

Sentiment Analysis

Waleed Alfaifi

1. Deeper Preprocessing Steps

- Lowercasing: normalize text to reduce feature sparsity (e.g., 'Good' vs 'good').
- Remove URLs, mentions, hashtags, HTML: drop non-semantic noise that skews features.
- Keep only letters: strip numbers/symbols when they add little sentiment value in this domain.
- Tokenization: split text into tokens for downstream filtering.
- Stopwords removal: discard frequent non-informative words (e.g., 'the', 'is').
- Stemming (Porter): conflate inflected forms (e.g., 'running' → 'run') to generalize features.
- Store both original and clean_text columns to preserve traceability.

2. Feature Representation (TF-IDF)

- I use TF-IDF with n-grams (1,2) to capture single words and short phrases (e.g., 'not good').
- min_df=5: drop very rare tokens to reduce noise; max_df=0.9: drop overly common tokens.
- The result is a high-dimensional sparse matrix ideal for linear classifiers.

3. Why Logistic Regression & Linear SVM?

- Both are strong baselines for text classification in high-dimensional sparse spaces.
- Efficient training; robust generalization with L2 regularization.
- Logistic Regression optimizes log-loss (probabilistic outputs).
- Linear SVM optimizes hinge-loss; often edges out LR on sparse TF-IDF features.
- class_weight='balanced' mitigates label imbalance without resampling.

4. Test Results (Best Model: Linear SVM)

Class	Precision	Recall	F1-score	Support
negative	0.87	0.85	0.86	1377
neutral	0.58	0.62	0.60	465
positive	0.68	0.68	0.68	354
accuracy			0.77	2196
macro avg	0.71	0.71	0.71	2196
weighted avg	0.78	0.77	0.77	2196

Confusion Matrix (Test):

