Gov 2018: Lab 10 Latent Dirichlet allocation (LDA)

Your name:

Apr 5, 2022

Overview

This exercise is based on "Text Mining with R: A Tidy Approach" and a tutorial from Language Technology and Data Analysis Laboratory (University of Queensland).

Latent Dirichlet allocation (LDA) is one of the most common algorithms for topic modeling. Without diving into the math behind the model, we can understand it as being guided by two principles.

- Every document is a mixture of topics.
- Every topic is a mixture of words.

LDA is a mathematical method for estimating both of these at the same time: finding the mixture of words that is associated with each topic, while also determining the mixture of topics that describes each document. There are a number of existing implementations of this algorithm, and we'll explore one of them in depth: `topicmodels::LDA()`.

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v ggplot2 3.3.5     v purrr   0.3.4
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Question 1: Data

In this lab, we will use the `AssociatedPress` dataset provided by the `topicmodels` package, as an example of a DocumentTermMatrix. This is a collection of 2246 news articles from an American news agency, mostly published around 1988.

```
library(topicmodels)
data("AssociatedPress")
```

Explore the data and report the number of documents and terms.

```
dim(AssociatedPress)
```

```
## [1]  2246 10473
```

Question 2: Fit the LDA model

Use the `LDA()` function from the `topicmodels` package, setting `k = 2` and `control = list(seed = 1234)`, to fit a two-topic LDA model with variational EM algorithm. (Alternatively, you can use Gibbs Sampling for the estimation by setting `method = "Gibbs"`.)

```
ap_lda <- LDA(AssociatedPress, k = 2, control = list(seed = 1234))
```

Question 3: Word-topic probabilities

Fitting the model was the "easy part": the rest of the analysis will involve exploring and interpreting the model using tidying functions from the tidytext package.

The `tidytext` package provides `tidy()` method, originally from the `broom` package, for extracting the per-topic-per-word probabilities, denoted β, from the model.

```
library(tidytext)
ap_topics <- tidy(ap_lda, matrix = "beta")
head(ap_topics)
```

```
## # A tibble: 6 x 3
##   topic term          beta
##   <int> <chr>        <dbl>
## 1     1 aaron     1.69e-12
## 2     2 aaron     3.90e- 5
## 3     1 abandon   2.65e- 5
## 4     2 abandon   3.99e- 5
## 5     1 abandoned 1.39e- 4
## 6     2 abandoned 5.88e- 5
```

3.1

What is the probabilities of the term `market` being generated from each of topic 1 and topic 2?

```
ap_topics %>%
  filter(term == "market")
```

```
## # A tibble: 2 x 3
##   topic term        beta
##   <int> <chr>      <dbl>
## 1     1 market 0.00333
## 2     2 market 0.0000156
```

