

Regulatory Documents via LDA - Documentation

Sebastian Knigge

18 8 2019

Contents

1 Setup

2 Import data

3 LDA

3.1	Wordclouds	
3.1.1	Wordclouds using <i>tf-idf</i>	
3.1.2	Wordclouds using <i>tf</i>	
3.2	Embedding via <i>tf-idf</i>	
3.3	Misclassification Rates	

4 Optimal value of Reduction via *tfidf*

5 Coherence Cloud

1 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)
library(openxlsx)
library(keras)
library(foreach)
library(doParallel)
```

2 Import data

The Documents had to be preprocessed. For the documents wp2.5 all list of contents had to be deleted, because they were the same in each of these documents. No more adjustments had to be made.

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

```
# getting the right order
setwd('..')
documents <- read.xlsx("Docs_classes.xlsx")[,2]
classes <- read.xlsx("Docs_classes.xlsx")[,c(1,3)]
documents <- paste0(documents, ".pdf")
documents %>% as.data.frame() %>% stargazer(summary=FALSE,
                                             header = FALSE,
                                             title="Document Titles")
```

Table 1: Document Titles

	.
1	admin-wp1.1_analysis_legal_institutional_environment_final.pdf
2	admin-wp1.2_good_practices_final.pdf
3	admin-wp2.1_estimation_methods1.pdf
4	admin-wp2.2_estimation_methods2.pdf
5	admin-wp2.3-estimation_methods3.pdf
6	admin-wp2.4_examples.pdf
7	admin-wp2.5_alignment.pdf
8	admin-wp2.5_editing.pdf
9	admin-wp2.5_greg.pdf
10	admin-wp2.5_imputation.pdf
11	admin-wp2.5_macro_integration.pdf
12	admin-wp2.5_matching.pdf
13	admin-wp2.6_good_practices.pdf
14	admin-wp2.6_guidelines.pdf
15	admin-wp3.1_quality1.pdf
16	admin-wp3.2_quality2.pdf
17	admin-wp3.3_quality.pdf
18	admin-wp3.4_quality.pdf
19	admin-wp3.5_quality_measures.pdf
20	admin-wp3_coherence.pdf
21	admin-wp3_growth_rates.pdf
22	admin-wp3_suitability1.pdf
23	admin-wp3_suitability2.pdf
24	admin-wp3_suitability3.pdf
25	admin-wp3_uncertainty.pdf
26	admin-wp5_frames.pdf
27	admin-wp5_frames_examples.pdf
28	admin-wp5_frames_recommendation.pdf

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

```
setwd(path)
```

Following functions are used to set up and analyze the pdfs. When cleaning up data, we have to take into account certain circumstances of the regulatory documents. For example, there are many formulas and technical abbreviations in the documents. Every variable, every estimator, and every index is included as a single word in the bag of words. These terms sometimes have a big influence on the documents, because they are very specific for individual documents and occur quite often. To avoid this, we exclude all mixed words with characters and numeric values, as well as all terms with special characters (e.g. Greek letters).

```
read_pdf_clean <- function(document){
  # This function loads the document given per name
  # and excludes the stop words
  pdf1 <- pdf_text(file.path(path, document)) %>%
    strsplit(split = "\n") %>%
    do.call("c",.) %>%
    as_tibble() %>%
    unnest_tokens(word,value) %>%
    # also exclude all words which include numbers and special characters
    filter(grepl("[a-z]+$", word))
  # load stopwords library
  data(stop_words)
  # stop words are excluded via anti_join
  pdf1 %>%
    anti_join(stop_words)
}

plot_most_freq_words <- function(pdf, n=7){
  # 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 using a for loop:

```
setwd(path)
# initial 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,.)
}
```

3 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()
# dimensions
c(N,M) %<-% dim(dtm)
N; M

```

```
## [1] 28
```

```
## [1] 8254
```

We use term frequency 2 embedding because in the example with the Gutenberg Data, it turned out to be advantageous with regard to the “predictive power” of the LDA algorithm.

```

dtm_tf2 <- dtm %>%
  # reduce by low frequencies
  dtm_remove_lowfreq(minfreq = 2)
ncol(dtm_tf2 )

```

```
## [1] 5842
```

Using the function LDA sets up the model and prediction/evaluation is done via *predict()*. But first of all it shall be verified whether the Predict function actually delivers the same classification as the export of the gamma matrix directly from the LDA model. Therefore both gamma matrices of the single functions are compared. Table 2 displays the output of the gamma matrix received by the *predict()* function and Table 3 displays the gamma matrix returned by the LDA model itself.

