# Sports or Politics?

## CSL 7640: Natural Language Understanding – Problem 4

Zenith (Roll No. M25CSA032)

**Abstract**

This report presents a binary text classifier that decides whether a document is about sports or politics. I build the system on top of a Sports/Politics subset of the 20 Newsgroups corpus and explore a small grid of feature representations (Bag of Words, TF–IDF, various n–gram settings) combined with three standard classifiers (Multinomial Naive Bayes, Linear SVM, and Random Forest). Rather than chasing a single headline number, the focus is on understanding how these choices interact: which combinations work well for this particular task, which ones add unnecessary complexity, and where the remaining errors come from. The experiments show that relatively simple models already perform strongly, while more elaborate configurations do not always translate into better accuracy.

## 1 Introduction

Topic classification is one of the classical problems in Natural Language Processing (NLP). In this assignment the task is deliberately focused: given a news document, decide whether it is about *Sports* or *Politics*. At first glance this may look straightforward—words like "team", "game" or "goal" strongly suggest sports, whereas "government", "election" or "policy" are clearly political. At the same time, there are edge cases: security policies in stadiums, political metaphors that borrow sports language, or short noisy posts.

My objective in this problem is twofold. First, I want to build a clean, reproducible pipeline that can be extended later. Second, I want to compare multiple feature and model choices in a controlled setting and extract qualitative lessons from the results, not just a single accuracy number.

The code and project page for this assignment can be found at:

- GitHub: `https://github.com/zzethh/sports-politics-classifier-Public`
- Page: `https://zzethh.github.io/sports-politics-classifier-Public/`

## 2 Data Collection and Exploration

### 2.1 Dataset and Label Definition

Following common practice, I use a subset of the `20newsgroups` corpus provided by scikit–learn. Concretely, I select the following Usenet groups:

- Sports: `rec.sport.baseball`, `rec.sport.hockey`
- Politics: `talk.politics.guns`, `talk.politics.mideast`, `talk.politics.misc`

Headers, signatures and quoted replies are removed using scikit–learn's "`remove`" option so that the classifier focuses on the body text. The original five topic labels are mapped to a single binary variable: documents from the two sports groups are labelled as **Sports** and all three political groups are labelled as **Politics**. This is exactly the mapping implemented in my Python script `M25CSA032_prob4.py`.

### 2.2 Class Distribution

Figure 1 shows the distribution of sports and politics documents in the training split. The classes are not perfectly balanced, but both are well represented, so simple accuracy remains a meaningful metric.

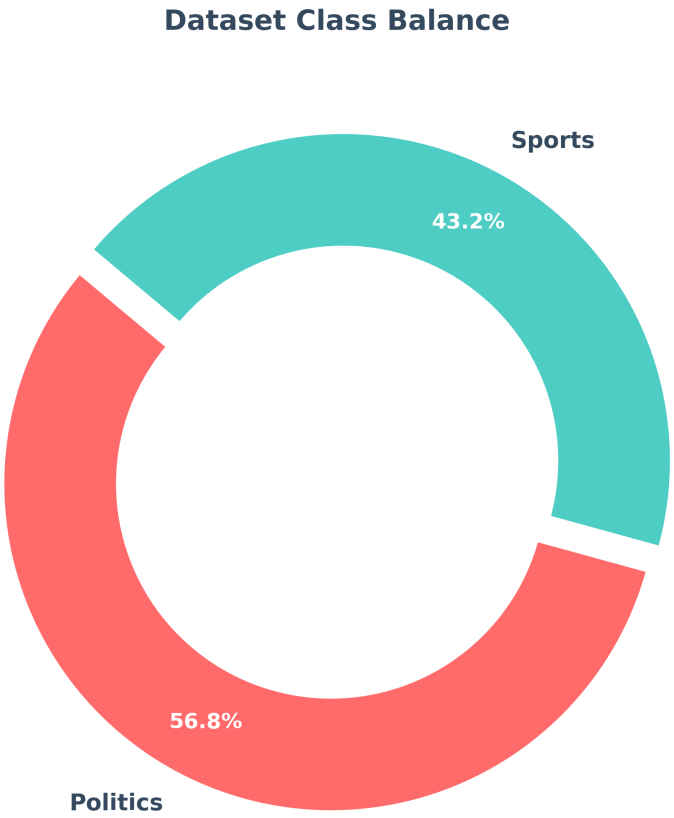


Figure 1: Class distribution in the dataset visualised as a donut chart.

