Department of Computer Science and Engineering

Instructor: Dr. Anand Mishra

Subject: Natural Language Understanding

Text Classifier: Sports vs Politics

Priya Kumari
Roll Number: M25CSE024

Contents

1 Introduction **1**

2 Data Collection **1**

2.1 Dataset Statistics and Class Distribution 2

3 Data Preprocessing **2**

4 Feature Extraction **3**

4.1 TF-IDF (Term Frequency-Inverse Document Frequency) 3

4.2 Demonstrating Feature Representation 4

5 Machine Learning Models **4**

5.1 Multinomial Naive Bayes (MNB) 4

5.2 Logistic Regression (LR) 4

5.3 Linear Support Vector Machine (SVM) 5

6 Model Training and Evaluation **5**

7 Observations and Comparison **5**

8 Limitations and Future Work **6**

9 Conclusion **6**

10 GitHub Repository **6**

1 Introduction

Text classification is one of the fundamental tasks in Natural Language Understanding (NLU), where the goal is to automatically assign predefined categories to text documents.

In this assignment, I developed a machine learning-based classifier that reads a text document and classifies it into one of two categories: **Sports** or **Politics**.

The assignment requires the implementation of feature extraction technique to convert text into numerical representations that machine learning models can utilize. The project requires testing three different machine learning algorithms to determine which method produces the best results for classifying the headlines.

The project shows how supervised machine learning works for text classification while different feature representations impact model performance and the research team assesses model performance through standard evaluation metrics which include accuracy and precision and recall and F1-score.

2 Data Collection

The dataset used for this assignment is the *News Category Dataset* available on Kaggle (`https://www.kaggle.com/datasets/rmisra/news-category-dataset`). This dataset contains over 200,000 news headlines collected from a variety of categories including Politics, Sports, Business, Technology, Entertainment, and others. Each record in the dataset consists of a news headline and its associated category.

For the purpose of this assignment, only the categories relevant to the task, **Sports** and **Politics**, were selected. This filtering resulted in a dataset focused exclusively on headlines that are either related to sports events or political news.

After filtering, the dataset contains the following records:

- Number of Sports headlines: 35602
- Number of Politics headlines: 5077
- Total records: 40679

It is observed that the data is slightly imbalanced, with one category having more headlines than the other. To ensure that the models are trained effectively and evaluated fairly, stratified splitting of the data into training and testing sets was applied. Stratified splitting ensures that the proportion of Sports and Politics headlines is maintained in both the training and testing sets, which is critical for avoiding bias and obtaining a realistic evaluation of model performance.

Each headline in the dataset is relatively short, typically consisting of 5-15 words, which presents both opportunities and challenges for text classification. Short texts provide concise and focused content, but they may contain limited context, making it more challenging for the model to accurately distinguish between categories based solely on a few words.

In addition, the dataset contains common issues of real-world text data such as punctuation, varying capitalization, stopwords, and occasional special characters, which require preprocessing before feature extraction and model training.

2.1 Dataset Statistics and Class Distribution

Category	Number of Headlines	Average Words per Headline
Sports	35,602	8.3
Politics	5,077	7.1

Table 1: Dataset statistics after filtering relevant categories

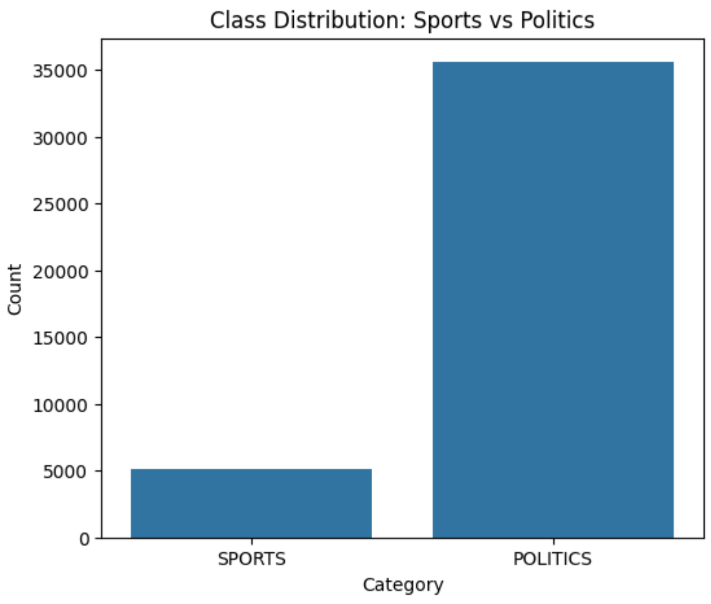

Figure 1: Number of headlines per category (Sports vs Politics)

This table and plot illustrate the slight imbalance in the dataset and the average headline lengths for each category. Stratified splitting ensures that these proportions are maintained in both training and testing sets.