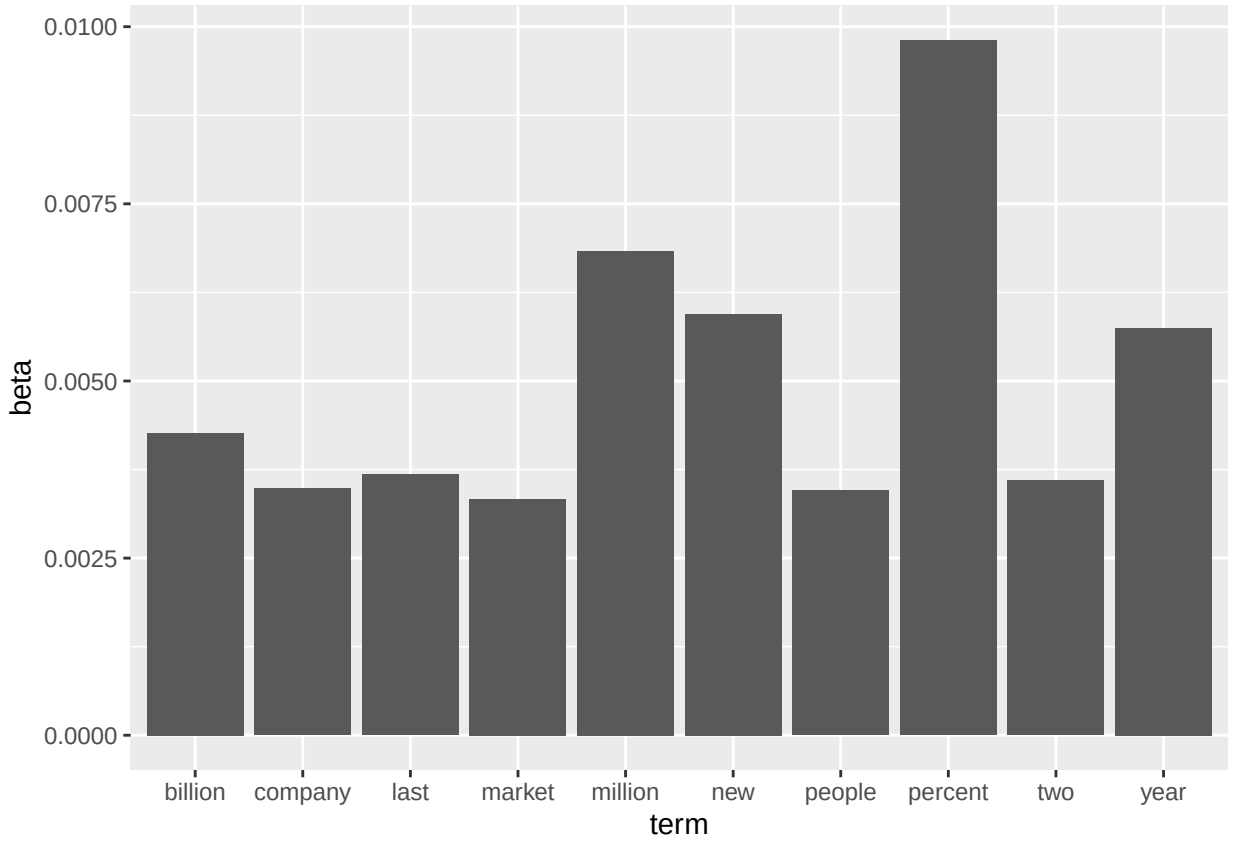
3.2

Find the 10 terms that are most common within each topic. Visualize the result as barplots of `beta`. Hand-label the two topics based on this result.

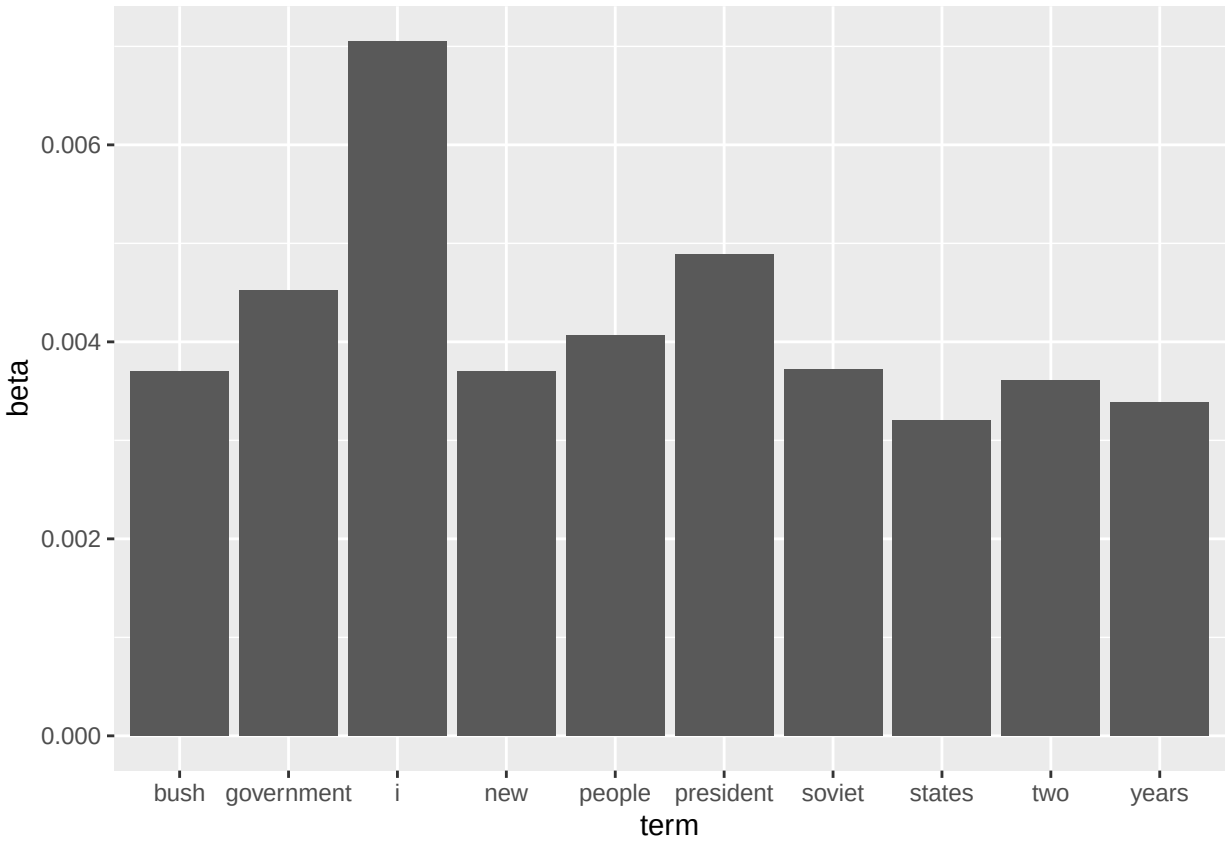
Hint: you may use `slice_max()` from `dplyr` package.

```
ap_topics %>%
  filter(topic == 1) %>%
  arrange(desc(beta)) %>%
  slice(1:10) %>%
```

```
  ggplot(aes(x = term, y = beta)) +
  geom_col()
```




```
ap_topics %>%
  filter(topic == 2) %>%
  arrange(desc(beta)) %>%
  slice(1:10) %>%
  ggplot(aes(x = term, y = beta)) +
  geom_col()
```