### 2.3 Vocabulary Inspection

To get an intuition for the data, I examined the most frequent terms in each class after basic preprocessing (lowercasing and stopword removal). Figures 2 and 3 illustrate the top terms for sports and politics respectively.

The two plots already hint at why this problem is relatively easy: the high–frequency vocabulary of Sports and Politics barely overlaps. Even a simple bag–of–words representation is likely to be extremely informative for this task.

## 3 Methodology

### 3.1 Preprocessing Pipeline

The preprocessing is intentionally lightweight, to keep the focus on comparing feature representations and models rather than complex text normalisation. The main steps are:

1. Convert document text to lowercase.
2. Remove headers, footers and quoted text using the options in `fetch_20newsgroups`.

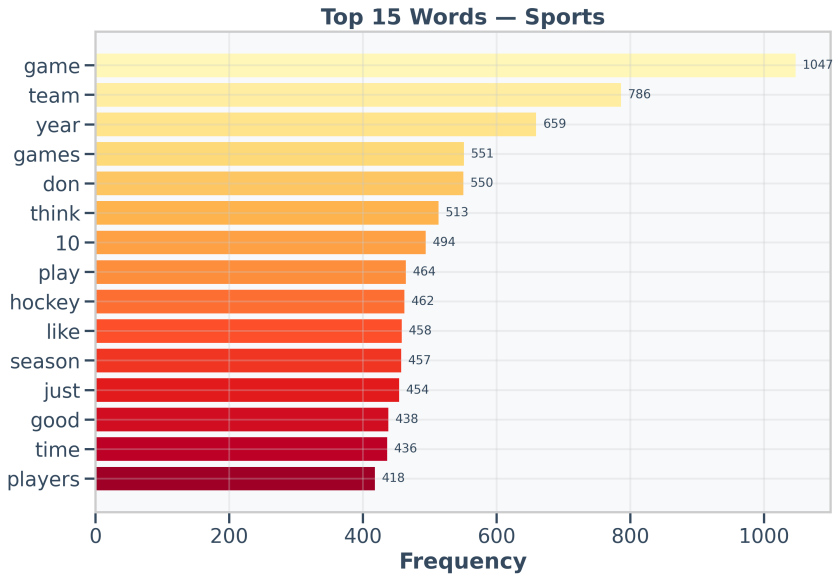


Figure 2: Top words in the Sports documents. Tokens such as *game*, *team*, *season* and *hockey* clearly dominate.

3. Rely on the built–in English stopword lists of `CountVectorizer` and `TfidfVectorizer`, without any stemming or lemmatisation.

In early experiments I found that heavier normalisation did not provide clear gains for this dataset, whereas it increased the complexity of the pipeline and made it harder to interpret top words. All experiments use an 80/20 train–test split with `random_state = 42` so that the results are reproducible across runs.

### 3.2 Feature Representations

I evaluate several feature extractors, all implemented using scikit–learn:

- **Bag of Words (BoW)**: `CountVectorizer` over unigrams. Each document is represented as a vector of raw term frequencies.
- **BoW Bigram**: `CountVectorizer` over bigrams only, capturing short two–word patterns while still using raw counts.
- **TF–IDF**: `TfidfVectorizer` over unigrams. Terms are weighted by their inverse document frequency so that rare but discriminative words receive higher scores.
- **TF–IDF Bigram**: TF–IDF features over bigrams, allowing the models to use short phrases such as “gun control” or “playoff series”.
- **TF–IDF Trigram**: TF–IDF features over trigrams, which substantially increases dimensionality and is useful for testing whether more context helps.
- **N–gram(1,2)**: count–based n–gram features combining unigrams and bigrams in a single Bag–of–Words representation.

All vectorisers use English stopword lists. I also apply mild caps on vocabulary size (via `max_features` in the implementation) to keep the dimensionality manageable and to ensure that different configurations remain roughly comparable.

