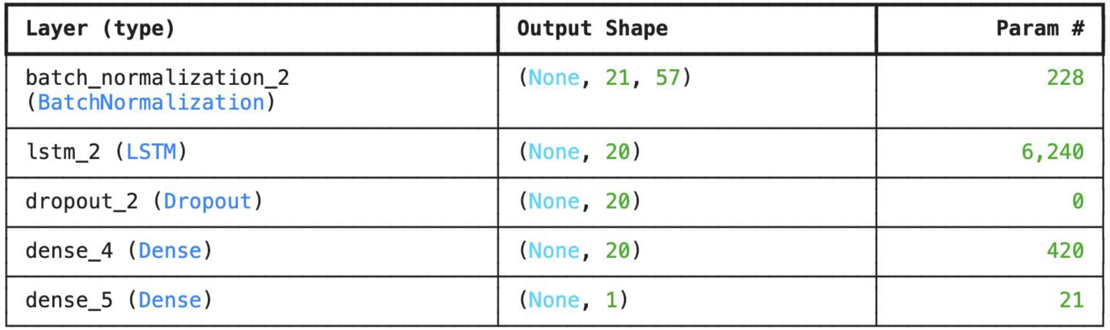
Logistic Regression Coefficients



Model 2 – RNN (GRU, LSTM)



Layer (type)	Output Shape	Param #
batch_normalization (BatchNormalization)	(None, 14, 57)	228
gru (GRU)	(None, 20)	4,740
dropout (Dropout)	(None, 20)	0
dense (Dense)	(None, 20)	420
dense_1 (Dense)	(None, 1)	21

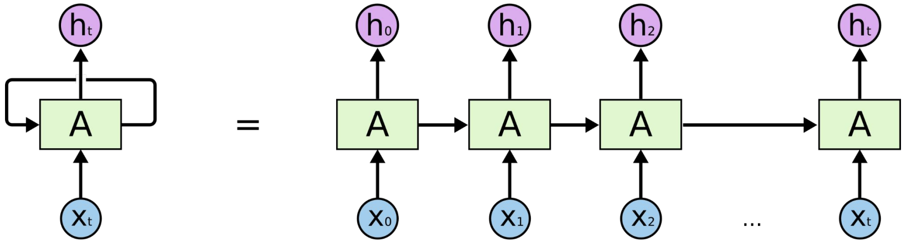


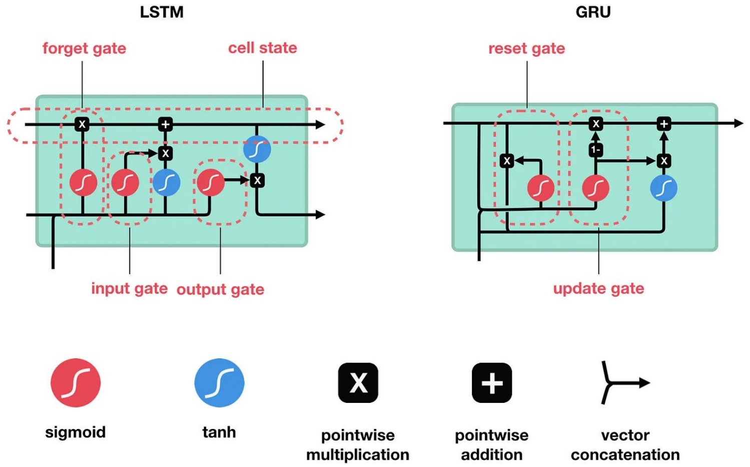
Layer (type)	Output Shape	Param #
batch_normalization_2 (BatchNormalization)	(None, 21, 57)	228
lstm_2 (LSTM)	(None, 20)	6,240
dropout_2 (Dropout)	(None, 20)	0
dense_4 (Dense)	(None, 20)	420
dense_5 (Dense)	(None, 1)	21

- Article representations through mean of document vectors
- One-hot encoded: section_name, news_desk, type_of_material
- Numeric: word_count, num_subjects, num_persons, num_glocs

