LDA Implementation in R

Promothesh Chatterjee*

Let's understand how to implement LDA in R with our original example which contains hotel reviews. We first create our DTM using quanteda package.

```
library(tidyverse)
library(quanteda)
library(tidytext)
library(topicmodels)
library(stm)
hotel_raw <- read_csv("Data/hotel-reviews.csv")
set.seed(1234)
hotel_raw<-hotel_raw[sample(nrow(hotel_raw), 1000), ]# take a small sample


hotel_raw_token2 <- tokens(hotel_raw$Description, what = "word",
                        remove_numbers = TRUE, remove_punct = TRUE,
                        remove_symbols = TRUE)

#Create DTM, but remove terms which occur in less than 1% of all documents
# and more than 90%
hotel_raw_dfm<-hotel_raw_token2 %>%
  tokens_remove(stopwords(source = "smart")) %>%
  #tokens_wordstem() %>%
  tokens_tolower() %>%
  dfm()%>%
  dfm_trim(min_docfreq = 0.01, max_docfreq = 0.90, docfreq_type = "prop")

#hotel_raw_dfm<-dfm_tfidf(hotel_raw_dfm, scheme_tf = "prop", scheme_df = "inverse", base = 10)

#hotel_raw_dfm[1:5,1:5]
hotel_demo_token1<-as.matrix(hotel_raw_dfm)
# dim(hotel_demo_token1)
hotel_demo_token1[1:3,1:8]
```

```
##        features
## docs    hotel hotels boston deluxe suite bathtub shower half
##   text1     2      1      1      1     1       1      1    1
##   text2     2      0      0      0     1       0      0    0
##   text3     2      0      0      0     0       0      0    0
```

Running LDA

When we first run LDA on a dataset, we have no idea what value should be chosen for the number of topics to estimate K. We look at this issue later, for now let's just go with K=15.

```
library(topicmodels)
set.seed(12345)
K <- 15
hotel_lda <-LDA(hotel_raw_dfm, K, method="Gibbs", control=list(iter = 200, verbose = 25))
```

```
## K = 15; V = 1000; M = 1000
## Sampling 200 iterations!
## Iteration 25 ...
## Iteration 50 ...
## Iteration 75 ...
## Iteration 100 ...
## Iteration 125 ...
## Iteration 150 ...
## Iteration 175 ...
## Iteration 200 ...
## Gibbs sampling completed!
```

Let's first extract and plot the per-topic-per-word probabilities, called β ("beta"), from the model using tidytext package.

```
term_topics <- tidy(hotel_lda, matrix = "beta")#Topic Term Probabilities # we tidy it up to use tidytex
term_topics
```

```
## # A tibble: 15,000 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 hotel 0.0000277
##  2     2 hotel 0.00195
##  3     3 hotel 0.0000327
##  4     4 hotel 0.00749
##  5     5 hotel 0.141
##  6     6 hotel 0.0370
##  7     7 hotel 0.0131
##  8     8 hotel 0.0000310
##  9     9 hotel 0.0000294
## 10    10 hotel 0.0000309
## # i 14,990 more rows
```

