Sentiment analysis and LSF

Replication of Lima-Lopes (2020) - Part 2

Rodrigo Esteves de Lima-Lopes
State University of Campinas
rll307@unicamp.br

Contents

1 Introduction

This file will discuss specifically how to prepare our data for replicating Lima-Lopes (2020). Please note that we will not make the whole corpus available here due to copyright. If you choose to reproduce this paper all articles have to be scraped by you. It is also important to observe that some articles might have been taken offline (actually by the time of this replication some already have).

Our objective is threefold:

1. Prepare the corpora for sentiment analysis
2. Prepare the corpora for network analysis using R scripts
3. Prepare and export the corpora for network analysis using other software, more specifically Gephi. Unfortunately, a tutorial on Gephi is beyond the scope of this tutorial.

1.1 Packages

```
library(quanteda)
library(quanteda.textplots)
library(quanteda.textstats)
```

In this tutorial we are going to use three packages:

- `quanteda` - for text processing
- `quanteda.textplots` - for plotting graphs using `quanteda`
- `quanteda.textstats` - for text statistics

2 Building the corpora

Our first step is to load a personalised stopword list and convert it to a vector:

```
My.Stopwords <- readr::read_csv(file.choose())
My.Stopwords  <- My.Stopwords |>
  purrr::as_vector()
```

In the code above:

1. `readr::read_csv(file.choose())` calls the package `readr`, its function `read_csv` which imports a *csv* file. The nested function `file.choose()` opens a window for choosing such a list.
2. `purrr::as_vector()` class the function `as_vector()` from the package `purrr` in order to convert my stopword list into a vector.

The next step is to make a corpus from each newspaper and a general containing all articles. A `corpus` is a formal class to run `quanteda` commands.

```
# one by one
TS.Corpus <- corpus(TS.df, text_field = 'Content')
DS.Corpus <- corpus(DS.df, text_field = 'Content')
TT.Corpus <- corpus(TT.df, text_field = 'Content')

#General
News.df <- rbind(TT.df,TS.df, DS.df)
News.Corpus <- corpus(News.df, text_field = 'Content')
```

Please note:

- The `field_command` tells `quanteda` what should be considered the text.
- The `doc_id` variable is now the formal identification of each text. Our next step is to create a `token` variable for each corpus. A `token` is an index of words and their positions within each file. Please note that we are also doing some cleaning, deleting numbers, URLs and punctuation. If your research depends on this elements, please keep them.

```
TS.tokens  <- tokens(TS.Corpus,
                    what = "word",
                    remove_punct = TRUE,
                    remove_symbols = TRUE,
                    remove_numbers = TRUE,
                    remove_url = TRUE,
                    split_hyphens = FALSE,
                    include_docvars = TRUE,
                    padding = FALSE,
                    verbose = TRUE
                    )

DS.tokens <- tokens(DS.Corpus,
                    what = "word",
                    remove_punct = TRUE,
                    remove_symbols = TRUE,
                    remove_numbers = TRUE,
                    remove_url = TRUE,
                    split_hyphens = FALSE,
                    include_docvars = TRUE,
                    padding = FALSE,
                    verbose = TRUE
                    )
TT.tokens <- tokens(TT.Corpus,
                    what = "word",
                    remove_punct = TRUE,
                    remove_symbols = TRUE,
                    remove_numbers = TRUE,
                    remove_url = TRUE,
                    split_hyphens = FALSE,
                    include_docvars = TRUE,
                    padding = FALSE,
                    verbose = TRUE
                    )
News.tokens <- News.Corpus |>
  tokens(what = "word",
         remove_punct = TRUE,
         remove_symbols = TRUE,
         remove_numbers = TRUE,
         remove_url = TRUE,
         split_hyphens = FALSE,
         include_docvars = TRUE,
         padding = FALSE,
         verbose = TRUE
  )
```

Now we are going to keep on cleaning and preparing the corpora. In the sequence below we: 1. Clean the corpora from two stopword lists: our personalised list and `quanteda`s built in list 1. Lemmatise our copora, using the lists available in the package `lexicon` - In this case, we do a simple substitution

