Naive Bayes Sentiment Classifier

This project implements a Naive Bayes Classifier to classify the sentiment of tweets as Positive, Negative, or Neutral. The classifier is trained on labeled tweets using the Bag-of-Words (BOW) model, and it supports Laplace smoothing to handle unseen words. It can work with n-gram models for improved accuracy.

My Output

```
(base) lizhuoyang@MacBookPro PA3ProvidedCode % python SentimentNaiveBayes.py
(base) lizhuoyang@MacBookPro PA3ProvidedCode % python SentimentNaiveBayes.py
Total Sentences correctly:  775
Predicted correctly:  673
Accuracy: 86.83871%
```

How to Avoid "Divide by Zero" Warning

Add 1 to the count of each word in the training data, ensuring that no word has a probability of 0.

Project Structure

```
./data-sentiment/
    ├── train/
    │   ├── Positive.txt
    │   ├── Negative.txt
    │   └── Neutral.txt
    └── test/
        ├── Positive.txt
        ├── Negative.txt
        └── Neutral.txt

SentimentNaiveBayes.py    # Main Python script
classifier.pkl            # Serialized trained classifier (saved after
training)
README.md                 # Documentation (this file)
```

How It Works

This project uses the Naive Bayes algorithm to classify sentences based on their sentiment. Each sentence is processed using the following steps:

1. Training Phase:

- The classifier reads training data from labeled files (Positive, Neutral, Negative).
- It builds a vocabulary from the training data and computes:
- Prior probabilities for each class (P(Class)).
- Conditional probabilities of words given each class (P(Word | Class)).

- Laplace smoothing is applied to prevent divide-by-zero errors for unseen words.
- The trained classifier is serialized and saved as ***classifier.pkl***.

2. Testing Phase:

- The classifier loads the trained model.
- It uses the log of probabilities to classify each test sentence into Positive, Neutral, or Negative.
- The program calculates and prints the accuracy of predictions on the test set.

2. Testing Phase:

- The classifier loads the trained model.
- It uses the log of probabilities to classify each test sentence into Positive, Neutral, or Negative.
- The program calculates and prints the accuracy of predictions on the test set.

How to Run

Training the Classifier

1. Open *SentimentNaiveBayes.py* and ensure the TASK variable is set to 'train':

```
TASK = 'train'
```

2. Run the script in the terminal to train the classifier and save it to classifier.pkl:

```
python SentimentNaiveBayes.py
```

Testing the Classifier

1. After training, set the TASK variable to 'test':

```
TASK = 'test'
```

2. Run the script to evaluate the classifier's performance on the test data:

```
python SentimentNaiveBayes.py
```