Comparative Analysis of ML Techniques for Text Classification

Problem 4: Sports vs. Politics

Name: Viswanadhapalli Sujay
Roll Number: B22CS063

February 15, 2026

Abstract

This report documents the design, implementation, and evaluation of a binary text classifier capable of distinguishing between "Sports" and "Politics" documents. Leveraging the 20 Newsgroups dataset, I conducted a comparative study of three supervised learning algorithms: Multinomial Naive Bayes (MNB), Logistic Regression (LR), and Linear Support Vector Machines (SVM). Furthermore, I analyzed the impact of different feature extraction techniques, specifically Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), including n-gram analysis. The experimental results demonstrate that the **Linear SVM with TF-IDF unigram features** yields the highest performance, achieving an F1-score of 0.9653 and an accuracy of 95.99% on the held-out test set. This report details the data collection, preprocessing methodology, theoretical underpinnings of the chosen models, and a comprehensive error analysis.

1 Introduction

Text classification is a foundational task in Natural Language Understanding (NLU). It involves assigning predefined categorical labels to free-text documents based on their content. The objective of this assignment is to construct a robust system capable of automatically classifying text documents into one of two distinct domains: **Sports** or **Politics**.

The proliferation of digital content—from news articles to social media feeds—has made manual categorization impossible. Machine Learning (ML) offers a solution by learning patterns from labeled data to predict the category of unseen documents. In this project, I explore the end-to-end pipeline of a text classification system:

1. **Data Ingestion:** Collecting and structuring raw text data from the 20 Newsgroups corpus.
2. **Preprocessing:** Cleaning noise to isolate semantic content.
3. **Feature Engineering:** Converting variable-length text into fixed-length numerical vectors using Vector Space Models (VSM).
4. **Model Selection:** Training and evaluating probabilistic (Naive Bayes) and geometric (SVM) classifiers.

I hypothesize that while Sports and Politics share some common general vocabulary (e.g., "win", "loss", "race"), the specific domain terminology (e.g., "touchdown" vs. "legislation") is distinct enough for linear models to achieve high accuracy without deep neural networks.

2 Theoretical Background

To provide context for my model selection, I briefly outline the mathematical foundations of the three algorithms compared in this study.

2.1 Multinomial Naive Bayes (MNB)

Naive Bayes is a probabilistic classifier based on Bayes' Theorem, assuming conditional independence between features (words). For a document d and class c, the probability is proportional to:

$$P(c|d) \propto P(c) \prod_{i=1}^{n} P(w_i|c) \tag{1}$$

Where $P(c)$ is the prior probability of the class, and $P(w_i|c)$ is the likelihood of word w_i appearing in class c. Despite the "naive" assumption that word occurrences are independent (which is rarely true in language contexts where "White" predicts "House"), MNB is computationally efficient and often serves as a strong baseline for NLU tasks.

2.2 Logistic Regression (LR)

Logistic Regression is a discriminative model that estimates the probability that a given input vector $\mathbf{x}$ belongs to class $y = 1$ (Politics) versus $y = 0$ (Sports) using the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$.

$$P(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + b) \tag{2}$$

Unlike Naive Bayes, Logistic Regression learns the weights $\mathbf{w}$ that maximize the likelihood of the training data. It is particularly useful for interpretability, as the magnitude of a weight w_j directly corresponds to the influence of word j on the decision boundary.

2.3 Linear Support Vector Machine (SVM)

The Linear SVM seeks to find the optimal hyperplane $\mathbf{w}^T\mathbf{x}+b = 0$ that separates the two classes with the maximum margin. The objective is to minimize the Hinge Loss:

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i + b)) \tag{3}$$

SVMs are historically effective for text classification because they are robust in high-dimensional spaces where the number of features (vocabulary size) often exceeds the number of samples.