RNNs

- Simple RNN: Hidden layer includes recurrent connection as part of input, introducing memory
- LSTM: introduce “gates” to decide what information to keep, add, and forget, so error
- GRU: Keep gates idea, but get rid of memory cell → expose hidden state



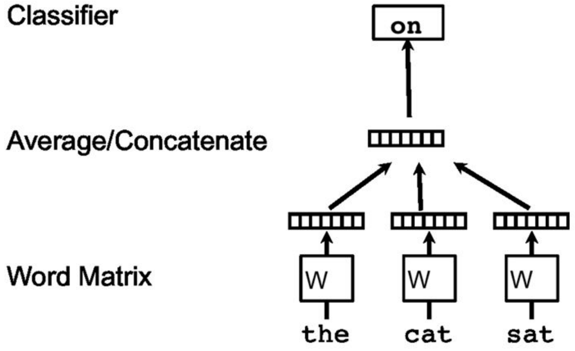


Document Vectors

- Word embeddings: capture similarities between words; paragraph embeddings: capture similarities between documents

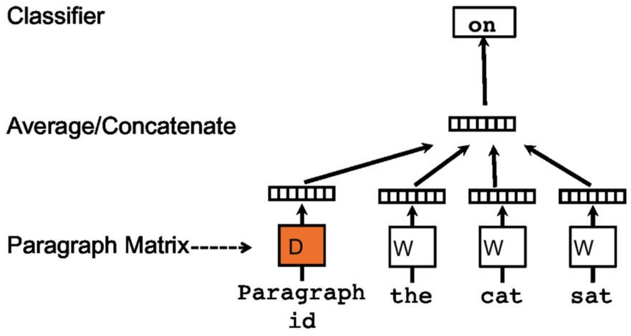
- ”A memory remembering what’s missing from the current context– or the topic of the paragraph”

- Both paragraph and word embeddings are trained; word vector matrix shared across all documents, but paragraph embeddings unique to each document