```

tim1 <- Sys.time()
set.seed(123)
documents_lda <- LDA(dtm_tf2, method = "Gibbs",
                     k = 7, control = list(seed = 1234))
tim2 <- Sys.time()
u1 <- tim2 - tim1

```

```
prediction5 <- predict(documents_lda, newdata=dtm_tf2, type="topic")
```

```
prediction5 <- merge(prediction5, classes, by.x="doc_id", by.y="No")
```

```

prediction5 %>%
  select(doc_id,topic_001,topic_002,topic_003,topic_004,topic_005,
         topic_006, topic_007) %>%
  mutate_each(funs(as.numeric),
              doc_id,topic_001,topic_002,topic_003,topic_004,
              topic_005, topic_006, topic_007) %>%
  arrange(desc(-doc_id)) %>%
  round(2) %>%
  stargazer(summary=F, rownames = F, header = F,
            title="Gamma matrix for predict function", label="predict")

```

```

ext_gamma_matrix <- function(model=documents_lda){
  # get gamma matrix for chapter probabilities
  chapters_gamma <- tidy(model, matrix = "gamma")
  # get matrix with probabilities for each topic per chapter
  spreaded_gamma <- chapters_gamma %>% spread(topic, gamma)
  spreaded_gamma %>%
    mutate_each(funs(as.numeric), document,1,2,3,4,5,6,7) %>%
    arrange(desc(-document))
}

```

Table 2: Gamma matrix for predict function

doc_id	topic_001	topic_002	topic_003	topic_004	topic_005	topic_006	topic_007
1	0	0.950	0	0.030	0.010	0	0.010
2	0.010	0.760	0.010	0.160	0.030	0.030	0.010
3	0.160	0.020	0.060	0.030	0.630	0.050	0.050
4	0.180	0.010	0.030	0.020	0.710	0.020	0.040
5	0.720	0	0.080	0.010	0.160	0.020	0.010
6	0.600	0.010	0.230	0.030	0.120	0.010	0.010
7	0.830	0.010	0.080	0.010	0.040	0.020	0.020
8	0.760	0.020	0.030	0.010	0.130	0.030	0.020
9	0.750	0.010	0.090	0.030	0.060	0.030	0.030
10	0.900	0	0.030	0.010	0.050	0.010	0.010
11	0.690	0.010	0.170	0.010	0.100	0.020	0.010
12	0.800	0.010	0.030	0.010	0.120	0.030	0.010
13	0.010	0.080	0.010	0.410	0.470	0.020	0.010
14	0.170	0.010	0.030	0.030	0.730	0.020	0.010
15	0.010	0.020	0.010	0.020	0.180	0.750	0.010
16	0.010	0	0.770	0	0.120	0.080	0.010
17	0.010	0.010	0.100	0.030	0.050	0.040	0.760
18	0.010	0	0.770	0	0.130	0.080	0.010
19	0.060	0	0.450	0.010	0.100	0.320	0.060
20	0.030	0.010	0.040	0.020	0.040	0.840	0.020
21	0.010	0	0.940	0.010	0.020	0.010	0.010
22	0.010	0.010	0.910	0.020	0.030	0	0.010
23	0.010	0.010	0.820	0.030	0.040	0.050	0.050
24	0.010	0.010	0.860	0.010	0.080	0.010	0.010
25	0.130	0	0.820	0.010	0.010	0.020	0.010
26	0.010	0.010	0.010	0.100	0.160	0.060	0.650
27	0	0.100	0.010	0.690	0.020	0.010	0.180
28	0.010	0.060	0.010	0.670	0.020	0.180	0.050

```
ext_gamma_matrix(documents_lda) %>%
  round(2) %>%
  stargazer(summary=F, rownames = F, header=F,
    title="Gamma matrix extracted from model",
    label="extract")
```

Table 3: Gamma matrix extracted from model

document	1	2	3	4	5	6	7
1	0	0.94	0	0.03	0.01	0	0.01
2	0.01	0.75	0	0.16	0.03	0.03	0.01
3	0.15	0.02	0.07	0.03	0.62	0.05	0.05
4	0.18	0.01	0.03	0.03	0.7	0.02	0.04
5	0.71	0.01	0.09	0	0.16	0.02	0.01
6	0.59	0.02	0.23	0.02	0.12	0.01	0.01
7	0.8	0.01	0.07	0.01	0.04	0.03	0.03
8	0.73	0.01	0.05	0.01	0.14	0.04	0.02
9	0.73	0.01	0.09	0.02	0.07	0.03	0.05
10	0.88	0	0.04	0.01	0.06	0	0.02
11	0.67	0.01	0.19	0.01	0.1	0.01	0.01
12	0.79	0.01	0.04	0.01	0.12	0.02	0.01
13	0.01	0.08	0.01	0.4	0.47	0.02	0.01
14	0.18	0.01	0.03	0.04	0.72	0.02	0.01
15	0.02	0.02	0.02	0.02	0.18	0.73	0.02
16	0.01	0.01	0.76	0.01	0.13	0.08	0.01
17	0.01	0.01	0.1	0.03	0.05	0.05	0.74
18	0.01	0	0.76	0	0.13	0.08	0.01
19	0.07	0.01	0.45	0.01	0.1	0.31	0.06
20	0.05	0.01	0.03	0.02	0.05	0.82	0.03
21	0.01	0	0.92	0.01	0.02	0.02	0.01
22	0.01	0	0.89	0.02	0.04	0.01	0.02
23	0.01	0.01	0.82	0.03	0.04	0.05	0.04
24	0.02	0.01	0.87	0.01	0.08	0.01	0.01
25	0.13	0.01	0.81	0.02	0.01	0.02	0.01
26	0.01	0.02	0.01	0.11	0.16	0.06	0.64
27	0.01	0.09	0.01	0.68	0.03	0.01	0.17
28	0.01	0.06	0.02	0.66	0.02	0.19	0.05

The tables below summarize which document refers to which topic, according to the LDA model (term frequency 2 embedding).

3.1 Wordclouds

To check what topics tackle which context, we produce wordclouds using the *tf-idf* and the *tf* itself.