3.3

As an alternative, we could consider the terms that had the *greatest difference* in β between topic 1 and topic 2. This can be estimated based on the log ratio of the two: $\log_2\left(\frac{\beta_2}{\beta_1}\right)$. A log ratio is useful because it makes the difference symmetrical: β_2 being twice as large leads to a log ratio of 1, while β_1 being twice as large results in -1.

To constrain it to a set of especially relevant words, filter for relatively common words, such as those that have a β greater than 1/1000 in at least one topic. Calculate the log ratio of β and report the 10 maximum and 10 minimum terms. Briefly discuss the results.

```
beta_wide <- ap_topics %>%
  mutate(topic = paste0("topic", topic)) %>%
  pivot_wider(names_from = topic, values_from = beta) %>%
  filter(topic1 > .001 | topic2 > .001) %>%
  mutate(log_ratio = log2(topic2 / topic1))

# 10 terms for topic 2
beta_wide %>%
  arrange(desc(log_ratio)) %>%
  .[1:10,]
```

```
## # A tibble: 10 x 4
##    term           topic1  topic2 log_ratio
##    <chr>           <dbl>   <dbl>     <dbl>
##  1 democratic   9.89e-19 0.00147      50.4
##  2 dukakis      8.43e-17 0.00175      44.2
```

```
##  3 republican   1.77e-13 0.00116      32.6
##  4 gorbachev    2.37e-13 0.00122      32.3
##  5 trial        3.28e-11 0.00114      25.0
##  6 vote         1.18e-10 0.00118      23.3
##  7 campaign     3.49e-10 0.00173      22.2
##  8 senate       4.63e- 9 0.00140      18.2
##  9 presidential 2.86e- 8 0.00104      15.2
## 10 prison       1.95e- 7 0.00111      12.5
```

```
# 10 terms for topic 1
beta_wide %>%
  arrange(log_ratio) %>%
  .[1:10,]
```

```
## # A tibble: 10 x 4
##    term     topic1   topic2 log_ratio
##    <chr>     <dbl>    <dbl>     <dbl>
##  1 yen     0.00107 2.50e-32     -95.1
##  2 index   0.00120 3.53e-24     -68.2
##  3 cents   0.00159 7.81e-24     -67.5
##  4 average 0.00135 1.76e-12     -29.5
##  5 stock   0.00252 6.07e-11     -25.3
##  6 dollar  0.00176 1.47e-10     -23.5
##  7 prices  0.00264 1.20e- 8     -17.7
##  8 rate    0.00152 1.10e- 8     -17.1
##  9 fell    0.00142 2.09e- 7     -12.7
## 10 rates   0.00152 3.96e- 7     -11.9
```

Question 4: Document-topic probabilities

Besides estimating each topic as a mixture of words, LDA also models each document as a mixture of topics. We can examine the per-document-per-topic probabilities, denoted as γ (θ from the lecture slides), with the `matrix = "gamma"` argument to `tidy()`.

