Quantitative Text Analysis
Exercise 6: Document Classification

28th July 2014, Essex Summer School

Kenneth Benoit and Paul Nulty

In today's lab we will use R to understand and apply document classification, using a classic computer science dataset of movie reviews, (Pang and Lee 2004).

Instructions

1. The movies corpus has an attribute 'lab' that labels each text as either `pos` or `neg` according to the original imdb.com archived newspaper review star rating. We will begin by examining the conditional probabilities at the word level.

   (a) Load the movies dataset and examine the attributes:

   ```
   data(movies)
   names(movies$attribs)
   summary(movies)
   ```

   (b) Make a dfm from the corpus, grouping the documents by the 'lab' attribute

   (c) What is the overall probability of the class `pos` in the corpus? (Hint: `table(movieDfm)`)

   (d) Words with very low overall frequencies in a corpus of this size are unlikely to be good general predictors. Remove words that occur less than twenty times using `dfmTrim`. Since the dfm now effectively contains only two documents (positive text and negative text), you will need to specify `minDoc=1`.

   (e) Calculate the word-level probabilities; i.e. the probability of the class `pos` given the word, for each word.

   (f) How would the word probabilities be affected if we had many more examples of positive reviews than negative?

   (g) Inspect the word-level probabilities. If the vector containing your probabilities is called `PwordGivenPos`, then use `PwordGivenPos['excellent']` to see the probability of the class `pos` given the word *excellent*. You can sort the vector by the probabilities and inspect the highest and lowest probabilities as follows:

   ```
   sortedProbs <- pPosgivenWord[order(pPosgivenWord)]
   head(sortedProbs, n=50)
   tail(sortedProbs, n=50)
   ```

2. Now we will use `quanteda`'s naive bayes function to run a prediction on the movie reviews.

   (a) The movie corpus contains 1000 positive examples followed by 1000 negative examples. When extracting training and testing labels, we want to get a mix of positive and negative in each set, so first we need to shuffle the corpus. You can do this with the `corpusSample` command:

   ```
   movies <- corpusSample(movies, size=2000, replace=FALSE)
   ```

If the subsequent training and testing commands are running too slowly on the lab machines, try using a smaller `size` than the full 2000.

(b) Next, make a dfm from the shuffled corpus, and make training labels. In this case, we are using 1500 training labels, and leaving the remaining 500 unlabelled to use as a test set.

```
movieDfm <- dfm(movies)
trainclass <- factor(c(movies$attribs$lab[1:1500], rep(NA, 500)))
```

(c) Now we run the training and testing commands, and compare the predictions for the documents with the actual document labels for the test set.

```
movieNb <- naiveBayesText(as.matrix(movieDfm), trainclass)
movPreds <- predict(movieNb)
table(movPreds$docs$nb.predicted[1500:2000],movies$attribs$lab[1500:2000])
```

(d) Compute, for the last classification: (Hint - the row named '1' in the table output corresponds to the name of the first column)

i. precision;

ii. recall;

iii. $F1$; and

iv. accuracy. How does accuracy change if you adjust the size of the training set relative to the test set?