fidelity LDA

Lintong Li, Jiahao Liu，Kaiwei Xiao, Zijia Wang

2022-11-12

Introduction

This is the report which is aimed at analyzing the most common words from IMDB reviews, and then trying to estimate the types of movies among these reviews.

Package Used

tidyverse

tidytext

janeaustenr

stringr

ggplot2

topicmodels

tm

Methodology

First, we plot the distribution of term frequency which is larger than 0.1, and we found that there is a very long tail to the right of this novel which means existing those extremely rare words!

Then we use the idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents. Calculating tf-idf attempts to find the words that are important (i.e., common) in a text, but not too common.

Moreover, we use the ggraph package for visualizing our networks. We plot networks of these co-occurring words so we can see these relationships better.

Finally, we do Latent Dirichlet allocation. The β tells us the probability of that term being generated from that topic for that document. It is the probability of that term (word) belonging to that topic. Notice that some of the values for β are very, very low, and some are not so low. Next, let's examine which topics are associated with which description fields (i.e., documents). We will look at a different probability for this, γ, the probability that each document belongs in each topic.

```
library(tidyverse)

## — Attaching packages ──────────────────────────────── tidyverse 1.
3.2 —
## ✔ ggplot2 3.3.6      ✔ purrr   0.3.5
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.4.1
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## — Conflicts ──────────────────────────────── tidyverse_conflict
s() —
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()

library(tidytext)
library(janeaustenr)
library(stringr)
library(ggplot2)
library(topicmodels)
library(tm)

## Loading required package: NLP
##
## Attaching package: 'NLP'
##
## The following object is masked from 'package:ggplot2':
##
##     annotate

## Import the IMDB data, and transfer the type of dataset as tibble.
IMDB.Dataset <- read_csv("IMDB Dataset.csv", show_col_types = F)
IMDB <- tibble(IMDB.Dataset)

## Add IMDB data with column docs by using mutate() function.
IMDB <- IMDB  %>%  mutate(docs = c(1:length(IMDB$review)))

## In order to better analysis our IMDB review words, we apply stop words to
delete the words that might distract our evaluation.

## Import stop words data set.
data(stop_words)
```

```r
new = stop_words$word
new = data.frame(new)