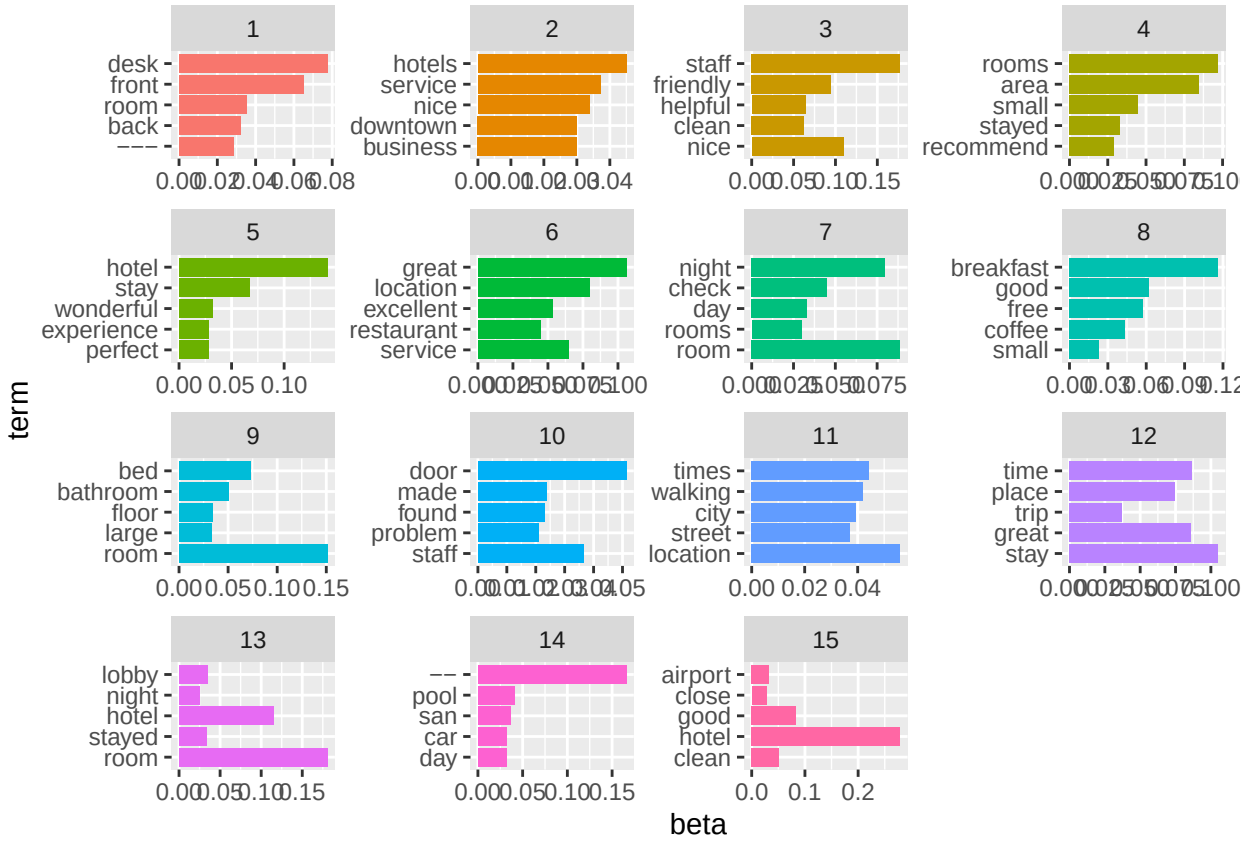
```
top_terms <- term_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 5) %>%
  arrange(topic, -beta)
top_terms
```

```
## # A tibble: 75 x 3
## # Groups:   topic [15]
```

```
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 desk     0.0775
##  2     1 front    0.0648
##  3     1 room     0.0352
##  4     1 back     0.0319
##  5     1 ---      0.0285
##  6     2 hotels   0.0450
##  7     2 service  0.0370
##  8     2 nice     0.0339
##  9     2 business 0.0300
## 10     2 downtown 0.0300
## # i 65 more rows
```

```
library(ggplot2)

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



In addition to estimating each topic as a mixture of words, each document is also modeled as a mixture of topicsin LDA. We can assign these per-document-per-topic probabilities (called γ ("gamma")) to a variable to examine those using matrix = "gamma" argument to function tidy().

