Task 2: Naive Bayes Classifier

In the previous task we only used selected words for the classification, and we relied on the manual work that went into the lexicon to decide which words were likely to be in reviews with a particular sentiment. The sentiment lexicon is general: it is supposed to be applicable to any English text. We will now create a statistical classifier, learning which words are positive and negative in movie reviews. Unlike the lexicon approach, this classifier uses **all** of the words in the text.

Step 0: Data preparation.

Use the dataset from Task 1 for training and for today's testing/development (i.e., the development set). There are a further 200 reviews that you do not have access to at the moment (**held-out** data) that you will use for more formal evaluation in a subsequent session. You can use `DataSplit.java` to split the 1800 document dataset into a training set and a development set. There should be 200 documents in your development set (balanced: 100 positive, 100 negative) and 1600 in the training set (also balanced).

Once you have done the split, rerun your code from Task 1 on the 200 examples in the development set so you can later see whether you get an improvement with the Naive Bayes classifier.

Step 1: Parameter estimation.

Your first task in developing the Naive Bayes classifier is to calculate probabilities of each word we observe in the text, namely: P(word|Pos-text) and P(word|Neg-text). Write a program that computes these probabilities from the training data. Then compute the logs of these probabilities.

Step 2: Classification.

Now use the probabilities to apply the classification argmax formula of Naive Bayes to the development set:

$$c_{NB} = \underset{c \in pos,neg}{argmax} \{logP(c) + \sum_{i \in positions} logP(w_i|c)\}$$

positions is the set of indexes into the words in the document. $P(c)$ is the probability of a document belonging to a given class. For a balanced data set, like the one here, this probability is the same for each class but this will not

always be the case. The argmax formula gives you a decision (a classification) for each document, either negative or positive.

How well did this program perform? As before, we can compare the system's performance to the truth but we should use only the development set to do this.

Step 3: Smoothing.

Have you noticed any issues when using the log probabilities calculated in Step 1?

When using a Naive Bayes classifier, you need to consider what to do with unseen words – words which occur in the development/test data but which do not occur in the training data at all. You will also need to consider what to do with words which were seen only in one class of document, either positive or negative ones. Write down possible solutions to this problem.

Modify your calculation of log probabilities to implement add-one smoothing – add one to all the counts.

> Example: Say a word *jabberwocky* appeared in you training set only once, in a positive review. These are the counts you should record.

	positive	negative
unsmoothed	1	0
add-one	2	1

How does your classifier perform now?

Once you have successfully developed and tested a system which incorporates add-one smoothing, you may submit your code to the online tester. Remember that, even if you do not base your code on the Exercise2Tester.java, you will find it useful to look at that to see what we expect you to submit. Once your code has passed, you may contact a demonstrator to obtain Tick 2.