TCSS 456 Assignment 2

NOTE: Be sure to adhere to the University's **Policy on Academic Integrity** as discussed in class. Programming assignments are to be written individually and submitted programs must be the result of your own efforts. Any suspicion of academic integrity violation will be dealt with accordingly

Assignment Details:

For this assignment, we will implement a Naïve Bayes Classifier to perform Sentiment Analysis on Movie Reviews. Given a movie review you will decide if it has a positive sentiment or a negative one. To summarize you will:

- Train a naive bayes model on a sentiment analysis task
- Test using your model
- Compute ratios of positive words to negative words
- Do some error analysis
- Predict on your own movie reviews

Step 1: Process the Data

Write a method called `process_data()` that does the following for you:

- **Remove noise**: You will first want to remove noise from your data -- that is, remove words that don't tell you much about the content. These include all common words like 'I, you, are, is, etc...' that would not give us enough information on the sentiment. A stopwords.txt document is provided with list of all the stopwords.

  Note: If you are planning on implementing your assignment in Python. You can download stopwords using NLTK library.

```
import NLTK
from nltk.corpus import stopwords

nltk.download('stopwords')
print(stopwords.words('english'))
```

- You also want to remove all the punctuation from your data. The reason for doing this is because we want to treat words with or without the punctuation as the same word, instead of treating "happy", "happy?", "happy!", "happy," and "happy." as different words.

Example for `process_data()`:

```
review = "This is a great movie !"
print(process_data(review))

[great, movie]
```

Step 2: Create a Dictionary

Write a method called `count_reviews()` that takes a list of movie reviews as input parameter, cleans all of them up, and returns a dictionary.

- The key in the dictionary is a tuple containing the word and its class label, e.g. ("happy",1).

  Note: positive words are labeled 1, and negative words are labeled 0

- The value the number of times this word appears in the given collection of reviews (an integer).

Example for `count_reviews()`:

Given a list of reviews `["i am rather excited", "you are rather happy"]` in the, the method will return a dictionary that contains the following key-value pairs:

```
{("rather", 1): 2 ("happy", 1): 1 ("excited", 1): 1}
```

Step 3: Train your Naïve Bayes Model

In class we learned about the Naïve Bayes classifier for text classification. The Naïve Bayes classifier takes a short time to train and has a short prediction time.

So how do you train a Naive Bayes classifier?

- The first part of training a naive bayes classifier is to identify the number of classes that you have.
- Prior Probability: You will create a probability for each class. $P(D_{pos})$ is the probability that the document is positive. $P(D_{neg})$ is the probability that the document is negative. Use the formulas as follows and store the values in a dictionary:

$$P(D_{pos}) = \frac{D_{pos}}{D}$$

$$P(D_{neg}) = \frac{D_{neg}}{D}$$

  Where D is the total number of documents, or reviews in this case, D_{pos} is the total number of positive reviews and D_{neg} is the total number of negative reviews.
- Positive and Negative Probability of a Word: To compute the positive probability and the negative probability for a specific word in the vocabulary, we'll use the following inputs:
  - $freq_{pos}$ and $freq_{neg}$ are the frequencies of that specific word in the positive or negative class. In other words, the positive frequency of a word is the number of times the word is counted with the label of 1.

- $Npos$ and N_{neg} are the total number of positive and negative words for all documents (for all reviews), respectively.
- V is the number of unique words in the entire set of documents, for all classes, whether positive or negative.

We'll use these to compute the positive and negative probability for a specific word using this formula:

$$P(W_{pos}) = \frac{freq_{pos} + 1}{N_{pos} + V}$$

$$P(W_{neg}) = \frac{freq_{neg} + 1}{N_{neg} + V}$$

Note: We are applying Add-1 or Laplace smoothing.

Now that you know how to train the Naïve Bayes Classifier, we will compute the `logprior` and `loglikelihood`. Write a method called `train_naiveBayes()` that takes the dictionary (from step 2) and reviews as input parameters and returns `logprior` and dictionary containing `loglikelihoods` for each word.

- Calculate the `logprior = log(`$P(D_{pos})$`) - log(`$P(D_{neg})$`)`
- Create a dictionary to store the `loglikehoods` for each word. The key is the word, the value is the `loglikelihood` of that word.
  `loglikelihood(W) =` $\log\left(\frac{\text{P(Wpos)}}{\text{P(Wneg)}}\right)$

Step 4: Test your Naïve Bayes

Now that we have the `logprior` and `loglikelihood (dictionary)`, we can test the naive bayes classifier by making predictions on some reviews!

- Write a method called `predict_naiveBayes()` to make predictions on a review:
  - The method takes in the `review`, `logprior`, `loglikelihood` as input parameters
    - Note: Use the `logprior`, `loglikelihood dictionary` calculated from step 3. Additionally, review needs to be cleaned i.e. use `process_data()` method.
  - It returns the probability that the review belongs to the positive or negative class.
  - For each review, sum up loglikelihoods of each word in the review.
  - Also add the `logprior` to this sum to get the predicted sentiment of that review.

$$p = logprior + \sum_{i}^{N}(loglikelihood_i)$$

```
test_review = "Great movie !"
p = predict_naiveBayes(test_review, logprior, loglikehood)
print('The expected output is', p)
The expected output is 1.5740278623499175
```

- Write a method called `test_naiveBayes()` to check the accuracy of your predictions:
    - The method takes in your `test_reviews` (test dataset of movie reviews, `logprior`, and `loglikelihood` (dictionary) as input parameters.
        - Note: use `predict_naiveBayes()` to make predictions for each review in `test_reviews`.
    - It returns the accuracy of your classifier.
        - Note: if the `predict_naiveBayes() > 0 assign label 1 (for positive; else assign label 0 (negative)`

```
Accuracy = (# of tweets classified correctly)/ (total # of tweets), or

Accuracy = 1 – error, where error = average of the absolute values of the
differences between the predicted label and the actual label
```

```
print('Naïve Bayes accuracy=0.4%f', test_naiveBayes(test_reviews, logprior,
loglikehood)

Naive Bayes accuracy = 0.9940
```

Step 5: Error Analysis

Write a method called `error_analyis()` that will print an error table of misclassified reviews to an output file `error_analysis.txt`

Sample Output:

Truth	Predicted	Review
0	1	The movie itself was to me a huge disappointment.
1	0	I really enjoyed this film.

Step 6: Predict sentiments for Lion King Movie Review

Predict sentiments for the Lion King Movie Reviews. The text file has been provided. Print the review and the sentiment (use `predict_naiveBayes()`) to a text file `LionKing_Output.txt`

Data Files:

The following data files have already been given to you:

- `pos` folder with positive reviews
- `neg` folder with negative reviews
- `stopwords.txt` file
- `LionKing_MovieReviews.txt` file

This dataset is collected from https://www.cs.cornell.edu/people/pabo/movie-review-data/ as a text file and converted into a CSV file. A Readme document is provided with the dataset to better understand the features.

Note:

- Split sentences into ***Training*** **(80%)** & ***Testing (20%)*** set. Example, if there are overall 1000 sentences in the text file, then use 800 sentences for ***training*** *and* rest 200 sentences for ***testing***.
- Lastly, please free to change method headers i.e. method parameters and return value to match your coding logic. Please feel free to add additional helper methods.

Submission Guidelines:

Submit your files on Canvas using the Programming Assignment 2 submission Link. **You will submit a zip file containing:**

- Turn in your source code: `Naïve_SA.java/ Naïve_SA.py`
- The following output files:
  - `error_analysis.txt`
  - `LionKing_Output.txt`