```
hotel_documents <- tidy(hotel_lda, matrix = "gamma")
hotel_documents
```

```
## # A tibble: 15,000 x 3
##    document topic  gamma
##    <chr>    <int>  <dbl>
##  1 text1        1 0.0392
##  2 text2        1 0.0441
##  3 text3        1 0.0847
##  4 text4        1 0.0358
##  5 text5        1 0.0516
##  6 text6        1 0.0513
##  7 text7        1 0.0487
##  8 text8        1 0.0437
##  9 text9        1 0.187
## 10 text10       1 0.0585
## # i 14,990 more rows
```

```
top_documents <- hotel_documents %>%
  group_by(topic) %>%
  slice_max(gamma, n = 5) %>%
  ungroup() %>%
  arrange(topic, -gamma)
top_documents
```

```
## # A tibble: 75 x 3
##    document topic gamma
##    <chr>    <int> <dbl>
##  1 text374      1 0.533
##  2 text617      1 0.306
##  3 text481      1 0.304
##  4 text797      1 0.271
##  5 text301      1 0.263
##  6 text434      2 0.262
##  7 text147      2 0.249
##  8 text21       2 0.231
##  9 text935      2 0.223
## 10 text571      2 0.208
## # i 65 more rows
```

The output gives the estimated proportion of words from a particular text in a particular topic. For example, the model estimates that only about 4% of the words in text1 were generated from topic 1.

We can evaluate a particular document of interest. For example, text6 has about 11% of the words in document 1 were generated from topic 1. Let's examine that:

```
tidy(hotel_raw_dfm) %>% # taking the original dfm
  filter(document == 'text6') %>%
  arrange(desc(count))
```

```
## # A tibble: 14 x 3
##    document term      count
##    <chr>    <chr>     <dbl>
##  1 text6    good          2
##  2 text6    staff         1
##  3 text6    stay          1
##  4 text6    clean         1
##  5 text6    breakfast     1
##  6 text6    rooms         1
##  7 text6    small         1
##  8 text6    door          1
##  9 text6    decent        1
## 10 text6    bar           1
## 11 text6    pool          1
## 12 text6    lobby         1
## 13 text6    ac            1
## 14 text6    enjoyed       1
```

The LDA algorithm can also help us understand which term in particular document is assigned to which topic. The more words in a document are assigned to a particular topic, generally, the more weight (gamma) will go on that document-topic classification. We can use the augment() function to find out this information.

