Text Classification:

Sports vs Politics Document Classifier

Neeraj Mansingh

B23CS1095

Natural Language Understanding

Abstract

This report presents a comprehensive study on binary text classification, specifically distinguishing between sports and politics-related documents. We implemented and evaluated three distinct machine learning algorithms using the 20 Newsgroups dataset as our foundation. The study examines how different feature extraction methods—Bag of Words, Bigrams, and TF-IDF—impact classification performance across Naive Bayes, Logistic Regression, and K-Nearest Neighbors classifiers. Our experimental results demonstrate that simpler feature representations combined with probabilistic models achieve superior accuracy, with Naive Bayes on Bag of Words features reaching 96.97% accuracy. The findings challenge common assumptions about feature complexity and model sophistication in text classification tasks.

1. Introduction

Text classification stands as one of the fundamental problems in natural language processing, with applications ranging from spam detection to sentiment analysis. The ability to automatically categorize documents into predefined classes has become increasingly important as the volume of digital text continues to grow exponentially. This project tackles a specific instance of the binary classification problem: distinguishing between sports-related and politics-related news articles.

The motivation behind choosing this particular classification task stems from the inherent linguistic differences between these two domains. Sports articles typically discuss games, scores, players, and teams using domain-specific terminology. Political discourse, on the other hand, involves policy discussions, governmental affairs, and ideological debates with its own distinct vocabulary. The contrast between these domains makes them ideal candidates for exploring how different machine learning approaches handle topical text classification.

This study makes several contributions to understanding text classification performance. First, we implement three fundamentally different classification algorithms from scratch, avoiding black-box libraries to ensure deep understanding of the underlying mechanisms. Second, we systematically evaluate how different feature extraction techniques affect each classifier's performance. Third, we analyze unexpected results that challenge conventional wisdom about feature engineering in NLP tasks.

2. Data Collection and Dataset Description

2.1 Dataset Selection

For this classification task, we utilized the 20 Newsgroups dataset, a well-established benchmark in text classification research. Rather than using the entire corpus, we focused on a strategic subset that aligned with our binary classification objective. The dataset is publicly available and widely used in academic research, making our results reproducible and comparable to existing literature.

The 20 Newsgroups dataset contains approximately 20,000 newsgroup documents partitioned across 20 different newsgroups. We selected five specific newsgroups that clearly represent our two target categories:

Sports Category:

- rec.sport.baseball - discussions about baseball games, players, and statistics
- rec.sport.hockey - hockey-related conversations and news

Politics Category:

- talk.politics.guns - debates about gun control and Second Amendment issues
- talk.politics.mideast - discussions about Middle Eastern politics
- talk.politics.misc - miscellaneous political discussions

2.2 Dataset Statistics

The final dataset consists of 4,618 documents total. This represents a balanced collection suitable for binary classification, though there is slight imbalance due to the different sizes of the original newsgroups. The documents vary in length from short posts of a few sentences to lengthy discussions spanning multiple paragraphs. Each document includes the original newsgroup headers, author information, and the message content itself.

2.3 Data Preprocessing

The preprocessing pipeline was kept intentionally simple to evaluate the raw performance of different algorithms without excessive feature engineering. Our approach included:

- Case normalization - converting all text to lowercase to ensure case-insensitive matching
- Tokenization - extracting word tokens using regular expressions that match word boundaries
- Feature limitation - restricting vocabulary to the 500 most frequent terms to manage computational complexity

Notably absent from our preprocessing are stemming, lemmatization, and stopword removal. This decision was deliberate - we wanted to assess whether these commonly applied techniques are necessary for achieving good classification performance. The results, as we will see, suggest that simple tokenization can be surprisingly effective.

2.4 Train-Test Split

We employed a standard 80-20 train-test split with random shuffling. This means 3,694 documents were used for training and 924 documents for testing. The split was performed randomly rather than chronologically, which is appropriate given that newsgroup posts do not have strong temporal dependencies. Each experimental run uses a new random split, which introduces some variance in results but provides a more realistic evaluation of model performance.

3. Methodology and Implementation

3.1 Feature Extraction Techniques

Feature extraction transforms raw text into numerical vectors that machine learning algorithms can process. We implemented three different feature extraction approaches to understand their impact on classification performance.