```
plot_wordcloud <- function(corpus, selection="ALL",
  max.words=50, i, freq="tfidf",
  scale=c(3,0.2)){
  # 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)
```

Table 4: Documents for Topic 1

Topic	doc_id	Group
1	10	4
1	11	4
1	12	4
1	5	2
1	6	3
1	7	4
1	8	4
1	9	4

Table 5: Documents for Topic 2

Topic	doc_id	Group
2	1	1
2	2	1

Table 6: Documents for Topic 3

Topic	doc_id	Group
3	16	5
3	18	5
3	19	5
3	21	6
3	22	6
3	23	6
3	24	6
3	25	6

Table 7: Documents for Topic 4

Topic	doc_id	Group
4	27	7
4	28	7

Table 8: Documents for Topic 5

Topic	doc_id	Group
5	13	3
5	14	4
5	3	2
5	4	2

Table 9: Documents for Topic 6

Topic	doc_id	Group
6	15	5
6	20	5

Table 10: Documents for Topic 7

Topic	doc_id	Group
7	17	5
7	26	7

```

# 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)
}
# plotting
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=scale,
  main="Title", use.r.layout = TRUE)
# set a title = document
text(x=0.5, y=1, paste("Topic", i))
}

```

A second possibility is to extract the *tf-idfs* of the words linked to the topics directly. First you have to map the topics to the documents within the tidytext format. This is the only way the *tfidf_tf* matrix can be set up for the individual topics.

```

plot_wordcloud_topic <- function(corpus, topic_select=1, prediction=prediction5,
  max.words=50,
  scale=c(3,0.2)){
  # get the correct topic mapping via the first two columns
  # of the prediction matrix
  new_corpus <- prediction %>%
    transmute(doc_id=as.numeric(doc_id), topic) %>%
    right_join(corpus, c("doc_id"="document"))
  # setting up a tibble which returns tfidf and tf and frequency for
  # the whole corpus

```



```

tfidf <- new_corpus %>% count(topic, word, sort = TRUE) %>%
  bind_tf_idf(word, topic, n) %>%
  # filter for all selected topics
  # use either ft or tfidf
  filter(topic==topic_select)
# plotting
wordcloud(words = tfidf$word, freq = tfidf$tf_idf,
           min.freq = 1,max.words=max.words, random.order=FALSE,
           colors=brewer.pal(8, "Dark2"), scale=scale,
           main="Title", use.r.layout = TRUE)
# set a title = document
text(x=0.5, y=1, paste("Topic", topic_select))
}

```

3.1.1 Wordclouds using *tf-idf*

For getting specific and more individual words for each cloud, we use the *tf-idf* in the first step. Now there are two methods to create the word clouds for the *tf-idf* measure. Once by aggregating the individual *tf-idfs* of the documents, as it is done in the following.

```

par(mfrow=c(2,4))
par(mar=c(1,1,1,1))
set.seed(123)
plot_wordcloud(corpus, selection=ind1[,1], i=1, scale=c(1.7,0.001),
               max.words = 45)
plot_wordcloud(corpus, selection=ind2[,1], i=2)
plot_wordcloud(corpus, selection=ind3[,1], i=3, scale=c(1.6,0.005),
               max.words = 45)
plot_wordcloud(corpus, selection=ind4[,1], i=4)
plot_wordcloud(corpus, selection=ind5[,1], i=5, scale=c(1.8,0.001),
               max.words = 40)
plot_wordcloud(corpus, selection=ind6[,1], i=6, scale=c(1.8,0.001),
               max.words = 40)
plot_wordcloud(corpus, selection=ind7[,1], i=7, scale=c(2.1,0.3),
               max.words = 45)

```



Now using the second approach, when applying the *tf-idf* measure to the mapped corpus.

```
par(mfrow=c(2,4))
par(mar=c(1,1,1,1))
set.seed(123)
plot_wordcloud_topic(corpus, topic_select=1, scale=c(1.9,0.0006),
  max.words = 40)
plot_wordcloud_topic(corpus, topic_select=2)
plot_wordcloud_topic(corpus, topic_select=3, scale=c(1.6,0.005),
  max.words = 40)
plot_wordcloud_topic(corpus, topic_select=4, max.words = 50, scale=c(2.3, 0.2))
plot_wordcloud_topic(corpus, topic_select=5, scale=c(1.8,0.01),
  max.words = 40)
plot_wordcloud_topic(corpus, topic_select=6, scale=c(2,0.1),
  max.words = 45)
plot_wordcloud_topic(corpus, topic_select=7, scale=c(2.5,0.5),
  max.words = 50)
```



Although the second procedure, using *tf-idf* distributions for the individual topics, seems to be more intuitive, the two figures are surprisingly similar.

3.1.2 Wordclouds using *tf*

The same can be done using the regular term frequency.