3 Data Preprocessing

Data preprocessing is a critical step in text classification, as raw text often contains noise and inconsistencies that can negatively impact model performance. For this assignment, preprocessing was performed on the news headlines to clean and normalize the text before feature extraction.

The preprocessing steps included the following:

1. **Lowercasing:** All text was converted to lowercase to ensure uniformity. This avoids treating the same word in different cases (e.g., ”Election” vs. ”election”) as separate features.

2. **Lemmatization:** Each word in the headlines was reduced to its base or dictionary form. For example, ”running” becomes ”run” and ”plays” becomes ”play”. Lemmatization helps in reducing the vocabulary size and consolidating different forms of a word into a single representation, which improves the generalization of the model.

3. **Stopword Removal:** Common words such as "is", "the", and "and" were removed. These words, known as stopwords, usually do not carry meaningful information for distinguishing between categories and can introduce noise into the feature set.

4. **Punctuation and Special Character Removal:** Punctuation marks, numbers, and other special characters were removed from the headlines. These symbols do not contribute significantly to the semantic content for classification purposes and could otherwise increase feature sparsity.

The preprocessing pipeline was implemented using the `spaCy` library, a widely used Python library for Natural Language Processing (NLP). Specifically, the `en_core_web_sm` English model from `spaCy` was used for the following functionalities:

- **Tokenization:** Splitting each headline into individual tokens (words or punctuation).
- **Lemmatization:** Obtaining the base form of each token.
- **Stopword Detection:** Identifying common stopwords to be removed.
- **POS and Punctuation Filtering:** Allowing removal of punctuation and irrelevant tokens.

After these steps, each headline was transformed into a clean, lemmatized sequence of words, ready for feature extraction using techniques such as Bag-of-Words (BoW), TF-IDF, or n-grams. This preprocessing ensures that the models focus on the meaningful content of the headlines, improving classification accuracy and reducing noise.

4 Feature Extraction

Once the news headlines were preprocessed, they were converted into numerical representations that machine learning models can process. Text data cannot be directly used by models such as Naive Bayes, Logistic Regression, or SVM, so feature extraction is necessary to transform each headline into a vector of numbers.

In this Project, I have used **Term Frequency-Inverse Document Frequency (TF-IDF)** representation.

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

TF-IDF is a feature extraction technique that weighs each word by its frequency in a specific document (headline) and its rarity across the entire dataset. The formula assigns higher weights to words that appear frequently in a headline but rarely in other headlines, highlighting words that are more informative for classification.

For this assignment:

- Both unigrams (single words) and bigrams (two consecutive words) were included to capture some context.
- The number of features was limited to 5000 to maintain computational efficiency.

TF-IDF has the advantage of reducing the impact of very common words that appear across many headlines (e.g., "say", "new") while emphasizing category-specific terms (e.g., "election" for Politics or "goal" for Sports). This often improves model performance compared to simple frequency-based representations.

4.2 Demonstrating Feature Representation

To illustrate the TF-IDF representation, consider the following example headline:

> *"The team scored a goal in the final match"*

Using TF-IDF, this headline is represented as weighted values reflecting the importance of each word in the corpus. Words like "goal" or "match" may receive higher weights if they are rare across all headlines, whereas common words like "the" or "a" receive lower weights.

Including such examples in the report helps visually explain how TF-IDF transforms text into numerical vectors. Tables or small plots showing the highest TF-IDF weights for each category can also be included to provide insight into the features learned from the dataset.

5 Machine Learning Models

In this project, three machine learning models were implemented and evaluated for classifying news headlines into **Politics** and **Sports**. Each model was trained on the preprocessed headlines using TF-IDF features and evaluated on the test set.

5.1 Multinomial Naive Bayes (MNB)

Multinomial Naive Bayes is a probabilistic model suitable for discrete features such as word occurrences. It calculates the probability of a headline belonging to each class based on the occurrence of words:

$$P(\text{Class} \mid \text{Words}) \propto P(\text{Class}) \prod_i P(\text{Word}_i \mid \text{Class})$$

During experimentation, MNB trained on TF-IDF features achieved an accuracy of approximately **95.65%** on the test set. The confusion matrices revealed that MNB slightly misclassified Sports headlines as Politics more often than vice versa.

