Regulatory documents via LDA

Sebastian Knigge

18 8 2019

Initial Setup

Following libraries are used in the code:

```
library(dplyr)
library(tidytext)
library(pdftools)
library(tidyr)
library(stringr)
library(tidytext)
library(udpipe)
library(topicmodels)
library(ggplot2)
library(wordcloud)
library(tm)
library(SnowballC)
library(RColorBrewer)
library(RCurl)
library(XML)
```

In this code reulatory documents are red in and processed via LDA. This first part focusses on reading in the pdf documents.

```
# getting the right directiory
library(here)
setwd("../")
path <- getwd() %>%
  file.path("TextDocs")
documents <- list.files(path)
```

Following functions are used to set up and analyze the pdfs.

```
read_pdf_clean <- function(document){
  # This function loads the document given per name
  # and excludes the stop words inclusive numbers
  pdf1 <- pdf_text(file.path(path, document)) %>%
    strsplit(split = "\n") %>%
    do.call("c",.) %>%
    as_tibble() %>%
    unnest_tokens(word,value) %>%
    # apply a filter for ^
    filter(!grepl("^",word))
  # load stopword library
  data(stop_words)
  # add own words to stop word library - here the numbers from 1 to 10
  new_stop_words <- tibble(word=as.character(0:9),
                           lexicon=rep("own",10)) %>%
    bind_rows(stop_words)
  pdf1 %>%
    anti_join(new_stop_words)
}

plot_most_freq_words <- function(pdf, n=5){
  # plots a bar plot via ggplot
  pdf %>% count(word) %>% arrange(desc(n)) %>% head(n) %>%
    ggplot(aes(x=word,y=n)) +
    geom_bar(stat="identity")+
    # no labels for x and y scale
    theme(axis.title.y=element_blank(),
          axis.title.x=element_blank())
}
```

Now we can read in all documents in a for loop:

```
# inital set up for the corpus
pdf1 <- read_pdf_clean(documents[1])
corpus <- tibble(document=1, word=pdf1$word)
# adding the documents iteratively
for (i in 2:length(documents)){
  pdf_i <- read_pdf_clean(documents[i])
  corpus <- tibble(document=i, word=pdf_i$word) %>% bind_rows(corpus,.)
}
```

LDA

The LDA model is applied. First the document term matrix has to be set up.

```
dtm <- corpus %>% count(document, word, sort = TRUE) %>%
  select(doc_id=document, term=word, freq=n) %>%
  document_term_matrix()
```

Using the function LDA sets up the model and prediction/evaluation is done via predict(). The tables below summarize which document refers to which topic, according to the LDA model.

```
set.seed(123)
documents_lda <- LDA(dtm,
                     k = 5, control = list(seed = 1234))
prediction5 <- predict(documents_lda, newdata=dtm, type="topic")$topic
```