Use `tidy()` method to extract γ. What is the probability of the words in document 6 being generated from topic 2? Check the most common words in that document to confirm the result.

```
ap_docs <- tidy(ap_lda, matrix = "gamma")
ap_docs %>%
  filter(document == 6)
```

```
## # A tibble: 2 x 3
##   document topic    gamma
##      <int> <int>    <dbl>
## 1        6     1 0.000588
## 2        6     2 0.999
```

```
tidy(AssociatedPress) %>%
  filter(document == 6) %>%
  arrange(desc(count))
```

```
## # A tibble: 287 x 3
##    document term           count
##       <int> <chr>          <dbl>
##  1        6 noriega           16
```

```
##  2        6 panama            12
##  3        6 jackson            6
##  4        6 powell             6
##  5        6 administration     5
##  6        6 economic           5
##  7        6 general            5
##  8        6 i                  5
##  9        6 panamanian         5
## 10        6 american           4
## # ... with 277 more rows
```

Question 5: Tuning the number of topics, k

So far, we have been using two-topic LDA model. For parameterized models such as LDA, the number of topics k is the most important parameter to define in advance. How an optimal k should be selected depends on various factors. If k is too small, the collection is divided into a few very general semantic contexts. If k is too large, the collection is divided into too many topics of which some may overlap and others are hardly interpretable.

To select an optimal number of topics, `ldatuning::FindTopicsNumber()` fit models with a range of k and compute four metrics to compare the results.

Since this may take a significant amount of time, we will use a smaller subset (only 10 documents) in this exercise.

Create a subset of dtm with only 10 documents and run `FindTopicsNumber()` with following arguments:

- `topics = seq(from = 2, to = 15, by = 1)`: the range of number of topics to compare different models
- `metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014")`
- `method = "Gibbs"`: we will use Gibbs sampling to fit each model
- `control = list(seed = 1234)`: set the seed number
- `verbose = TRUE`: print the progress and warnings

Visualize the result with `FindTopicsNumber_plot()` and briefly discuss the result.

Hint: The best number of topics shows low values for `CaoJuan2009`/`Arun2010` and high values for `Griffith2004`/`Deveaud2014` (optimally, several methods should converge and show peaks and dips respectively for a certain number of topics).

```
library(ldatuning)
```

[Optional] Question 6: Alternative LDA implementations

The `LDA()` function in the `topicmodels` package is only one implementation of the latent Dirichlet allocation algorithm. For example, the `mallet` package (Mimno 2013) implements a wrapper around the MALLET Java package for text classification tools, and the `tidytext` package provides tidiers for this model output as well.

The `mallet` package takes a somewhat different approach to the input format. For instance, it takes non-tokenized documents and performs the tokenization itself, and requires a separate file of stopwords. This means we have to collapse the text into one string for each document before performing LDA.

