Problem 4: Sports or Politics Text Classification

Classical NLP Pipeline using Bag of Words, N-grams, and TF-IDF

Krish Jain
B23CM1019
Course : Natural Language Understanding

GitHub Repository: https://github.com/b23cm1019/NLU_Ass1
GitHub Page: https://b23cm1019.github.io/NLU_Ass1/

February 15, 2026

Abstract

This report presents an end-to-end binary text classification system that predicts whether a news article belongs to *Sport* or *Politics*. Articles were scraped from **The Guardian** using category-specific URL lists. After cleaning and preprocessing, three feature representation methods were implemented independently: Bag of Words (BoW), N-gram counts, and TF-IDF. Four classical machine learning models were evaluated: Multinomial Naive Bayes, Logistic Regression, Linear SVC, and SGD Classifier. The best test performance was obtained by BoW + Logistic Regression with accuracy 0.9969 and macro F1 0.9969. This report covers data collection, preprocessing rationale, feature engineering theory, model theory, quantitative comparison, interpretation of results, and limitations.

1 Problem Statement

The assignment requires a classifier that reads a text document and predicts one of two classes:

- **Politics**
- **Sport**

Required components include:

1. Data collection and dataset analysis.
2. Feature representation using BoW, N-grams, and TF-IDF.
3. Comparison of at least three ML techniques.
4. Quantitative evaluation and discussion.
5. Limitations of the system.

2 Data Collection from The Guardian

2.1 Source and Category Filtering

Data was collected from **The Guardian** (https://www.theguardian.com) using two input CSV files:

- `guardian_politics_links.csv`
- `guardian_sports_links.csv`

Each CSV contains URL paths pointing to category pages/articles. Category filtering was done at source level:

- Politics links came from Guardian "/politics/..." URLs.
- Sports links came from Guardian "/sport/..." URLs.

Thus, labels are category-grounded by origin.

2.2 Libraries and Frameworks Used for Data Collection

- **Python**: core scripting language.
- **pandas**: reading URL CSV files and storing scraped dataset.
- **requests**: downloading web pages via HTTP.
- **BeautifulSoup (bs4)**: parsing HTML and extracting title/body text.
- **concurrent.futures**: parallel scraping for faster collection.
- **pathlib**: robust path handling for outputs.

2.3 Scraping Procedure

The scraper (`scrape_guardian.py`) follows:

1. Read URL paths from CSV.
2. Convert relative paths to full The Guardian URLs.
3. Send HTTP request with browser-like user-agent.
4. Parse page and extract:
    - title (og:title or page title fallback)
    - article body paragraphs from main content selectors
5. Join paragraphs to a single document string.
6. Assign class label (`politics` or `sport`) based on source file.
7. Save raw scraped records to `data/raw/guardian_articles.csv`.

2.4 Raw-to-Final Dataset Description

Final modeling dataset statistics (after cleaning and balancing in preprocessing stage):

- **Number of classes**: 2
- **Class labels**: politics (0), sport (1)
- **Total samples**: 2180 rows/documents
- **Samples per class**: politics = 1090, sport = 1090

  Each row/document contains:

- `label`: class name
- `text`: full article text
- (plus metadata in raw stage such as URL/title/word count)

3 Stage 2: Text Preprocessing

After scraping, raw text is not directly suitable for feature extraction. The preprocessing stage (`prepare_features.py`) transforms noisy web text into normalized, learnable inputs.

3.1 Why Preprocessing is Required

Web text includes markup artifacts, links, punctuation noise, and inconsistent casing. Without normalization:

- vocabulary size explodes unnecessarily,
- same word appears in multiple forms (e.g., "Vote", "vote"),
- models overfit on noise instead of topic content.