```
par(mfrow=c(2,4))
par(mar=c(1,1,0.5,1))
set.seed(123)
plot_wordcloud(corpus, selection=ind1[,1], i=1, freq="tf", scale=c(2,0.05))
plot_wordcloud(corpus, selection=ind2[,1], i=2, freq="tf", scale=c(2,0.05))
plot_wordcloud(corpus, selection=ind3[,1], i=3, freq="tf", scale=c(2,0.03),
               max.words = 40)
plot_wordcloud(corpus, selection=ind4[,1], i=4, freq="tf", scale=c(2,0.3))
plot_wordcloud(corpus, selection=ind5[,1], i=5, freq="tf", scale=c(2.5,0.1))
plot_wordcloud(corpus, selection=ind6[,1], i=6, freq="tf")
plot_wordcloud(corpus, selection=ind7[,1], i=7, freq="tf", scale=c(2.5,0.2),
               max.words = 45)
```

[illegible][illegible]

Now it is interesting to see if embedding via *tf-idf* will cluster other groups or the same. So we will reduce the Document Term Matrix to 5717 words, which amounts to a reduction by 50%.

```
tim1_tfidf <- Sys.time()
dtm_50 <- dtm %>% dtm_remove_tfidf(top=(0.5*M))
set.seed(123)
documents_lda_2 <- LDA(dtm_50, method="Gibbs",
                        k = 7, control = list(seed = 1234))
tim2_tfidf <- Sys.time()
u2 <- tim2_tfidf - tim1_tfidf

prediction5_2 <- predict(documents_lda_2, newdata=dtm_50, type="topic")
prediction5_2 <- merge(prediction5_2, classes, by.x="doc_id", by.y="No")
# compare topic 1 with topic 2, 3, 4 and 5
ind1_2 <- prediction5_2 %>% filter(topic==1) %>% select(doc_id, Group)
ind2_2 <- prediction5_2 %>% filter(topic==2) %>% select(doc_id, Group)
ind3_2 <- prediction5_2 %>% filter(topic==3) %>% select(doc_id, Group)
ind4_2 <- prediction5_2 %>% filter(topic==4) %>% select(doc_id, Group)
ind5_2 <- prediction5_2 %>% filter(topic==5) %>% select(doc_id, Group)
ind6_2 <- prediction5_2 %>% filter(topic==6) %>% select(doc_id, Group)
ind7_2 <- prediction5_2 %>% filter(topic==7) %>% select(doc_id, Group)