## Combining with our IMDB data set, we use rbind() function to add these spe
cific stop words, which are meaningless or repeated, into stop_word data.
stop_w = rbind(new,"movie", "film", "films", "movies", "acting", "act", "role
",
              "actor", "actors", "scenes","scene", "character","br","cast",
              "characters", "make", "director", "10", "watch","watching", "2
")
colnames(stop_w) <- c("word")
#stop_words <- rbind(stop_words,c("br","Smart" ))
#stop_words = c(,"my","custom","words")

##tf-idf
book_words <- IMDB %>%
  unnest_tokens(word, review) %>
  anti_join(stop_w)%>%
  anti_join(stop_words)%>%
  count(docs, word, sort = TRUE)

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

## We calculate the total words in each novel here, for later use.
total_words <- book_words %>%
  group_by(docs) %>%
  summarize(total = sum(n))

book_words <- left_join(book_words, total_words)

## Joining, by = "docs"

## Then use row_number() to find the rank and rank column here tells us the r
ank of each word within the frequency table.
freq_by_rank <- book_words %>%
  group_by(docs) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

## First we look at term frequency (tf), which means how frequently a word oc
curs in a document.

## And term's inverse document frequency (idf), which decreases the weight fo
r commonly used words and increases the weight for words that are not used ve
ry much in a collection of documents.

## Thirdly, combining with term frequency to calculate a term's tf-idf (the t
wo quantities multiplied together), the frequency of a term adjusted for how
```

```
rarely it is used.
book_tf_idf <- book_words %>%
  bind_tf_idf(word, docs, n)

## We want delete the uncommon words that hardly use in this document. So, se
lect the words only with the term frequency is larger than 0.1.
book_tf_idf_new <- filter(book_tf_idf, tf > 0.1)

## We look at the distribution of term frequency(tf), n/total for each novel,
 the number of times a word appears in a novel divided by the total number of
 terms (words) in that novel.
ggplot(book_tf_idf_new, aes(tf, fill = docs)) +
  geom_histogram(show.legend = FALSE, color = 'black', fill = 'light blue') +
 geom_density(alpha=.2, fill="#FF6666") + labs(title = 'Density of word tf bi
gger than 0.1')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```
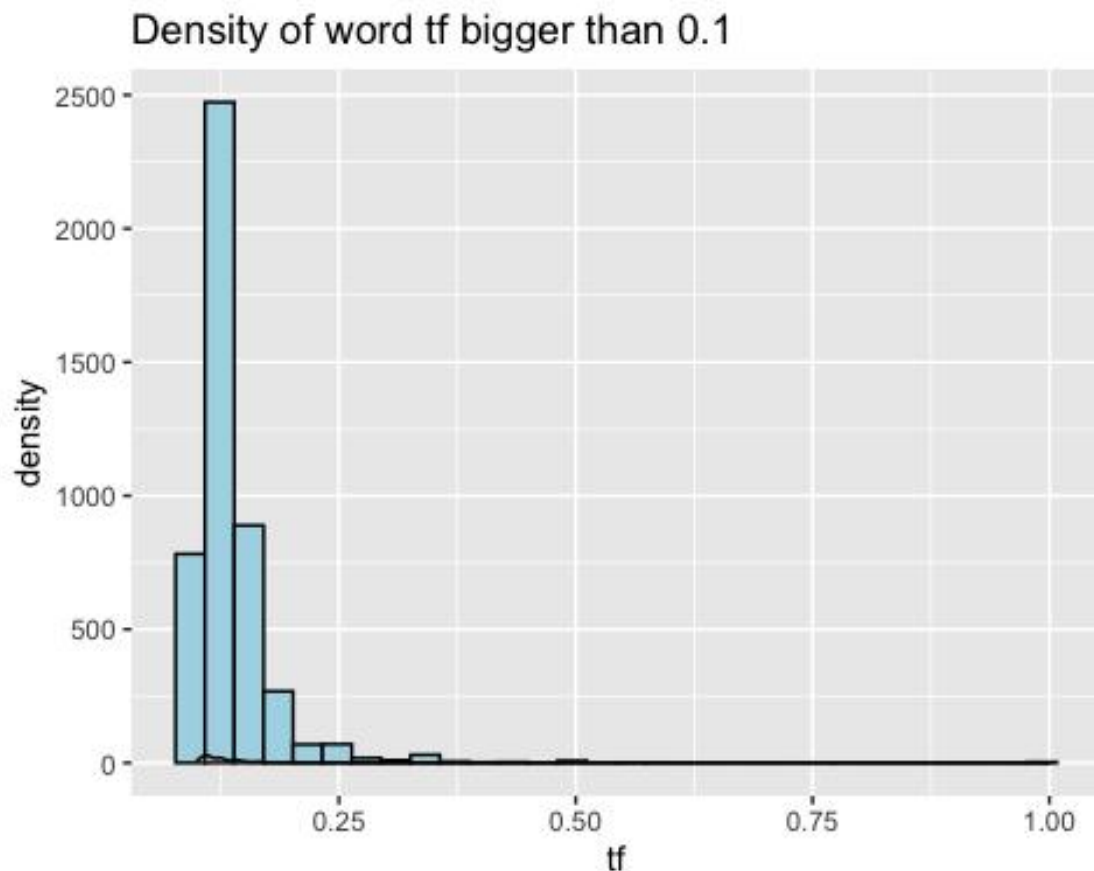

First, we plot the distribution of term frequency which is larger than 0.1, and we found that there is a very long tail to the right of this novel which means existing those extremely rare words!

