Describing text

9 May 2019

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.

```
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"
```

```
#load libraries
library(quanteda)
library(stringr)

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

#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, "  ", " ")

#construct a corpus
corpus <- corpus(speeches)
```

Describing text

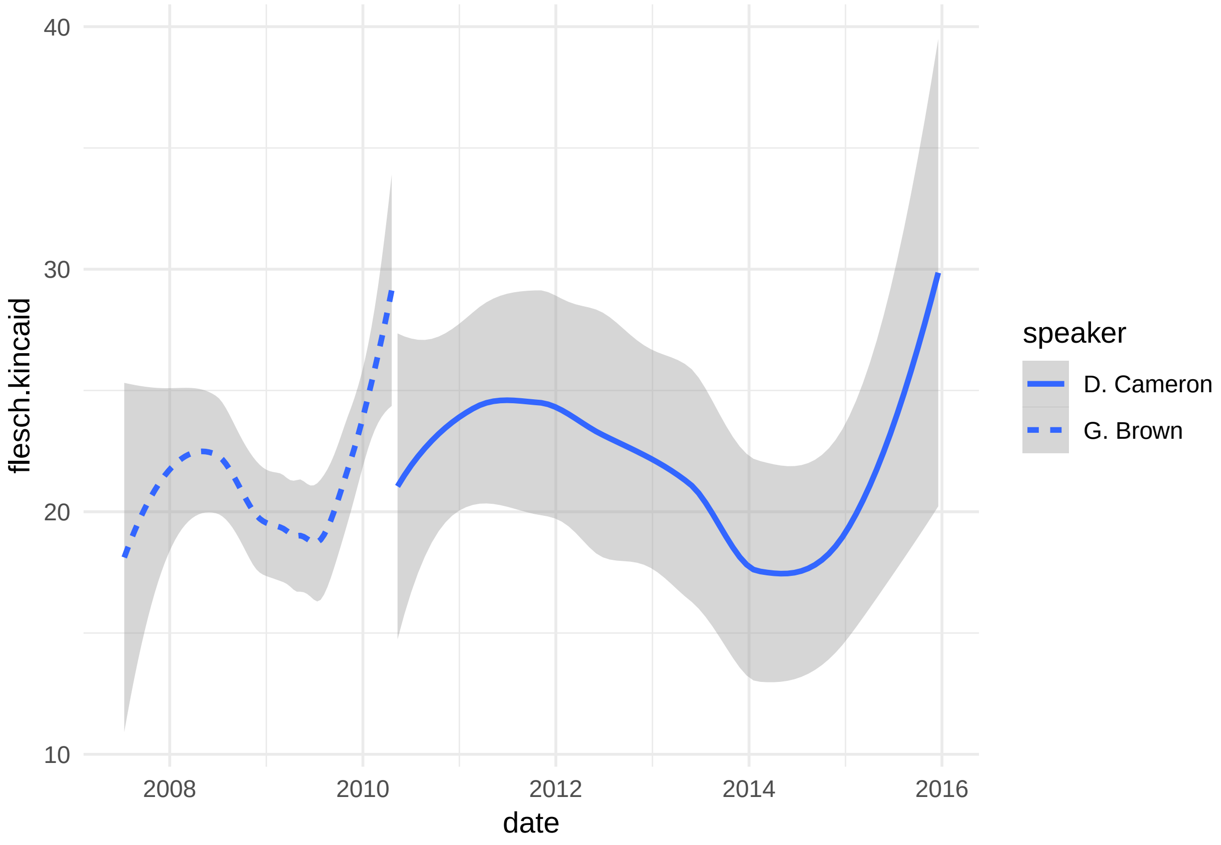
Let's first take a look at language readability, as measured by Flesch Kincaid. `Quanteda` contains quite a number of readability indices, which can be called with the `textstat_readability()` function. Since Flesch Kincaid is a weighted average of word length and sentence length it requires the corpus to contain interpunction.

```
docvars(corpus, "flesch.kincaid") <- textstat_readability(corpus, "Flesch.Kincaid")[,2]
```

Let's plot readability of PM speeches over time:

```
library(ggplot2)

#turn the date variable in a date format instead of character format
docvars(corpus, "date") <- as.Date(docvars(corpus, "date"), "%d-%m-%Y")

#let's look at Cameron and Brown
corpus.cameron.brown <- corpus_subset(corpus, speaker != "T. Blair")

#make plot using ggplot
```

```r
complexity.plot <- ggplot(docvars(corpus.cameron.brown),
                          aes(x = date,
                              y = flesch.kincaid,
                              linetype = speaker))

complexity.plot <- complexity.plot +
  geom_smooth() +
  theme_minimal()
print(complexity.plot)
```



Question: How would you interpret this plot?