ext_gamma_matrix(documents_lda_2) %>%
  round(2) %>%
  stargazer(summary=F, rownames = F, header=F,
             title="Gamma matrix extracted from model for embedding with tfidf",
             label="extract2")
```

Table 11: Documents for Topic 1

Topic_embedding_0.8	doc_id	Group
1	10	4
1	11	4
1	12	4
1	5	2
1	6	3
1	7	4
1	8	4
1	9	4

Table 12: Documents for Topic 2

Topic_embedding_0.8	doc_id	Group
2	15	5
2	25	6

Table 13: Documents for Topic 3

Topic_embedding_0.8	doc_id	Group
3	13	3
3	14	4
3	3	2
3	4	2

Table 14: Documents for Topic 4

Topic_embedding_0.8	doc_id	Group
4	1	1
4	2	1

Table 15: Documents for Topic 5

Topic_embedding_0.8	doc_id	Group
5	16	5
5	18	5
5	19	5
5	21	6
5	22	6
5	23	6
5	24	6

Table 16: Documents for Topic 6

Topic_embedding_0.8	doc_id	Group
6	20	5
6	28	7

Table 17: Documents for Topic 7

Topic_embedding_0.8	doc_id	Group
7	17	5
7	26	7
7	27	7

Table 18: Gamma matrix extracted from model for embedding with tfidf

document	1	2	3	4	5	6	7
1	0	0	0	0.99	0	0	0
2	0.01	0.02	0.03	0.86	0.01	0.07	0.01
3	0.16	0.02	0.54	0.01	0.18	0.03	0.05
4	0.2	0.02	0.68	0.02	0.02	0.02	0.04
5	0.85	0.03	0.04	0.01	0.07	0.01	0.01
6	0.69	0	0.02	0.01	0.25	0.01	0.01
7	0.85	0.04	0.02	0.01	0.05	0.01	0.03
8	0.86	0.02	0.03	0.02	0.03	0.01	0.03
9	0.72	0.06	0.08	0.01	0.06	0.03	0.05
10	0.95	0	0.01	0	0.02	0.01	0
11	0.73	0.01	0.03	0.01	0.19	0.01	0.01
12	0.9	0.02	0.03	0.01	0.02	0.01	0.01
13	0.01	0.02	0.76	0.15	0	0.01	0.04
14	0.24	0.02	0.68	0.02	0.02	0.01	0.01
15	0.01	0.83	0.09	0.02	0.02	0.01	0.02
16	0.01	0.06	0.01	0.01	0.9	0.01	0.01
17	0.01	0.05	0.01	0.01	0.07	0.01	0.84
18	0.01	0.06	0.01	0	0.89	0.01	0.02
19	0.04	0.42	0.02	0.01	0.43	0.02	0.07
20	0.01	0.05	0.02	0.01	0.01	0.88	0.01
21	0.01	0.01	0.01	0.01	0.96	0.01	0.01
22	0.01	0.01	0.01	0.01	0.94	0.01	0.01
23	0	0.02	0.03	0.01	0.9	0	0.03
24	0.02	0.01	0.02	0.01	0.92	0.01	0.01
25	0.01	0.91	0.01	0.01	0.05	0.01	0.01
26	0.01	0.01	0.06	0.03	0.02	0.05	0.82
27	0.01	0.01	0.03	0.19	0	0.02	0.75
28	0.01	0.03	0.04	0.05	0.01	0.73	0.13

We want to give an overview over the clustered documents using the *tf-idf* embedding.

Table 19: Documents for Topic 1

Topic	doc_id	Group
1	10	4
1	11	4
1	12	4
1	5	2
1	6	3
1	7	4
1	8	4
1	9	4

Table 20: Documents for Topic 2

Topic	doc_id	Group
2	15	5
2	25	6

Table 21: Documents for Topic 3

Topic	doc_id	Group
3	13	3
3	14	4
3	3	2
3	4	2

Table 22: Documents for Topic 4

Topic	doc_id	Group
4	1	1
4	2	1

Table 23: Documents for Topic 5

Topic	doc_id	Group
5	16	5
5	18	5
5	19	5
5	21	6
5	22	6
5	23	6
5	24	6

We want to produce wordclouds again. This time using the *tf-idf* embedding for clustering via LDA. The first plot shows the aggregated *tf-idfs*.

Table 24: Documents for Topic 6

Topic	doc_id	Group
6	20	5
6	28	7

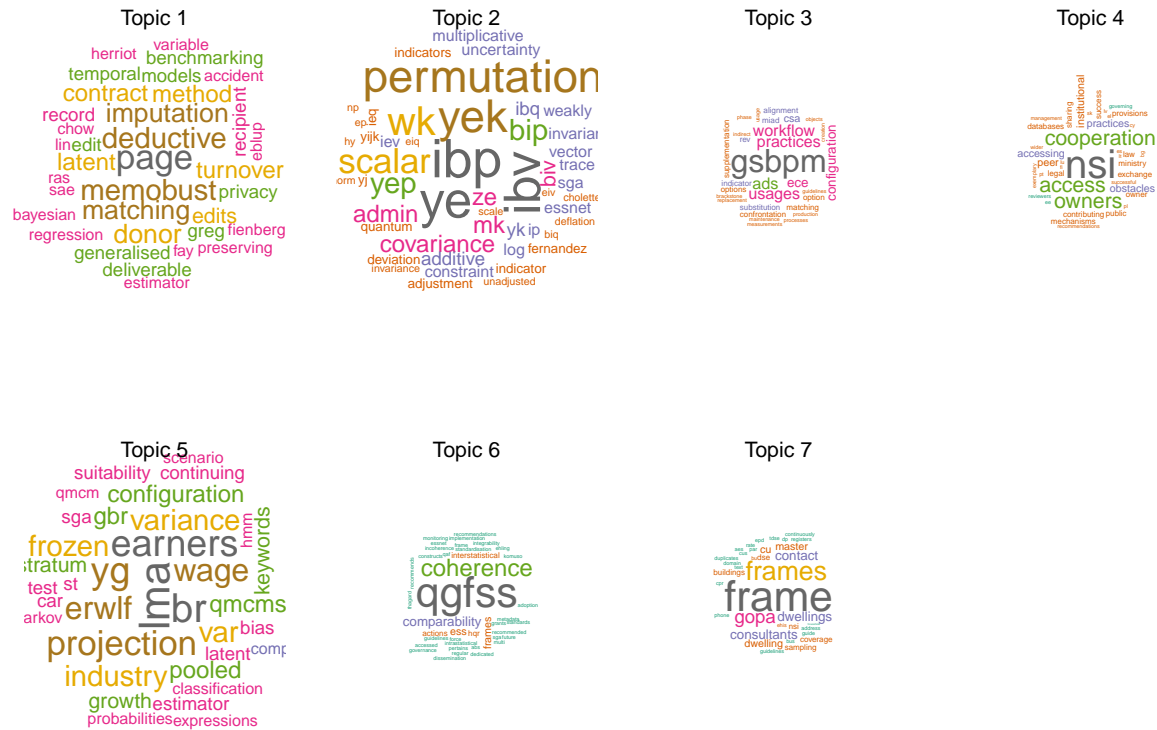
Table 25: Documents for Topic 7

Topic	doc_id	Group
7	17	5
7	26	7
7	27	7

