CSCE 638

Programming Assignment 2

Name: Sun Yul Lee

ID: 824007724

Objective

The objective of this assignment is to perform sentiment analysis. The programs train a Naïve Bayes classifier and a Perceptron classifier on the imdb1 data set. The programs need to classify an entire movie review as positive or negative.

Implementation

- Naïve Bayes

I add one list for holding documents information and two integer variables for counting documents labeled ‘pos’ and counting documents labeled ‘neg’ in the initializer of the class NaiveBayes.

```
self.docs = []
self.num_pos = 0
self.num_neg = 0
```

Figure 1. Added instance variables of a class NaiveBayes

- def addExample(self, klass, words)

  When a new document with its label is added, this function increments the number of documents with the corresponding label. Then it creates a dictionary. It sets ‘klass’ and ‘words’ as the dictionary’s keys and sets a value of the keys with the given parameters. Then the function adds the dictionary to the list of documents information.

- der classify(self, words)

  First, this function checks FILTER_STOP_WORDS value. If the value is true, the function removes stop words from the given list of words. In the Naïve Bayes classifier, I applied Laplace smoothing, and underflow prevention using log space. With the number of documents with the label ‘pos’ and the number of documents with the label ‘neg’ from adding examples, the function calculates a probability of each class. Then, it creates a dictionary variable bag_words to hold number of each word in documents with the label ‘pos’ and number of each word in documents with the label ‘neg’. If BOOLEAN_NB value is true, the function removes all duplicates in the list of words, then creates bag_words. After it add all words of examples in bag_words, the function uses the information in bag_words and the given list of words to classifies a class of a given document using Naïve Bayes method. It removes all duplicates in the given list of words if BOOLEAN_NB value is true, then classifies a class of a given document.

- Perceptron

I add one list for holding documents information, one dictionary for holding words information, and two integer variables for counting documents labeled ‘pos’ and counting documents labeled ‘neg’ in the initializer of the class Perceptron.

```
self.docs = []
self.bag_words = {}
self.num_pos = 0
self.num_neg = 0
```

Figure 2. Added instance variables of a class Perceptron

- def addExample(self, klass, words)

When a new document with its label is added, this function increments the number of documents with the corresponding label. Then it creates a dictionary. It sets ‘klass’ and ‘words’ as the dictionary’s keys and sets a value of the keys with the given parameters. Then the function adds the dictionary to the list of documents information. The function uses the given list of words to update the instance variable bag_words to initialize weight for each word and hold information that whether a word is found in documents labeled ‘pos’, documents labeled ‘neg’, or both.

- def classify(self, words)

This function uses Maxent method to classify a class of a given document. During the implementation, the function got an out of range error in math.exp() method. Therefore, the function compares sum of weights for a class ‘pos’ and sum of weights for a class ‘neg’ for classification since the original probability of a class is higher when sum of weights for a class is higher.

- def train(self, split, iterations)

First, this function adds all example documents to setup data for training. Then, the function calls function classify() for each example document. It updates weights of words in the instance variable bag_words if a predicted class is different from an actual class of the document. The weight update process in the function implements the parameter averaging.

Compile and Execution

- Naïve Bayes

To execute the program, user needs to type “python NaiveBayes.py <data_dir>”, “python NaiveBayes.py -f <data_dir>”, or “python NaiveBayes.py -b <data_dir>” in a command line prompt where <data_dir> is a directory of a folder which contains data set.

```
(base) C:₩Users₩etional₩Desktop₩2020₩spring₩CSCE638₩SentimentAnalyzer_Spring21₩SentimentAnalyzer_Spring21₩python>
python NaiveBayes.py C:/Users/etional/Desktop/2020/spring/CSCE638/SentimentAnalyzer_Spring21/SentimentAnalyzer_Sp
ring21/data/imdb1
```

Figure 3. Executing the Naïve Bayes classifier without removing stop words in my device

```
(base) C:₩Users₩etional₩Desktop₩2020₩spring₩CSCE638₩SentimentAnalyzer_Spring21₩SentimentAnalyzer_Spring21₩python>
python NaiveBayes.py -f C:/Users/etional/Desktop/2020/spring/CSCE638/SentimentAnalyzer_Spring21/SentimentAnalyzer
_Spring21/data/imdb1
```

Figure 4. Executing the Naïve Bayes classifier with removing stop words in my device

Figure 5. Executing the Binarized Naïve Bayes classifier in my device

- Perceptron

  To execute the program, user needs to type "python Perceptron.py <data_dir> <iteration>" in a command line prompt where <data_dir> is a directory of a folder which contains data set and <iteration> is a number of iterations for training.