We can use the `kwic()` to function to obtain the context in which certain keywords appear. Let's take a look at how David Cameron talked about Syria:

```r
corpus.cameron <- corpus_subset(corpus, speaker == "D. Cameron")
tokenized.corpus.cameron <- tokens(corpus.cameron)

syria.kw <- kwic(tokenized.corpus.cameron, 'syria')
nrow(syria.kw)
```

```
## [1] 428
```

```r
head(syria.kw)
```

```
##
##  [text1, 253] to resettle 22,000 refugees from | Syria |
##  [text1, 641] the efforts of the International | Syria |
##  [text1, 649]            to end the conflict in | Syria |
##  [text5, 581]    to the humanitarian crisis in | Syria |
##  [text5, 734]                  - in Iraq and in | Syria |
##  [text7, 734]           its importance? On the | Syria |
##
##  over 2 years and to
##  Support Group to end the
##  through a political process.And we
##  - providing£ 1.1 billion.And
##  where British fighter jets struck
##  vote, can you just
```

The search for keywords can be generalized using wild card expressions for pattern matches using `valuetype = "glob`. The search string can also contain regular expressions. In that case use `valuetype = "regex`.

```
syria.kw <- kwic(tokenized.corpus.cameron, 'syria*', valuetype = "glob")
nrow(syria.kw)
```

```
## [1] 614
```

```
head(syria.kw)
```

```
##
##  [text1, 162]   in humanitarian assistance for the |    Syrian    |
##  [text1, 253]     to resettle 22,000 refugees from |    Syria     |
##  [text1, 295] United Kingdom would resettle 20,000 |    Syrian    |
##  [text1, 316]          we have resettled over 1,000 |    Syrian    |
##  [text1, 378]             all those affected by the |    Syrian    |
##  [text1, 607]         the instability in Libya and | Syria.That's |
##
##  conflict and deployed HMS Enterprise
##  over 2 years and to
##  refugees during this Parliament.And we
##  refugees from camps in Turkey
##  crisis with the conference we
##  why it's so important that
```

Using `kwic(tokenized.corpus.cameron, 'syria*', valuetype = "glob")`, the number of hits increases considerably, now including versions on the syria keyword.

```
table(syria.kw$keyword)
```

```
##
##              Syria      Syria-related          Syria.And
##                428                  1                  2
##      Syria.Britain          Syria.But        Syria.Could
##                  1                  3                  1
##          Syria.HMS            Syria.I           Syria.It
##                  1                  3                  1
##          Syria.Let           Syria.Mr          Syria.Now
##                  3                  1                  4
##           Syria.So         Syria.That       Syria.That's
##                  1                  1                  1
##        Syria.There         Syria.This           Syria.We
##                  1                  2                  6
##         Syria.Well Syria.Yes.Recently    Syria.Yes.There
##                  1                  1                  1
##            Syria's          SyriaGood             Syrian
##                  10                    1                  121
##             Syrians         Syrians.That        Syrians.Third
##                  16                    1                    1
```

As you can see, there are a few instances of incorrect interpunction, where one sentence ends with Syria but there is no white space in between it and the next sentence.

Question: How would you use 'stringr' and regular expression to ensure proper white spaces in between sentences?

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

```
corpus.dfm <- dfm(corpus, stem = TRUE, remove=stopwords("english"), remove_punct=TRUE)
#let's select those tokens that appear in at least 10 documents
corpus.dfm = dfm_trim(corpus.dfm, min_docfreq = 10)
corpus.dfm
```

```
## Document-feature matrix of: 787 documents, 4,205 features (89.5% sparse).
```

The `textstat_simil()` function lets you calculate the cosine similarity between individual speeches.

```
similarity <- textstat_simil(corpus.dfm, margin = "documents" , method = "cosine")
```

You can inspect which speeches are most similar to each other according to their cosine similarity score like so:

```
similarity <- as.matrix(similarity)
which(similarity == max(similarity), arr.ind = TRUE)
```

```
##         row col
## text654 654 652
## text652 652 654
```

You can also use vector correlations as a similarity measure:

```
similarity <- textstat_simil(corpus.dfm, margin = "documents" , method = "correlation")
similarity <- as.matrix(similarity)
which(similarity == max(similarity), arr.ind = TRUE)
```

```
##         row col
## text719 719 717
## text717 717 719
```

Question: Cosine similarity and correlation point to different most similar speeches. Take a look at those speeches. What could be a reason that these scores differ?

Instead of similarity between speeches, you can also calculate the similarity between features. For example, what words are most similar to Syria?

```
similarity <- textstat_simil(corpus.dfm, c("syria"), margin = "features" , method = "correlation")
similarity <- as.matrix(similarity)
similarity <- similarity[order(similarity[,1], decreasing = TRUE),]
head(similarity, 10)
```

```
##     syria     assad    syrian   opposit    geneva   embargo     lough
## 1.0000000 0.8188982 0.7231585 0.5687997 0.5300491 0.5228524 0.4950468
##   transit     putin       ern
## 0.4735480 0.4529267 0.4342788
```

Instead of the similarity, you can calculate the euclidian distance between words or speeches:

```r
distance <- textstat_dist(corpus.dfm, margin = "documents" , method = "euclidean")
distance <- as.matrix(distance)
which(distance == max(distance), arr.ind = TRUE)
```

```
##         row col
## text679 679 634
## text634 634 679
```

For more similarity or distance measures check the `textstat_simil` and `textstat_dist` reference manual by typing `?textstat_simil` or `?textstat_dist` in your console.