3.2 Preprocessing Steps Implemented

1. **Lowercasing**: convert all text to lowercase.
2. **URL removal**: remove embedded links.
3. **Email-like token removal**: remove noisy contact patterns.
4. **Non-alphabetic filtering**: keep only alphabetic tokens/spaces.
5. **Whitespace normalization**: collapse repeated spaces.
6. **Duplicate control**: remove repeated URL/text entries.
7. **Length filtering**: keep documents with at least 30 tokens to avoid extremely short/noisy pages.
8. **Label normalization**: map

$$\texttt{politics} \to 0, \quad \texttt{sport} \to 1$$

3.3 Train/Validation/Test Split

A stratified split preserves class ratio in each partition:

- Train: 1526
- Validation: 327
- Test: 327

This allows fair model selection and reliable final evaluation.

4 Feature Representation: Theory + Implementation

Three feature representations were implemented **independently** to compare their strengths and weaknesses.

4.1 Bag of Words (BoW)

Theory: BoW represents a document as term-frequency counts of vocabulary words (order ignored). **Implementation:** unigram CountVectorizer (`ngram_range=(1,1)`), English stopword filtering.

Capabilities:

- simple and effective baseline,
- strong for topic classification with discriminative keywords.

Limitations:

- ignores word order/context,
- high frequency common words can dominate if not controlled.

4.2 N-gram Counts (2–3 grams)

Theory: N-grams encode local token sequence patterns. Bi-grams/trigrams capture short phrases like "prime minister", "world cup".

Implementation: CountVectorizer with `ngram_range=(2,3)`.

Capabilities:

- captures phrase-level semantics better than pure unigrams,
- useful when class cues are multi-word expressions.

Limitations:

- sparse and very high-dimensional space,
- many rare n-grams reduce generalization if data is limited.

4.3 TF-IDF (Unigram)

Theory: TF-IDF downweights common terms and upweights class-informative rare terms:

$$\text{tfidf}(t, d) = \text{tf}(t, d) \cdot \log \frac{N}{df(t)}$$

where $df(t)$ is document frequency.

Implementation: unigram TF-IDF with sublinear TF.

Capabilities:

- strong signal-to-noise filtering,
- often robust across news domains.

Limitations:

- still context-limited (bag assumption),
- semantics beyond lexical overlap are not modeled.

4.4 Feature Space Sizes

- BoW vocabulary: 22,575
- N-grams (2–3) vocabulary: 88,630
- TF-IDF vocabulary: 22,575

The n-gram space is much larger, increasing sparsity and computational burden.

5 ML Techniques: Why Chosen and How They Work

Four classical models were used (assignment requires at least three):

5.1 Multinomial Naive Bayes

How it works: probabilistic generative classifier with conditional independence assumption over token features. **Why suitable:** fast baseline for count-based text features. **Known weakness:** independence assumption is often too strong for nuanced text.

5.2 Logistic Regression

How it works: discriminative linear model minimizing logistic loss; outputs class probabilities. **Why suitable:** strong performance on sparse high-dimensional text, stable optimization, interpretable weights.

5.3 Linear SVC

How it works: maximum-margin linear classifier. **Why suitable:** robust in sparse spaces, often competitive/high-performing in document classification.

5.4 SGD Classifier (log-loss)

How it works: incremental optimization using stochastic gradient descent for linear decision boundaries. **Why suitable:** scalable, efficient on large sparse matrices, often near-LogReg performance with lower training cost.

6 Quantitative Comparison

6.1 Full Results

Feature	Model	Val F1	Test Acc	Test F1
bow	LogisticRegression	0.9939	0.9969	0.9969
tfidf	SGDClassifier	1.0000	0.9969	0.9969
tfidf	LinearSVC	1.0000	0.9939	0.9939
bow	LinearSVC	0.9969	0.9939	0.9939
bow	SGDClassifier	0.9969	0.9939	0.9939
tfidf	LogisticRegression	1.0000	0.9939	0.9939
ngrams_2_3	SGDClassifier	0.9969	0.9908	0.9908
ngrams_2_3	LinearSVC	0.9939	0.9908	0.9908
ngrams_2_3	LogisticRegression	0.9908	0.9878	0.9878
ngrams_2_3	MultinomialNB	0.9878	0.9878	0.9878
bow	MultinomialNB	0.9969	0.9878	0.9878
tfidf	MultinomialNB	0.9847	0.9847	0.9847