```
## We use count() function to find the most common words.
df <- book_tf_idf_new %>%
  group_by(word) %>%
  count(sort = TRUE)

## We likely want to change all of the keywords to either lower or upper case
 to get rid of duplicates like "MOVIE" and "Movie".
df <- df %>%
  mutate(word = toupper(word))

##Then We use pairwise_count() from the widyr package to count how many times
 each pair of words occurs together in a title or description field.
library(widyr)
word_pairs <- book_words %>%
  pairwise_count(word, docs, sort = TRUE, upper = FALSE)
```

The idea of tf-idf is to find the important words for the content of each document by decreasing the weight for commonly used words and increasing the weight for words that are not used very much in a collection or corpus of documents.

```
## We use ggplot package to plot the 15 most common words in this review docu
ments.
df_new <- head(df,15)
ggplot(df_new, aes(x = n, fill = word)) + geom_histogram() + labs(title = 'To
p 15 Most Frequent Words in Reviews')

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

Top 15 Most Frequent Words in Reviews
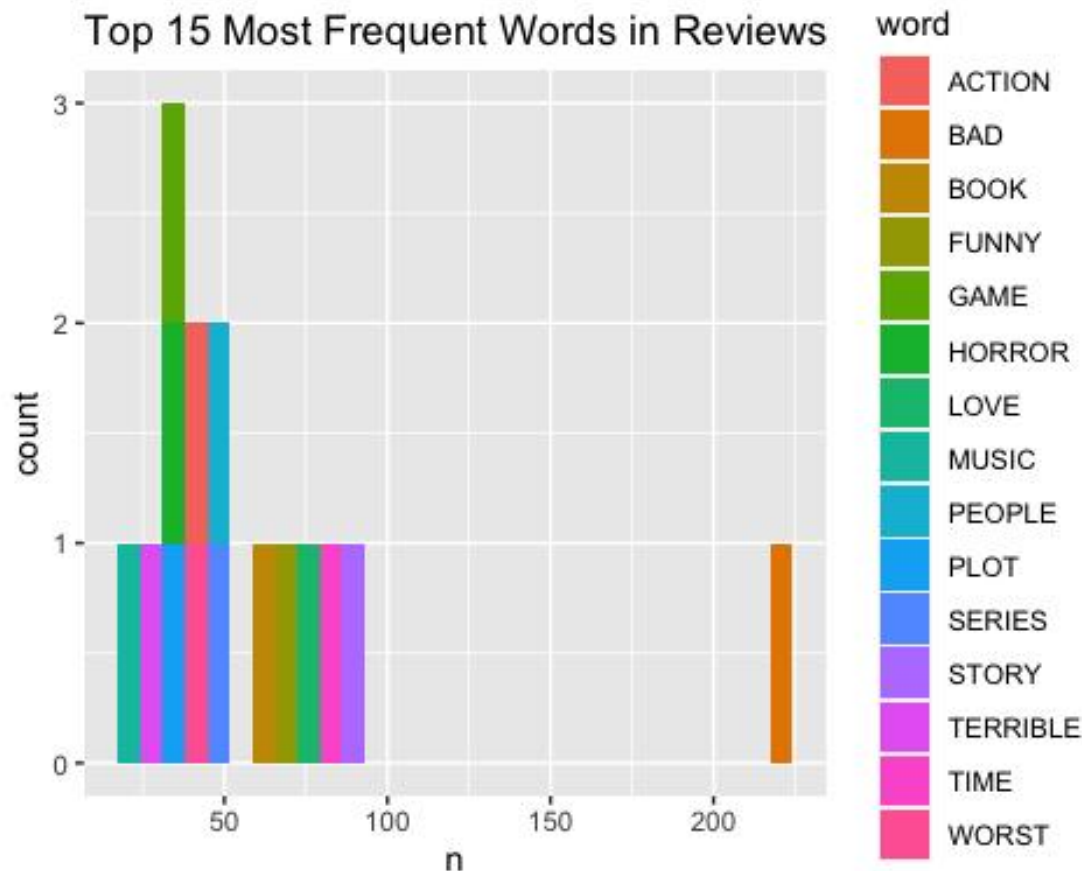