3 Data Collection and Analysis

3.1 Dataset Description

I utilized the **20 Newsgroups dataset**, a standard benchmark dataset for text classification available via 'scikit-learn'. The dataset consists of approximately 20,000 newsgroup documents partitioned across 20 different newsgroups.

To satisfy the problem statement, I filtered the dataset to form a binary classification task:

- **Class 0 (Sports):** Aggregated from 'rec.sport.baseball' and 'rec.sport.hockey'.
- **Class 1 (Politics):** Aggregated from 'talk.politics.guns', 'talk.politics.mideast', and 'talk.politics.misc'.

3.2 Exploratory Analysis

The dataset was split into a training set (80%) and a testing set (20%) using stratified sampling to ensure the class distribution remained consistent between training and evaluation.

Split	Total Documents	Split Ratio
Training Set	3,694	80%
Test Set	924	20%
Total	**4,618**	**100%**

Table 1: Dataset Split Statistics

3.3 Preprocessing Pipeline

Raw text is unstructured and noisy. I implemented a rigorous preprocessing pipeline to clean the data:

1. **Metadata Removal:** I stripped headers, footers, and quotes from the newsgroup posts. This is a critical step; without it, the model might learn to classify based on email domains (e.g., distinguishing academic emails in politics vs. ISP emails in sports) rather than the actual content.
2. **Lowercasing:** All text was converted to lowercase to treat "Election" and "election" as the same token.
3. **Noise Removal:** I used Regular Expressions (Regex) to remove:
   - URLs (e.g., 'http://...')
   - Email addresses
   - Non-alphanumeric characters (punctuation)
4. **Whitespace Normalization:** Multiple spaces and line breaks were collapsed into single spaces.

4 Feature Extraction Methodology

Machine learning algorithms require numerical input. I compared three strategies to convert text into numerical vectors.

4.1 Bag of Words (BoW)

The BoW model creates a vocabulary of all unique words in the corpus and represents each document as a vector of word counts. While simple, BoW suffers from the issue that frequent, non-informative words (stopwords) can dominate the feature space.

4.2 TF-IDF Representation

To address the limitations of BoW, I employed **Term Frequency-Inverse Document Frequency (TF-IDF)**. This technique weights words based on their importance.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right) \tag{4}$$

Where $tf_{i,j}$ is the frequency of word i in document j, N is the total number of documents, and df_i is the number of documents containing word i. This effectively down-weights common words like "the" or "is" and up-weights discriminative terms like "stadium" or "amendment".

4.3 N-Grams

I also experimented with **Bigrams** (pairs of consecutive words). This allows the model to capture local context. For example, "White House" has a political meaning distinct from the colors "white" and "house" appearing separately. I tested a range of ‘(1, 1)‘ (unigrams only) and ‘(1, 2)‘ (unigrams and bigrams).

5 Quantitative Comparisons & Results

5.1 Experimental Setup

All models were trained using Python’s ‘scikit-learn‘ library. I used a fixed random seed (‘42‘) to ensure reproducibility. The primary evaluation metric chosen was the **F1-Score**. In binary classification, accuracy can be misleading if classes are imbalanced. F1 provides a harmonic mean of Precision (P) and Recall (R):

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{5}$$

5.2 Results Summary

The performance of the six trained pipelines is summarized in Table 2 below.

Table 2: Comparative Performance of ML Models (Sorted by F1 Score)

Feature Representation	Classifier	Accuracy	F1 Score	ROC AUC
TF-IDF (Unigram)	**Linear SVM**	**0.9599**	**0.9653**	**0.9932**
TF-IDF (Unigram + Bigram)	Linear SVM	0.9556	0.9619	0.9924
BoW (Unigram)	Naive Bayes	0.9545	0.9613	0.9861
BoW (Unigram)	Logistic Regression	0.9513	0.9576	0.9885
TF-IDF (Unigram)	Naive Bayes	0.9351	0.9457	0.9921
TF-IDF (Bigram)	Naive Bayes	0.8929	0.9135	0.9861

5.3 Detailed Analysis

1. Superiority of SVM: The **Linear SVM with TF-IDF** unigram features emerged as the best performing model. This aligns with theoretical expectations; SVMs maximize the margin between classes, which is highly effective in high-dimensional sparse spaces created by text data. The ROC AUC of 0.9932 indicates near-perfect separation capability.

