# LDA Documentation

*Sebastian Knigge*

*6 8 2019*

## Setup

Following packages were used in this script:

```
# loading packages
library(gutenbergr)
library(dplyr)
library(tidyr)
library(stringr)
library(tidytext)
library(udpipe)
library(topicmodels)
library(ggplot2)
library(parallel)
library(foreach)
```

## Get data

### Definition of functions

This is the essential step for setting up the LDA model. These functions include the sampling procedure from the *gutenbergr* library.

```
sampling_books <- function(seed=1234, n=20){
  # sample n books from the whole library
  set.seed(seed)
  gutenberg_works() %>%
    # select works with title
    dplyr::filter(!is.na(title)) %>%
    # set the sample sitze
    sample_n(n) %>%
    # set a special download link
    gutenberg_download(
      mirror = "http://mirrors.xmission.com/gutenberg/")
}

set_up_books <- function(n_books=4, seed=1992){
  # initial book sample
  books <- sampling_books(n=n_books, seed=seed)
  by_chapter <- books %>%
    group_by(gutenberg_id) %>%
    # split in chapters
    mutate(chapter = cumsum(str_detect(text, regex("^chapter ", ignore_case = TRUE)))) %>%
    ungroup() %>%
    # exclude books without chapters
```

```
    dplyr::filter(chapter > 0)
  return(by_chapter)
}

shorten_titles <- function(titles){
  # shorten very long book titles by setting
  # a subset of characters of the first line
  # of the title
  sub_inds <- titles %>%
    regexpr(pattern="\\n|\\r")-1
  sub_inds[sub_inds<0] <- nchar(titles)[sub_inds<0]
  titles %>%
    substr(1,sub_inds)
}

get_titles <- function(x, n_books){
  # get the sampled gutenberg_ids
  unique_ids <- x %>%
    select(gutenberg_id) %>%
    unique() %>% unlist()
  # get the titles
  titles <- gutenberg_works() %>%
    dplyr::filter(gutenberg_id %in% unique_ids) %>%
    select(gutenberg_id, title, author) %>%
    mutate(title=shorten_titles(title))
  # get the number of gutenberg ids
  len <- nrow(titles)
  if(n_books!=len) warning(paste("--- ",n_books-len,
                                 " books have 0 chapters --- "))
  # the output as a list
  ret <- list(
    titles=titles,
    len=len
  )
  return(ret)
}

append_by_chapter <- function(x=by_chapter, n_books, seed_index=1){
  # append the books matrix until
  # we get the desired number of books n_books
  titles <- get_titles(x, n_books)
  n <- titles$len
  while (n<n_books) {
    book2add <- sampling_books(n=1, seed=seed_index)
    by_chapter_add <- book2add %>%
      group_by(gutenberg_id) %>%
      # split in chapters
      mutate(chapter = cumsum(str_detect(text, regex("^chapter ", ignore_case = TRUE)))) %>%
      ungroup() %>%
      # exclude books without chapters
      dplyr::filter(chapter > 2)
    titles2add <- get_titles(by_chapter_add, 1)
    # adding the book to by_chapter if there are chapters in the
```

```
    # book plus it is not in the data already
    if (titles2add$len==1) if(!titles2add$titles$gutenberg_id%in%titles$titles$gutenberg_id) {
      x <- bind_rows(x, by_chapter_add)
    }
    n<-get_titles(x, n)$len
    seed_index <- seed_index+1
  }
  return(x)
}

exclude_stop_words <- function(x){
  # unite chapter and document title
  by_chapter_word <- x %>%
    unite(document, gutenberg_id, chapter) %>%
    # split into words
    unnest_tokens(word, text)
  # import tibble stop words
  data(stop_words)
  # find document-word counts
  word_counts <- by_chapter_word %>%
    # exclude stop words
    anti_join(stop_words) %>%
    # count each word by chapter
    count(document, word, sort = TRUE) %>%
    ungroup()
  return(word_counts)
}

convert_to_dtm <- function(x, minfq = 2){
  # get into a format lda can handle
  chapters_dtm <- x %>%
    select(doc_id=document, term=word, freq=n) %>%
    document_term_matrix() %>%
    # reduce by low frequencies
    dtm_remove_lowfreq(minfreq = minfq)
  return(chapters_dtm)
}


convert_to_dtm_2 <- function(x, n=n, minfq = 2, top=10000){
  # get into a format lda can handle
  chapters_dtm <- x %>%
    select(doc_id=document, term=word, freq=n) %>%
    document_term_matrix() %>%
    # reduce by low frequencies
    dtm_remove_tfidf(top=top)
  return(chapters_dtm)
}
```

