Report: Naive Bayes Text Classification

1. Tokenizer

For the tokenizer implementation, I went with a simple approach, focusing on

- **Lowercasing**: All text was converted to lowercase to ensure uniformity across the dataset.
- **Punctuation Removal**: I removed all punctuation marks using Python’s `string.punctuation`. The rationale behind this was to simplify the input by eliminating non-essential symbols.
- **Tokenization (Whitespace Splitting)**: Once the text was cleaned, it was split into individual words based on whitespace.

Why I Didn't Use Stopword Removal:
Though I initially considered removing stopwords (common words like "the", "is", etc.), I found that stopword removal didn't significantly impact performance in my experiments. Since stopwords could occasionally carry sentiment (e.g., "not" in "not good"), so I decided to retain all words in the text.

---

2. Results

I evaluated my Naive Bayes classifier on the validation set using four key metrics: accuracy, precision, recall, and F1 score.

Here are the metrics calculated for the validation set with an alpha (Laplace smoothing parameter) value of 0.1 and also tried a loop of many values but found 0.1 to be the best:

- **Accuracy**: 85.26%
- **Precision**: 85.44%
- **Recall**: 84.99%
- **F1 Score**: 85.22%

These metrics show that the model did perform kind of well on sentiment classification, over 85% accuracy. The F1 score of 85.22% which says that we have a good balance between precision and recall. Precision is a bit higher than the recall value, which I believe it translates to the fact that model is more cautious about predicting the positive class. However, it still maintains good recall.

Interpretation of the Results:

- The high accuracy means the model predicts good sentiment for most reviews.

- We also have a balance between precision and recall indicating that the model performs well across both classes ("positive" and "negative") without heavily favoring one over the other.

---

3. Naive Bayes

Training Procedure

During training, the Naive Bayes model estimates two key probabilities:

1. **Prior Probabilities**: The likelihood of each class (positive or negative) in the training set.
    - These probabilities were computed by counting the number of documents in each class and dividing by the total number of documents.
2. **Conditional Probabilities**: The likelihood of a word occurring given a particular class.
    - This is achieved by counting the occurrences of each word in the documents belonging to a particular class, while also applying Laplace smoothing with an alpha value of 0.1. The smoothing ensures that we avoid issues with zero probabilities for unseen words in the test data.

Laplace smoothing was necessary because there were many words in the vocabulary that didn’t appear in all classes. Without smoothing, the absence of these words in some classes would lead to zero probabilities and cause problems during inference.

Inference Procedure

For inference, I used the following steps:

1. **Logarithmic Probabilities**: To avoid numerical underflow when multiplying many small probabilities, I used the logarithms of the prior and conditional probabilities. This to prevent values from becoming too small to represent accurately.
2. **Prediction**: For each document, the model calculates the total log probability for both the positive and negative classes. The way I do this is by summing the log prior probability and the log conditional probabilities for each word in the document.
    - Words present in the document contribute the log probability of being present given the class.
    - Words absent from the document contribute the log probability of being absent given the class.
3. **Class Assignment**: The class with the highest total log probability is selected as the predicted class for the document.

I believe the approach is pretty good and robust and also efficient.

Conclusion

In this assignment, I successfully implemented a Naive Bayes text classifier for sentiment analysis using a bag-of-words binary feature representation. The model achieved strong results on the validation set, with an accuracy of over 85%. I found that using Laplace smoothing with an alpha value of 0.1 improved the stability of the model, preventing issues with zero probabilities.

While the model performs well, future improvements could include experimenting with different feature representations (e.g., TF-IDF or continuous features) or exploring more sophisticated models like logistic regression or deep learning approaches. However, given the simplicity and effectiveness of Naive Bayes, it remains a solid baseline for text classification tasks.