assignment_5_topic_analysis

Marie Rivers

2022-05-05

```
library(forcats)
library(ggplot2)
library(ggraph)
library(here)
library(igraph) #network plots
library(ldatuning)
library(LDAvis)
library(lubridate) #working with date data
library(pdftools)
library(quanteda)
library(quanteda.textstats)
library(quanteda.textplots)
library(readr)
library(readtext) #quanteda subpackage for reading pdf
library(reshape2)
library(stringr)
library(tidyr) #text analysis in R
library(tidytext)
library(tidyverse)
library(tm)
library(topicmodels)
library("tsne")
library(widyr) # pairwise correlations
```

```
files <- list.files(path = here("movie_scripts"),
                    pattern = "pdf$", full.names = TRUE)

movie_scripts <- lapply(files, pdf_text)

movie_scripts_pdf <- readtext(file = here("movie_scripts", "*.pdf"),
                              docvarsfrom = c("metadata", "filenames", "filepaths"),
                              sep = "_")

#creating an initial corpus containing our data
script_corp <- corpus(x = movie_scripts_pdf, text_field = "text" )
summary(script_corp) %>%
  knitr::kable(caption = "Summary of Movie Script Corpus")
```

Table 1: Summary of Movie Script Corpus

Text	Types	Tokens	Sentences
before_the_flood.pdf	2540	13634	863
Dont-Look-Up-Read-The-Screenplay.pdf	4620	28016	2825
Gore_Inconvient_Truth_Transcript.pdf	2245	10936	685

```
toks <- tokens(script_corp, remove_punct = TRUE, remove_numbers = TRUE)

# xxx...add custome stop words such as character names from Don't Look Up
add_stops <- c(stopwords("en"),"xxx", "yyy", "zzz")
toks1 <- tokens_select(toks, pattern = add_stops, selection = "remove")
```

Convert to a document-feature matrix

```
dfm_comm <- dfm(toks1, tolower = TRUE)
dfm <- dfm_wordstem(dfm_comm)
dfm <- dfm_trim(dfm, min_docfreq = 2) #remove terms only appearing in one doc (min_termfreq = 10)

print(head(dfm))
```

```
## Document-feature matrix of: 3 documents, 1,377 features (20.12% sparse) and 0 docvars.
##                                       features
## docs                                   flood mark first memori frame poster
##   before_the_flood.pdf                     8    1    10      1     2      1
##   Dont-Look-Up-Read-The-Screenplay.pdf     1    1    17      2     1      3
##   Gore_Inconvient_Truth_Transcript.pdf     4    1    11      2     0      0
##                                       features
## docs                                   stare everi night went
##   before_the_flood.pdf                     2    14     3    4
##   Dont-Look-Up-Read-The-Screenplay.pdf     8     8    49    2
##   Gore_Inconvient_Truth_Transcript.pdf     0    11     1   21
## [ reached max_nfeat ... 1,367 more features ]
```