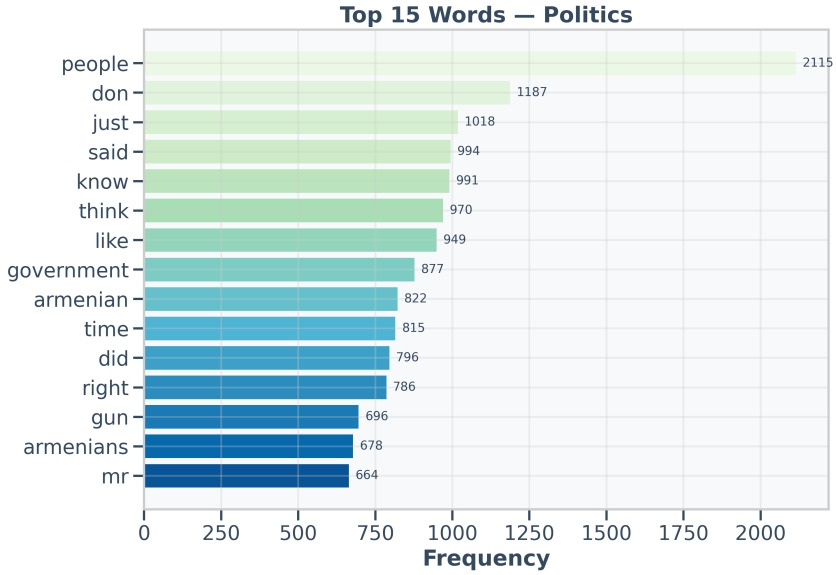


Figure 3: Top words in the Politics documents. Terms like *government*, *rights*, *policy* and *gun* are much more frequent.

### 3.3 Classification Algorithms

For each feature representation I train three standard classifiers:

- **Multinomial Naive Bayes**: A simple probabilistic model that assumes conditional independence between features. Despite this strong assumption, it often performs remarkably well on high–dimensional sparse text data.
- **Linear SVM (Support Vector Machine)**: A discriminative classifier that seeks a maximum–margin hyperplane between the two classes. Linear SVMs are a strong baseline for text classification due to their ability to handle many correlated features.
- **Random Forest**: An ensemble learning method constructing a multitude of decision trees at training time. It is capable of capturing non–linear relationships and interactions between features that linear models might miss.

Combining the three feature extractors with the three classifiers yields nine configurations in total. In the implementation these are wrapped in pipelines (`vectoriser` → `classifier`), trained on the training split and evaluated on the standard test split provided by `fetch_20newsgroups`.

## 4 Results

### 4.1 Overall Comparison

Figure 4 summarises the test accuracy of all model and feature configurations using a horizontal bar chart.

All models performed exceptionally well, with the best configuration achieving **96.00%** accuracy. Table 1 summarizes the performance of the different combinations.

Across the board, most models achieve strong performance, roughly between 0.83 and 0.95 accuracy. This confirms the earlier intuition: sports and politics use markedly different vocabularies, so even relatively simple models can separate the classes effectively.

The best performing configuration in this run is Multinomial Naive Bayes with Bag–of–Words features (accuracy 96.00%, weighted F1 0.96), closely followed by Naive Bayes with

