Data

- Dataset: 20 Newsgroups (CMU subset).
- Split: 50% / 50% per class (stratified).
- Examples: Train = 9,998, Test = 9,999.
- Vocabulary (training): 105,428 unique tokens.
- Preprocessing: lowercase, regex alphanumeric tokenization, optional small stopword removal.

Method

- Tokenizer: simple regex-based alphanumeric token extraction; stopwords may be removed in code.
- Model: Multinomial Naive Bayes implemented from first principles.
  - Class prior: $P(y) = N_y / N$.
  - Likelihood with Laplace smoothing:
    $P(w \mid y) = (\text{count}(w, y) + \alpha) / (\text{total\_tokens}_y + \alpha \cdot V)$
    where $\alpha = 1.0$ and V is the vocabulary size.
- Prediction: sum $\log(P(y)) + \Sigma \log(P(w \mid y))$ over tokens to avoid underflow; predict argmax over classes.
- Implementation detail: token counts and log-probabilities are computed directly in Python.

Experimental details

- Split reproducibility: seed = 42.

- Training aggregates per-class token counts, computes log-likelihoods, and caches unseen-token log-probabilities.

- Evaluation metrics: Accuracy, Macro F1, Micro F1, per-class precision/recall/F1, and confusion matrix.

Results (main figures)

- Accuracy: 0.8702

- Macro F1: 0.8661

- Micro F1: 0.8702

- Train examples: 9,998 — Test examples: 9,999

- Vocabulary size: 105,428

Sample per-class performance (selected examples)

- rec.sport.hockey — F1 ≈ 0.967

- sci.crypt — F1 ≈ 0.960

- comp.graphics — F1 ≈ 0.794

- talk.religion.misc — F1 ≈ 0.660

(Full per-class metrics and confusion matrix are available in the project reports/ directory.)

7. Observations & short error analysis

- The classifier does particularly well on narrowly-scoped topics (sports, some science categories) where topic-specific vocabulary strongly signals the class.

- Errors concentrate among semantically close groups (for example, several comp.* classes and some political/religion subcategories). Overlap in technical terms and common vocabulary across these topics increases misclassification rates.

- Long-tailed tokens and rare proper nouns can skew likelihoods for individual classes. Possible improvements (not required for this assignment) include vocabulary pruning, stemming, or TF–IDF weighting prior to classification.

Reproducibility & how to run

Create the split (one of the options below):

  - If you have the tarball:
    python bin/prepare_data.py --tar_gz /path/to/20_newsgroups.tar.gz --out data --seed 42
  - If you already extracted the dataset to a folder:
    python bin/prepare_data.py --extracted_root /path/to/20_newsgroups --out data --seed 42
  - Or run the tolerant splitter:
    python auto_split.py

Train and evaluate:
python main.py --data_root data --alpha 1.0 --report reports/report.md

There is a convenience script to run both steps:

./run_all.sh