Reading in text data. Creating and visualizing a dfm

8 May 2019

In this document we will go through the steps of going from raw texts to a document term matrix that can be analyzed.

Reading in data

Let's take a look a set of UK prime minister speeches from the EUSpeech dataset.

NB: Use setwd() to set the working directory to the folder that contains English speeches in the file speeches_uk.csv. Read in the speeches as follows using the `read.csv()` function:

```
Sys.setlocale(locale = "en_US.UTF-8")
```

```
## [1] "en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8"
```

```
speeches <- read.csv(file = "speeches_uk.csv",
                     header = TRUE,
                     stringsAsFactors = FALSE,
                     sep = ",",
                     encoding = "UTF-8")
```

Let's take a look at the structure of this dataset:

```
str(speeches)
```

```
## 'data.frame':    787 obs. of  6 variables:
##  $ X      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ text   : chr  "<p>This European Council has focused on 3 issues - the UK renegotiation, migration
##  $ title  : chr  "EU Council: PM press conference" "PM statement in Poland: 10 December 2015" "PM st
##  $ date   : chr  "18-12-2015" "10-12-2015" "09-12-2015" "07-12-2015" ...
##  $ country: chr  "Great Britain" "Great Britain" "Great Britain" "Great Britain" ...
##  $ speaker: chr  "D. Cameron" "D. Cameron" "D. Cameron" "D. Cameron" ...
```

As you can see, the corpus contains 787 speeches and variables containing meta data like speaker, country, date, etc. Take a look at a few speeches. Let's do some very light cleaning on these speeches, using the `stringr` library.

```
library(stringr)

#remove html tags
speeches$text <- str_replace_all(speeches$text, "<.*?>", "")
#replace multiple white spaces with single white spaces
speeches$text <- str_replace_all(speeches$text, "  ", " ")
```

Question: If these were a proper data analysis, what other steps would you take to further clean the data?

Our speeches object is currently a dataframe. To be able to apply functions in `quanteda` on this object it needs to recognize it as a corpus. For this, you use the `corpus()` function

```
library(quanteda)
corpus <- corpus(speeches)

#the ndoc function displays the number of documents in the corpus
ndoc(corpus)
```

```
## [1] 787
```

Metadata such as speaker, date, etc. are stored in a corpus object as docvars, and can be accessed like so (we'll use the `head()` function to limit the output):

```
#date
head(docvars(corpus, "date"), 10)
```

```
##  [1] "18-12-2015" "10-12-2015" "09-12-2015" "07-12-2015" "07-12-2015"
##  [6] "01-12-2015" "28-11-2015" "23-11-2015" "19-11-2015" "16-11-2015"
```

```
#use the unique() function to check the number of unique speakers
unique(docvars(corpus, "speaker"))
```

```
## [1] "D. Cameron" "G. Brown"   "T. Blair"
```

```
#use the table() function to check the number of speeches for each speaker
table(docvars(corpus, "speaker"))
```

```
##
## D. Cameron   G. Brown   T. Blair
##        493        283         11
```

```
#subsetting a corpus is easy using the corpus_subset() function
cameron.corpus <- corpus_subset(corpus, speaker == "D. Cameron")
ndoc(cameron.corpus)
```

```
## [1] 493
```

Sometimes it can be useful to tokenize your corpus. You can do this using the `tokens()` function.

```
tokens.speech <- tokens(corpus)
```

NB: the `tokens()` function generates a list that contains token vectors for all documents in the corpus. A list is an `R` object that can contain vectors of various lengths and types and as such is different from a dataframe which can contain variables of multiple types but with equal length.

Let's turn the corpus into a dfm using the `dfm()` function in `quanteda`:

```
corpus.dfm <- dfm(corpus, stem = FALSE, remove=stopwords("english"), remove_punct=TRUE)

#you can check the number of features in the dfm using the dim() function
dim(corpus.dfm)
```

```
## [1]   787 44533
```

Quanteda makes it very easy to inspect a dfm. For example, the `topfeatures()` function displays the most occurring features:

```
topfeatures(corpus.dfm, 20)
```

```
##    people       can     think     world      want      make      just
##     10628      9617      8848      5443      5119      4967      4817
##       now      also      need       one   country       new     going
##      4632      4512      4423      4342      4342      4305      4257
## countries   britain     prime        us  minister      work
##      4077      4056      4045      3921      3898      3884
```

You can check the number of features in the dfm using the dim() function:

```
dim(corpus.dfm)
```

```
## [1]   787 44533
```

There are over 44,000 features in this dfm. Let's select those tokens that appear in at least 10 documents by using the `dfm_trim()` function

```
corpus.dfm = dfm_trim(corpus.dfm, min_docfreq = 10)
dim(corpus.dfm)
```

```
## [1]  787 6104
```

As you can see, this reduces the size of the dfm considerably. However, be mindful that applying such arbitrary cutoffs may

NB: Because most words don't occur in most documents, a dtm often contains many zeroes (sparse). Internally, `quanteda` stores the dfm in a sparse format, which means that the zeroes are not stored, so you can create a dtm of many documents and many words without running into memory problems.