Note:

- good seperation for 4 topics:
  - seed=12345
  - seed=54321

- for 6 books:
  - seed=222
  - seed 101
- for 10 books:
  - seed=54321
  - seed=123456

## Sample corpus

Now we can use all these functions to get to the initial corpus sample. In this example 6 books are choosen.

```
n_books <- 6
by_chapter <- set_up_books(n_books=n_books, seed=222)
get_titles(by_chapter, n_books)
```

```
## Warning in get_titles(by_chapter, n_books): --- 1 books have 0 chapters ---
```

```
## $titles
## # A tibble: 5 x 3
##   gutenberg_id title                                    author
##          <int> <chr>                                    <chr>
## 1           11 Alice's Adventures in Wonderland         Carroll, Lewis
## 2         3096 Beatrice                                 Haggard, H. Rider~
## 3        25603 Detailed Minutiae of Soldier life in the~ McCarthy, Carlton
## 4        47402 Along Alaska's Great River               Schwatka, Frederi~
## 5        49675 Hawkins Electrical Guide v. 5 (of 10)    Hawkins, N. (Nehe~
##
## $len
## [1] 5
```

The function set_up_books() returns a warning that one book seems to consist of only one chapter. In order to get a copus consisting out of 6 books, the function append_by_chaper() is used, which fills up the corpus to the desired number of books.

```
appended_by_chapter <- append_by_chapter(x=by_chapter, n_books = n_books)
word_counts <- exclude_stop_words(appended_by_chapter)
```

```
## Joining, by = "word"
```

In table 1 the sampled titles for the book sample with the seed 222 are displayed. It appears through the function append_by:chapter() one book was added, called “My Novel” — Volume 04“.

```
titles <- get_titles(appended_by_chapter, n_books)
titles$titles %>% stargazer(summary=FALSE, font.size = "footnotesize",
                            header=FALSE, title="Book-titles", rownames=FALSE,
                            label="titles:6books")
```

Table 1: Book-titles

| gutenberg_id | title | author |
|---|---|---|
| 11 | Alice’s Adventures in Wonderland | Carroll, Lewis |
| 3096 | Beatrice | Haggard, H. Rider (Henry Rider) |
| 7705 | "My Novel" — Volume 04 | Lytton, Edward Bulwer Lytton, Baron |
| 25603 | Detailed Minutiae of Soldier life in the Army of Northern Virginia, 1861-1865 | McCarthy, Carlton |
| 47402 | Along Alaska’s Great River | Schwatka, Frederick |
| 49675 | Hawkins Electrical Guide v. 5 (of 10) | Hawkins, N. (Nehemiah) |

## Reduction of the dimensionality

In the set up we have another parameter to adjust. The function *convert_to_dtm* takes the parameter *minfq*, which is used to reduce the "bag of words" (i.e. dimensionality). *minfq* is the minimum frequency for the bag of words dictionary. I will refer to this as "embedding". Let us set it to 2 in this case, meaning that we include a word only if the frequency is 2 or more.

```
chapters_dtm <- convert_to_dtm(word_counts, minfq=2)
ncol(chapters_dtm)
```

```
## [1] 10685
```

Let us compare it to the case if including all words.

```
chapters_dtm_all <- convert_to_dtm(word_counts, minfq=0)
ncol(chapters_dtm_all)
```

```
## [1] 17961
```