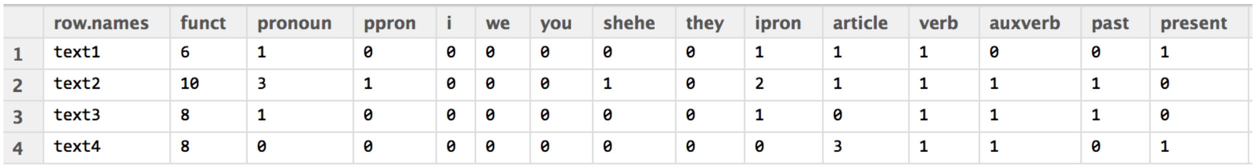
```
TS.tokens <- tokens_remove(TS.tokens, c(stopwords("english"),My.Stopwords))
TS.tokens <- tokens_replace(TS.tokens,
                            pattern = lexicon::hash_lemmas$token,
                            replacement = lexicon::hash_lemmas$lemma) |>
  tokens_tolower()

DS.tokens <- tokens_remove(DS.tokens, c(stopwords("english"),
                                        My.Stopwords))
DS.tokens <- tokens_replace(DS.tokens,
                            pattern = lexicon::hash_lemmas$token,
                            replacement = lexicon::hash_lemmas$lemma) |>
  tokens_tolower()

TT.tokens <- tokens_remove(TT.tokens, c(stopwords("english"),My.Stopwords))
TT.tokens <- tokens_replace(TT.tokens,
                            pattern = lexicon::hash_lemmas$token,
                            replacement = lexicon::hash_lemmas$lemma) |>
  tokens_tolower()


News.tokens <- tokens_remove(News.tokens, c(stopwords("english"),My.Stopwords))
News.tokens <- tokens_replace(News.tokens,
                            pattern = lexicon::hash_lemmas$token,
                            replacement = lexicon::hash_lemmas$lemma) |>
  tokens_tolower()
```

In order to get our data ready for the network analysis we will:

1. Creating a `dfm`
2. Choosing the more frequent lemmas (150 in this study)

A DFM is a space matrix which tells us which words are in each text of a corpus. It also tells us the frequency of such words.

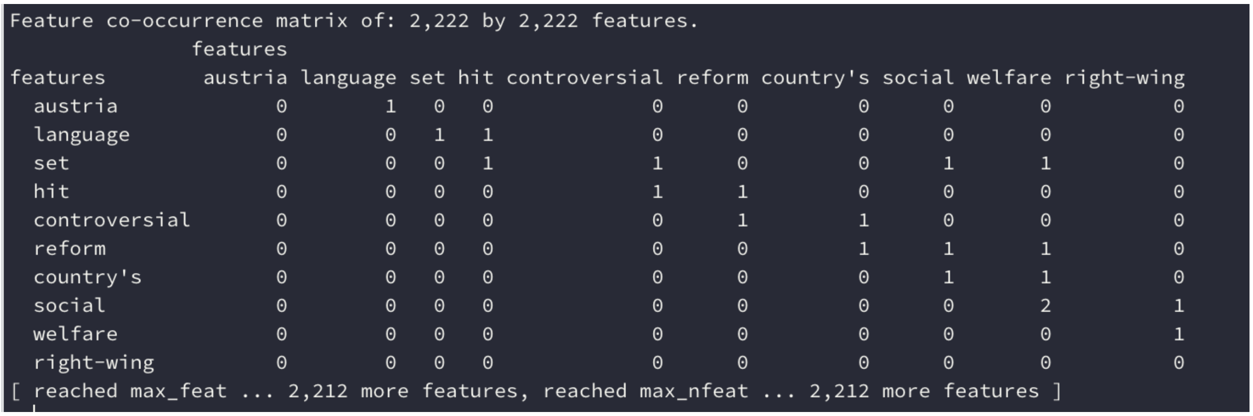
```r
news.dfm <- dfm(News.tokens)
news.top <- names(topfeatures(news.dfm, 150))

TT.dfm <- dfm(DS.tokens)
TT.top <- names(topfeatures(DS.dfm, 150))

DS.dfm <- dfm(DS.tokens)
DS.top <- names(topfeatures(DS.dfm, 150))

TS.dfm <- dfm(TS.tokens)
TS.top <- names(topfeatures(TS.dfm, 150))
```



