Practical: Cable News Clip Classification

Matt Krasnow
mdkrasnow@college.harvard.edu

April 25th, 2025

This practical assignment focuses on classifying cable news clips by their source channel based solely on the transcript text. The project involves implementing multiple machine learning approaches to distinguish between 31 different news channels, evaluating their performance, and analyzing the strengths and limitations of different modeling techniques.

1 Part A: Feature Engineering, Baseline Models

1.1 Approach

For our initial exploration, we implemented two feature representation techniques. CountVectorizer creates a bag-of-words representation counting word frequencies with max_features=5000 and min_df=2. TfidfVectorizer uses similar parameters but weighs terms by their importance relative to the corpus. Both approaches transform text data into numerical features that can be processed by our models. TF-IDF has the advantage of downweighting common words while highlighting distinctive terms.

We used scikit-learn's LogisticRegression with max_iter=1000 and random_state=42. The model predicts output probabilities by learning weights for each feature-class combination. For multi-class classification (31 channels), it uses a one-vs-rest approach with the following mathematical formulation:

For each class k, logistic regression computes:

$$P(y = k|\mathbf{x}) = \frac{e^{\mathbf{w}_k \cdot \mathbf{x} + b_k}}{\sum_{j=1}^{K} e^{\mathbf{w}_j \cdot \mathbf{x} + b_j}} \tag{1}$$

Where $\mathbf{w}_k$ is the weight vector for class k, $\mathbf{x}$ is the feature vector, b_k is the bias term for class k, and K is the number of classes (31 in our case).

1.2 Results

The performance metrics show that Logistic Regression with CountVectorizer achieved 50.44% validation accuracy, while Logistic Regression with TfidfVectorizer reached 54.00% validation accuracy. Per-class performance varies significantly across channels as shown in Table 1.

Model	Overall Acc (%)	Top-3 and Bottom-3 Classes (F1)
LR + CountVectorizer	50.44	**Top:** BLOOMBERG (0.75), KDTV (0.83), KSTS (0.78) **Bottom:** RT (0.00), NTV (0.00), PRESSTV (0.00)
LR + TfidfVectorizer	54.00	**Top:** BLOOMBERG (0.73), BBCNEWS (0.64), KDTV (0.69) **Bottom:** RT (0.00), PRESSTV (0.00), KQED (0.11)

Table 1: Baseline model performance on validation data with best and worst performing channels.

1.3 Discussion

TF-IDF representation outperformed CountVectorizer likely because news transcripts contain many common words across channels, but distinctive vocabulary and phrases are better indicators of channel identity. TF-IDF emphasizes channel-specific terminology by reducing the impact of universally common words. This suggests the importance of term distinctiveness over raw frequency counts.

The baseline models perform reasonably well considering the challenging nature of the task (31 classes). Stronger performance on channels like BLOOMBERG (F1: 0.75/0.73), CNBC, and FBC suggests these channels have more distinctive language patterns, while poor performance on smaller channels (RT, NTV) indicates insufficient training examples. The class imbalance in our dataset likely contributes to this disparity, with majority classes benefiting from more training data.

2 Part B: More Modeling

2.1 First Step

2.1.1 Approach

We implemented a Random Forest classifier with default parameters (n_estimators=100, random_state=42) using the better-performing TF-IDF features. Random Forests construct multiple decision trees during training and use majority voting for classification.

Each tree in a Random Forest is built as follows:

1. Randomly sample n examples from the training data with replacement
2. At each node:
   (a) Randomly select m features without replacement
   (b) Split the node using the feature that provides the best split according to the Gini impurity
   (c) Repeat until maximum depth is reached or no improvement can be made
3. Aggregate predictions from all trees via majority voting

This approach theoretically excels at handling complex, high-dimensional text data by reducing overfitting through feature randomization and ensemble averaging.

Model	Overall Acc (%)	Top-3 and Bottom-3 Classes (F1)
LR + TfidfVectorizer	54.00	**Top:** BLOOMBERG (0.73), BBCNEWS (0.64), KDTV (0.69) **Bottom:** RT (0.00), PRESSTV (0.00), KQED (0.11)
Random Forest	43.07	**Top:** KDTV (0.80), MSNBCW (0.68), KSTS (0.63) **Bottom:** KQED (0.00), NTV (0.00), RT (0.00)

Table 2: Comparison of Logistic Regression and Random Forest performance.

2.1.2 Results

Random Forest with default parameters achieved 43.07% validation accuracy, significantly worse than our baseline Logistic Regression (54.00%). It shows high precision for some channels (ALJAZ, DW, SFGTV at 1.00) but poor recall, as shown in Table 2.

