CS114 Homework 3

Student Name: Lizzie Liang

Table of content

Introduction of the implementation

The initial features are not just *great*, *poor*, and *long*. The model uses all words except stop words and punctuations from the training set. The method _remove_stopwords() was used to remove stop words and punctuations. The running time for including all words as features is around 5 minutes.

The initial evaluation metrics using all words are shown as following. The overall accuracy is at 78%.

Evaluation metrics for all words except for stop words and punctuations:

```
Number of features:  45,483

Positive:
Precision:              0.7957
Recall:                 0.7475
F1-Score:               0.7708

Negative:
Precision:              0.7664
```

```
Recall:                 0.8119
F1-Score:               0.7885
Accuracy:               0.78
```

Evaluation metrics when the model uses only three features *great, poor,* and *long*:

```
Number of features:  3

Positive:
Precision:              0.52258
Recall:                 0.81818
F1-Score:               0.6378

Negative:
Precision:              0.6
Recall:                 0.26733
F1-Score:               0.36986
Accuracy:               0.54
```

How to run the program?

1. Create a naiveBayes object.
2. Call the feature_selection method and pass in the train set and number of features. The decision on the number of features need to rely on trial and error.
3. Call the train method and pass in the train set and selected features from step 2.
4. Call the test method and save the result for evaluation.
5. Call the evaluate method and pass in the results from step 4 to the evaluate method.

Sample code:

```
if __name__ == '__main__':
    nb = NaiveBayes()
    selected_features = nb.feature_selection('movie_reviews/train',
4200)
    nb.train('movie_reviews/train', selected_features)
    results = nb.test('movie_reviews/dev')
    nb.evaluate(results)
```

Feature selection

The feature_selection() method was implemented using the likelihood ratio. It caps the features to the top n features based on the order of the likelihood ratio for each word. The number n is a parameter defined by the user.

Implementing the likelihood ratio for feature selection doesn't improve the accuracy. As one can imagine, the initial accuracy of 0.78 was based off all words, and selecting feature means dropping some less important features. Thus, it is possible that we lose some information from the features that we dropped. Therefore, it is not counterintuitive that the accuracy goes down when the feature selection method is not very impressive.

Following numbers show the evaluation metrics with only 100 features selected by the likelihood ratio feature selection method. As can be seen, the accuracy is around 69%, which is almost 10% lower than using all words.

Evaluation metrics using 100 features selected by likelihood ratio method:

```
Number of features:  100

Positive:
Precision:              0.65041
Recall:                 0.80808
F1-Score:               0.72072

Negative:
Precision:              0.75325
Recall:                 0.57426
F1-Score:               0.65169
Accuracy:               0.69
```

To solve this problem, we then take the approach of using the words that we intuitively know that they are going to have positive impact on the model. These words are words from the sentiment dictionary. The sentiment dictionary could be long and therefore increases the model running time. Thus, the feature selection method comes in handy.

The sentiment dictionary is from UIC (https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html). In order to build the sentiment dictionary for the model to use, the program reads a positive dictionary called

positive-words.txt and a negative dictionary called negative-words.txt. These two files are included in the submission.

In summary, the new approach is to start with using the words that are in the train set and also appear in the sentiment dictionary. And then apply feature selection method to the existing sentiment words to get the words that gives most information in terms of classification.

Evaluation metrics using only sentiment words:

```
Number of features:  4414

Positive:
Precision:              0.85714
Recall:                 0.78788
F1-Score:               0.82105

Negative:
Precision:              0.80734
Recall:                 0.87129
F1-Score:               0.8381
Accuracy:               0.83
```

Apply the feature selection method on the sentiment word features:

(n = 4200)

```
Number of features:  4200

Positive:
Precision:              0.8587
Recall:                 0.79798
F1-Score:               0.82723

Negative:
Precision:              0.81481
Recall:                 0.87129
F1-Score:               0.84211
Accuracy:               0.835
```

The best accuracy I got was ***83.5%*** for this assignment, by using 4200 features selected by the likelihood ratio method.

Other

I spent around 20 hours on assignment 3. There was nothing surprising about assignment 3 but fixing a few small bug did cost me many hours. I started out with a long running time but later on figured it can definitely be reduced significantly through reconstruct the variables that are used to store counts and probabilities. I am sure that there are still plenty of room to improve the running time of my model.