```
assignments <- augment(hotel_lda)
assignments
```

```
## # A tibble: 37,576 x 3
##    document term    .topic
##    <chr>    <chr>    <dbl>
##  1 text1    hotel       15
##  2 text1    hotels       2
##  3 text1    boston       2
##  4 text1    deluxe       7
##  5 text1    suite        9
##  6 text1    bathtub     13
##  7 text1    shower       9
##  8 text1    half         7
##  9 text1    glass        6
## 10 text1    wall         7
## # i 37,566 more rows
```

How to find the appropriate number of topics?

In our analysis we randomly took the number of topics as 15. How do we know that is appropriate. In general, research has shown that as the number of topics increase, the topic words start appearing across multiple topics making inferring underlying themes difficult. Other than eyeballing a few topics, there are a few statistical methods such as measuring perplexity, coherence. In addition there are R packages like ldatuning which calculate different metrics to estimate the most preferable number of topics for LDA model.

Perplexity indicates the level of fit between probability distribution and the underlying dataset. Since, topic models assume different probability distributions, we can use perplexity to determine the optimal number of topics. One thing to note here is that the more the number of topics one asks from the LDA model, lower will be the perplexity. Hence, one has to use domain knowledge and make a judgment call. Topicmodels package includes the function perplexity() which calculates this value for a given model.

Another measure could be coherence, which indicates whether the term for a particular topic appear together.

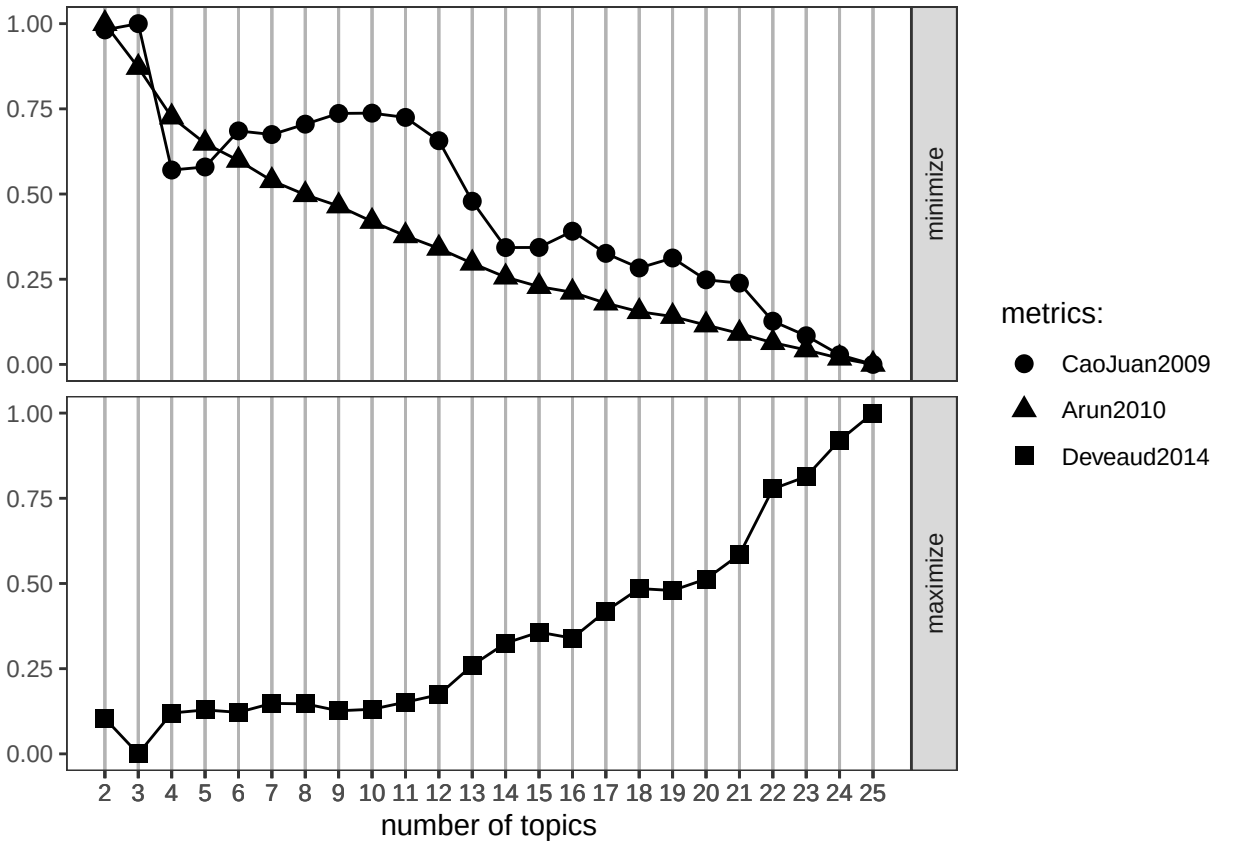
Topic numbers using LDAtuning package

```
library(ldatuning)
# Compute the K value "scores" from K=2 to K=25
result <- FindTopicsNumber(
    hotel_raw_dfm,
    topics = seq(from = 2, to = 25, by = 1),
    metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
    method = "VEM",#Gibbs can used too
    control = list(seed = 1971),
    mc.cores = 2L,
    return_models= TRUE,
    verbose = TRUE
)
```

```
## 'Griffiths2004' is incompatible with 'VEM' method, excluded.
## fit models... done.
## calculate metrics:
##   CaoJuan2009... done.
##   Arun2010... done.
##   Deveaud2014... done.
```

```
# Griffiths2004: Implement scoring algorithm. In order to use this algorithm, the LDA model
# MUST be generated using the keep control parameter >0 (defaults to 50) so that the logLiks
# vector is retained

# Plot the scores, with graphs labeled "minimize" or
# maximize based on whether it's a metric you want
# to minimize or maximize
FindTopicsNumber_plot(result)
```



Topic numbers using Perplexity

You can play around with number of topics and compute perplexity. In general, perplexity is used to compare models with different number of topics. The model with the lowest perplexity is generally considered the "best".

