Quantitative Text Analysis
Exercise 2: Text Processing and Words as Features

22nd July 2014, Essex Summer School

Kenneth Benoit and Paul Nulty

In today's lab we will use quanteda to apply some standard pre-processing steps to our text. quanteda provides functions for tokenization, stemming, stopword removal, and the creation of a document-term matrix.

Getting Started

Open RStudio and load the quanteda library. This will always be the command `library(quanteda)`. quanteda includes a built-in data object consisting of a simple string of text to help you examine the effects of basic preprocessing functions. Load this text with the command `data(exampleString)`. You can read the string by returning it to the top-level environment by simply typing its name at the prompt, in other words,

```
> library(quanteda)     # loads the library
> data(exampleString)   # loads the data object that is part of quanteda
> exampleString         # dumps the example string to the screen
```

Detailed Instructions

1. **Cleaning and Tokenizing**

   (a) Often, before applying other pre-processing steps, the text is 'cleaned' slightly, perhaps by removing punctuation, digits, and converting to lower case. Look at the documentation for quanteda's `clean` command (`?clean`) and use the command on the `exampleString`. Can you think of cases where cleaning could introduce homonymy?

   (b) In order to count word frequencies, we first need to split the text into words through a process known as *tokenization.* Look at the documentation for quanteda's `tokenize` command using the built in help function. You can get the help instructions for any command by typing it preceded by a "?" at the prompt, e.g. (`?tokenize`). Use the `tokenize` command on `exampleString`, and examine the results that are printed to the screen. Are there cases where it is unclear where the boundary between two words lies?

2. **Stopwords and stemming**

   (a) Look at the documentation for the `stopwordsRemove` command. This function can be applied to any string — try it out at the command prompt, on the example string.

   (b) For stemming, quanteda uses the Porter stemmer algorithm implemented in the SnowballC package. To see how this function works, load the SnowballC library and look at the documentation. The `wordStem` command can be used on a list of tokens. Apply it to the list of tokens returned by `tokenize` in 1(b).

3. **Document Feature Matrix**

(a) In order to perform statistical analysis such as document scaling, we must extract a matrix associating values for certain features with each document. In quanteda, we use the dfm function to produce such a matrix. Read the documentation for dfm, and use the command to create a document-feature matrix from the example string.

(b) For comparative analysis, we want to observe matrixes over multiple documents. Load the Irish budget speeches corpus with the command data(iebudgets) and make a document-feature matrix from it. Take a subset of the corpus, with only the 2010 texts, with this command:

```
> data(iebudgets)
> iebudgets2010 <- subset(iebudgets, year==2010)
> ie2010Dfm <- dfm(iebudgets2010)
```

Inspect the resulting matrix by clicking on the environment pane, and by indexing at the command prompt. How many times does the word 'the' occur in the third document?

(c) Try the stopwords and stem arguments to dfm and observe the effect on the word columns in the resulting matrices.

(d) quanteda provides a dfmTrim function allow for the document-feature matrix to be trimmed according to the overall frequency or document frequency of a feature. On the Irish budget speeches corpus, what is the effect on the matrix if we remove words that occur less than 5 times overall, or in fewer than 3 documents?

(e) As an alternative to quanteda provides a dfmTrim function allow for the document-feature matrix to be trimmed according to the overall frequency or document frequency of a feature. On the Irish budget speeches corpus, what is the effect on the matrix if we remove words that occur less than 5 times overall, or in fewer than 3 documents?

(f) The group parameter to the dfm function allows us to treat texts aggregated by a particular attribute as the unit of analysis, rather than single documents. Create a dfm grouped by the party variable and inspect the matrix.

4. **tf-idf Weighting**

(a) Now we will convert the document-term matrix into a matrix of tf-idf weights. Remember these definitions:

term frequency (for us) the relative (or *normalized*) term frequency, defined as the count of the term divided by the total count of terms for the document. From a dfm object, this will be the cell divided by the row marginals (document totals).

document frequency the total number of documents a term appears in. In R, this can be obtained by coercing the logical value TRUE to a value of 1 and summing it. In other words, you can sum the Boolean to test that a term's frequency across documents (in columns of the dfm) is > 0. Remember we will take the inverse of document frequency and log this quantity.

Compute the tf-idf matrix for the dfm you already created. Here are some clues:

```
> tf <- myDfm / rowSums(myDfm)  # rowSums is the total tokens per document
> idf <- log(nrow(myDfm)) - log(colSums(myDfm > 0) + 1)  # study this carefully
                                                    # what if we don't add the 1?
                                                    # why are we subtracting the logs?
```

(b) Now compare this to the built-in function's version, using `tfidf(myDfm)`. Is it the same?

(c) Compare the tf-idf values for the first speech, using

```
> plot(myDfm[1,], tfidf(myDfm)[1,], log="x", xlab="log(Term Frequency)", ylab="tf-idf")
```

What happens if we do not display the term frequencies on the log scale?