Table 1: Documents for Topic 1

Group	Doc
1	5
1	6
1	10
1	16
1	21
1	23
1	27
1	28

Table 2: Documents for Topic 2

Group	Doc
2	1
2	4

Table 3: Documents for Topic 3

Group	Doc
3	12
3	13
3	15
3	19
3	20
3	22
3	24
3	26

Table 4: Documents for Topic 4

Group	Doc
4	3
4	7
4	9
4	11
4	14
4	17
4	25

Table 5: Documents for Topic 5

Group	Doc
5	2
5	8
5	18

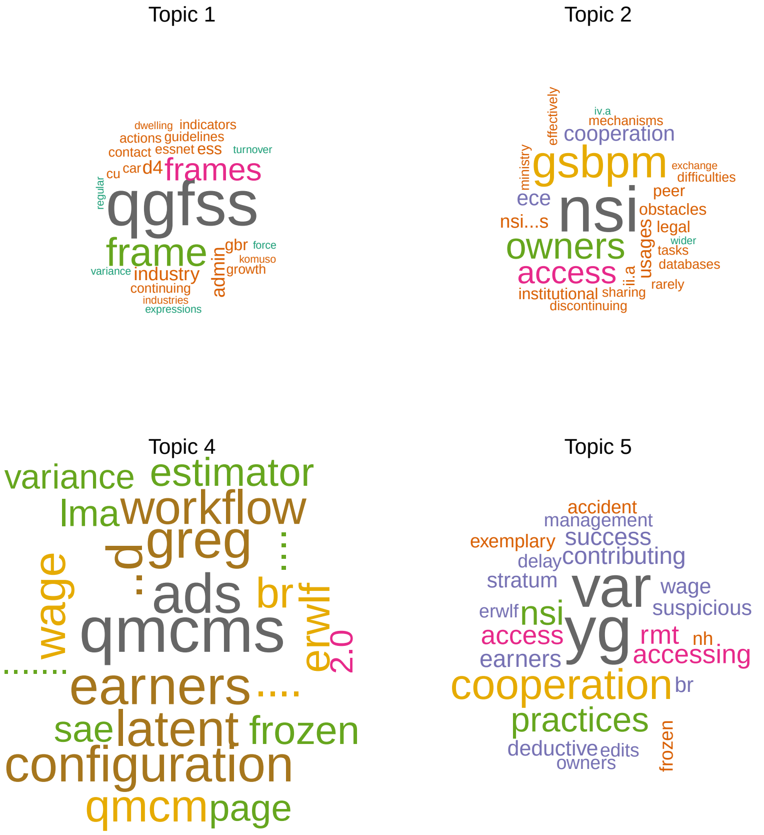
Wordclouds

To check what topics tackle which context, we produce wordclouds using the TFIDF and the TF itself.

```
plot_wordcloud <- function(corpus, selection="ALL", max.words=25, i, freq="tfidf"){
  # setting up a tibble which returns tfidf and tf and frequency for
  # the whole corpus
  tfidf <- corpus %>% count(document, word, sort = TRUE) %>%
    bind_tf_idf(word, document, n)
  # include all documents for selection if selection="ALL"
  if (all(selection=="ALL")) {
    selection <- corpus %>%
      select(document) %>%
      unique() %>%
      unlist() %>%
      sort()}
  # filter for all selected documents
  # use either ft or tfidf
  if (freq=="tfidf"){
    dtm_selected <- tfidf %>% filter(document%in%selection) %>%
      select(word, tf_idf) %>% count(word, wt=tf_idf, sort=TRUE)
  } else {
    dtm_selected <- tfidf %>% filter(document%in%selection) %>%
      select(word, tf) %>% count(word, wt=tf, sort=TRUE)
  }
  wordcloud(words = dtm_selected$word, freq = dtm_selected$n, min.freq = 1,
            max.words=max.words, random.order=FALSE,
            colors=brewer.pal(8, "Dark2"), scale=c(3,0.2),
            main="Title", use.r.layout = TRUE)
  text(x=0.5, y=1, paste("Topic", i))
}
```

For getting specific and more individual words for each cloud, we use the TFIDF in the first step.

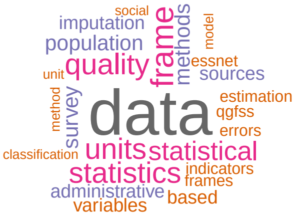
```
# compare topic 1 with topic 2, 3, 4 and 5
ind1 <- which(prediction5==1)
ind2 <- which(prediction5==2)
ind3 <- which(prediction5==3)
ind4 <- which(prediction5==4)
ind5 <- which(prediction5==5)

par(mfrow=c(2,3))
par(mar=c(1,1,0.5,1))
plot_wordcloud(corpus, selection=ind1, i=1)
plot_wordcloud(corpus, selection=ind2, i=2)
plot_wordcloud(corpus, selection=ind3, i=3)
plot_wordcloud(corpus, selection=ind4, i=4)
plot_wordcloud(corpus, selection=ind5, i=5)
```



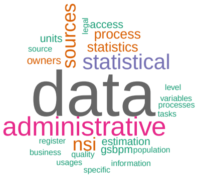
The same can be done using the regular term frequency.

```r
par(mfrow=c(2,3))
par(mar=c(1,1,0.5,1))
plot_wordcloud(corpus, selection=ind1, i=1, freq="tf")
plot_wordcloud(corpus, selection=ind2, i=2, freq="tf")
plot_wordcloud(corpus, selection=ind3, i=3, freq="tf")
plot_wordcloud(corpus, selection=ind4, i=4, freq="tf")
plot_wordcloud(corpus, selection=ind5, i=5, freq="tf")
```

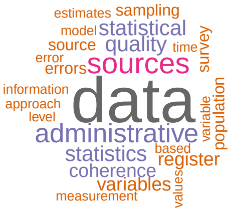
Topic 1



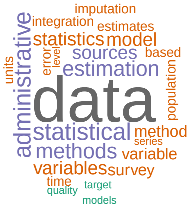
Topic 2



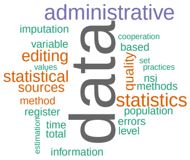
Topic 3



Topic 4



Topic 5


administrative
imputation
cooperation
variable
based
editing
set
values
practices
statistical
quality
nsi
sources
methods
data
method
statistics
register
population
errors
time
total
level
estimation
information