Visualization in quanteda

`Quanteda` contains some very useful functions to plot your corpus in order get a feel for it. For example, it is easy to construct a wordcloud to see which features appear most often in your corpus (NB: don't overinterpret a word cloud)

```
textplot_wordcloud(corpus.dfm, max_words=100)
```

A more informative frequency plot can be constructed as follows (using the `ggplot2` library):

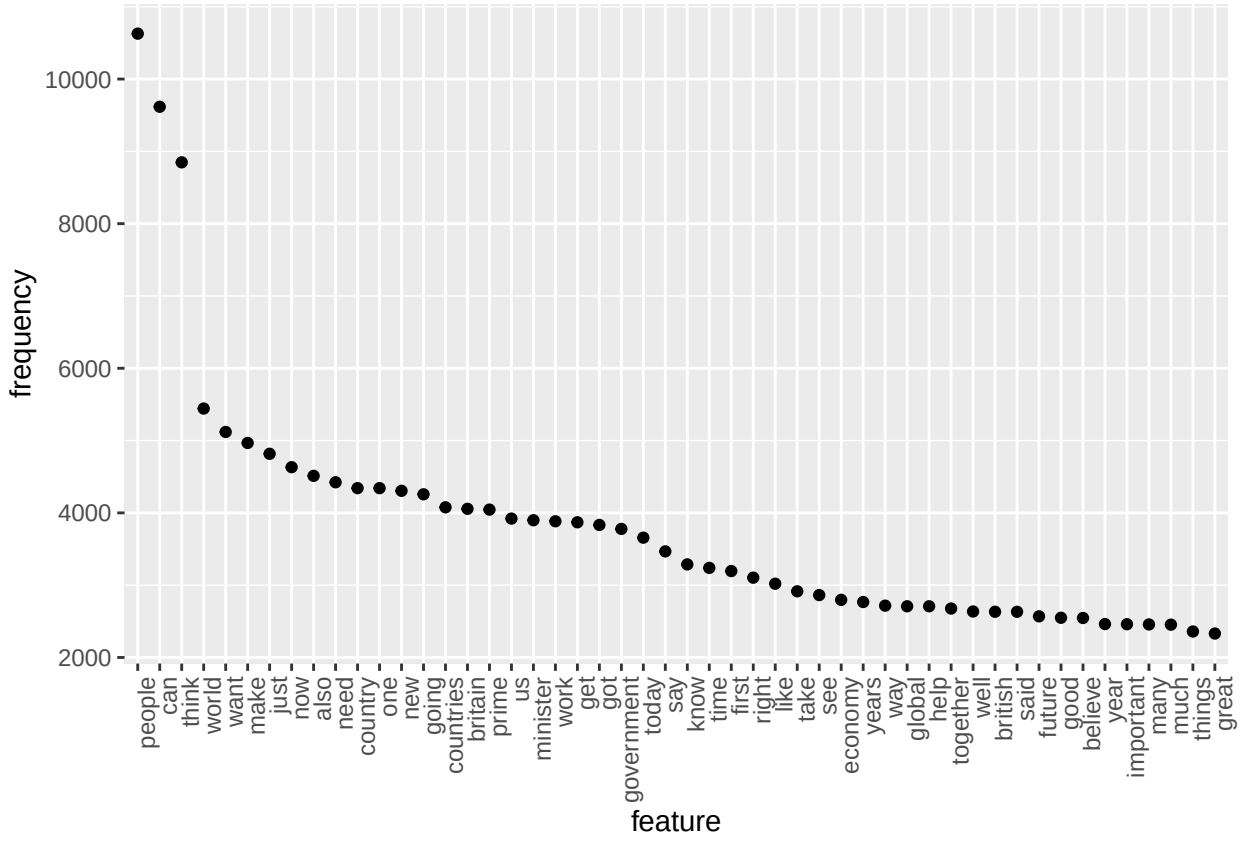
```
library(ggplot2)

corpus.dfm.features <- textstat_frequency(corpus.dfm, n = 50)

# Sort by reverse frequency order
corpus.dfm.features$feature <- with(corpus.dfm.features , reorder(feature, -frequency))

ggplot(corpus.dfm.features, aes(x = feature, y = frequency)) +
```

```
    geom_point() +
    theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

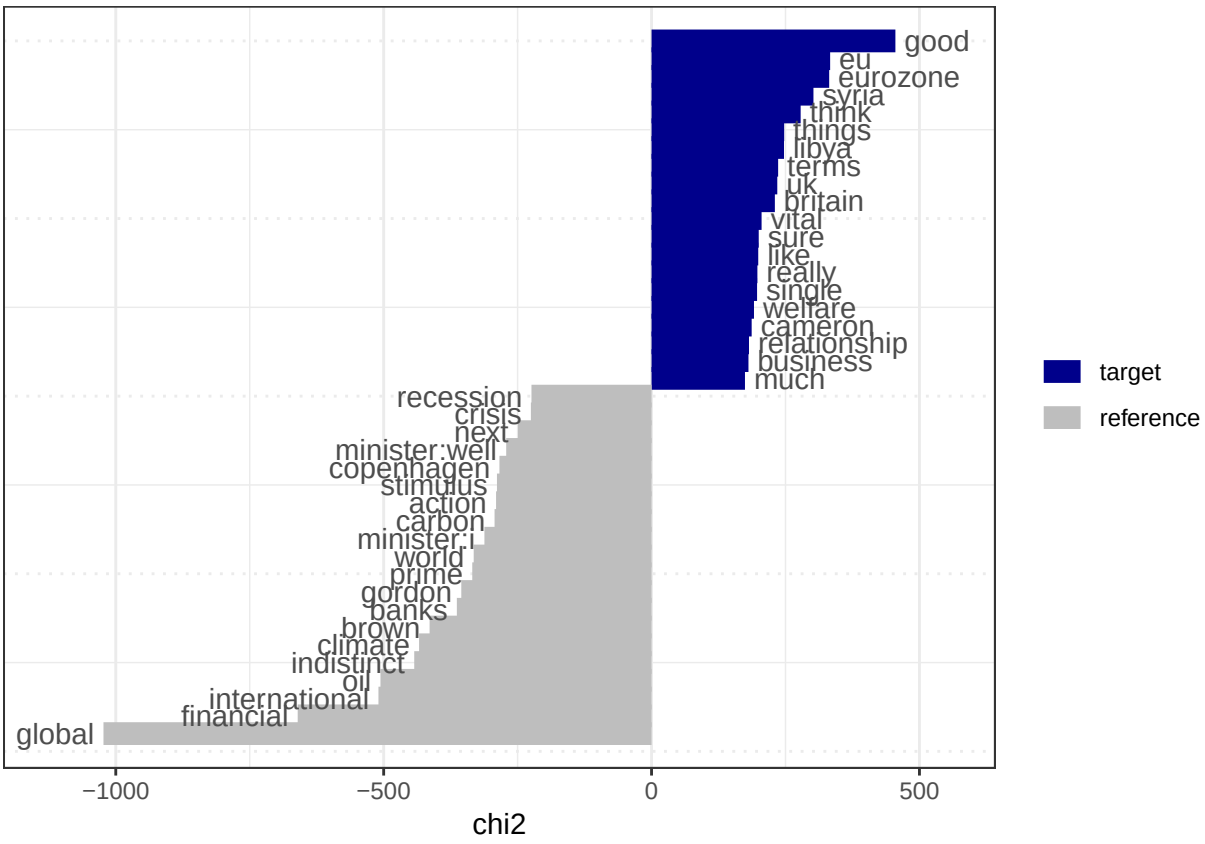


Let's say we are interested in which words are spoken relatively more often by David Cameron than by Tony Blair and Gordon Brown. For this we can use `textstat_keyness()` and `textplot_keyness()` functions.

```
cameron <- docvars(corpus, "speaker") == "D. Cameron"
comparison <- textstat_keyness(corpus.dfm, cameron)
head(comparison)
```

```
##    feature     chi2 p n_target n_reference
## 1     good 455.1106 0     1970         579
## 2       eu 333.2907 0      783         123
## 3 eurozone 331.5308 0      466          11
## 4    syria 302.1670 0      428          11
## 5    think 278.2528 0     5761        3087
## 6   things 247.5477 0     1708         651
```

```
textplot_keyness(comparison)
```



Question How would you interpret this plot?

Exercise

Take a look at this Quanteda tutorial for plotting frequency plots. Let's say you are interested in how often (in relative terms) David Cameron uses the word 'british' versus how often Gordon Brown uses it. Write some code to display this comparison in one plot.