3.1.1 Bag of Words (BoW)

The Bag of Words model represents each document as a vector of word frequencies, disregarding grammar and word order while maintaining multiplicity. For a vocabulary of size V, each document becomes a V-dimensional vector where element i contains the count of how many times word i appears in the document. Despite its simplicity, BoW captures important information about document content through word frequency distributions.

The implementation builds a vocabulary dictionary during the fit phase by selecting the 500 most frequent terms across all training documents. During transformation, each document is tokenized and its words are counted, with the counts mapped to the corresponding vocabulary positions. This approach is computationally efficient and surprisingly effective for many text classification tasks.

3.1.2 Bigrams

Bigram features capture sequential word pairs, providing some contextual information that unigrams miss. For example, 'supreme court' as a bigram carries different meaning than 'supreme' and 'court' separately. The bigram model considers all consecutive two-word sequences in the document, creating a feature space of word pairs rather than individual words.

Our implementation generates bigrams by sliding a window of size 2 across the token sequence. The vocabulary consists of the 500 most frequent bigrams observed in the training set. While bigrams can capture some phrasal patterns and collocations, they dramatically increase the feature space sparsity compared to unigrams, which can hurt performance with limited training data.

3.1.3 TF-IDF (Term Frequency-Inverse Document Frequency)

TF-IDF weighs terms based on their importance to a document relative to the entire corpus. The term frequency component measures how often a term appears in a document, while the inverse document frequency component down-weights terms that appear frequently across many documents. The formula is:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \log(N / \text{DF}(t))$$

where TF(t,d) is the frequency of term t in document d, N is the total number of documents, and DF(t) is the number of documents containing term t. This weighting scheme emphasizes discriminative terms while reducing the influence of common words that appear everywhere.

3.2 Classification Algorithms

3.2.1 Naive Bayes Classifier

Naive Bayes applies Bayes' theorem with the naive assumption that features are

conditionally independent given the class label. For text classification, this means we assume the presence of one word is independent of the presence of other words given the document category. While this assumption is clearly violated in natural language, Naive Bayes often performs remarkably well in practice.

The classifier computes the posterior probability of each class given the document features using:

$$P(\text{class}|\text{document}) \propto P(\text{class}) \times \prod P(\text{word}|\text{class})^{\text{count}}$$

We implement this using log probabilities to avoid numerical underflow. The class prior P(class) is estimated from the training data proportions. Feature probabilities P(word|class) are estimated using maximum likelihood with Laplace smoothing (add-one smoothing) to handle words not seen during training. The smoothing adds a count of 1 to every feature in every class, preventing zero probabilities.

3.2.2 Logistic Regression

Logistic Regression is a discriminative linear classifier that models the probability of class membership using the logistic function. Unlike Naive Bayes which models the joint distribution P(X,Y), logistic regression directly models the conditional probability P(Y|X). The model learns a weight vector w and bias term b such that:

$$P(\text{class}=1|x) = \sigma(w \cdot x + b) = 1 / (1 + e^{-(w \cdot x + b)})$$

We optimize the model parameters using gradient descent. The implementation runs for 300 iterations with a learning rate of 0.1. At each iteration, we compute predictions for all training examples, calculate the gradient of the log-loss with respect to each weight, and update the weights in the direction that reduces the loss. The sigmoid function is clamped to prevent numerical overflow for large positive or negative values.

3.2.3 K-Nearest Neighbors (KNN)

K-Nearest Neighbors is a non-parametric instance-based learning algorithm. Unlike Naive Bayes and Logistic Regression which learn explicit models during training, KNN simply stores the training data. Classification is performed by finding the k closest training examples to the test point and taking a majority vote among their labels.

We use Euclidean distance as the similarity metric and set k=5. During prediction, for each test document, we compute its distance to all training documents, identify the 5 nearest neighbors, and assign the most common class label among those neighbors. While simple conceptually, KNN can be computationally expensive for large datasets since it requires distance calculations to all training points for each prediction.