```
ldaResult <- posterior(hotel_lda)# have a look a some of the results (posterior distributions)
attributes(ldaResult)
```

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

```
library(text2vec)
text2vec::perplexity(hotel_raw_dfm, ldaResult$terms, ldaResult$topics)
```

```
## [1] 397.5237
```

Optional Reading- Coherence

Let's use R package textmineR to look at text coherence. In contrast to other packages, textmineR has outputs R-squared as a measure of fit. R-squared has same interpretation as a regular regression model (how much proportion of variance is explained by the model).

Probabilistic coherence measures how associated words are in a topic, controlling for statistical independence.

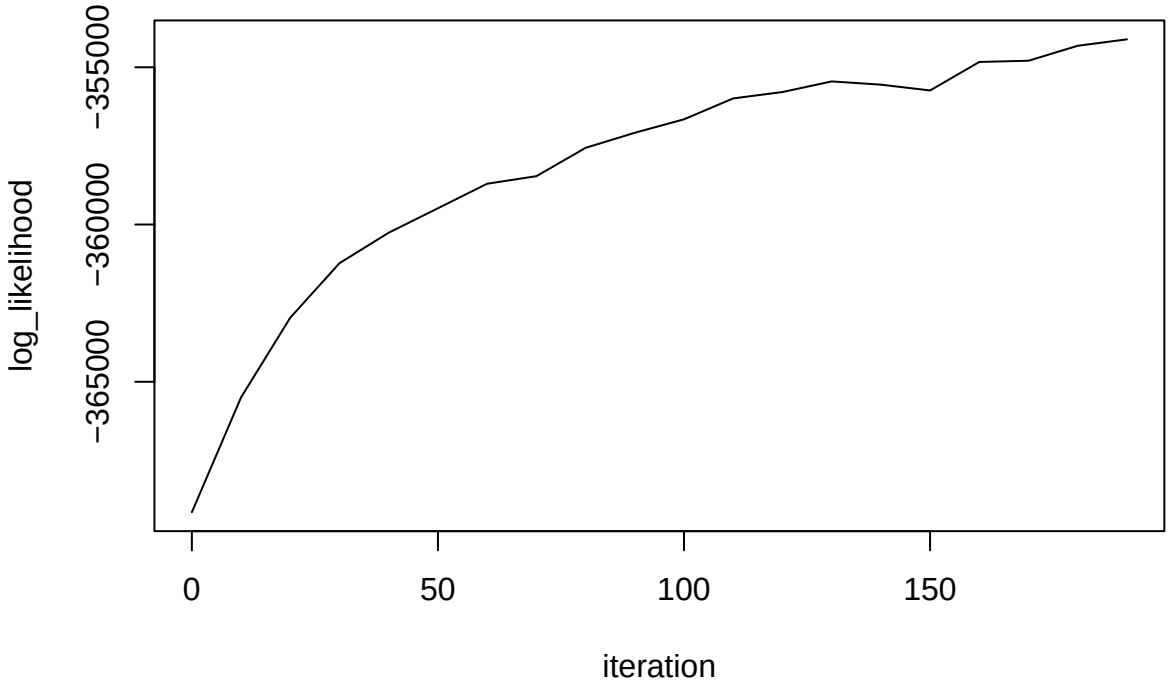
```
library(textmineR)
set.seed(12345)

model <- FitLdaModel(dtm = hotel_raw_dfm,
                     k = 20,
                     iterations = 200, # I usually recommend at least 500 iterations or more
                     burnin = 180,
                     alpha = 0.1,
                     beta = 0.05,
                     optimize_alpha = TRUE,
                     calc_likelihood = TRUE,
                     calc_coherence = TRUE,
                     calc_r2 = TRUE,
                     cpus = 2)# allows two cores for processing

