For this exercise, three preprocessing strategies were tested: using raw, unprocessed text; applying basic NLTK-based cleaning; and employing more aggressive spaCy processing. Classification was performed using two models: Naive Bayes and Logistic Regression. Text was vectorized using the TF-IDF method, and SelectKBest was used to select the most informative features. Due to the extremely small dataset of five samples, the results are more for illustration. However, they reveal essential patterns in text classification workflows.

Preprocessing Impact on Text Analysis

The first approach, using raw text without any preprocessing, retained all original casing, punctuation, stopwords, and contractions. This introduced a high level of noise and led to poor performance for Naive Bayes, which scored an accuracy of zero. The model struggled to distinguish sentiment due to the overwhelming presence of irrelevant tokens. Logistic Regression, however, achieved perfect accuracy, though this result is likely due to the very small test size and does not indicate general reliability. The second strategy, using NLTK preprocessing, involved converting text to lowercase, removing common stopwords, and filtering tokens. This provided a balanced pipeline that preserved essential words while reducing noise. Both Naive Bayes and Logistic Regression achieved perfect accuracy under this setup, suggesting that meaningful tokens like “great,” “boring,” and “loved” played a key role in distinguishing sentiment. The third strategy involved spaCy preprocessing, which was more aggressive. It removed stopwords and punctuation, applied lemmatization, and followed linguistic rules for token filtering. Logistic Regression continued to perform well with this approach, again achieving perfect accuracy. However, Naive Bayes failed to classify correctly, scoring zero accuracy. This suggests that spaCy may have went too far when cleaning the text, removing short but semantically important tokens necessary for count-based models like Naive Bayes.

Naive Bayes proved to be highly sensitive to token availability and quantity, whereas Logistic Regression demonstrated greater robustness to various preprocessing levels. These findings suggest that preprocessing decisions should be closely aligned with the intended model. A more aggressive pipeline may benefit certain algorithms, like Logistic Regression, that are capable of handling sparse or weighted feature sets. In contrast, probabilistic models like Naive Bayes require a richer set of token occurrences to perform well.

Feature Extraction Findings

|  |  |  |
| --- | --- | --- |
| Preprocessing | Naïve Bayes Accuracy | Logistic Regression Accuracy |
| Raw Text | 0.00 | 1.00 |
| NLTK Processed | 1.00 | 1.00 |
| spaCy Processed | 0.00 | 1.00 |

The exercise utiulized two main vectorization techniques: CountVectorizer and TfidfVectorizer. CountVectorizer simply tallies the frequency of each token, making it more suitable for models that rely on token counts, such as Naive Bayes. TfidfVectorizer, on the other hand, weights tokens based on their inverse document frequency, downweighting common words and amplifying infrequent but potentially more informative ones. This approach worked particularly well with Logistic Regression, which can leverage these weighted features more effectively.

To enhance feature quality, SelectKBest was applied using a chi-squared statistical test to identify the most discriminative features. When k was set to 5, the top features included emotionally polarized words such as “boring,” “fantastic,” “hated,” “loved,” and “weak.” These words showed strong correlations with sentiment labels and thus improved model performance. However, increasing the number of selected features began to introduce more neutral terms like “characters” and “film,” which are less directly tied to sentiment. This dilution can reduce classification accuracy, especially in models that are sensitive to irrelevant features. Overall, these results support the idea that a smaller, well-chosen feature set can outperform a larger, noisier one. The combination of TF-IDF vectorization and SelectKBest feature selection offers a strong baseline for future text classification tasks.

Classification Performance

Logistic Regression consistently outperformed Naive Bayes across all preprocessing strategies. When no preprocessing was applied, Naive Bayes achieved zero accuracy, whereas Logistic Regression scored perfectly. Under NLTK preprocessing, both models performed well. However, spaCy preprocessing again uncovered the limitations of Naive Bayes. The prominent difference in performance shows how the behavior of a model is influenced by the back-and-forth of preprocessing and feature representation. Logistic Regression thrives even with aggressively filtered text due to its ability to handle sparse, weighted input. On the other hand, Naive Bayes relies heavily on a high number of distinct tokens. Ultimately performing poorly when these are significantly reduced. A necessary point is that all of these accuracy figures stem from a single sample test set. Which makes them unreliable for definitive conclusions. These results should be viewed as indicative trends, not statistically significant outcomes.

Challenges Encountered

With only five labeled movie reviews and a single-sample test set, accuracy results varied wildly (0% to 100%) depending on the preprocessing and model configuration. To address this, future work should aim for at least 50 to 100 labeled samples and incorporate more rigorous evaluation methods like Stratified K-Fold cross-validation. Another significant challenge was finding the right balance in preprocessing. Over-cleaning text, as observed with spaCy, removed useful sentiment-bearing tokens, particularly harming Naive Bayes. Conversely, no cleaning also led to performance issues in both models. This suggests the need for a controlled and balanced preprocessing pipeline, perhaps one based on NLTK with custom, task-specific filters. We also observed overfitting during evaluation. Some models achieved perfect accuracy, which is unlikely to reflect real-world performance given the tiny dataset. Addressing this will require larger datasets and more comprehensive evaluation metrics, including confusion matrices and F1-scores, rather than relying solely on accuracy.

Conclusion

This exercise clearly shows that preprocessing, feature selection, and model choice are all deeply connected in sentiment analysis. How you preprocess text significantly impacts the quality of your features, which then directly influences how well your model performs. It was revealed that Logistic Regression was more resilient to various preprocessing strategies. In contrast, Naive Bayes needed a richer and properly cleaned set of tokens to perform well. When combined with TF-IDF, feature selection helped the models focus on the most informative tokens, boosting overall performance. To verify these findings and gain more concrete insight, future research will need larger datasets and more robust validation methods.