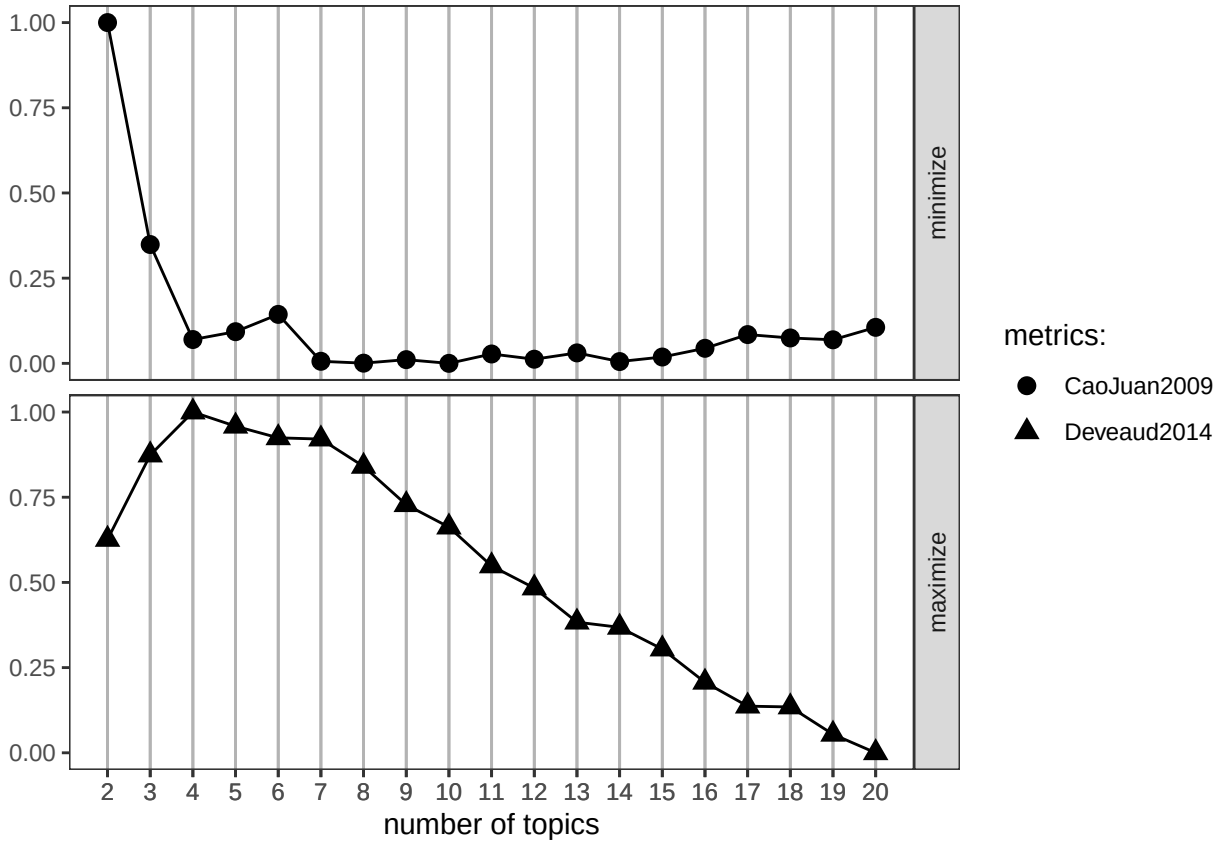
```
#remove rows (docs) with all zeros...for the topic model you can't have zeros
sel_idx <- slam::row_sums(dfm) > 0
dfm <- dfm[sel_idx, ]
#comments_df <- dfm[sel_idx, ]
```

```
result <- FindTopicsNumber(
  dfm,
  topics = seq(from = 2, to = 20, by = 1),
  metrics = c("CaoJuan2009",  "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 77),
  verbose = TRUE
)
```

```
## fit models... done.
## calculate metrics:
##   CaoJuan2009... done.
##   Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```




```
k <- 7 # k is the number of topics

topicModel_k7 <- LDA(dfm, k, method="Gibbs", control=list(iter = 500, verbose = 25))
```

```
## K = 7; V = 1377; M = 3
## Sampling 500 iterations!
## Iteration 25 ...
## Iteration 50 ...
## Iteration 75 ...
## Iteration 100 ...
## Iteration 125 ...
## Iteration 150 ...
## Iteration 175 ...
## Iteration 200 ...
## Iteration 225 ...
## Iteration 250 ...
## Iteration 275 ...
## Iteration 300 ...
## Iteration 325 ...
## Iteration 350 ...
```

```
## Iteration 375 ...
## Iteration 400 ...
## Iteration 425 ...
## Iteration 450 ...
## Iteration 475 ...
## Iteration 500 ...
## Gibbs sampling completed!
```

```
#nTerms(dfm_comm)

tmResult <- posterior(topicModel_k7)
attributes(tmResult)
```

```
## $names
## [1] "terms"  "topics"
```

```
#nTerms(dfm_comm)
beta <- tmResult$terms   # get beta from results
dim(beta)                # K distributions over nTerms(DTM) terms# lengthOfVocab
```