	row.names	funct	pronoun	ppron	i	we	you	shehe	they	ipron	article	verb	auxverb	past	present
1	text1	6	1	0	0	0	0	0	0	1	1	1	0	0	1
2	text2	10	3	1	0	0	0	1	0	2	1	1	1	1	0
3	text3	8	1	0	0	0	0	0	0	1	0	1	1	1	0
4	text4	8	0	0	0	0	0	0	0	0	3	1	1	0	1

Figure 1: DFM example

Now our last pre-processing step: creating a fcm and filtering it with our top dfm results. A fcm is a matrix of co-occurrence of words inside a document or corpus.



```
Feature co-occurrence matrix of: 2,222 by 2,222 features.
               features
features        austria language set hit controversial reform country's social welfare right-wing
  austria             0        1   0   0             0      0         0      0       0          0
  language            0        0   1   1             0      0         0      0       0          0
  set                 0        0   0   1             1      0         0      1       1          0
  hit                 0        0   0   0             1      1         0      0       0          0
  controversial       0        0   0   0             0      1         1      0       0          0
  reform              0        0   0   0             0      0         1      1       1          0
  country's           0        0   0   0             0      0         0      1       1          0
  social              0        0   0   0             0      0         0      0       2          1
  welfare             0        0   0   0             0      0         0      0       0          1
  right-wing          0        0   0   0             0      0         0      0       0          0
[ reached max_feat ... 2,212 more features, reached max_nfeat ... 2,212 more features ]
```

Figure 2: FCM example

```r
TS.fcm <- fcm(
  TS.tokens,
  context = 'window',
  count = "frequency",
  window = 2L,
  weights = NULL,
  ordered = FALSE,
  tri = TRUE
)
TS.fcm.top <- fcm_select(TS.fcm , pattern = TS.top)

DS.fcm <- fcm(
  DS.tokens,
  context = 'window',
  count = "frequency",
  window = 2L,
  weights = NULL,
  ordered = FALSE,
  tri = TRUE
)

DS.fcm.top <- fcm_select(DS.fcm , pattern = DS.top)

TT.fcm <- fcm(
  TT.tokens,
  context = 'window',
  count = "frequency",
  window = 2L,
  weights = NULL,
  ordered = FALSE,
  tri = TRUE
)

TT.fcm.top <- fcm_select(TT.fcm , pattern = TT.top)

News.fcm <- fcm(
  News.tokens,
  context = 'window',
  count = "frequency",
  window = 2L,
  weights = NULL,
  ordered = FALSE,
  tri = TRUE
)

News.fcm.top <- fcm_select(News.fcm , pattern = news.top)
```

3 Plotting a network

First, we will export our data in order to plot them using Gephi:

```
#as a matrix
TS.CC <- as.matrix(TS.fcm.top)
DS.CC <- as.matrix(DS.fcm.top)
TT.CC <- as.matrix(TT.fcm.top)
News.CC <- as.matrix(News.fcm.top)

# Then exporting
write.csv(TS.CC,"TS.csv")
write.csv(DS.CC,"DS.csv")
write.csv(TT.CC,"TT.csv")
write.csv(News.CC,"News.csv")
```

then plot each `fcm` using `textplot_network()` command

```
textplot_network(News.fcm.top, min_freq = 4, edge_color = "orange", edge_alpha = 0.8, edge_size = 5)
textplot_network(TS.fcm.top, min_freq = 4, edge_color = "green", edge_alpha = 0.8, edge_size = 5)

textplot_network(DS.fcm.top, min_freq = 3, edge_color = "darkblue", edge_alpha = 0.8, edge_size = 5)

textplot_network(TT.fcm.top, min_freq = 3, edge_color = "red", edge_alpha = 0.8, edge_size = 3)
```

The results are:

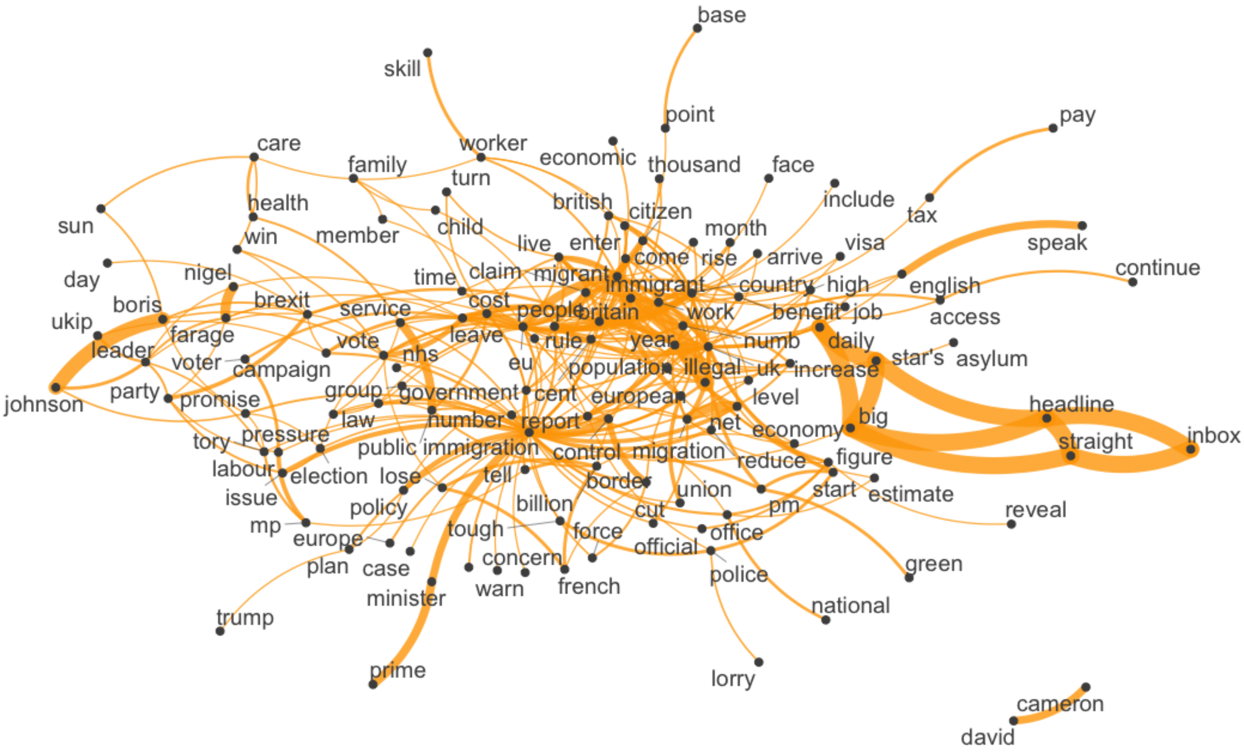


Figure 3: General Corpus