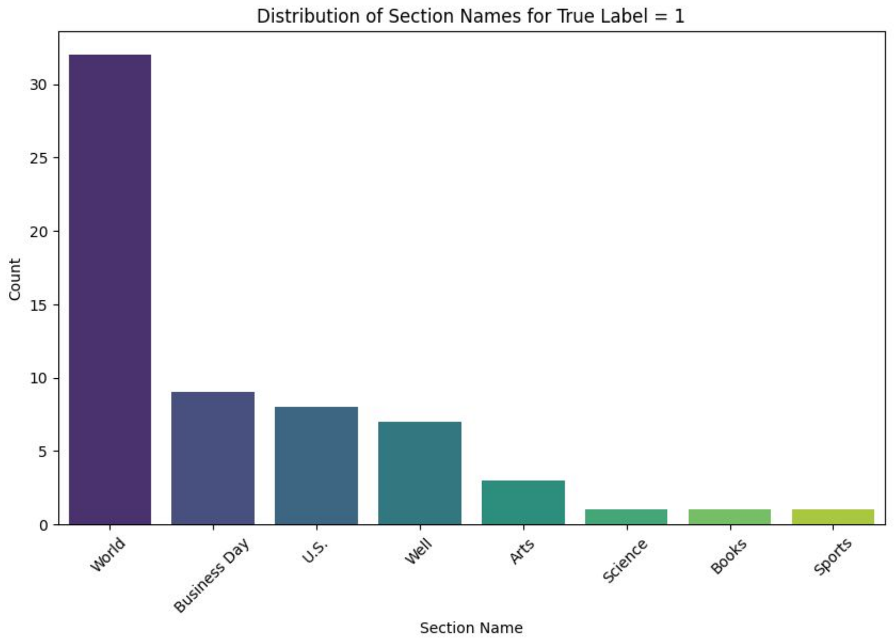


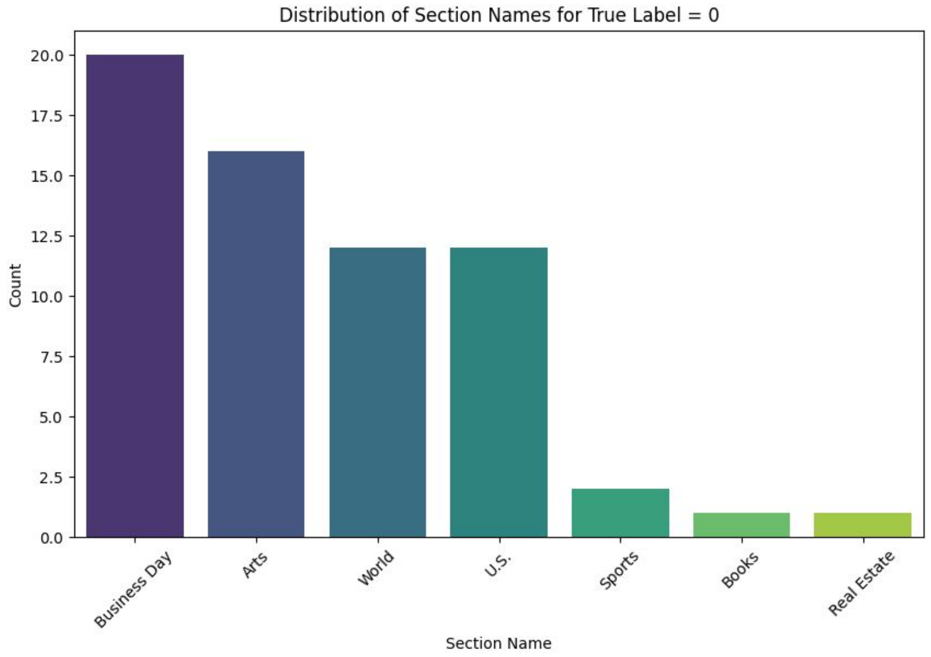
$$p(w_t|w_{t-k}, ..., w_{t+k}) = \frac{e^{y_{w_t}}}{\sum_i e^{y_i}}$$