```
## [1]    7 1377
```

```
terms(topicModel_k7, 10)
```

```
##       Topic 1      Topic 2 Topic 3   Topic 4    Topic 5 Topic 6  Topic 7
##  [1,] "year"       "know"  "last"    "show"     "just"  "chang"  "presid"
##  [2,] "ice"        "now"   "use"     "open"     "go"    "climat" "dr"
##  [3,] "one"        "can"   "new"     "actual"   "like"  "see"    "orlean"
##  [4,] "warm"       "think" "come"    "hit"      "say"   "will"   "look"
##  [5,] "global"     "yeah"  "know"    "cours"    "right" "peopl"  "two"
##  [6,] "happen"     "call"  "big"     "want"     "back"  "world"  "night"
##  [7,] "temperatur" "come"  "first"   "pictur"   "time"  "carbon" "tell"
##  [8,] "lot"        "got"   "togeth"  "question" "well"  "thing"  "offic"
##  [9,] "now"        "gonna" "percent" "hear"     "earth" "can"    "june"
## [10,] "water"      "need"  "move"    "look"     "us"    "much"   "later"
```

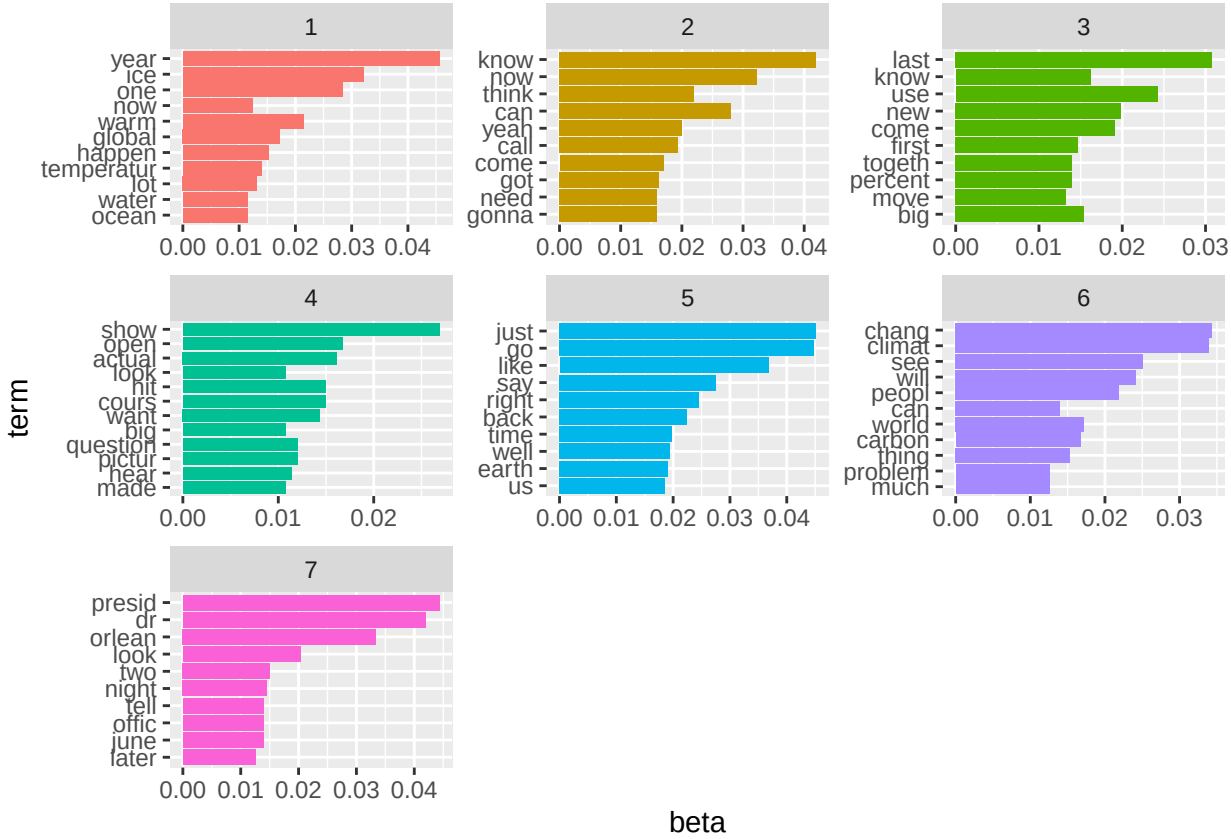
```
theta <- tmResult$topics
beta <- tmResult$terms
vocab <- (colnames(beta))
```