```

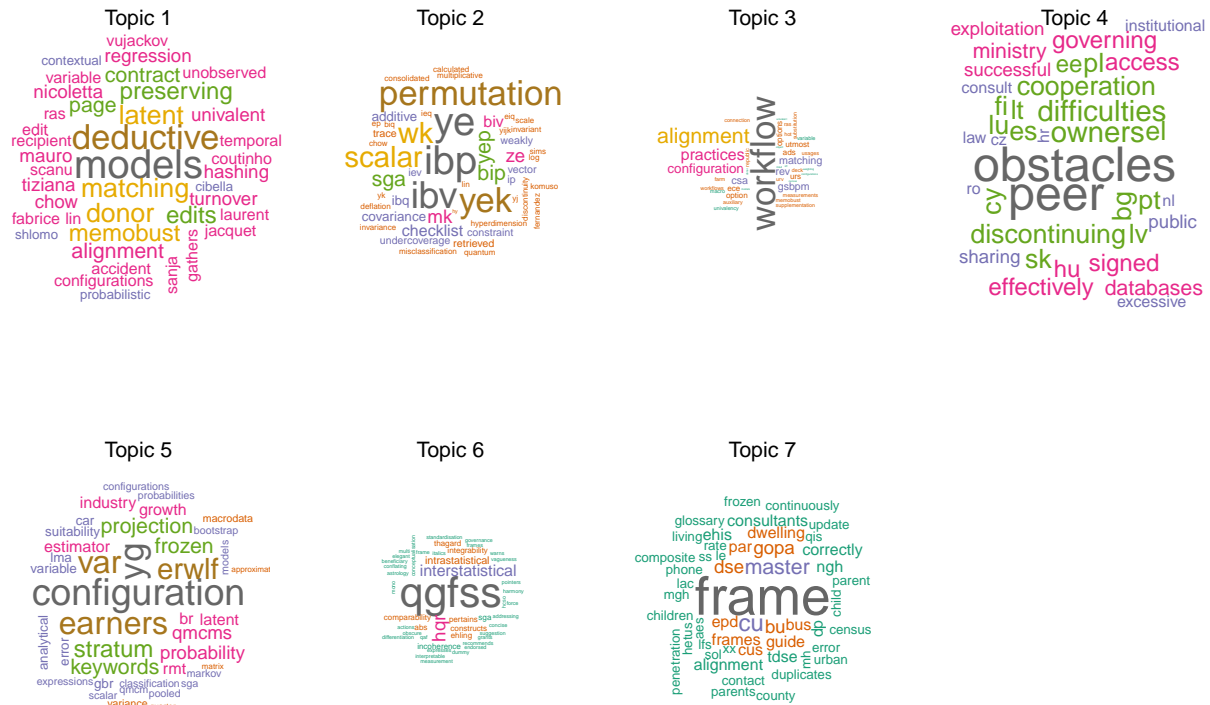
par(mfrow=c(2,4))
par(mar=c(1,1,0.5,1))
set.seed(123)
plot_wordcloud(corpus, selection=ind1_2[,1], i=1, scale=c(1.7,0.002),
               max.words = 35)
plot_wordcloud(corpus, selection=ind2_2[,1], i=2, scale=c(2.5,0.1))
plot_wordcloud(corpus, selection=ind3_2[,1], i=3, scale=c(1.5,0.001),
               max.words = 30)
plot_wordcloud(corpus, selection=ind4_2[,1], i=4, scale=c(1.9,0.05),
               max.words = 40)
plot_wordcloud(corpus, selection=ind5_2[,1], i=5, scale=c(2,0.02),
               max.words = 35)
plot_wordcloud(corpus, selection=ind6_2[,1], i=6, scale=c(2,0.1),
               max.words = 40)
plot_wordcloud(corpus, selection=ind7_2[,1], i=7, scale=c(2,0.03),
               max.words = 35)

```

The second plot shows the *tf-idf*s for each individual topic.

```
par(mfrow=c(2,4))
par(mar=c(1,1,1,1))
set.seed(123)
plot_wordcloud_topic(corpus, topic_select=1, scale=c(1.9,0.0006),
  max.words = 40, prediction=prediction5_2)
plot_wordcloud_topic(corpus, topic_select=2, prediction=prediction5_2,
  scale=c(1.9,0.0006))
plot_wordcloud_topic(corpus, topic_select=3, scale=c(1.6,0.005),
  max.words = 40, prediction=prediction5_2)
plot_wordcloud_topic(corpus, topic_select=4, max.words = 40,
  scale=c(2.3, 0.02), prediction=prediction5_2)
plot_wordcloud_topic(corpus, topic_select=5, scale=c(1.8,0.01),
  max.words = 40, prediction=prediction5_2)
plot_wordcloud_topic(corpus, topic_select=6, scale=c(2,0.1),
  max.words = 45, prediction=prediction5_2)
plot_wordcloud_topic(corpus, topic_select=7, scale=c(2.5,0.5),
  max.words = 50, prediction=prediction5_2)
```



3.3 Misclassification Rates

Now we use the validation measure we used for Example 1.