2.1.3 Discussion

The surprising underperformance of Random Forest suggests that the relationship between features and channel classification may be more linear than expected. The model might be overfitting to specific patterns in the training data. Important features (like "cnn", "ai", channel names) indicate the model relies on explicit channel mentions rather than subtler linguistic patterns.

This result challenges common assumptions that more complex nonlinear models always outperform linear ones. In text classification tasks with high dimensionality, simpler linear models might better capture the decision boundaries when features are already highly engineered (as with TF-IDF). The random forest's ensemble approach may be focusing too heavily on rare but distinctive words that don't generalize well.

2.2 Hyperparameter Tuning and Validation

2.2.1 Approach

We performed hyperparameter tuning for both model classes. For Logistic Regression, we used Grid Search over C values [0.1, 1.0, 10.0] and solvers ['lbfgs', 'saga']. For Random Forest, we employed Randomized Search over n_estimators [50, 100, 200], max_depth [10, 20, None], min_samples_split [2, 5], min_samples_leaf [1, 2], and max_features ['sqrt', 'log2'].

We used 2-fold cross-validation to balance between computational efficiency and reliable evaluation. For the Logistic Regression, the regularization parameter C controls the strength of regularization (lower C means stronger regularization). For Random Forest, parameters like max_depth control tree complexity, while n_estimators determines the number of trees in the ensemble.

2.2.2 Results

As shown in Table 3, tuned Logistic Regression achieved 54.20% validation accuracy (best params: C=10.0, solver='lbfgs'). Tuned Random Forest reached 44.39% validation accuracy. Logistic Regression remained the superior model even after tuning.

Model	Accuracy (%)	Improvement	Best Parameters
LR (Default)	54.00	–	–
LR (Tuned)	54.20	+0.20%	C=10.0, solver='lbfgs'
RF (Default)	43.07	–	–
RF (Tuned)	44.39	+1.32%	n_estimators=200, max_depth=None, min_samples_split=2, min_samples_leaf=1, max_features='sqrt'

Table 3: Hyperparameter tuning results showing modest improvements.

2.2.3 Discussion

Hyperparameter tuning yielded modest improvements. For Logistic Regression, a higher C value (10.0) indicates less regularization was beneficial, suggesting that the model wasn't overfitting with the default parameters. For Random Forest, deeper trees with more estimators improved performance, but still couldn't match the linear model. The persistent gap suggests fundamental characteristics of the data favor linear decision boundaries.

Our validation strategy using 2-fold cross-validation provided a reasonable estimate of model performance while maintaining computational efficiency. The modest gains from hyperparameter tuning (+0.20% for LR, +1.32% for RF) suggest that our initial parameter choices were reasonable, but Random Forest benefited more from optimization. Despite this, the performance gap between model classes persisted, reinforcing our conclusion that this particular text classification task is better suited to linear models.

3 Final Write-up and Reflections

3.1 Discussion:

Data Pipeline: Our data pipeline involved preprocessing news clip transcripts by removing HTML tags and entity references using regular expressions. We explored two feature extraction methods: CountVectorizer and TF-IDF Vectorizer, both with 5000 maximum features. The TF-IDF representation proved more effective by prioritizing distinctive terms over common words, which better captured channel-specific language patterns. This suggests that relative word importance is more predictive than pure frequency for this task.

Model Selection: We chose Logistic Regression as our baseline and Random Forest as our nonlinear model, based on their proven effectiveness for text classification tasks. Surprisingly, the linear model consistently outperformed the more complex Random Forest (54.00% vs. 43.07%), suggesting that channel classification may rely more on presence of key terms than complex interactions between features. This unexpected result demonstrates the importance of testing both simple and complex models rather than assuming more complexity equals better performance.

Model Tuning: Our tuning approach used Grid Search for Logistic Regression and Randomized Search for Random Forest, focusing on parameters most likely to affect performance. For efficiency, we reduced cross-validation folds from 3 to 2 and limited parameter combinations. Tuning yielded modest improvements (LR: 54.20%, RF: 44.39%), with larger gains for Random Forest, though it still underperformed Logistic Regression. The best LR configuration (C=10.0) indicates

that less regularization improved performance, suggesting the model benefited from increased flexibility.