We also want to compare this to a reduction of the word dictionary by the tfidf. For the sake of comparison, the reduction is made to the same value as used above via minfreq=2 (i.e. 10685 words).

```
chapters_dtm_tfidf <- convert_to_dtm_2(word_counts, top=10685)
ncol(chapters_dtm_tfidf)
```

```
## [1] 10685
```

## Apply the LDA model on the full corpus

Set up the LDA model for the shrinked embedding corpus via frequency 2.

```
tim1 <- Sys.time()
chapters_lda <- LDA(chapters_dtm,
                    k = n_books, control = list(seed = 1234))
tim2 <- Sys.time()
u_1 <- tim2-tim1
```

In comparison set up the LDA model for the full word embedding corpus.

```
tim1 <- Sys.time()
chapters_lda_all <- LDA(chapters_dtm,
                    k = n_books, control = list(seed = 1234))
tim2 <- Sys.time()
u_all <- tim2-tim1
```

Third LDA setup is for the shrinkage of the dictionary by TFIDF.

```
tim1 <- Sys.time()
chapters_lda_tfidf <- LDA(chapters_dtm,
                    k = n_books, control = list(seed = 1234))
tim2 <- Sys.time()
u_tfidf <- tim2-tim1
chapters_lda
```

```
## A LDA_VEM topic model with 6 topics.
```

Now we evaluate the model all in once:

```
ext_gamma_matrix <- function(model){
  # get gamma matrix for chapter probabilities
  chapters_gamma <- tidy(model, matrix = "gamma")
  # split joint name of book and chapter
  chapters_gamma <- chapters_gamma %>%
    separate(document, c("gutenberg_id", "chapter"), sep = "_", convert = TRUE)
  # get matrix with probabilities for each topic per chapter
  gamma_per_chapter <- chapters_gamma %>%
    spread(topic, gamma)
  return(chapters_gamma)
}

validate_LDAclassification <- function(x){
  #First we'd find the topic that was most associated with
  # each chapter using top_n(), which is effectively the
  # "classification" of that chapter
  chapter_classifications <- x %>%
    group_by(gutenberg_id, chapter) %>%
    top_n(1, gamma) %>%
    ungroup()

  # We can then compare each to the "consensus"
  # topic for each book (the most common topic among its chapters),
  # and see which were most often misidentified.
  book_topics <- chapter_classifications %>%
    count(gutenberg_id, topic) %>%
    group_by(gutenberg_id) %>%
    # just keep the most frequent one
    top_n(1, n) %>%
    ungroup() %>%
    # keep title called census and topic
    transmute(consensus = gutenberg_id, topic)

  # check the fraction of missclassification
  chapter_classifications %>%
    inner_join(book_topics, by = "topic") %>%
    # missmatches
    dplyr::filter(gutenberg_id != consensus)%>%
    nrow()/nrow(chapter_classifications)
}
```

Now we exclude 3 times the beta matrix and evaluate the most likely result with the real results. Depending on how good the LDA will seperate the books, this influences the goodness of the fit.

```
misc.rate_1 <- ext_gamma_matrix(chapters_lda) %>%
  validate_LDAclassification()
```

```
misc.rate_all <- ext_gamma_matrix(chapters_lda_all) %>%
  validate_LDAclassification()
```

```
misc.rate_tfidf <- ext_gamma_matrix(chapters_lda_tfidf) %>%
  validate_LDAclassification()
```

Following matrix gives an overview:

```
performance_matrix <- data.frame(freq2.embedding=c(misc.rate_1, u_1),
           all.embedding=c(misc.rate_all, u_all),
           tfidf=c(misc.rate_tfidf, u_tfidf))
rownames(performance_matrix) <- c("missc. rate", "time")
performance_matrix %>%  stargazer(summary=FALSE, header=F)
```

Table 2:

| | freq2.embedding | all.embedding | tfidf |
|---|---|---|---|
| missc. rate | 0.636 | 0.636 | 0.636 |
| time | 12.487 | 13.067 | 12.678 |

Surprisingly, the run-time to fit the LDA model for the embedding using all words, does not take way longer than the embedding using a lower frequency. The fitting of the reduced bag of words via ftidf takes even longer. Thus from here on the full embedding is used.