$$y = b + Uh(w_{t-k}, ..., w_{t+k}; W)$$

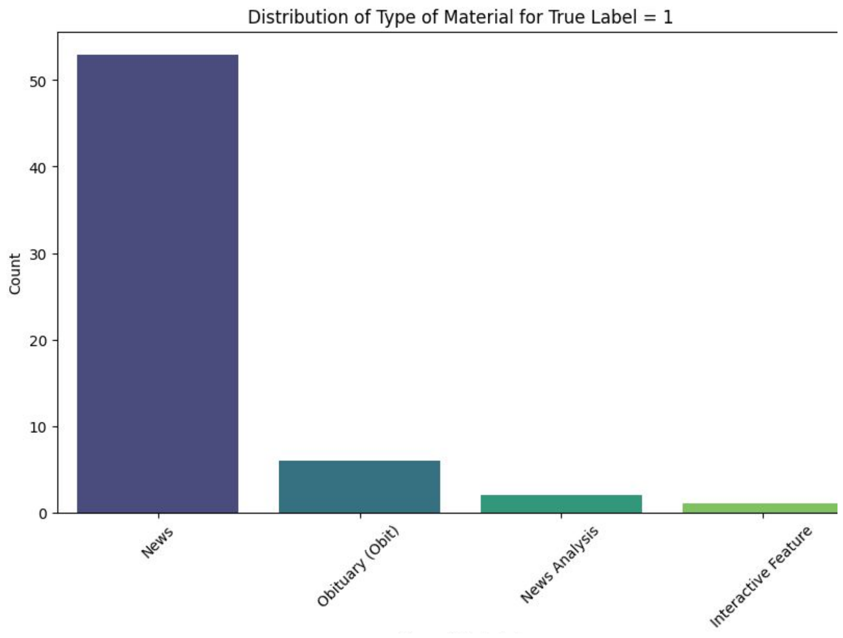


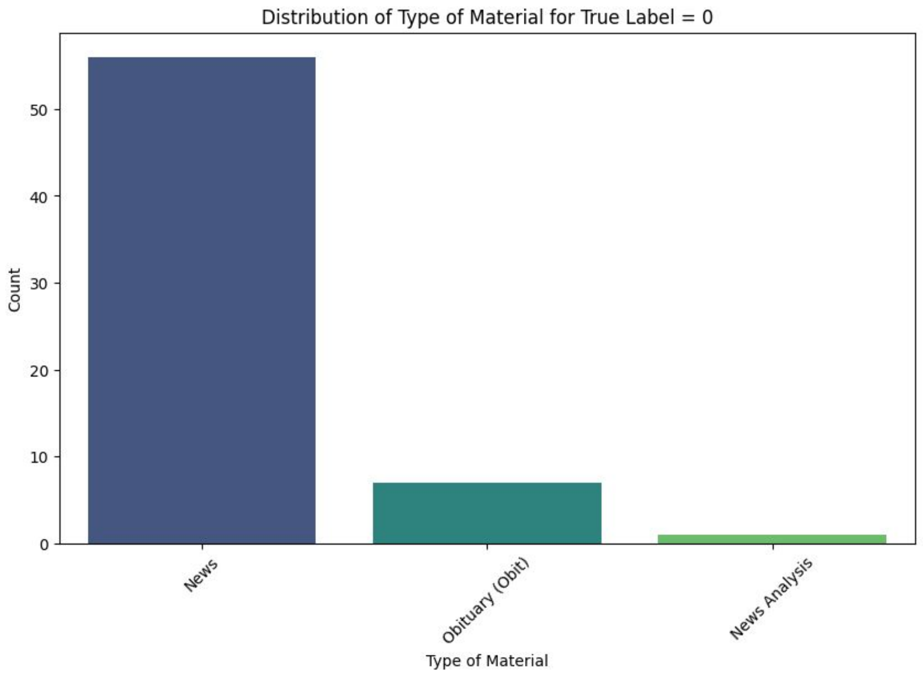
Exploring Misclassifications: Section Name



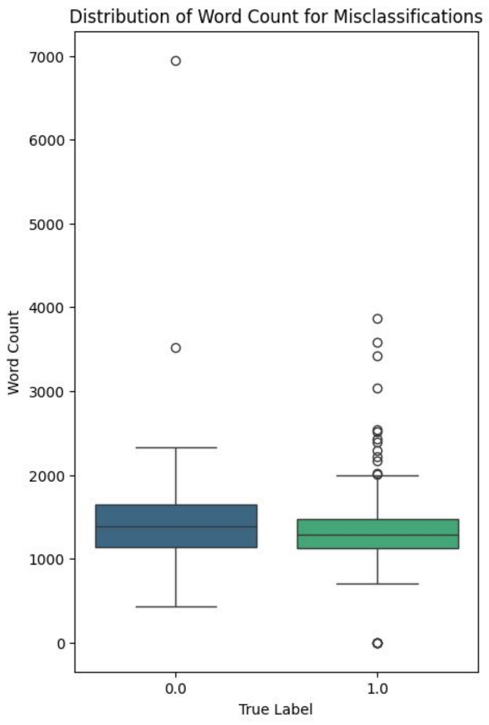


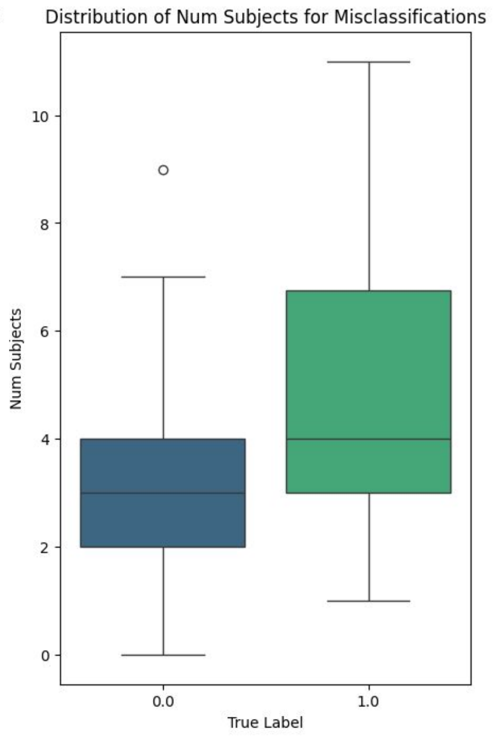
Exploring Misclassifications: Material Type