6.2 Best Configuration

- **Feature:** BoW
- **Model:** Logistic Regression
- **Test Accuracy:** 0.9969
- **Test Macro F1:** 0.9969
- **Confusion Matrix:** [[163, 1], [0, 163]]

7 Detailed Interpretation and Analysis

7.1 Why BoW + Logistic Regression Became Best

This dataset is topic-focused and lexically separable. Sports and politics contain strong topic words that unigram counts already capture well. Logistic Regression then finds a robust linear boundary with good regularization behavior in sparse space. Hence high precision and recall for both classes are expected.

7.2 Why TF-IDF is Also Very Strong

TF-IDF + SGD also reached near-identical top performance. This is consistent with theory: TF-IDF suppresses very common terms and emphasizes class-informative terms. In news classification, this usually improves signal quality. The small difference from top BoW result is expected because both representations are already very informative for this binary topic setup.

7.3 Why N-grams(2–3) Underperformed Slightly

Although n-grams add phrase context, the feature space increased to 88,630 dimensions, causing:

- higher sparsity,
- many rare phrase features,
- weaker statistical support for each phrase feature.

Given moderate sample size, this can reduce generalization slightly compared to unigram-based representations.

7.4 Model-wise Behavior

- **Linear discriminative models** (LogReg, LinearSVC, SGD) consistently outperformed Naive Bayes.
- **MultinomialNB** remained competitive but lower, likely because independence assumption is restrictive.
- **SGD** offered near-top quality with efficient optimization, validating it as a practical NLP model.

7.5 Validation vs Test Pattern

Some runs show perfect validation but slightly lower test performance. This is not abnormal; it indicates split-level variance and confirms why final conclusions should be based on test metrics, not validation alone.

8 System Demonstration Feature

Beyond training comparison, the system supports interactive prediction:

- Input modes: direct text or document file.
- User selects feature representation (`bow`, `ngrams_2_3`, `tfidf`).
- User selects trained model (`MultinomialNB`, `LogisticRegression`, `LinearSVC`, `SGDClassifier`).
- Output: predicted class (`politics`/`sport`) and probabilities (if model supports `predict_proba`).

This satisfies assignment-level usability for testing custom documents.

9 Limitations and Failure Cases

9.1 Feature Representation Limitations

- **BoW/TF-IDF:** no deep context, no syntax/discourse understanding.
- **N-grams:** sparse and data-hungry; brittle to phrasing variation.

9.2 Classical ML Limitations

- Linear boundaries may miss non-linear semantics.
- Limited handling of irony, sarcasm, implicit topic shifts.
- Sensitive to domain shift (other publishers, writing styles, time periods).

9.3 Likely Difficult Cases

- Mixed-topic articles (e.g., sports governance/political decisions in sports bodies).
- Very short or headline-only texts.
- Opinion/editorial pieces with indirect wording.

10 Conclusion

An end-to-end classical NLP system for Sports vs Politics classification was successfully built and evaluated. The pipeline demonstrates strong engineering practice:

- source-specific data collection from The Guardian,
- structured preprocessing,
- theoretically grounded feature comparisons,
- multi-model evaluation with quantitative evidence.

Best performance came from BoW + Logistic Regression (test macro F1 = 0.9969), while TF-IDF + SGD was essentially equivalent. The results confirm that classical ML methods can achieve excellent performance on this NLP task without deep learning, provided that preprocessing and feature engineering are done carefully.

Reproducibility Commands

```
python scrape_guardian.py
python prepare_features.py --input data/raw/guardian_articles_balanced.csv
python train_models.py
python predict_text.py --feature tfidf --model LogisticRegression --text "..."
```