## Evaluate model on testing set

First we split the data randomly in training and testing samples.

```
split_for_fit <- function(data, test_ratio=0.1, seed=1234){
  # function to set up the training
  # and the testing set from the input data which
  # is an object from the function convert_to_dtm()
  set.seed(seed)
  N <- nrow(data)
  n_test <- (N*test_ratio) %>% ceiling
  test_ind <- sample(1:N, n_test)
  train_ind <- (1:N)[-test_ind]
  ret <- list(train=data[train_ind,],
              test=data[test_ind,])
  return(ret)
}
```

The fit_n_evaluate() function will fit the LDA model and evaluate the goodness of fit for an object of the function split_for_fit. The section Validation gives insights in the procedure of how the "consensus"" is set up.

```
fit_n_evaluate <- function(split, k=n_books){
  LDA_model <- LDA(split$train,
                           k = k, control = list(seed = 1234))
  # use the predict function of udpipe
  # the topic predict funtion already extract the most likely topics
  prediction <- predict(LDA_model, newdata=split$test) %>% .$topic
  # get "consensus" via maximum likelihood
  # first extract the gamma matrix of the model fitted on the training
  # data
  chapters_gamma <- ext_gamma_matrix(LDA_model)
  spreaded_gamma <- chapters_gamma  %>% spread(topic, gamma)
  # get pdfs
  plotm <- spreaded_gamma %>%
    group_by(gutenberg_id) %>%
```

```
    # note: pdfs are unnormalized
    summarise_at(2:(titles$len+1), sum)
  topic_link <- plotm %>%
    apply(1, function(x) which.max(x[2:length(x)])) %>%
    cbind(plotm$gutenberg_id) %>%
    as.data.frame()
  # exclude the
  consensus <- split$test %>%
    rownames() %>%
    substr(1,regexpr("_",.)-1) %>%
    as.numeric() %>%
    as.data.frame() %>%
    # merge it to the topic
    merge(topic_link, by.y="V2", sort=FALSE) %>%
    select(..y)
  # missclassification rate will be returned
  sum(consensus!=prediction)/length(prediction)
}
```

We now can evaluate several fits of a model for different splits. Here is a parallelization for the individual for loops applied.

```
# setting up how many cores to be used
useable_cores <- parallel::detectCores() - 1
# registering cluster
cl <- parallel::makeCluster(useable_cores)
doParallel::registerDoParallel(cl)
n <- 59
results <- foreach(i = 1:n, .combine = 'c', .export = ls(.GlobalEnv), .packages = c("dplyr", "udpipe",


  chapters_dtm %>%
    split_for_fit(seed=12*i) %>%
    fit_n_evaluate()


}
```

```
## Warning in e$fun(obj, substitute(ex), parent.frame(), e$data): already
## exporting variable(s): chapters_dtm, ext_gamma_matrix, fit_n_evaluate,
## n_books, split_for_fit, titles
```

```
parallel::stopCluster(cl)
```

This is the output of the simulation over 59 splits and fits. The mean is the final result:

```
results %>% summary
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1818  0.4545  0.4545  0.5347  0.6364  0.9091
```