Figure 6. Executing the Perceptron classifier with 1 iteration in my device

Figure 7. Executing the Perceptron classifier with 1000 iterations in my device

- # Result

- Naïve Bayes

  Without removing stop words, the Naïve Bayes classifier achieved an average accuracy of 0.817. With removing stop words, the Naïve Bayes classifier achieved an average accuracy of 0.811. The observation of the results shows that removing stop words affects average accuracy. It decreased the average accuracy of the Naïve Bayes classifier. The Binarized Naïve Bayes classifier achieved an average accuracy of 0.829, which is the highest accuracy among the three models.

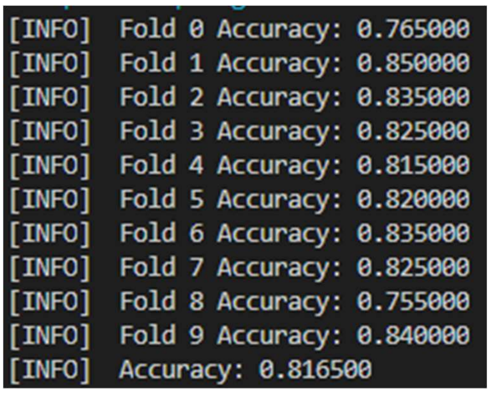

Figure 8. Result of the Naïve Bayes classifier without removing stop words

```
[INFO]  Fold 0 Accuracy: 0.765000
[INFO]  Fold 1 Accuracy: 0.825000
[INFO]  Fold 2 Accuracy: 0.815000
[INFO]  Fold 3 Accuracy: 0.830000
[INFO]  Fold 4 Accuracy: 0.795000
[INFO]  Fold 5 Accuracy: 0.830000
[INFO]  Fold 6 Accuracy: 0.835000
[INFO]  Fold 7 Accuracy: 0.835000
[INFO]  Fold 8 Accuracy: 0.760000
[INFO]  Fold 9 Accuracy: 0.820000
[INFO]  Accuracy: 0.811000
```

Figure 9. Result of the Naïve Bayes classifier with removing stop words

```
[INFO]  Fold 0 Accuracy: 0.805000
[INFO]  Fold 1 Accuracy: 0.840000
[INFO]  Fold 2 Accuracy: 0.835000
[INFO]  Fold 3 Accuracy: 0.825000
[INFO]  Fold 4 Accuracy: 0.835000
[INFO]  Fold 5 Accuracy: 0.825000
[INFO]  Fold 6 Accuracy: 0.845000
[INFO]  Fold 7 Accuracy: 0.835000
[INFO]  Fold 8 Accuracy: 0.790000
[INFO]  Fold 9 Accuracy: 0.855000
[INFO]  Accuracy: 0.829000
```

Figure 10. Result of the Binarized Naïve Bayes classifier

- Perceptron

  With the iteration of 1, the Perceptron classifier achieves the average accuracy of 0.5. As the number of iterations is increased, the average accuracy is also increased. With the iteration of 5000, the Perceptron classifier can achieve the average accuracy of 0.830 which is better than the average accuracy of Binarized Naïve Bayes classifier.

```
[INFO]  Fold 0 Accuracy: 0.500000
[INFO]  Fold 1 Accuracy: 0.500000
[INFO]  Fold 2 Accuracy: 0.500000
[INFO]  Fold 3 Accuracy: 0.500000
[INFO]  Fold 4 Accuracy: 0.500000
[INFO]  Fold 5 Accuracy: 0.500000
[INFO]  Fold 6 Accuracy: 0.500000
[INFO]  Fold 7 Accuracy: 0.500000
[INFO]  Fold 8 Accuracy: 0.500000
[INFO]  Fold 9 Accuracy: 0.500000
[INFO]  Accuracy: 0.500000
```

