Reading in and cleaning the data

Tom Paskhalis

29 November, 2018

Introduction

Text data comes in many different forms and shapes. It can be manually transcribed by a researcher (worst-case scenario), extracted from PDF files, converted from DOC/DOCX/ODT/RTF and other text formats, downloaded from an online API in JSON, before it can be easily read in as a TXT file. In an ideal world we would only deal with text data that has been already pre-processed for us and we just need to load the respective CSV, RData or RDS file.

A few packages that we will be using for reading in the data:

```
library("pdftools")
library("stringr")
library("readtext")
library("quanteda")
```

```
## Package version: 1.3.14
```

```
## Parallel computing: 2 of 8 threads used.
```

```
## See https://quanteda.io for tutorials and examples.
```

```
##
## Attaching package: 'quanteda'
```

```
## The following object is masked from 'package:utils':
##
##     View
```

```
library("tabulizer")
```

If you don't have them installed, do `install.packages("package_name")`.

PDF/Online

We won't be working with the hardest case, where you would have to manually input all your data, but although extremely tedious, it's relatively straight-forward on the technical side. But let's try the second hardest case, PDF files downloaded from an online source. An example here would be *Draft Agreement on the withdrawal of the United Kingdom from the European Union*, published just a couple of weeks ago. It resides only here and an offline copy is available in the folder `data`. To parse the text layer of the PDF file we will use the package `pdftools`. We will later see how it can be done with `readtext`, which provides an interface to many more file formats.

```
draft <- pdftools::pdf_text("../data/draft_withdrawal_agreement_0.pdf")
length(draft)
```

```
## [1] 585
```

As you probably read in the news, the draft agreement is *585* pages long, so `pdf_text` returns a 585-element character vector, where each element is raw text from a page. Let's have a look at the first page:

```
first <- draft[1]
first
```

```
## [1] "                                                                14 November 2018\n
```

There are a few things to note. The text contains newline characters (`\n`) and an uniformative `& /en 1` at the bottom of the page. To remove them we'll use regular expressions in the base R (but check `stringr` package for newer implementation). Regular expression deserve a session their own, but we will limit our discussion to this example. Let's first remove all whitespaces. We're using a class of characters `[[:space:]]` as a pattern to replace, which refers to any kind of whitespace or invisible character. More information on regular expresions in R is available here. `+` at the end indicates that whitespace should occur one or more times.

```
first <- gsub("[[:space:]]+", " ", first)
first
```

```
## [1] " 14 November 2018 TF50 (2018) 55 - Commission to EU27 Subject: Draft Agreement on the withdrawal
```

Now, let's remove the page numbering. We're removing all numbers (`[0-9]`) that occur between 1 and 3 times (`{1,3}`) and follow the string `& /en`.

```
first <- gsub("& /en [0-9]{1,3}", "", first)
first
```

```
## [1] " 14 November 2018 TF50 (2018) 55 - Commission to EU27 Subject: Draft Agreement on the withdrawal
```

This looks much better! Don't you agree? Let's now apply this strategy to all pages in the document.

```
draft <- gsub("[[:space:]]+", " ", draft)
draft <- gsub("& /en [0-9]{1,3}", "", draft)
draft[3]
```

```
## [1] "RECALLING that, pursuant to Article 50 TEU, in conjunction with Article 106a of the Euratom Trea
```

Challenge 1

Easy mode Now extract all the Directives mentioned in the Withdrawal Agreement. You can either use `str_extract_all` function from `stringr` package. Otherwise, use `gregexpr` function from base R and pass its output to `regmatches`. The solution should be able to detect such directives as **Directive 92/84/EEC**, **Directive 2011/64/EU** and **Directive 2008/118/EC**. We can then use the extracted directives as a back-of-the-envelope 'topic model' to understand the areas that the draft Withrawal Agreement concentrated on.

Medium: Compute a frequency table of directives mentioned and return the first 10 and their corresponding frequencies. Also, show the total number of cited directives.

Advanced: Produce a bar chart showing the frequencies of the top 10 directives.

Subject expert Explore the issues to which the most mentioned directives refer.

Alternatives

readtext

While `pdf_text` from `pdftools` works well with the PDF documents, `readtext` offers a more versatile approach, allowing to pass different file formats, including PDF (`pdftools` is still used in the backend).

```
draft <- readtext::readtext("../data/draft_withdrawal_agreement_0.pdf")
head(draft)
```

```
## readtext object consisting of 1 document and 0 docvars.
## # data.frame [1 x 2]
##   doc_id                          text
## * <chr>                           <chr>
## 1 draft_withdrawal_agreement_0.pdf "\"          \"..."
```

After Merging all of the pages together, `readtext` returns a 2-column data frame. This approach is very useful for reading in a large number of documents, which all have identical structure, and creating a corpus out of them.

```
draft_corpus <- quanteda::corpus(draft)
summary(draft_corpus)
```

```
## Corpus consisting of 1 document:
##
##                             Text Types Tokens Sentences
##  draft_withdrawal_agreement_0.pdf  6713 127114      2961
##
## Source: /home/tpaskhalis/Decrypted/Git/VAM_Text_Analysis/code/* on x86_64 by tpaskhalis
## Created: Wed Nov 28 21:47:58 2018
## Notes:
```

```
draft_dfm <- quanteda::dfm(draft_corpus,
                           tolower = TRUE,
                           stem = FALSE,
                           remove_punct = TRUE)
summary(draft_dfm)
```

```
## Length  Class   Mode
##   5850    dfm     S4
```

tabulizer

On the other end of the spectrum, instead of lumping together all of the pages from the entire document, you might want to extract just a part of a specific page. The package `tabulizer` has a handy function `extract_text` that allows to pass the interactively drawn area of a document and extract just this part.

For example, let's extract the first sections of Articles 4,6,7 (article 5 doesn't have any). The pages for those articles are 11-15.

```
#  First we interactively locate areas that contain these sections
areas <- tabulizer::locate_areas("./data/draft_withdrawal_agreement_0.pdf",
                                 pages = seq(11,15))
```

```
# Manually define areas as interactive mode isn't available when knitting
areas <- list(
  c(top = 275.79010,
    left = 47.54186,
    bottom = 435.68968,
    right = 544.84421),
  NULL,
  c(top = 291.92675,
    left = 46.07489,
```

```
    bottom = 375.54396,
    right = 541.91028),
  c(top = 230.31407,
    left = 46.07489,
    bottom = 469.42995,
    right = 560.98087),
  c(top = 49.87693,
    left = 46.07489,
    bottom = 145.22989,
    right = 541.91028)
)
# Slightly convoluted setup needed to avoid pages with no areas defined
first_sections <- tabulizer::extract_text("../data/draft_withdrawal_agreement_0.pdf",
                                  pages = seq(11,15)[!unlist(lapply(areas, is.null))],
                                  area = `[`(areas, which(!unlist(lapply(areas, is.null)))))
first_sections
```

```
## [1] "1. The provisions of this Agreement and the provisions of Union law made applicable by this \nAg
## [2] " \n1. With the exception of Parts Four and Five, unless otherwise provided in this Agreement all
## [3] " \n1. For the purposes of this Agreement, all references to Member States and competent \nautho
## [4] "(c) the attendance in the meetings of the committees referred to in Article 3(2) of Regulation \
```