```
comment_topics <- tidy(topicModel_k7, matrix = "beta")

top_terms <- comment_topics %>%
  group_by(topic) %>%
  top_n(10, beta) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms
```

```
## # A tibble: 74 x 3
##    topic term         beta
##    <int> <chr>       <dbl>
##  1     1 year       0.0456
##  2     1 ice        0.0321
##  3     1 one        0.0284
##  4     1 warm       0.0214
##  5     1 global     0.0173
##  6     1 happen     0.0152
##  7     1 temperatur 0.0140
##  8     1 lot        0.0132
##  9     1 now        0.0124
## 10     1 water      0.0115
## # ... with 64 more rows
```

```
# beta is the probability of a term in a topic...highest beta or words most likely to be in topic
```

```
top_terms %>%
  mutate(term = reorder(term, beta)) %>%
  ggplot(aes(term, beta, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free") +
  coord_flip()
```



Assign names to the topics so we know what we are working with. We can name them by their top terms

```
top5termsPerTopic <- terms(topicModel_k7, 5)
topicNames <- apply(top5termsPerTopic, 2, paste, collapse=" ")
```

We can explore the theta matrix, which contains the distribution of each topic over each document

```
exampleIds <- c(1, 2, 3)
N <- length(exampleIds)
```

```
#lapply(epa_corp[exampleIds], as.character) #uncomment to view example text
# get topic proportions form example documents
topicProportionExamples <- theta[exampleIds,]
colnames(topicProportionExamples) <- topicNames
rownames(topicProportionExamples) <- c("Before the Flood", "Don't Look Up", "Inconvient Truth")
vizDataFrame <- melt(cbind(data.frame(topicProportionExamples), document=factor(1:N)), variable.name =
  mutate(document = case_when(
    document == 1 ~ "Before the Flood",
    document == 2 ~ "Don't Look Up",
    document == 3 ~ "Inconvient Truth"))
ggplot(data = vizDataFrame, aes(topic, value, fill = document), ylab = "proportion") +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1)) +
  coord_flip() +
  facet_wrap(~ document, ncol = N)
```

```
# named topics based on first 5 words
# first column show prevalence of each topic in the 1st document
# 1, 2, 3 are the 1st, 2nd, and 3rd documents and the plot shows how each topic is distributed within e
```

Here's a neat JSON-based model visualizer

```
library(LDAvis)
library("tsne")
svd_tsne <- function(x) tsne(svd(x)$u)
json <- createJSON(
  phi = tmResult$terms,
  theta = tmResult$topics,
  doc.length = rowSums(dfm),
  vocab = colnames(dfm),
  term.frequency = colSums(dfm),
  mds.method = svd_tsne,
  plot.opts = list(xlab="", ylab="")
)
```

```
## sigma summary: Min. : 33554432 |1st Qu. : 33554432 |Median : 33554432 |Mean : 33554432 |3rd Qu. : 335

## Epoch: Iteration #100 error is: 9.96570256782324

## Epoch: Iteration #200 error is: 0.230163585627043

## Epoch: Iteration #300 error is: 0.188260600300189

## Epoch: Iteration #400 error is: 0.158660472472083

## Epoch: Iteration #500 error is: 0.158651231649369

## Epoch: Iteration #600 error is: 0.15865122653676

## Epoch: Iteration #700 error is: 0.15865122652863

## Epoch: Iteration #800 error is: 0.158651226513877

## Epoch: Iteration #900 error is: 0.158651226492211

## Epoch: Iteration #1000 error is: 0.15865122646376
```

```
serVis(json)
```

```
## Loading required namespace: servr
```