Figure 11. Result with the iteration of 1

```
[INFO]  Fold 0 Accuracy: 0.510000
[INFO]  Fold 1 Accuracy: 0.545000
[INFO]  Fold 2 Accuracy: 0.535000
[INFO]  Fold 3 Accuracy: 0.525000
[INFO]  Fold 4 Accuracy: 0.510000
[INFO]  Fold 5 Accuracy: 0.515000
[INFO]  Fold 6 Accuracy: 0.520000
[INFO]  Fold 7 Accuracy: 0.525000
[INFO]  Fold 8 Accuracy: 0.590000
[INFO]  Fold 9 Accuracy: 0.565000
[INFO]  Accuracy: 0.534000
```

Figure 12. Result with the iteration of 10

```
[INFO]  Fold 0 Accuracy: 0.640000
[INFO]  Fold 1 Accuracy: 0.620000
[INFO]  Fold 2 Accuracy: 0.680000
[INFO]  Fold 3 Accuracy: 0.605000
[INFO]  Fold 4 Accuracy: 0.605000
[INFO]  Fold 5 Accuracy: 0.585000
[INFO]  Fold 6 Accuracy: 0.585000
[INFO]  Fold 7 Accuracy: 0.555000
[INFO]  Fold 8 Accuracy: 0.675000
[INFO]  Fold 9 Accuracy: 0.600000
[INFO]  Accuracy: 0.615000
```

Figure 13. Result with the iteration of 50

```
[INFO]  Fold 0 Accuracy: 0.605000
[INFO]  Fold 1 Accuracy: 0.645000
[INFO]  Fold 2 Accuracy: 0.700000
[INFO]  Fold 3 Accuracy: 0.630000
[INFO]  Fold 4 Accuracy: 0.660000
[INFO]  Fold 5 Accuracy: 0.620000
[INFO]  Fold 6 Accuracy: 0.630000
[INFO]  Fold 7 Accuracy: 0.580000
[INFO]  Fold 8 Accuracy: 0.700000
[INFO]  Fold 9 Accuracy: 0.650000
[INFO]  Accuracy: 0.642000
```

Figure 14. Result with the iteration of 100

```
[INFO]  Fold 0 Accuracy: 0.785000
[INFO]  Fold 1 Accuracy: 0.765000
[INFO]  Fold 2 Accuracy: 0.755000
[INFO]  Fold 3 Accuracy: 0.755000
[INFO]  Fold 4 Accuracy: 0.770000
[INFO]  Fold 5 Accuracy: 0.780000
[INFO]  Fold 6 Accuracy: 0.750000
[INFO]  Fold 7 Accuracy: 0.700000
[INFO]  Fold 8 Accuracy: 0.770000
[INFO]  Fold 9 Accuracy: 0.745000
[INFO]  Accuracy: 0.757500
```

Figure 15. Result with the iteration of 1000

```
[INFO]  Fold 0 Accuracy: 0.825000
[INFO]  Fold 1 Accuracy: 0.835000
[INFO]  Fold 2 Accuracy: 0.835000
[INFO]  Fold 3 Accuracy: 0.840000
[INFO]  Fold 4 Accuracy: 0.810000
[INFO]  Fold 5 Accuracy: 0.860000
[INFO]  Fold 6 Accuracy: 0.840000
[INFO]  Fold 7 Accuracy: 0.805000
[INFO]  Fold 8 Accuracy: 0.805000
[INFO]  Fold 9 Accuracy: 0.840000
[INFO]  Accuracy: 0.829500
```

Figure 16. Result with the iteration of 5000

- ## Solved problem

During the implementation of Maxent method in Perceptron classifier, the program got an out of range error in math.exp() method. When I inspected the variables to find a cause of the error, a variable 'wa' in the process of parameter averaging became a huge number. To solve the problem, I compare sum of weights for a class 'pos' and sum of weights for a class 'neg' for classification since the probability of a class is higher when sum of weights for a class is higher.