```
validate_LDClassification <- function(predict_table){
  # gamma_matrix ... an object of the function ext_gamma_matrix()
  # First we'd find the topic that was most associated with
  # each chapter
  conversion <- predict_table %>%
    select(Group, topic) %>%
    group_by(Group) %>%
    top_n(1,topic) %>%
    unique()

  consensus <- predict_table %>%
    left_join(conversion, by=c("topic"))
  consensus %>% filter(Group.x!=Group.y) %>%
    nrow()/nrow(consensus)
}
```

On both full bag of words and 50% embedding via *tf-idf*

```
predict_table <- prediction5 %>% select(doc_id, topic) %>%
  merge( y=classes, by.x=1, by.y=1)

( misc.rate_embedding2 <- validate_LDClassification(predict_table) )
```

```
## [1] 0.3421053
```

```
predict_table2 <- prediction5_2 %>% select(doc_id, topic) %>%
  merge( y=classes, by.x=1, by.y=1)
```

```
( misc.rate_embedding_tfidf <- validate_LDAClassification(predict_table2))

## [1] 0.3589744

performance_matrix <- data.frame(freq2.embedding=c(misc.rate_embedding2, u1),
                                tfidf.embedding=c(misc.rate_embedding_tfidf, u2))
rownames(performance_matrix) <- c("misc. rate", "time")
performance_matrix %>% stargazer(summary=FALSE, header=F, title = "LDA via Gibbs Sampling")
```

Table 26: LDA via Gibbs Sampling

	freq2.embedding	tfidf.embedding
misc. rate	0.342	0.359
time	43.467	16.994

4 Optimal value of Reduction via tfidf

In this example we want to know if there is a better value for the reduction of tfidf than 50%. We will examine it using the same approach as for the Gutenberg examples.

```
evaluation_for_embedding <- function(word_counts, frac=0.1, M=8254) {
  # embedding 3
  chapters_dtm_tfidf <- dtm %>% dtm_remove_tfidf(top=(frac*M))
  # fitting
  tim1 <- Sys.time()
  chapters_lda_tfidf_Gibbs <- LDA(chapters_dtm_tfidf, method="Gibbs",
                                k = 7, control = list(seed = 1234))
  tim2 <- Sys.time()
  u_tfidf_Gibbs <- tim2-tim1
  prediction5_2 <- predict(chapters_lda_tfidf_Gibbs,
                          newdata=chapters_dtm_tfidf, type="topic")
  prediction5_2 <- merge(prediction5_2, classes, by.x="doc_id", by.y="No")
  # calculating missclassification rate
  predict_table <- prediction5_2 %>% select(doc_id, topic) %>%
  merge( y=classes, by.x=1, by.y=1)

  misc.rate_tfidf_Gibbs <- validate_LDAClassification(predict_table)

  # function returns a vector including the misc. ratio and the fitting time
  return(c(misc.rate_tfidf_Gibbs,
          as.numeric(u_tfidf_Gibbs)))
}
```

```
# set up the fractions we want to use
# we use 0.1, 0.2, ..., 1
fractions <- seq(0.1,1,0.01)

# Use parallelization
# setting up how many cores to be used
useable_cores <- parallel::detectCores() - 1
# registering cluster
```

```

cl <- parallel::makeCluster(useable_cores)
doParallel::registerDoParallel(cl)
embedding_performance_matrix <- foreach(i = 1:length(fractions), .combine = 'rbind', .export = ls(.GlobalEnv))

  evaluation_for_embedding(dtm, frac=fractions[i]) %>%
    c(.,fractions[i])

}

## Warning in e$fun(obj, substitute(ex), parent.frame(), e$data): already
## exporting variable(s): classes, dtm, evaluation_for_embedding, fractions,
## validate_LDClassification