From the Top 15 Most Frequent Words in Reviews table, we get these words are ACTION, BAD, FUNNY, GAME, HORROR, LOVE, MUSIC, PEOPLE, PLOT, SERIES, STORY, TERRIBLE, TIME, and WORST. From above words, we could give a guess that people probably like watching movies about action, funny, game, love, music and so on. They watch movies with the aime at entertaining rather than learning something new. Or other words, to kill the time, which is also displayed in the Top 15 Most Frequent Words in Reviews table.

```
## We will again use the ggraph package for visualizing our networks. We plot
 networks of these co-occurring words so we can see these relationships bette
r.
library(igraph)

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tidyr':
##
##      crossing

## The following object is masked from 'package:tibble':
##
##      as_data_frame

## The following objects are masked from 'package:stats':
##
##      decompose, spectrum

## The following object is masked from 'package:base':
##
##      union

library(ggraph)
set.seed(1234)
word_pairs %>%
  filter(n >= 1800) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "cyan4")
+
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```
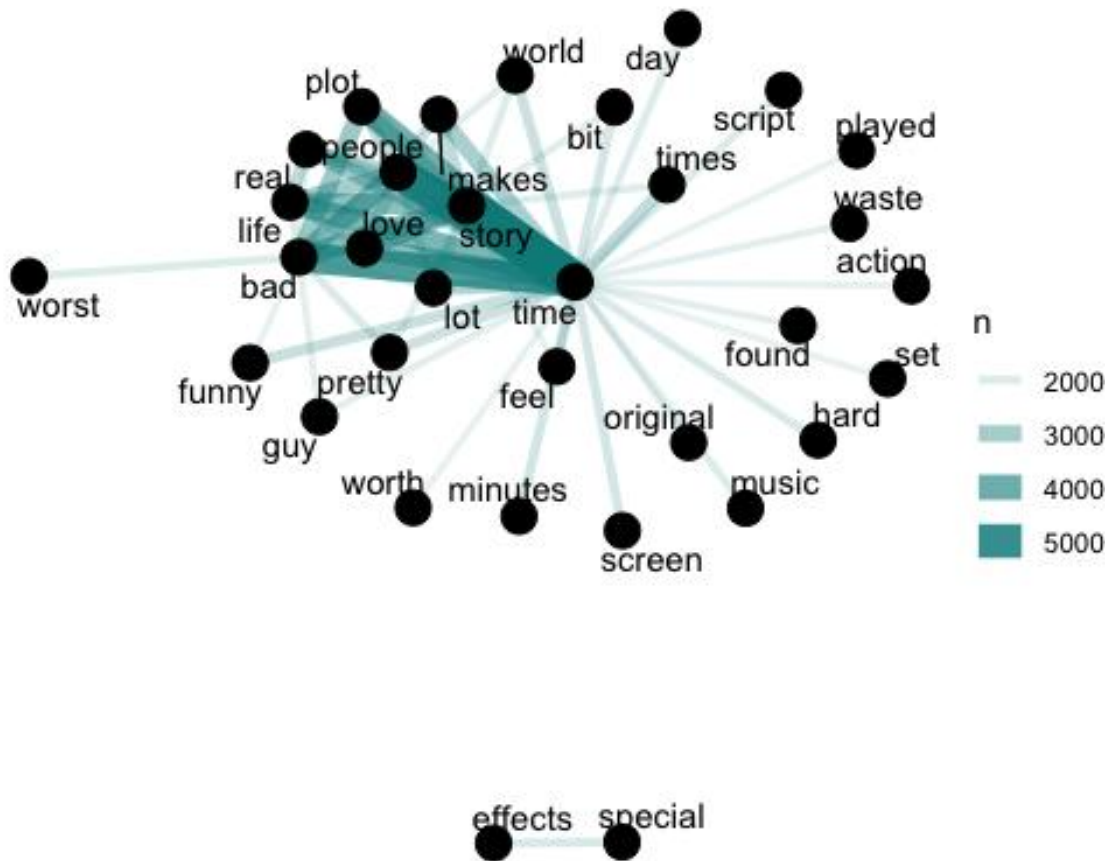
We see some clear clustering in this network of title words; word "time" is at center and is largely organized into several families of words that tend to go together. It is largely proved our guess that people watching movies is aimed at killing their leisure time, so they are prone to these relaxing types of movies.