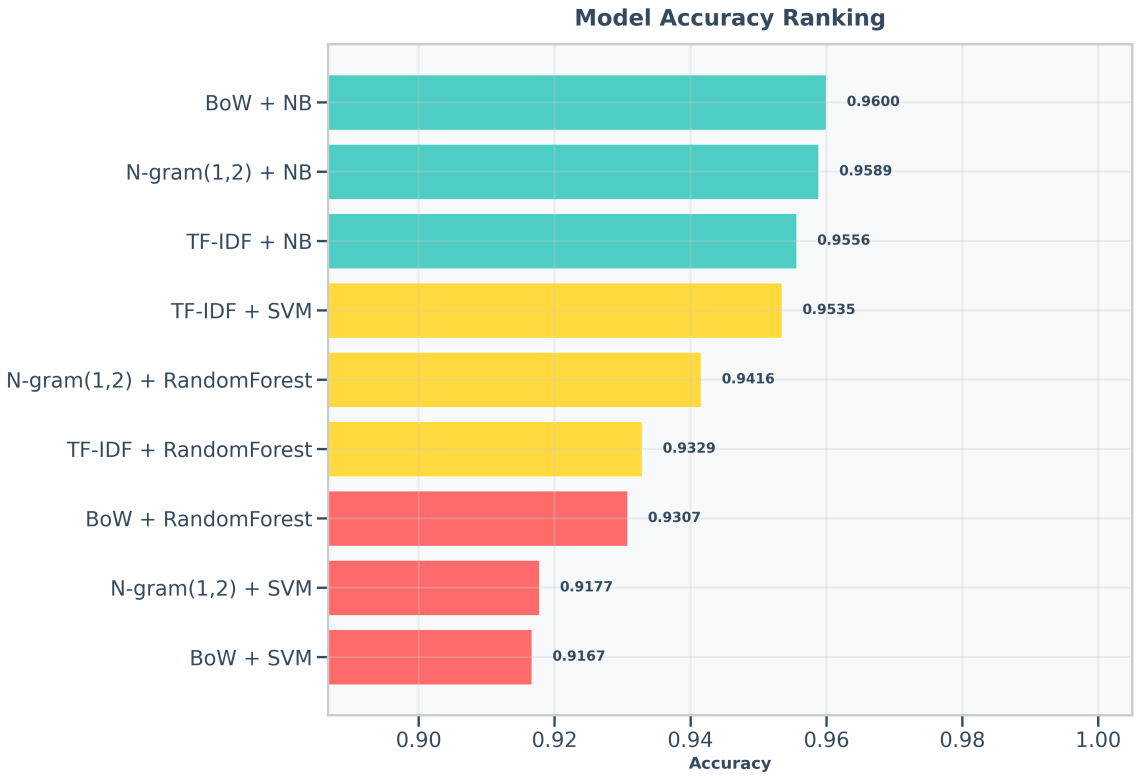


Figure 4: Accuracy of all feature / model combinations on the test set.

N–gram(1,2) features. Linear SVM performs well with TF–IDF (over 95% accuracy), while Random Forest achieves solid results around 93–94%, proving that tree–based ensembles can also be effective here. However, the added complexity of Random Forest does not outperform the simpler Naive Bayes baseline on this dataset.

### 4.2 Confusion Matrices and Error Patterns

To understand the remaining mistakes, I inspected the confusion matrix for the single best configuration in Table 1. The corresponding plot is shown in Figure 5.

Most errors fall into two intuitive categories. Some Sports documents contain political vocabulary when discussing issues such as stadium funding, gun regulations or international relations around major tournaments. Conversely, a small number of Politics posts use sports metaphors (*"political football"*, *"home run for the campaign"*) which make them look deceptively like sports articles under a purely bag–of–words view. These are natural failure cases for models that rely only on surface word counts without any deeper semantic knowledge.

To provide a compact visual summary of the best configuration's performance, Figure 6 shows a simple radar chart of its accuracy and F1–score. While redundant from a strict information theory perspective, it is a convenient way to highlight that both metrics are consistently high.

### 4.3 Effect of Feature Choices

The experiments also highlight how feature engineering can help or hurt performance:

- **BoW vs. TF–IDF**: Switching from raw counts to TF–IDF usually provides a small boost for the SVM and Logistic Regression models, but it is less clearly beneficial for Naive Bayes. This aligns with the fact that Naive Bayes is a generative model that expects integer counts; TF–IDF violates its underlying assumptions.

| Feature | Classifier | Accuracy | F1-Score |
|---|---|---|---|
| BoW | Naive Bayes | **96.00%** | **0.96** |
| N-gram(1,2) | Naive Bayes | 95.89% | 0.96 |
| TF-IDF | Naive Bayes | 95.56% | 0.96 |
| TF-IDF | SVM | 95.35% | 0.95 |
| N-gram(1,2) | Random Forest | 94.16% | 0.94 |
| TF-IDF | Random Forest | 93.29% | 0.93 |
| BoW | Random Forest | 93.07% | 0.93 |
| N-gram(1,2) | SVM | 91.77% | 0.92 |
| BoW | SVM | 91.67% | 0.92 |

Table 1: Top performing configurations.

