Text Classification

In class, we spent some time on text classification, in particular naive Bayes classifiers. We focused on these models not only because they are simple to implement and fairly effective, but also because of their similarity to widely used bag-of-words retrieval models such as BM25 and query likelihood.

Your task is to write a naive Bayes text categorization system to predict whether movie reviews are positive or negative. The data for this "sentiment analysis" task were first assembled and published in Bo Pang and Lillian Lee, "A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts", *Proceedings of the Association for Computational Linguistics*, 2004.

The data in `textcat.zip` are, when unzipped, organized in the following directory structure:

```
train/
train/neg/
train/pos/
dev/
dev/neg/
dev/pos/
test/
```

Each directory contains some `.txt` files, with one review per file. For the training and development data, the positive and negative reviews are known, and indicated by the subdirectory names. For the test data, all reviews are mixed together in a single directory.

The text has been pre-tokenized and lower-cased. All you have to do to get the individual terms in a review is to split the tokens by whitespace, a sequence of spaces and/or newlines. You don't need to remove stopwords or do any further stemming. Given the simple smoothing you will implement (see below), you should ignore any terms that occur **fewer than 5 times** in the combined positive and negative training data.

The main part of this assignment involves implementing a bag-of-words, naive Bayes classifier using add-1 (Laplace) smoothing. After estimating the parameters of the naive Bayes model using the data in `train`, you should output your predictions on the files in the `test` directory.

You should organize your system into two programs, which should be called as follows:

```
nbtrain <training-directory> <model-file>
```

This should read all the files in (subdirectories of) the supplied training directory and output the parameters to a model file, whose format is up to you (but see below). Then, you should predict the class of unseen test documents with:

```
nbtest <model-file> <test-directory> <predictions-file>
```

You should hand in:

1. Your source code and a `README` explaining how to (compile and) run it.
2. Your predictions file for the development data, which should list each filename in the `dev` directory, and the scores your model assigns to the possibilities of positive and negative reviews. In your `README`, you should also tell us what percentage of positive and negative reviews in the development data were correctly classified.
3. Your predictions file for the test data, which should list each filename in the `test` directory, and the scores your model assigns to the possibilities of positive and negative reviews.
4. A list of the **20 terms** with the highest (log) ratio of positive to negative weight.
5. A list of the **20 terms** with the highest (log) ratio of negative to positive weight.

As you might gather from the list above, the parameters of the model will be (log) probabilities of different terms given positive or negative review polarity. You can format your model file in any way you like, as long as you can extract the top 20 terms we request, although you may do that step by hand.

Extra Credit: 20 points

In the assignment above, we specified which features to use and how to smooth the probability distributions. For extra credit, try other features besides unigrams that occur five times or more, other smoothing methods and parameters, and so on. The basic naive Bayes training and testing procedure should remain the same, of course. You can use the development data to track your progress. Describe the new features you tried and their accuracy on the development data in your `README`. In addition to the results from the assignment above, turn in the modified code and its predictions on the test data.