4. Experimental Results and Analysis

4.1 Overall Performance Comparison

The table below summarizes the accuracy achieved by each combination of feature extraction method and classification algorithm on the test set:

Feature Type	Naive Bayes	Logistic Regression	KNN (k=5)
Bag of Words	96.97%	96.00%	85.50%
Bigrams	84.52%	83.33%	74.13%
TF-IDF	63.10%	54.65%	86.69%

These results reveal several interesting patterns that warrant detailed discussion.

4.2 Feature Type Analysis

4.2.1 Bag of Words Performance

Bag of Words features achieved the best overall results, particularly with Naive Bayes (96.97%) and Logistic Regression (96.00%). This is somewhat surprising given that BoW is the simplest feature representation. The strong performance suggests that for this classification task, word frequency information alone provides sufficient discriminative power. Sports documents contain distinctive vocabulary like 'hockey,' 'baseball,' 'game,' 'score,' and 'player,' while political documents use terms like 'government,' 'policy,' 'law,' and 'rights.' The simple counting of these domain-specific terms effectively separates the two classes.

Even KNN, which typically struggles with high-dimensional sparse data, achieved 85.5% accuracy with BoW features. This indicates that the document representations form reasonably well-separated clusters in the feature space.

4.2.2 Bigram Performance Degradation

Bigram features led to substantial performance drops across all three classifiers. Naive Bayes accuracy fell to 84.52%, Logistic Regression to 83.33%, and KNN to just 74.13%. This degradation likely stems from increased sparsity - with a vocabulary limited to 500 features, bigrams cover far fewer of the actual word pairs in documents compared to how well unigrams cover individual words. Many relevant bigrams in test documents were never seen during training, resulting in zero features for those positions.

Additionally, bigrams fragment the statistical signal. Instead of accumulating evidence from seeing 'supreme' and 'court' separately multiple times, the model only gets evidence from seeing 'supreme court' as a unit. With limited training data, this

fragmentation hurts generalization. The results suggest that for this task and dataset size, the additional context from word pairs does not outweigh the increased sparsity.

4.2.3 TF-IDF Unexpected Behavior

The TF-IDF results present the most puzzling pattern. While KNN achieved its best performance with TF-IDF (86.69%), both Naive Bayes (63.1%) and Logistic Regression (54.65%) performed poorly - worse than random guessing would suggest for a balanced binary classification problem. This dramatic failure requires careful analysis.

The poor performance likely results from the interaction between TF-IDF normalization and the assumptions of Naive Bayes. Naive Bayes expects count data and models P(word|class) as a multinomial distribution. TF-IDF produces real-valued, normalized features that violate these distributional assumptions. The log-probability calculations in Naive Bayes become meaningless when applied to TF-IDF values rather than counts.

For Logistic Regression, the failure might stem from the scale of TF-IDF values and the fixed learning rate. TF-IDF weights are typically small real numbers, and the gradient descent optimization may not converge properly with our chosen hyperparameters. The fixed learning rate of 0.1 might be inappropriate for the normalized TF-IDF feature space, leading to unstable optimization.

KNN's success with TF-IDF makes sense because it only relies on distance calculations, not distributional assumptions. TF-IDF provides better-scaled features for distance computation than raw counts, which tend to be dominated by document length. The normalization in TF-IDF helps KNN by making documents of different lengths more comparable.

4.3 Classifier Comparison

4.3.1 Naive Bayes Strengths

Naive Bayes achieved the highest overall accuracy (96.97% with BoW) and proved remarkably robust. The classifier's probabilistic framework naturally handles the sparse, high-dimensional feature spaces common in text classification. The independence assumption, while technically incorrect, works well because the strong signal from domain-specific vocabulary overwhelms the dependencies between words.

Another advantage of Naive Bayes is its computational efficiency. Training requires only a single pass through the data to compute word counts per class. Prediction involves simple probability calculations without iterative optimization. For applications requiring fast training and inference, Naive Bayes offers an excellent speed-accuracy tradeoff.

4.3.2 Logistic Regression Performance

Logistic Regression achieved competitive performance with BoW features (96.00%), nearly matching Naive Bayes. The discriminative approach of directly modeling P(class|

document) can be advantageous when the independence assumptions of Naive Bayes are severely violated. However, for this task, the simpler Naive Bayes approach proved equally effective.