```
###  Latent Dirichlet allocation.
imdb_dtm <- IMDB %>%
  unnest_tokens(word, review) %>%
  anti_join(stop_w)%>%
  count(docs, word) %>%
  cast_dtm(docs, word, n)

## Joining, by = "word"

## set a seed so that the output of the model is predictable
## A LDA_VEM topic model with 2 topics.
ap_lda <- LDA(imdb_dtm, k = 2, control = list(seed = 1234))

##The tidytext package provides this method for extracting the per-topic-per-
word probabilities, called ("beta"), from the model.
ap_topics <- tidy(ap_lda, matrix = "beta")

## We use dplyr's slice_max() to find the 10 terms that are most common withi
```
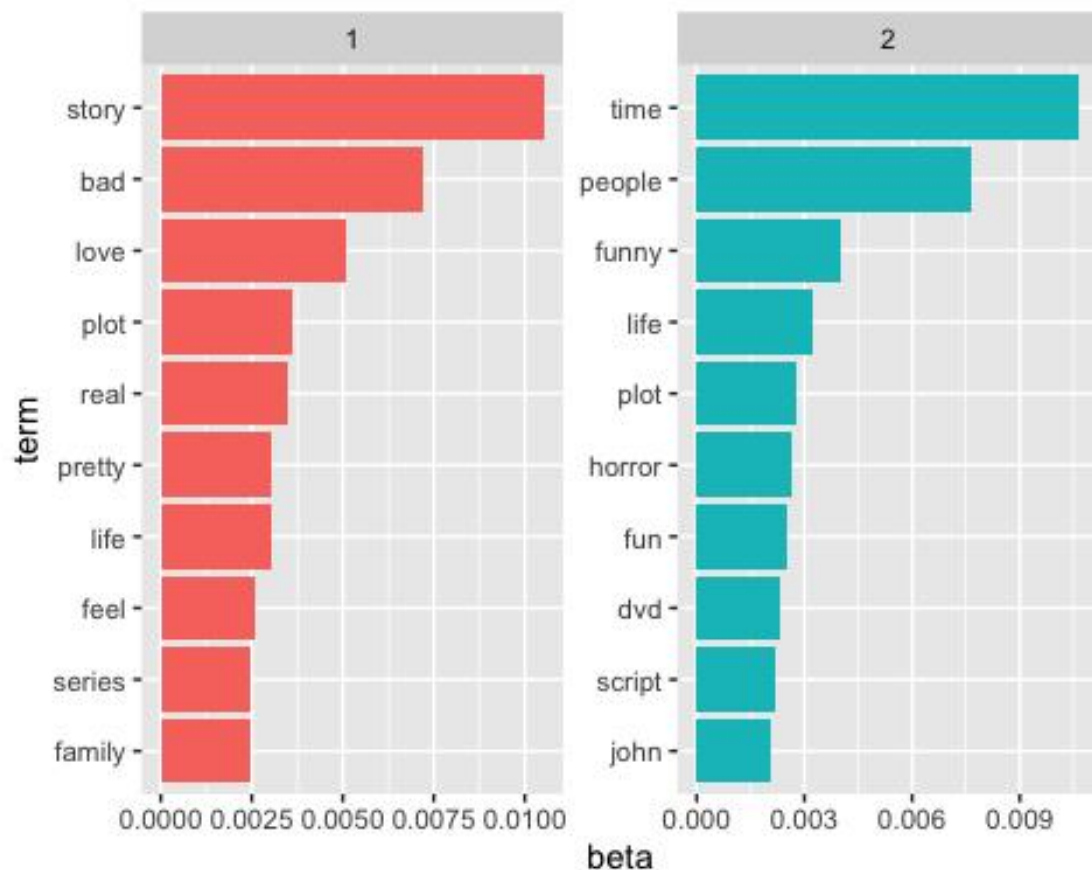
```r
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free", ncol = 5) +
  scale_y_reordered()
```


From above Top Term words table 1, we can see that the first movie contains words like story, love and life. We can infer that the type od first movie is family drama or story, which is suitable for families to watch together. However, Top Term words table 2, it contains word "horror" in the second movie, we guess it was a horror movie.

```r
library(tidyr)
library(hrbrthemes)
```

```
## NOTE: Either Arial Narrow or Roboto Condensed fonts are required to use th
ese themes.

##        Please use hrbrthemes::import_roboto_condensed() to install Roboto C
ondensed and

##        if Arial Narrow is not on your system, please see https://bit.ly/ari
alnarrow

ap_topics <- tidy(ap_lda, matrix = "beta")
ap_topics

## # A tibble: 237,846 × 3
##     topic term              beta
##     <int> <chr>            <dbl>
##  1      1 1             0.000864
##  2      2 1             0.00106
##  3      1 accustomed 0.0000179
##  4      2 accustomed 0.0000106
##  5      1 agenda        0.0000285
##  6      2 agenda        0.0000522
##  7      1 agreements 0.00000224
##  8      2 agreements 0.00000119
##  9      1 appeal        0.000134
## 10      2 appeal        0.000275
## # … with 237,836 more rows

## Consider the terms that had the greatest difference in beta between topic
1 and topic 2. This can be estimated based on the log ratio of the two.
## We visualize the words with the greatest differences between the two topic
s.
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))

head(beta_wide)