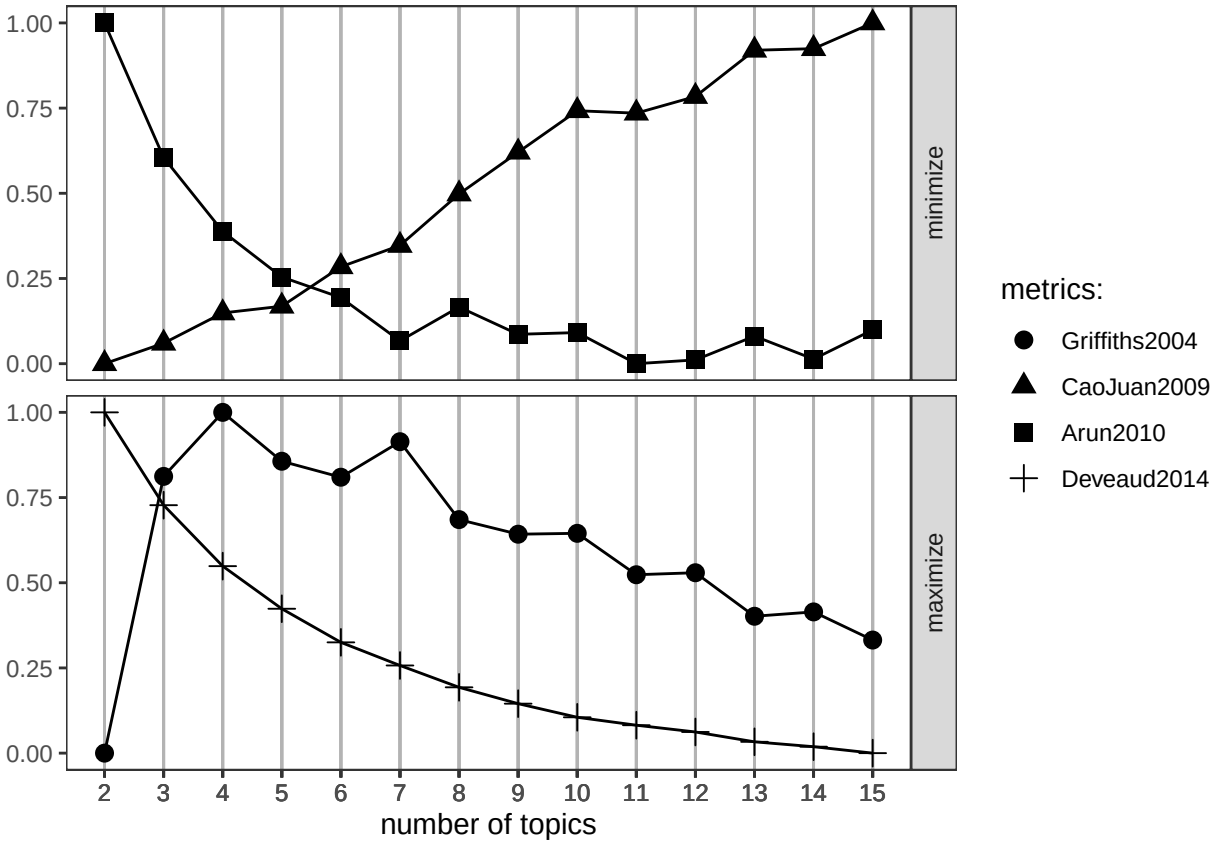
```
sub_dtm <- AssociatedPress[1:10, ]
result <- FindTopicsNumber(
  sub_dtm,
```

```
  topics = seq(from = 2, to = 15, by = 1),
  metrics = c("Griffiths2004", "CaoJuan2009", "Arun2010", "Deveaud2014"),
  method = "Gibbs",
  control = list(seed = 1234),
  verbose = TRUE
)
```

```
## fit models... done.
## calculate metrics:
##   Griffiths2004... done.
##   CaoJuan2009... done.
##   Arun2010... done.
##   Deveaud2014... done.
```

```
FindTopicsNumber_plot(result)
```

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



6.1 Preprocess

Use the following codes for preprocessing the data.

```
library(mallet)

# create a vector with one string per chapter
collapsed <- by_chapter_word %>%
```

```
  anti_join(stop_words, by = "word") %>%
  mutate(word = str_replace(word, "'", "")) %>%
  group_by(document) %>%
  summarize(text = paste(word, collapse = " "))

# create an empty file of "stopwords"
file.create(empty_file <- tempfile())
docs <- mallet.import(collapsed$document, collapsed$text, empty_file)

mallet_model <- MalletLDA(num.topics = 4)
mallet_model$loadDocuments(docs)
mallet_model$train(100)
```

6.2

Once the model is created, we can use the `tidy()` and `augment()` functions in an almost identical way using `LDA()`. This includes extracting the probabilities of words within each topic or topics within each document.

```
# word-topic pairs
tidy(mallet_model)

# document-topic pairs
tidy(mallet_model, matrix = "gamma")

# column needs to be named "term" for "augment"
term_counts <- rename(word_counts, term = word)
augment(mallet_model, term_counts)
```

We could use ggplot2 to explore and visualize the model in the same way we did the LDA output.