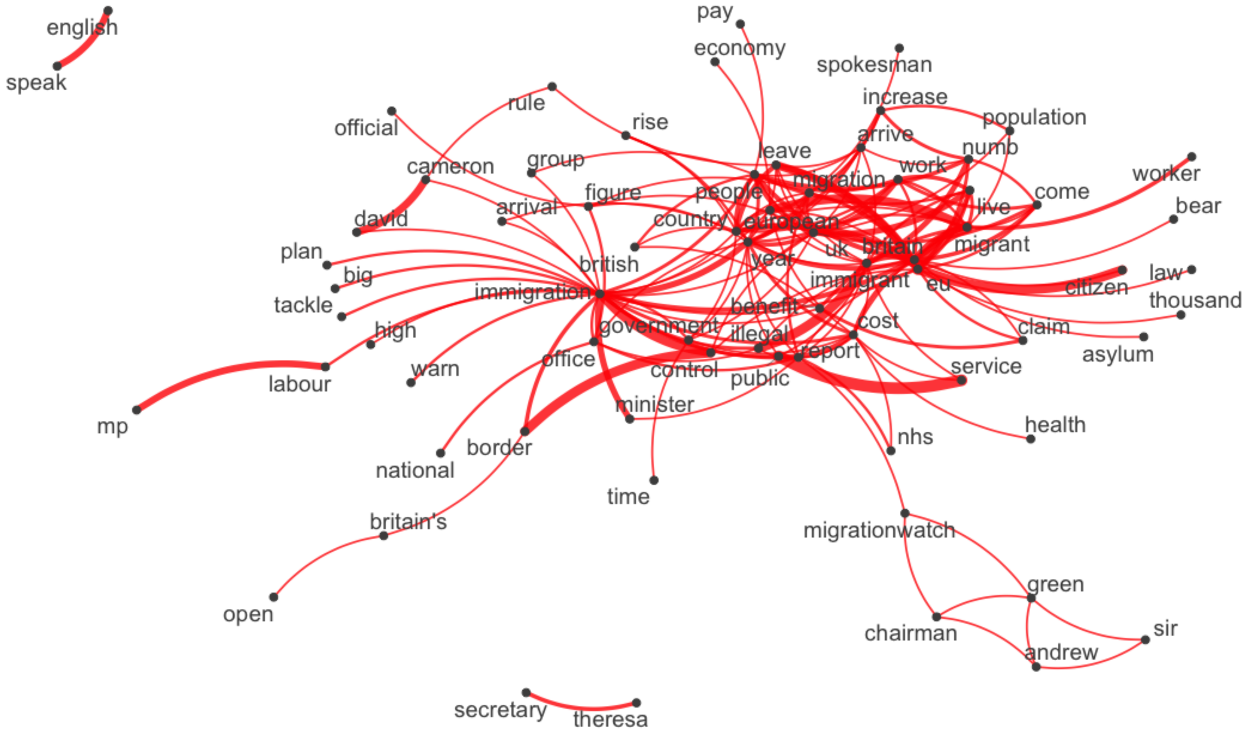


Figure 4: The Telegraph

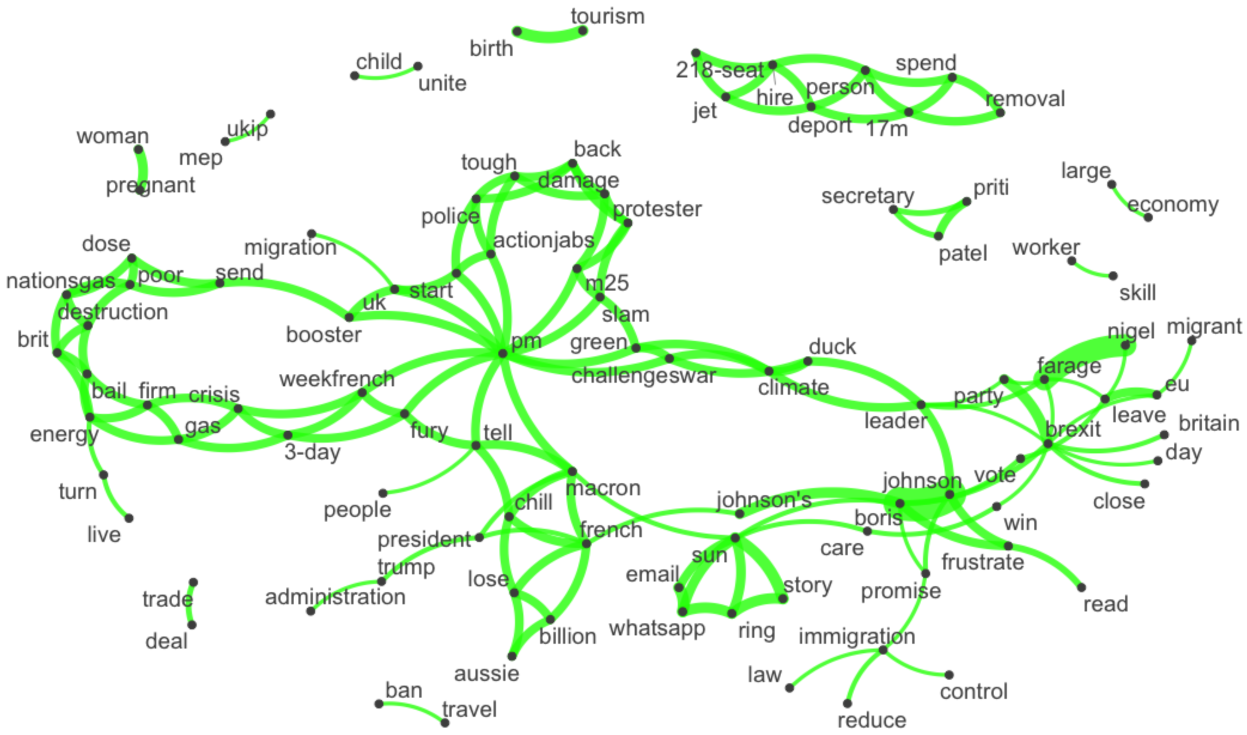


