Hasanat Jahan
CS381 - Homework 1 Report
Date: April 27, 2021

**Answer to Question 1**

Assume that you have trained a Na¨ive Bayes classifier for the task of sentiment classification (please refer to Chapter 6, pp. 1-9 in the J&M book). The classifier uses only bag-of-word features. Assume the following parameters for each word being part of a positive or negative movie review, and the prior probabilities are 0.4 for the positive class and 0.6 for the negative class.

Question: What class will Na¨ive Bayes assign to the sentence "I always like foreign films"? Show your work.

```
p(pos) = 0.4
p(neg) = 0.6
Let D = "I always like foreign films"

p(pos|D) = p(pos) * p(I|pos) * p(always|pos) * p(like|pos) *
p(foreign|pos) * p(films|pos)
= 0.4 * 0.09 * 0.07 * 0.29 * 0.04 * 0.08
= 0.00000233856

p(neg|D) = p(neg) * p(I|neg) * p(always|neg) * p(like|neg) *
p(foreign|neg) * p(films|neg)
= 0.6 * 0.16 * 0.06 * 0.06 * 0.15 * 0.11
= 0.0000057024
```

As p(neg|D) > p(pos|D), the model assigns the sentence D class "neg"

**Answer to Question 2**

**B.** The small input was created into feature vectors for both train and test.
The feature vectors were created by using preprocess.py.
Each file was represented using a vector of bag-of-word features. Feature vectors for test and train are are located in the folder small_movie_review/feature_vectors
Then NB.py was run, the parameters of the model was saved in movie-review-small.NB in small_movie_review

**C.** Running NB.py also tested the model on the test document. The output of the test was saved in small_movie_review/output.txt
The output is also displayed to the terminal. The evaluation of the model with Naive Bayes classifier with BOW features and add-one smoothing:

```
Example: 1
Comedy Prediction: -4.135238693248112
action Prediction: -3.765817515309918
Predicted Label: action
Actual Label: unknown
```

For the sake of checking the validity of NB.py for the large movie review dataset, these values were also hand calculated and checked.
This value can be found again by running NB.py and choosing option 1.

**D.** Similarly to the example above, using pre-process.py, each review was represented using a vector of bag-of-word features. Preprocess.py was run individually on the train and test folders and the feature vectors were saved in movie-review-HW2/feature_vectors

option 2 :First, I ran NB.py on the train and test feature vectors, outputted the parameters to movie-review-BOW.NB and outputted the result to output.txt in the movie-review-HW2 directory. This is option 2 in the terminal upon running NB.py

The accuracy of only BOW features with add one smoothing came to be: (also outputted to the terminal and the txt file):

```
The results have been written to output.txt
Num correct : 22257
Num incorrect : 2743
Num of documents : 25000
Accuracy: 89.02799999999999 %
```

option 3: Then I looked to optimize the sentiment analysis of the review. For sentiment analysis, the occurrence of a word might matter more than whether a word appears multiple times. Therefore, in NB.py, I adjusted the feature vectors to reflect a word occurring being counted only once per document for both the count of the word and in the total number of words in the training data. This is option 3 in the terminal.
The accuracy then improved by 0.768298%, therefore it seemed like there was an improvement to doing binary counts of words:

```
The results have been written to experiment-output.txt
Num correct : 22428
Num incorrect : 2572
Num of documents : 25000
Accuracy: 89.712 %
```

option 4 :Afterwards, I tried to remove redundant words like articles and prepositions including “the, a, an, from, across, along, in, by, upon, with, within, of, to” in preprocessing and running Naive Bayes classifier with BOW features without binary counts. I did not

exclude some specific prepositions like “against, above, below, behind, down, near, off, under” and more since these prepositions might signal sentiment. The feature vectors for this were different from the feature vectors for both models above as there were many words missing from the total word count. The feature vectors were then saved in the same folder with the other feature vectors but they have the word “experiment” prefixed to their filenames for clarity. This is option 4 in the terminal for the program. The accuracy decreased by 5.65215%:

```
The results have been written to experiment-output-1.txt
Num correct : 20999
Num incorrect : 4001
Num of documents : 25000
Accuracy: 83.99600000000001 %
```

So it would seem that removing redundant prepositions is not such a good idea for sentiment analysis, at least removing the specific prepositions in the list from the feature vectors did not yield a high accuracy like the previous models.

option 5: I also wanted to examine if the length of the test document had any effect on the prediction made by the model. In the calculations of the likelihoods for each class for each example, along with BOW features I also included the log of the word count of that particular test document. I had high hopes for this feature, because by intuition it would seem that document word counts might signal something about sentiment. When we leave reviews on Amazon, the negative reviews are usually much longer than the five-star positive “I love it!” reviews. However, after implementing this feature along with the original BOW features with add-one smoothing the accuracy for the model went down even more than it did for removing redundant words, and the accuracy decreased by 6.21378% . It might be that number of words in a movie review might not be a strong indicator of it’s sentiment. This is option 5 in the terminal for the running of the program:

```
The results have been written to experiment-output-2.txt
Num correct : 20874
Num incorrect : 4126
Num of documents : 25000
Accuracy: 83.49600000000001 %
```

Then I investigated the results for the original Naive Bayes classifier with BOW features and add one smoothing:

```
Percent of neg classes incorrectly labelled 30.076558512577474
%
Percent of pos classes incorrect labelled  69.92344148742254 %
Average word count per example in incorrect classification
170.62376959533358
```

```
Average word count per example in correct classification
200.92438334007278
```

With the model, positive classes were more likely to mislabelled as negative rather than negative classes mislabelled as positive. I investigated the average word counts of the correctly classified documents and the incorrectly classified documents and it seems that the average number of words in the incorrectly classified documents is 15.0806% less than the average number of words in the correctly classified documents. It could be that having more words in the test data allowed it to have a higher chance of being classified correctly.