# R-squared
# - only works for probabilistic models like LDA and CTM
model$r2
```

```
## [1] 0.2130964
```

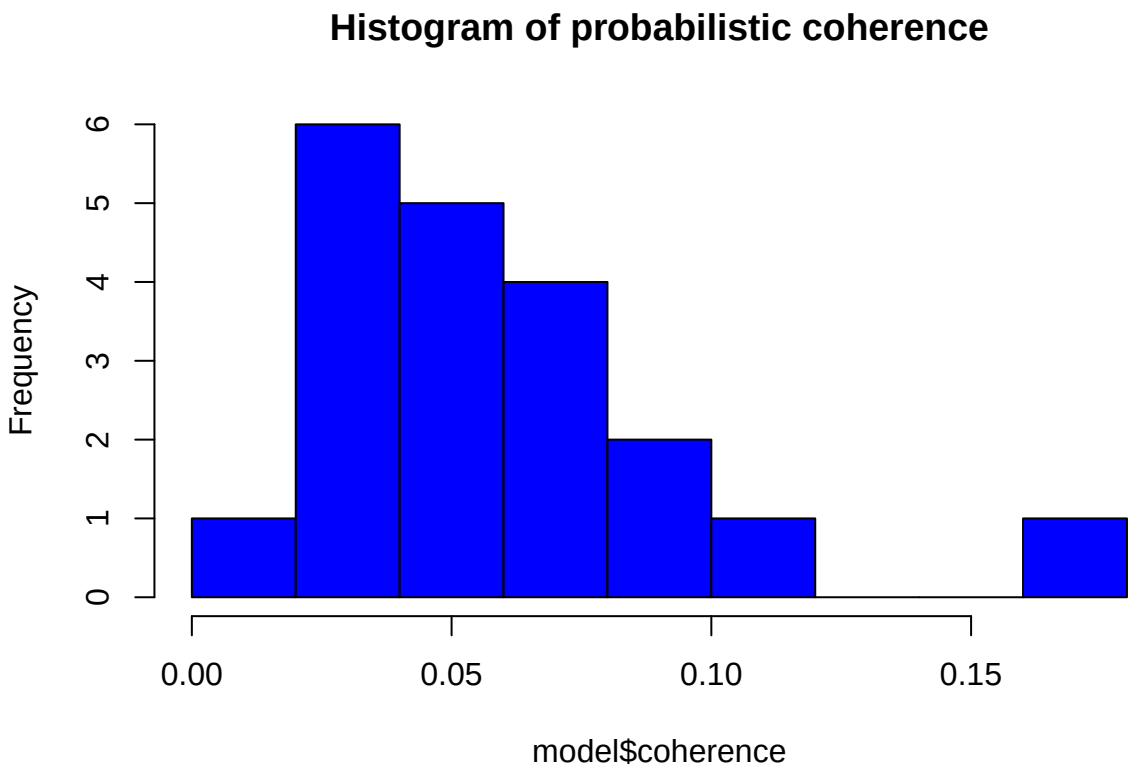
```
# log Likelihood (does not consider the prior)
plot(model$log_likelihood, type = "l")
```




```
summary(model$coherence)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.01982 0.03435 0.05486 0.06115 0.07752 0.16474
```

```
hist(model$coherence,
     col= "blue",
     main = "Histogram of probabilistic coherence")
```



Compare the coherence across different number of topics for better understanding.

Look at details here: https://rdrr.io/cran/textmineR/f/vignettes/c_topic_modeling.Rmd

Optional Reading -2

For an alternative treatment of topic models using R package STM, see Julia Silge's excellent blog: https://juliasilge.com/blog/evaluating-stm/.

Check these excellent resources which I borrow heavily from:

https://tm4ss.github.io/docs/Tutorial_6_Topic_Models.html

https://cfss.uchicago.edu/notes/topic-modeling/#importing-our-own-lda-model

https://stat.ethz.ch/CRAN/web/packages/ldatuning/vignettes/topics.html

http://jjacobs.me/tad/04_Topic_Modeling_ggplot2.html

https://knowledger.rbind.io/post/topic-modeling-using-r/

https://www.tidytextmining.com/topicmodeling.html