Figure 5: The Sun

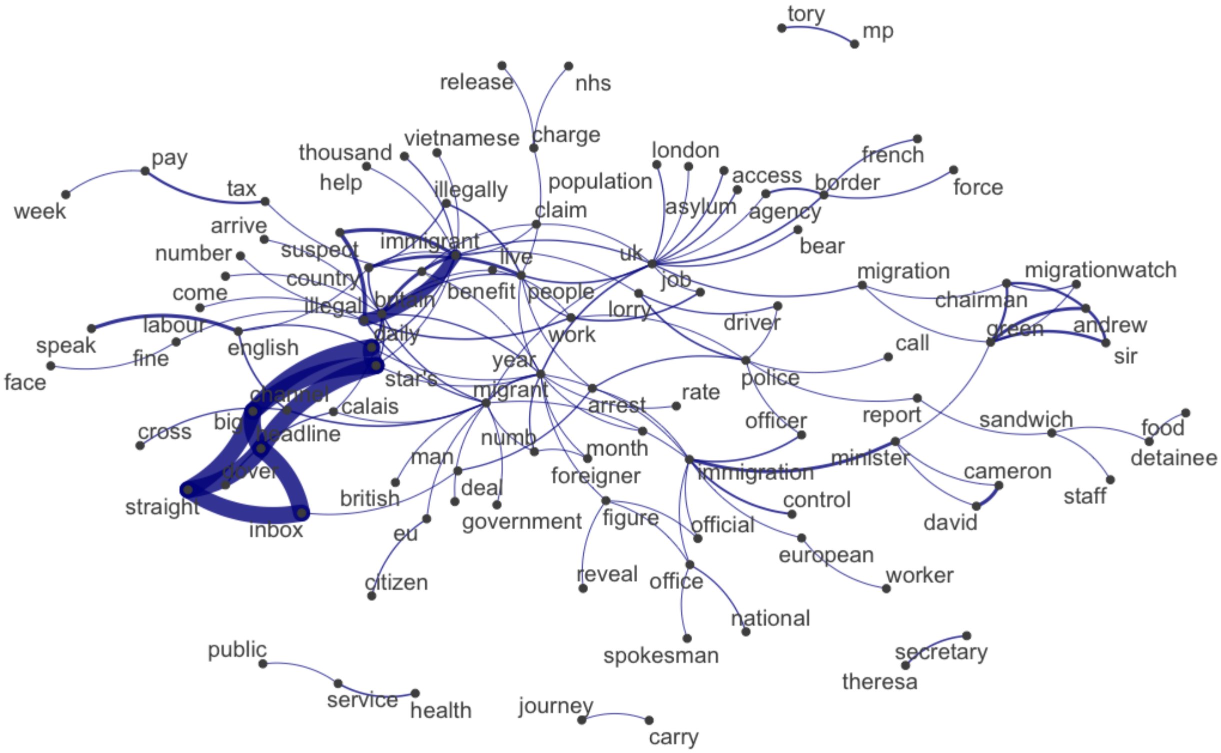


Figure 6: Daily Star