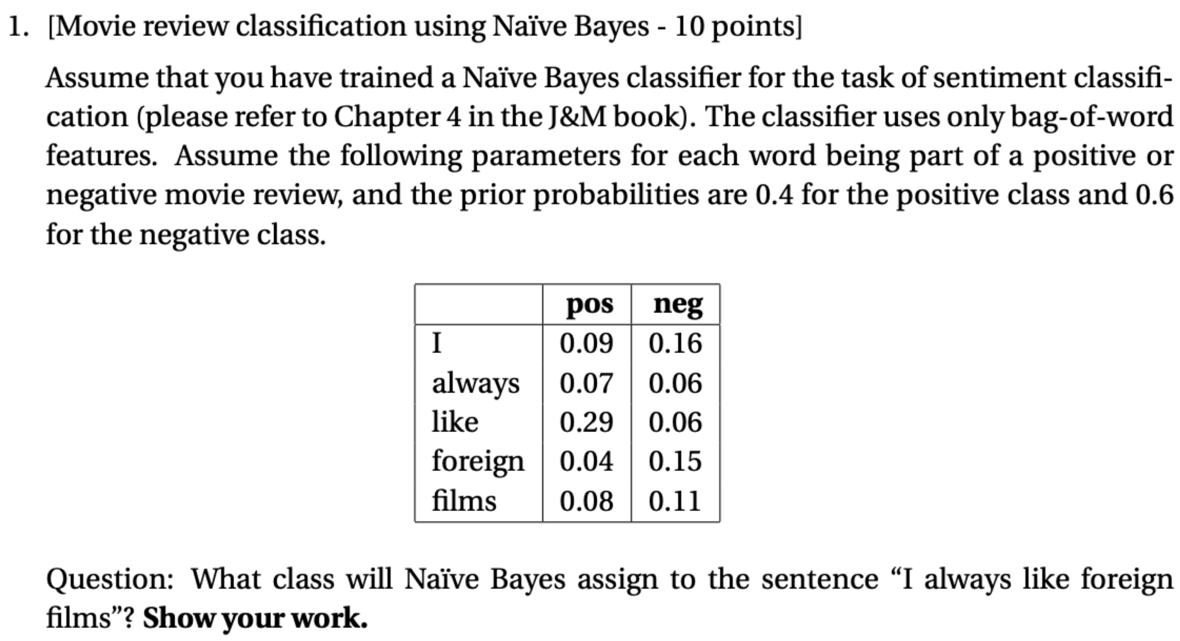

1. [Movie review classification using Naïve Bayes - 10 points]

   Assume that you have trained a Naïve Bayes classifier for the task of sentiment classification (please refer to Chapter 4 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.

	pos	**neg**
I	0.09	0.16
always	0.07	0.06
like	0.29	0.06
foreign	0.04	0.15
films	0.08	0.11

   Question: What class will Naïve Bayes assign to the sentence “I always like foreign films”? **Show your work.**

$$p(pos) * p(s|pos) = .4 * .09 * .07 * .29 * .04 * .08 = 2.34 * 10^{-6}$$
$$p(neg) * p(s|neg) = .6 * .16 * .06 * .06 * .15 * .11 = 5.7 * 10^{-6}$$

According to the product above, the Naïve Bayes model would assign this sentence to the negative class.

2a: Implemented in NB.py. Code is efficient and ran on my computer in under 1 minute. In order to run the code, you need the preprocessed files which in order to make the code fast, I pickled. Pickled bytestream files are much easier to load onto python than csv files.

2b: Parameters hard coded into testnb.py. The parameters are stored as dictionaries. The final result was that the test set belongs in the comedy class.

2c: comedy
I predict that the test set to be in the comedy class. The P(comedy) is 9.605448239171125 and the P(action) is 9.160289649922504.

You can get this result by running testnb.py. It is a ‘mini version’ of NB.py.

2d: Files preprocessed and pickled into vectors and dictionaries. You can easily generate this set by running preprocess.py and having the following directory structure in your current working directory:

```
#positive class of training data
path_to_pos_train = r'movie-review-HW2/aclImdb/train/pos'
#negative class of training data
path_to_neg_train = r'movie-review-HW2/aclImdb/train/neg'
#positive class of testing data
path_to_pos_test = r'movie-review-HW2/aclImdb/test/pos'
#negative class of testing data
path_to_neg_test = r'movie-review-HW2/aclImdb/test/neg'
```

The BOW features are stored in .pickle files instead of .nb files for easy python loading. Since that is the case, and these files are included in the zip file, you don’t need to run preprocess for the nb.py code to run.

Result analysis:

20376 of the 25000 test cases accurately. That is an 81.504% accuracy which fits the Naïve Bayes accuracy we were looking for in this project.

From my analysis of the incorrectly predicted reviews, it appears that authors of these reviews often used ambiguous language, making the review sound positive when they were giving a critique or using sarcasm. Also, it’s expected that Naïve Bayes wouldn’t give a really good accuracy, but 81% is pretty good.

Happy holidays!