2. The "Curse of Dimensionality" in Bigrams: Counter-intuitively, adding Bigrams (TF-IDF + Bigram) slightly reduced performance for the SVM (F1 dropped from 0.9653 to 0.9619) and significantly hurt the Naive Bayes model. This suggests that the dataset size (approx. 4,600 docs) is too small to support the massive increase in feature space that comes with bigrams. The model likely started overfitting to noise in the bigram features.

3. Robustness of Naive Bayes: Multinomial Naive Bayes performed surprisingly well with simple Bag-of-Words features (95.45% accuracy). This confirms why MNB is often the "baseline" for text tasks: it requires very little data to estimate parameters effectively. However, it struggled with TF-IDF, likely because MNB assumes integer counts (multinomial distribution), whereas TF-IDF produces continuous floats.

6 Error Analysis and Limitations

To better understand the limitations of my best model (Linear SVM), I analyzed the Confusion Matrix.

	Predicted Sports	**Predicted Politics**
Actual Sports	373 (TN)	26 (FP)
Actual Politics	11 (FN)	514 (TP)

Figure 1: Confusion Matrix for Linear SVC (TF-IDF)

6.1 Analysis of Misclassifications

The model committed 37 errors in total out of 924 test samples.

- **False Positives (26):** Sports articles misclassified as Politics. This likely happens in sports articles discussing rules, organizational bodies, or contracts, using terms like "law," "ruling," or "penalty" which overlap with political jargon.

- **False Negatives (11):** Politics articles misclassified as Sports. This is rarer but may occur in "horse-race" political coverage using metaphors like "winning," "losing," "race," or "team."

6.2 System Limitations

1. **Static Vocabulary:** The model relies on a fixed vocabulary learned during training. It cannot handle Out-Of-Vocabulary (OOV) words. If a new politician named "Zokub" appears, the model will not recognize the name as a feature.
2. **Context Insensitivity:** By using unigrams (Bag of Words approach), the model loses word order. The sentence "The President killed the bill" and "The bill killed the President" would have identical feature vectors, despite vastly different meanings.
3. **Domain Specificity:** The 20 Newsgroups dataset is from the 1990s. The language usage, political topics (e.g., Soviet Union, Clinton), and sports references are dated. The model would likely require retraining to perform well on modern social media text (tweets) or contemporary news.

7 Conclusion

In this assignment, I successfully designed and evaluated a text classification system for Sports vs. Politics. By comparing multiple algorithms and feature representations, I identified that a **Linear Support Vector Machine (SVM)** combined with **TF-IDF** feature extraction offers the most robust performance, achieving an F1 score of **0.9653**.

The study highlighted that while complex features (like bigrams) theoretically add context, they can degrade performance in smaller datasets due to sparsity. Simple, well-tuned linear models remain highly effective for distinct topic classification tasks.

Future improvements could involve:

- **Deep Learning:** Implementing Transformer-based models (like BERT) to capture semantic context better than TF-IDF.
- **Hyperparameter Tuning:** Using Grid Search to optimize the regularization parameter C in SVM and α in Naive Bayes.
- **Ensemble Methods:** Combining predictions from MNB and SVM to reduce variance.

References

[1] Pedregosa, F. et al., "Scikit-learn: Machine Learning in Python", *Journal of Machine Learning Research*, 12, pp. 2825-2830, 2011. `https://jmlr.csail.mit.edu/papers/v12/pedregosa11a.html`

[2] Lang, K., "Newsweeder: Learning to filter netnews", *Proceedings of the 12th International Conference on Machine Learning*, 1995. `http://qwone.com/~jason/20Newsgroups/`