The degradation with bigrams and TF-IDF suggests that Logistic Regression is more sensitive to feature engineering choices and hyperparameter settings. The fixed learning rate and iteration count may not be optimal for all feature representations. A production system would likely benefit from hyperparameter tuning and potentially more sophisticated optimization algorithms.

4.3.3 KNN Limitations

KNN consistently underperformed the parametric models with BoW and bigram features, though it achieved the best result with TF-IDF. The algorithm's main weakness is its sensitivity to the distance metric and feature scaling. In high-dimensional spaces, Euclidean distance becomes less meaningful as all points appear roughly equidistant from each other - a phenomenon known as the curse of dimensionality.

Additionally, KNN has significant computational costs during prediction. While training is instant (just storing the data), each prediction requires computing distances to all training examples. With 3,694 training documents and 500-dimensional feature vectors, this becomes expensive. For real-time applications, the prediction latency could be prohibitive.

5. System Limitations

5.1 Current Limitations

5.1.1 Limited Vocabulary Size

Restricting the vocabulary to 500 features was necessary for computational tractability but likely hurt performance. Many informative words are excluded, and the feature space may not capture the full richness of the document content. A production system would benefit from larger vocabularies (5,000-10,000 features) or more sophisticated feature selection methods that identify the most discriminative terms rather than simply taking the most frequent ones.

5.1.2 Binary Classification Scope

The system only handles two classes. Extending to multi-class classification would require modifications to all three algorithms. Naive Bayes extends naturally to multiple classes, but Logistic Regression would need either one-vs-rest or softmax regression, and KNN would need careful consideration of class imbalance in voting. Real-world applications often involve many categories, so this limitation is significant.

5.1.3 No Hyperparameter Optimization

All algorithms use fixed hyperparameters: k=5 for KNN, learning rate=0.1 and iterations=300 for Logistic Regression. These were chosen based on reasonable defaults rather than systematic tuning. The poor TF-IDF results for Logistic Regression might improve significantly with proper hyperparameter optimization. Cross-validation to select optimal hyperparameters would likely improve performance across the board.

5.1.4 Minimal Preprocessing

We deliberately kept preprocessing simple, but this may leave performance on the table. Techniques not explored include stopword removal (eliminating common words like 'the' and 'is'), stemming or lemmatization (reducing words to root forms), handling of punctuation and special characters, and removal of newsgroup headers and metadata. These techniques might help, though our results suggest simple approaches can be effective.

5.2 Domain Limitations

The system is specifically trained on sports vs politics classification. The models learn domain-specific vocabulary patterns that may not transfer to other classification tasks. A model trained on our data would likely fail on sentiment analysis or spam detection. Additionally, the newsgroup writing style differs from modern social media text, news articles, or formal documents. The system's performance on contemporary text sources is unknown and likely degraded.

6. Conclusion

This project successfully demonstrated that effective text classification does not always require complex feature engineering or sophisticated algorithms. Our experiments with sports vs politics document classification yielded several key insights:

First, simple Bag of Words features combined with Naive Bayes achieved excellent performance (96.97% accuracy), suggesting that word frequency alone provides strong discriminative signal for topical classification. The assumption that more complex features automatically improve performance proved incorrect - bigrams hurt all classifiers, and TF-IDF catastrophically failed with Naive Bayes despite being a standard recommendation in many tutorials.

Second, different feature representations suit different algorithms. While KNN performed best with normalized TF-IDF features, Naive Bayes and Logistic Regression strongly preferred raw count-based BoW features. This highlights the importance of matching feature engineering to algorithm assumptions rather than blindly applying standard pipelines.

Third, classical machine learning algorithms remain competitive for many text classification tasks. While deep learning dominates many NLP benchmarks, our results show that for well-defined classification problems with clear vocabulary distinctions, traditional approaches offer excellent accuracy with far less computational overhead and greater interpretability.

The project achieved its core objective of comparing multiple machine learning approaches across different feature representations. The implementation from scratch, while limiting us to simpler algorithms, ensured deep understanding of the underlying mechanics. Future work should address the limitations identified, particularly around hyperparameter optimization and evaluation metrics, while exploring how these findings generalize to other text classification domains.