## # A tibble: 6 × 4
##    term       topic1     topic2 log_ratio
##    <chr>       <dbl>      <dbl>     <dbl>
## 1 1         0.000864 0.00106       0.301
## 2 called    0.00132  0.000107     -3.62
## 3 classic 0.00143    0.000321     -2.16
## 4 death     0.000656 0.00127       0.955
## 5 main      0.00106  0.00120       0.184
## 6 pretty    0.00306  0.000504     -2.60

ggplot(data = beta_wide, aes(x = x)) + geom_density( aes(x = topic1 , y = ..d
ensity..), fill="#69b3a2" ) + geom_label( aes(x = 0.01, y = 500, label="Topic
```
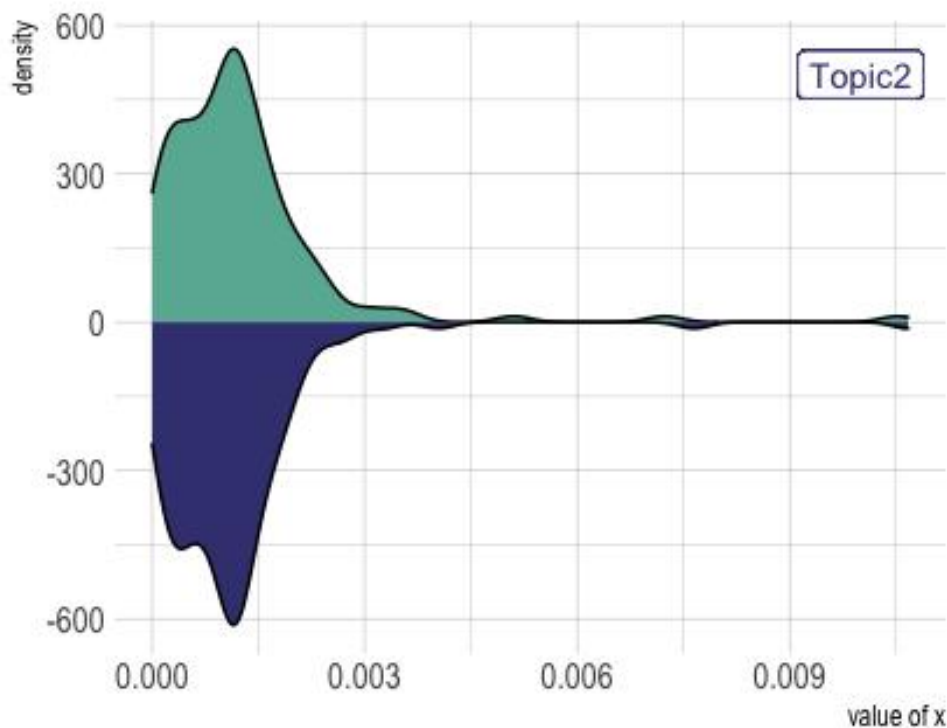
```
1"), color="#69b3a2") +
  # Bottom
  geom_density( aes(x = topic2, y = -..density..), fill= "#404080") +
  geom_label( aes(x = 0.01, y = 500,label="Topic2"), color="#404080") +
  theme_ipsum() +
  xlab("value of x")
```


From the density difference visulization plot, we get that in Topic 2, it is more concertrated at a word and has a higher density, compared with the Topic 1.