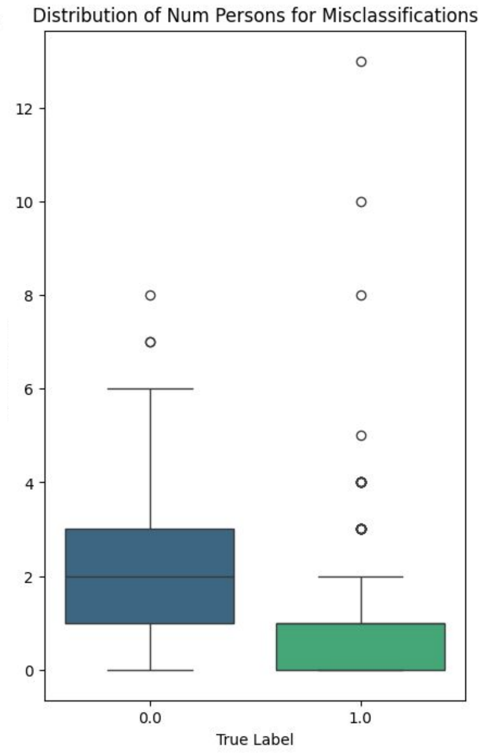


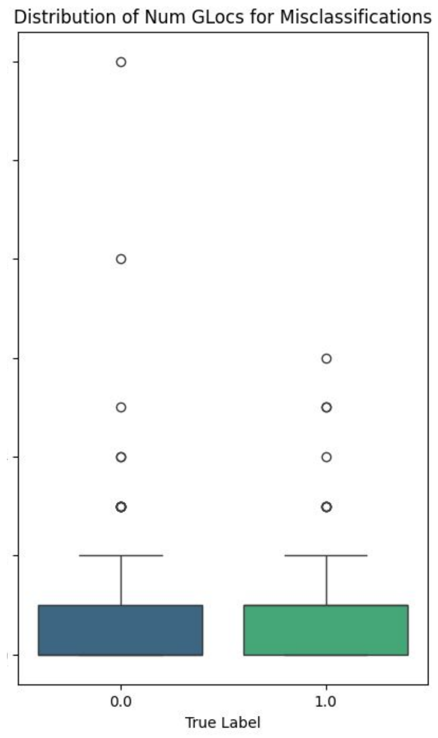


Numerical Features in Misclassification

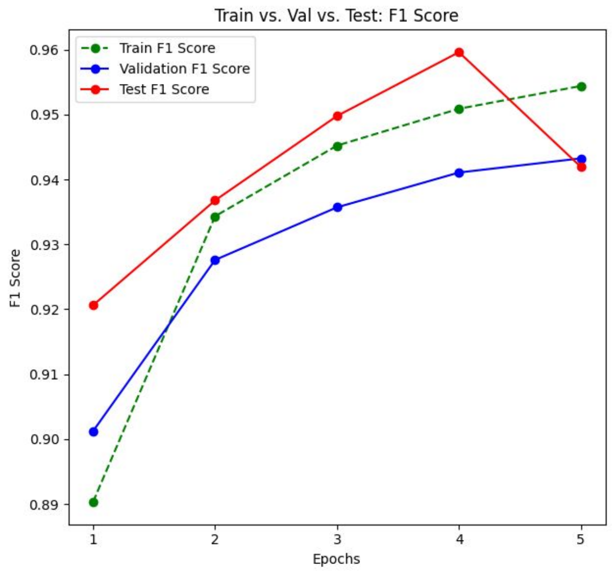


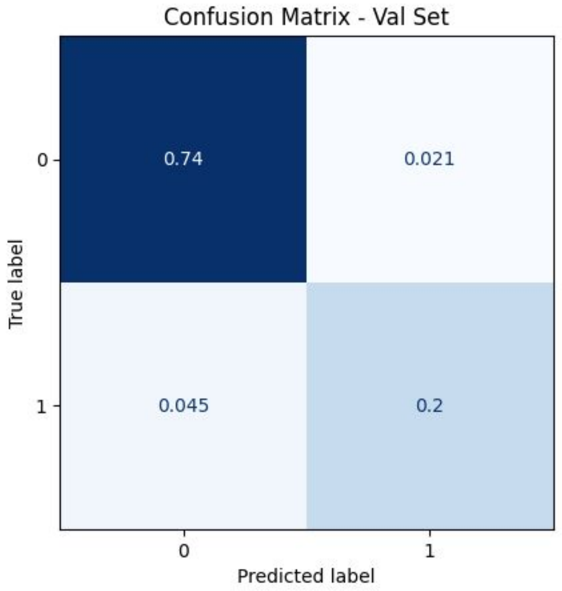


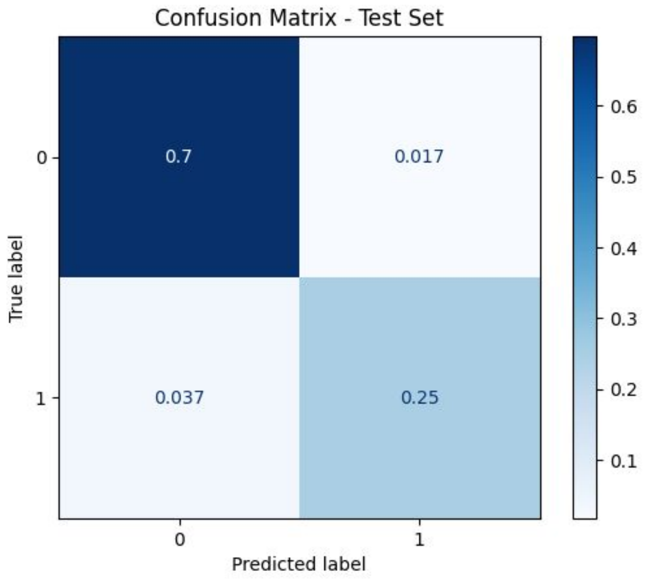




Results (GRU)


Train vs. Val vs. Test: F1 Score
Train F1 Score
Validation F1 Score
Test F1 Score
F1 Score
Epochs
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
1
2
3
4
5


Confusion Matrix - Val Set
True label
Predicted label
0
1
0.74
0.021
0.045
0.2


Confusion Matrix - Test Set
True label
Predicted label
0
1
0.7
0.017
0.037
0.25
0.6
0.5
0.4
0.3
0.2
0.1

XGBoost

Test/Train	Precision	Recall	F1-Score	Accuracy
All Features	81.7% / 81.8%	85.68% / 86.3%	83.6% / 84.0%	90.1% / 90.3%
Only Text With Embedding	68.46% / 78.3%	62.15% / 72.4%	65.1% / 75.2%	80.4% / 85.9%
Just text TF-IDF	76.5% / 79.0%	30.8% / 31.1%	43.9% / 44.7%	76.9% / 77.2%

Conclusions

- Key results
  - It is possible to predict NYT front page presence based off of key features in the dataset, but there is some over-reliance on word count
  - American politics, cities, governments more “important”
  - Overfitting: 1) RNNs (~10%), 2) Logistic Reg (~7%), 3) XGBoost (<5%)
- Avenues for future work
  - Understanding time dimension role in misclassification
  - Effective ways to process text
    - “Filler” words to be eliminated
    - Time reliance of document vector model
    - Vector autoregressive article representation to further explore semantic meaning
  - Maximizing interpretability, looking at topic cycles and trends