Bias-Variance Trade-off: The performance gap between training and validation accuracy indicates some overfitting, particularly in the Random Forest model. The default Random Forest likely overfit to training data patterns that didn't generalize well. Hyperparameter tuning helped reduce this gap by finding configurations with better generalization. For Logistic Regression, we increased the regularization strength during tuning to find the optimal balance. The linear model's superior performance suggests that added model complexity introduced variance without capturing meaningful patterns.

Evaluation: We primarily used accuracy as our evaluation metric, while also analyzing per-class precision, recall, and F1-scores to understand performance across different channels. Accuracy provides a straightforward overall assessment, but the significant class imbalance makes per-class metrics crucial for understanding model effectiveness. The classification reports revealed substantial performance variation across channels, with larger channels generally showing better results. This approach helped identify specific channels where the model struggles (RT, NTV, PRESSTV), guiding potential improvements.

Domain-specific Evaluation: Our analysis revealed clear patterns in how different news sources are classified. Channels with distinctive language patterns (BLOOMBERG, CNBC) were consistently well-classified, while channels with fewer examples or more generic language were often misclassified. The feature importance analysis showed that channel names appearing in transcripts were strong predictors, which makes intuitive sense as news programs often reference their own network. This domain insight suggests that channels with strong branding in their transcripts are easier to identify.

Design Review: In retrospect, we could have explored more sophisticated text preprocessing methods and addressed class imbalance more directly. Additional approaches worth considering include: using n-grams to capture phrases rather than just individual words, implementing class weighting to improve performance on underrepresented channels, and exploring more advanced embedding techniques beyond TF-IDF. Our decision to focus on two core model types provided clear comparative insights, but exploring ensemble methods might have yielded better performance.

Execution & Implementation: From a real-world perspective, the developed models could be deployed as part of a media monitoring system, but would require regular retraining as language patterns evolve. The main limitation is computational efficiency - while Logistic Regression is lightweight enough for production, transformer models like our Part C exploration would require more resources. Ethically, such classification systems could raise concerns about media monitoring and potential misuse for censorship, making transparency in deployment critical. For broader deployment, the model would benefit from more diverse training data across time periods.

4 Optional Exploration, Part C: Explore some more!

4.1 Approach

For our optional exploration, we implemented a transformer-based model using DistilBERT, a more computationally efficient version of BERT (Bidirectional Encoder Representations from Transformers). Transformers represent the state-of-the-art in NLP tasks through:

1. Self-attention mechanisms that capture contextual relationships between words

Model	Validation Accuracy (%)	Performance Gain vs. LR
Logistic Regression (TF-IDF)	54.20	–
Random Forest	44.39	-9.81%
DistilBERT (base)	59.59	+5.39%
DistilBERT (fine-tuned)	56.63	+2.43%

Table 4: Performance comparison of all implemented models.

2. Pre-training on massive text corpora, enabling transfer learning

3. Fine-tuning on specific downstream tasks

We constructed a custom dataset and dataloader with a maximum sequence length of 128 tokens, trained the model for 3 epochs using AdamW optimizer (learning rate 5e-5), and implemented a linear schedule with warmup. We also experimented with class-weighted fine-tuning to address class imbalance, providing more emphasis on underrepresented channels.

4.2 Results

The transformer approach achieved impressive results, outperforming our traditional models. DistilBERT reached 59.59% validation accuracy, showing significant improvement over Logistic Regression (54.20%) and Random Forest (44.39%). When fine-tuned with class weights to address imbalance, performance reached 56.63%.

The transformer model showed substantial improvements in classifying challenging channels compared to our baseline models. For example, its F1-score for RT improved to 0.34 (from 0.00), and KQED to 0.23 (from 0.11).

4.3 Discussion

The superior performance of transformer models demonstrates their effectiveness in capturing nuanced language patterns. Unlike traditional bag-of-words approaches, transformers understand contextual relationships between words and can identify subtle channel-specific language patterns. This capability proved especially valuable for this classification task, where the distinction between channels often involves writing style, tone, and topic framing rather than just vocabulary choices.

Interestingly, our fine-tuning approach with class weights actually reduced overall accuracy (56.63% vs. 59.59%). This suggests that while we improved performance on underrepresented classes, it came at the cost of performance on majority classes. The training and validation loss curves also revealed potential overfitting after the first epoch, as validation loss began increasing despite continued improvement in training loss.

The implementation highlighted important considerations for transformer models, including computational requirements (significantly higher than traditional models) and the need for careful fine-tuning. While DistilBERT offered the best performance, the computational efficiency of Logistic Regression (with only a 5% accuracy reduction) may make it preferable in production environments with resource constraints or real-time classification needs.