```
beta_wide %>%
  group_by(direction = log_ratio > 0) %>%
  slice_max(abs(log_ratio), n = 10) %>%
  ungroup() %>%
  mutate(term = reorder(term, log_ratio)) %>%
  ggplot(aes(log_ratio, term)) +
  geom_col() +
  labs(x = "Log2 ratio of beta in topic 2 / topic 1", y = NULL)
```

Log2 ratio of beta in topic 2 / topic 1
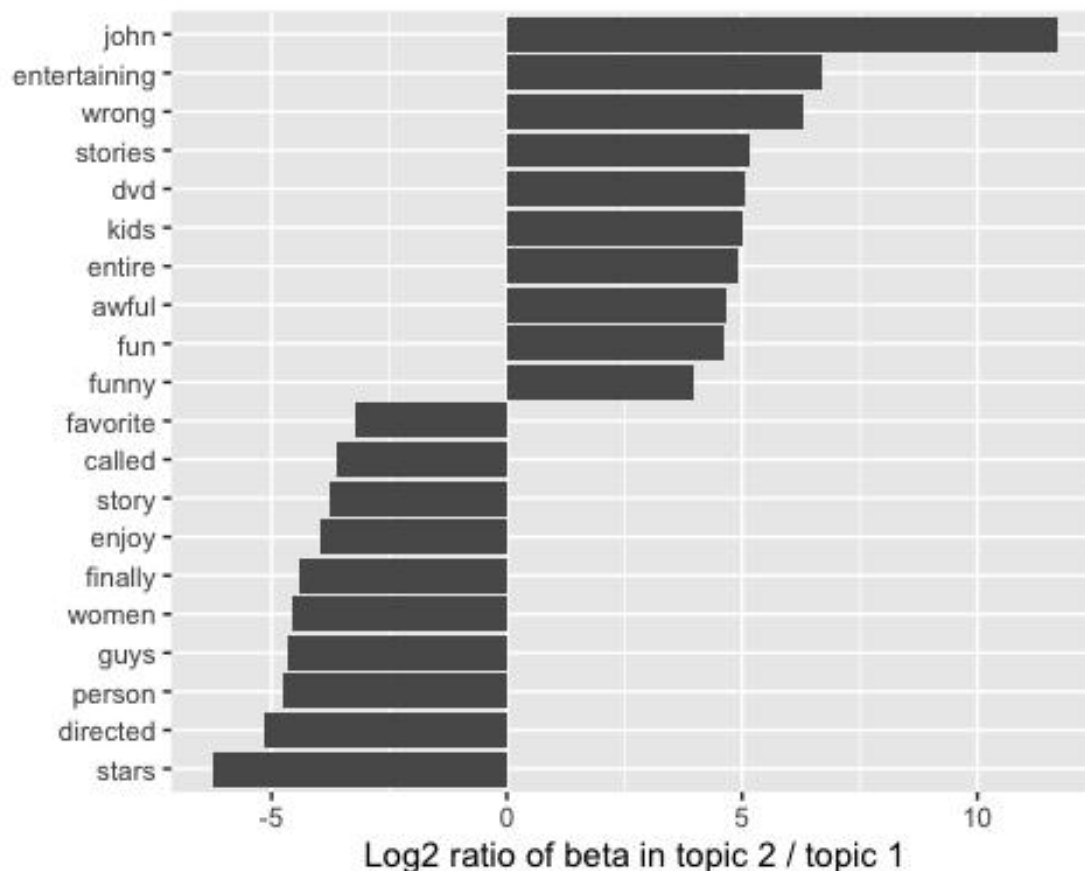
```r
## We then use the LDA() function to create a ten-topic model.
ap_lda <- LDA(imdb_dtm, k = 10, control = list(seed = 1234))