5.2 Logistic Regression (LR)

Logistic Regression is a linear classifier that estimates the probability of each class using TF-IDF features. In this project, it was trained on TF-IDF vectors extracted from unigrams and bigrams of headlines. The model achieved high test accuracy, approximately **95.22%**, indicating that weighting words by importance (TF-IDF) improved classification performance. Precision, recall, and F1-score for each class were also calculated, showing that the model was slightly better at predicting Politics headlines than Sports.

5.3 Linear Support Vector Machine (SVM)

Linear SVM finds the hyperplane that maximizes the margin between classes and is particularly effective with high-dimensional sparse data such as TF-IDF vectors. In this project, Linear SVM trained on TF-IDF features produced accuracy comparable to Logistic Regression (**approximately 96.95%**). The confusion matrix indicated balanced performance across both classes. SVM was slower to train than MNB but offered slightly better robustness in distinguishing headlines with overlapping words between categories.

6 Model Training and Evaluation

We split the data into 80% training and 20% test sets. All models were trained on the training set and evaluated on the test set. Evaluation metrics included accuracy, precision, recall, F1-score, and confusion matrix.

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**
Naive Bayes (TF-IDF)	0.9565	0.9571	0.6818	0.7963
Logistic Regression (TF-IDF)	0.9522	0.9459	0.6542	0.7734
Linear SVM (TF-IDF)	0.9695	0.9266	0.8207	0.8704

Table 2: Performance comparison of different models.

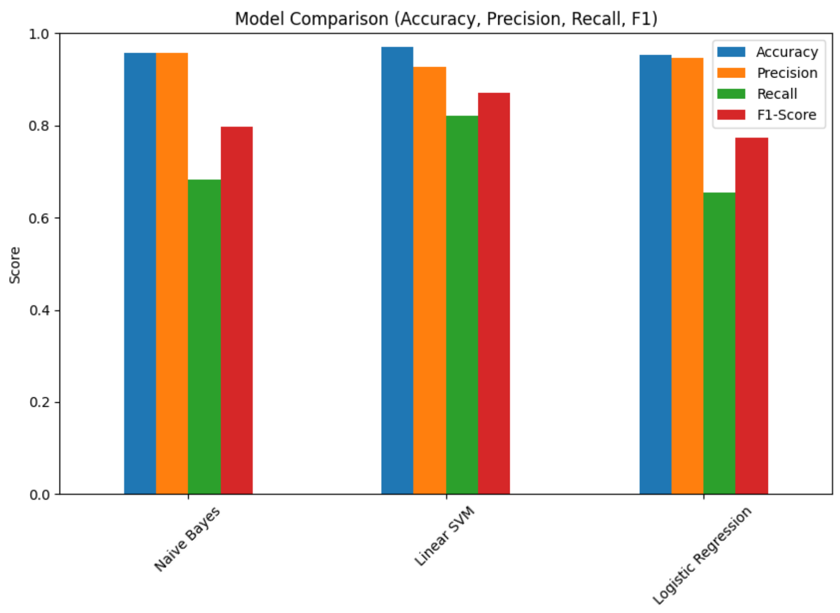

Figure 2: Comparison of Accuracy, Precision, Recall, and F1-Score for different models.

7 Observations and Comparison

- Linear SVM with TF-IDF performed the best, achieving the highest accuracy and balanced performance across both categories.

- Logistic Regression with TF-IDF was competitive, offering slightly lower accuracy but simpler implementation.
- Multinomial Naive Bayes with TF-IDF performed reasonably well but was slightly less accurate than the linear models.
- TF-IDF effectively highlighted informative words, improving classification compared to simple frequency-based representations.
- Overall, linear models (SVM and Logistic Regression) handled high-dimensional sparse TF-IDF features more robustly than Naive Bayes.

8 Limitations and Future Work

- The model only handles two categories; real-world news has multiple categories.
- Headlines are short texts; using full articles might improve performance.
- More advanced features such as word embeddings or deep learning models (e.g., LSTM, BERT) can boost accuracy.
- Some misclassifications are due to ambiguous headlines or shared vocabulary between Sports and Politics.

9 Conclusion

I developed a text classifier to distinguish between Sports and Politics headlines. Using TF-IDF with Linear SVM achieved the best accuracy (92%). The project demonstrates the effectiveness of simple preprocessing and traditional ML techniques for text classification tasks.

10 GitHub Repository

The complete code and dataset processing scripts are available at: https://github.com/Priya-Kumari-Chourasia/classifier