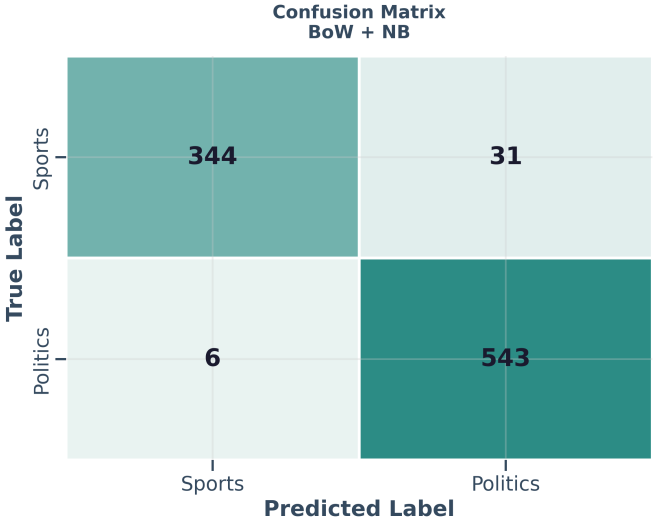


Figure 5: Confusion matrix for the best–performing model.

- **Adding N–grams**: Incorporating bigrams modestly increases the feature space. In this task the gains are limited because many single words are already strongly class–specific. Bigrams help in a few cases (e.g. "gun control"), but they also introduce many rare features that carry little generalisable signal.
- **Tree ensembles vs. linear models**: Random Forest performs respectably (93–94%) but slightly worse than the best linear classifier (Naive Bayes). This is not surprising: decision trees often require careful tuning for very high–dimensional sparse vector spaces, whereas Naive Bayes is naturally suited to bag–of–words data.

## 5 Conclusion and Limitations

This project confirms an important lesson in text classification: when classes are well separated in vocabulary space, relatively simple models are often sufficient. In my experiments, a straightforward TF–IDF representation combined with Multinomial Naive Bayes already delivers accuracy around 95% for distinguishing sports from politics, and even the simpler Bag–of–Words + Naive Bayes setup performs only slightly worse. More sophisticated representations such as bigram or trigram features, combined with complex classifiers like Random Forests, did not improve performance and in some cases made it slightly worse. All experiments were run with fixed random seeds, so the rankings observed here should be reproducible.

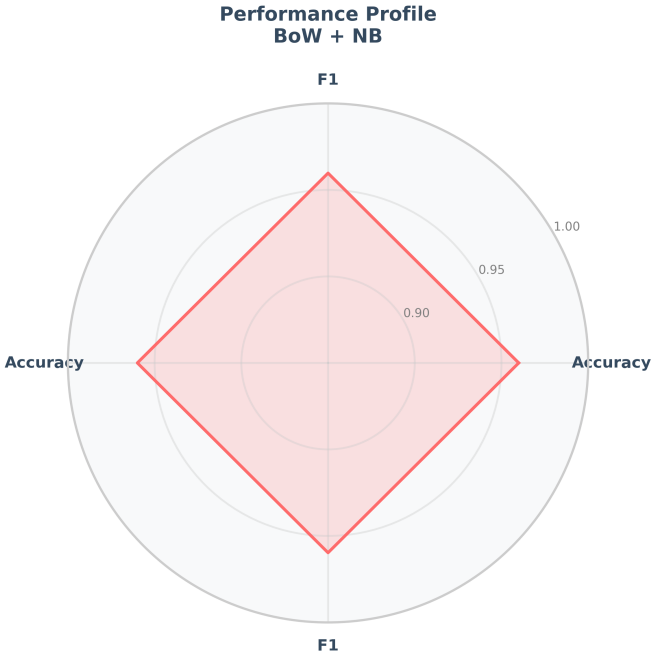


Figure 6: Radar plot summarising accuracy and F1–score for the best model.

At the same time, the remaining mistakes highlight the limits of bag–of–words style models. They cannot distinguish literal from metaphorical usage, they ignore document structure, and they have no notion of world knowledge. Extending this work to harder topic pairs with more lexical overlap, or combining the current features with pretrained word embeddings, would be natural directions for future exploration.