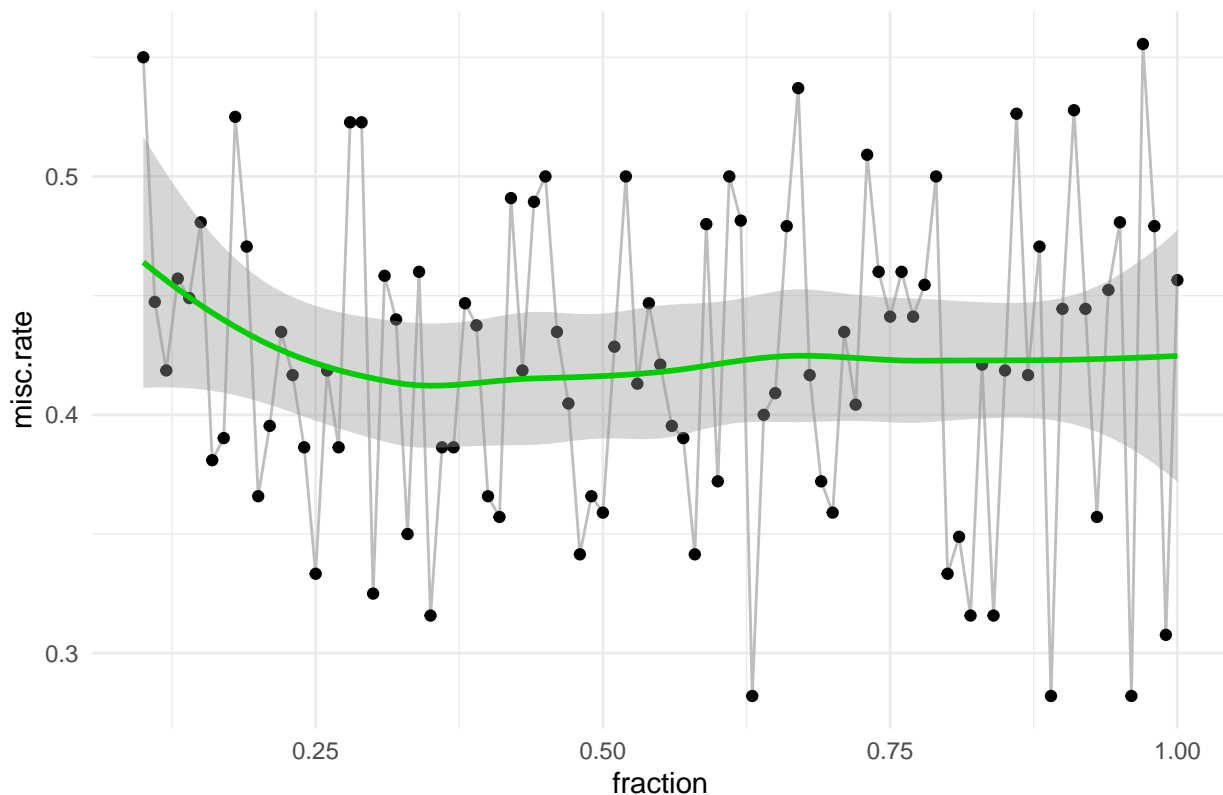
parallel::stopCluster(cl)

# adjust the resulting matrix to a data frame for plotting
colnames(embedding_performance_matrix) <- c("misc.rate", "time", "fractions")
embedding_performance_matrix <- as.data.frame(embedding_performance_matrix)

ggplot(data=embedding_performance_matrix,
       aes(x=fractions, y=misc.rate)) +
  geom_line(col="grey") +
  geom_point() +
  geom_smooth(col="green3", method="loess", span=0.7) +
  theme_minimal() +
  ggtitle("Misclassification rates for different types of tfidf-embeddings") +
  xlab("fraction")

```

Misclassification rates for different types of tfidf-embeddings



We can find an optimal value for *fraction* at the value around 0.35, so it might be interesting to compare the results of a fit using this optimized value of *fraction* to the other embeddings.

```
tim1_tfidf0.85 <- Sys.time()
dtm_0.85 <- dtm %>% dtm_remove_tfidf(top=(0.35*M))
set.seed(123)
documents_lda_0.85 <- LDA(dtm_0.85, method="Gibbs",
                          k = 7, control = list(seed = 1234))
tim2_tfidf0.85 <- Sys.time()
u0.85 <- tim2_tfidf0.85 - tim1_tfidf0.85

prediction0.85 <- predict(documents_lda_0.85, newdata=dtm_0.85, type="topic")
prediction0.85 <- merge(prediction0.85, classes, by.x="doc_id", by.y="No")

predict_table0.85 <- prediction0.85 %>% select(doc_id, topic) %>%
  merge( y=classes, by.x=1, by.y=1)

( misc.rate_embedding_0.85 <- validate_LDAclassification(predict_table0.85))
```

```
## [1] 0.3157895
```

In Table 27 you will find an overview of the performance of the different embedding methods including *tfidf* embedding of fraction 0.35.

```
performance_matrix <- data.frame(freq2.embedding=c(misc.rate_embedding2, u1),
                                tfidf_0.5.embedding=c(misc.rate_embedding_tfidf, u2),
                                tfidf_0.85.embedding=c(misc.rate_embedding_0.85, u0.85))
rownames(performance_matrix) <- c("misc. rate", "time")
performance_matrix %>% stargazer(summary=FALSE, header=F,
```

```
title = "LDA via Gibbs Sampling", label="comp0.85")
```

Table 27: LDA via Gibbs Sampling

	freq2.embedding	tfidf_0.5.embedding	tfidf_0.85.embedding
misc. rate	0.342	0.359	0.316
time	43.467	16.994	15.020

5 Coherence Cloud

[illegible]

```
## Warning in brewer.pal(max(3, ncol(term.matrix)), "Dark2"): n too large, allowed maximum for palette 1
## Returning the palette you asked for with that many colors
```