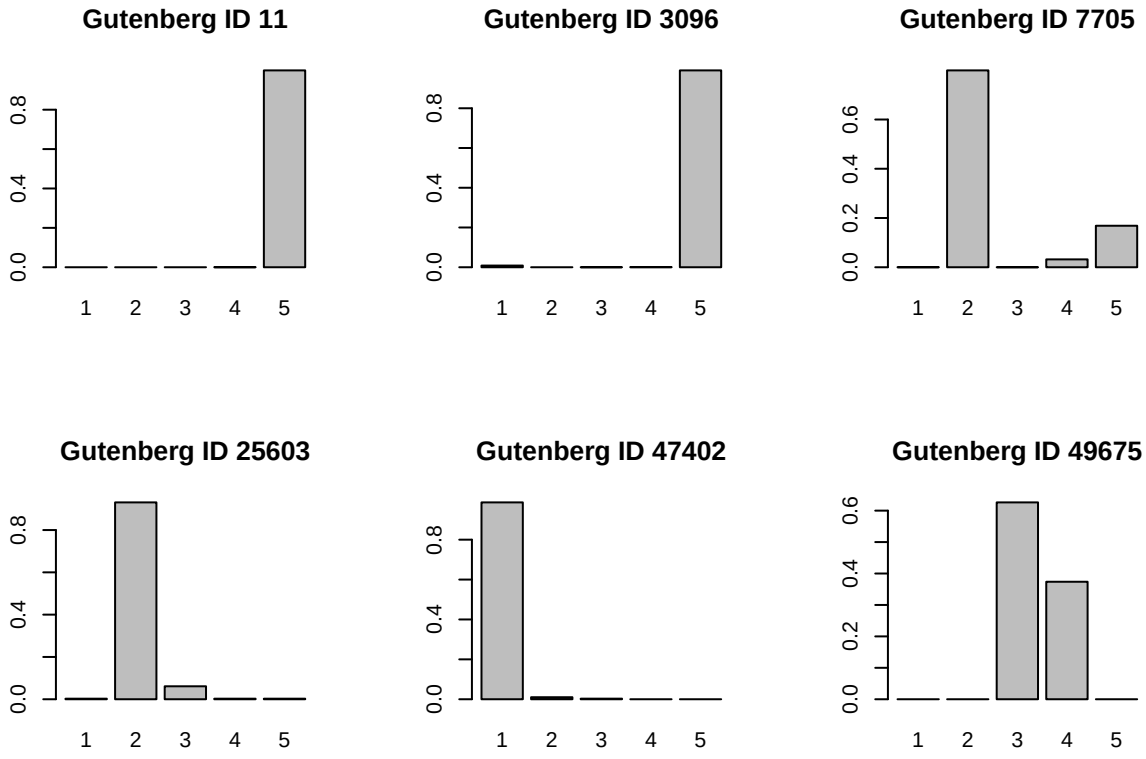
## Validation

Since the LDA algorithm just clusters into $k$ topics, it is necessary to evaluate which "topic" refers to which book, in order to calculate a missclassification rate. For each of the books we naively derive a distribution

of the assignment to the topics. This is done by accumulating the distributions for each chapter of the book. The most likely assignment of the LDA model is choosen as the "correct" topic. Now calculating the missclassification rate is basically the fraction of those chapters/documents not coinciding with the most likely assignment.

```
LDA_model <- LDA(chapters_dtm,
                          k = n_books, control = list(seed = 1234))
  # use the predict function of udpipe
  # the topic predict funtion already extract the most likely topics
prediction <- predict(LDA_model, newdata=chapters_dtm) %>% .$topic
  # get "consensus" via maximum likelihood
  # first extract the gamma matrix of the model fitted on the training
  # data
chapters_gamma <- ext_gamma_matrix(LDA_model)
spreaded_gamma <- chapters_gamma  %>% spread(topic, gamma)
  # get pdfs
plotm <- spreaded_gamma %>%
    group_by(gutenberg_id) %>%
    # note: pdfs are unnormalized
    summarise_at(2:(titles$len+1), sum)
  topic_link <- plotm %>%
    apply(1, function(x) which.max(x[2:length(x)])) %>%
    cbind(plotm$gutenberg_id) %>%
    as.data.frame()

par(mfrow=c(2,3))
for (i in 1:n_books){
  vec <- plotm[i,2:n_books] %>% unlist()
  barplot(vec/sum(vec), main=paste("Gutenberg ID", plotm[i,1]))
}
```



This does not exclude the fact that two books are assigned to the same topic. But still, from each of the plots of the "distributions" you can obtain how good the chapters of a single book are classified. The more of the mass of the distribution is on a single topic, the better the chapters of this topic are predicted.