ap_topics <- tidy(ap_lda, matrix = "beta")
## We use dplyr's slice_max() to find the top 10 terms within each topic
ap_top_terms <- ap_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)
## This tidy output lends itself well to a ggplot2 visualization
ap_top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  ggplot(aes(beta, term, fill = factor(topic))) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~ topic, scales = "free", ncol = 4) +
  scale_y_reordered()
```
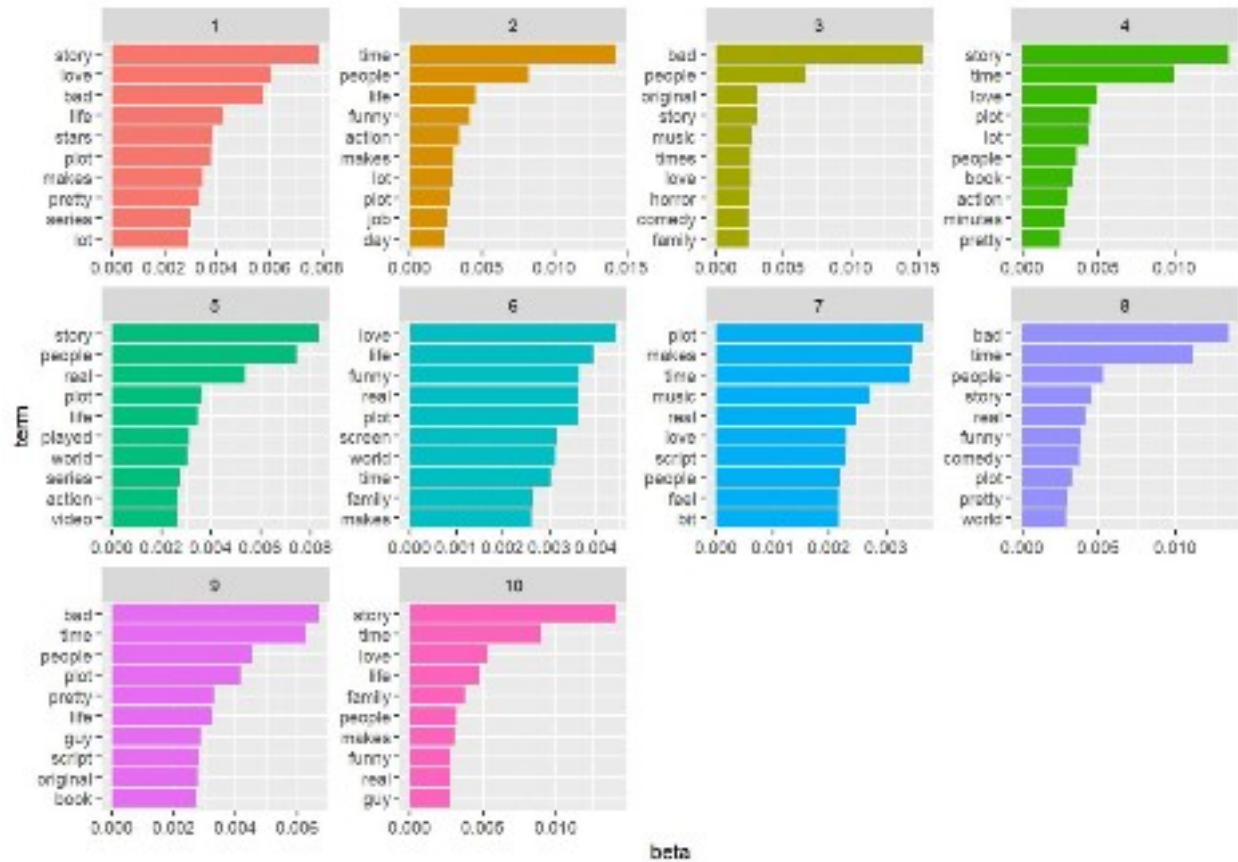
We get 10 topic top 10 words. Most of them contain words like time, love, life and so on. The β tells us the probability of that term being generated from that topic for that document. It is the probability of that term (word) belonging to that topic. Notice that some of the values for β are very, very low, and some are not so low.

```
tidy_lda <- tidy(ap_lda)
tidy_lda

## # A tibble: 1,189,230 × 3
##     topic term        beta
##     <int> <chr>      <dbl>
##  1      1 1       0.00136
##  2      2 1       0.00171
##  3      3 1       0.000214
##  4      4 1       0.000874
##  5      5 1       0.000227
##  6      6 1       0.0000530
##  7      7 1       0.00172
##  8      8 1       0.000877
##  9      9 1       0.000241
## 10     10 1       0.00236
## # … with 1,189,220 more rows
```

Let's examine the top 10 terms for each topic.

```
top_terms <- tidy_lda %>%
  group_by(topic) %>%
  slice_max(beta, n = 10, with_ties = FALSE) %>%
  ungroup() %>%
  arrange(topic, -beta)

top_terms

## # A tibble: 100 × 3
##    topic term        beta
##    <int> <chr>      <dbl>
##  1     1 story   0.00784
##  2     1 love    0.00604
##  3     1 bad     0.00574
##  4     1 life    0.00425
##  5     1 stars   0.00384
##  6     1 plot    0.00378
##  7     1 makes   0.00343
##  8     1 pretty  0.00333
##  9     1 series  0.00297
## 10     1 lot     0.00292
## # … with 90 more rows

top_terms %>%
  mutate(term = reorder_within(term, beta, topic)) %>%
  group_by(topic, term) %>%
  arrange(desc(beta)) %>%
  ungroup() %>%
  ggplot(aes(beta, term, fill = as.factor(topic))) +
  geom_col(show.legend = FALSE) +
  scale_y_reordered() +
  labs(title = "Top 10 terms in each LDA topic",
       x = expression(beta), y = NULL) +
  facet_wrap(~ topic, ncol = 4, scales = "free")
```

Top 10 terms in each LDA topic
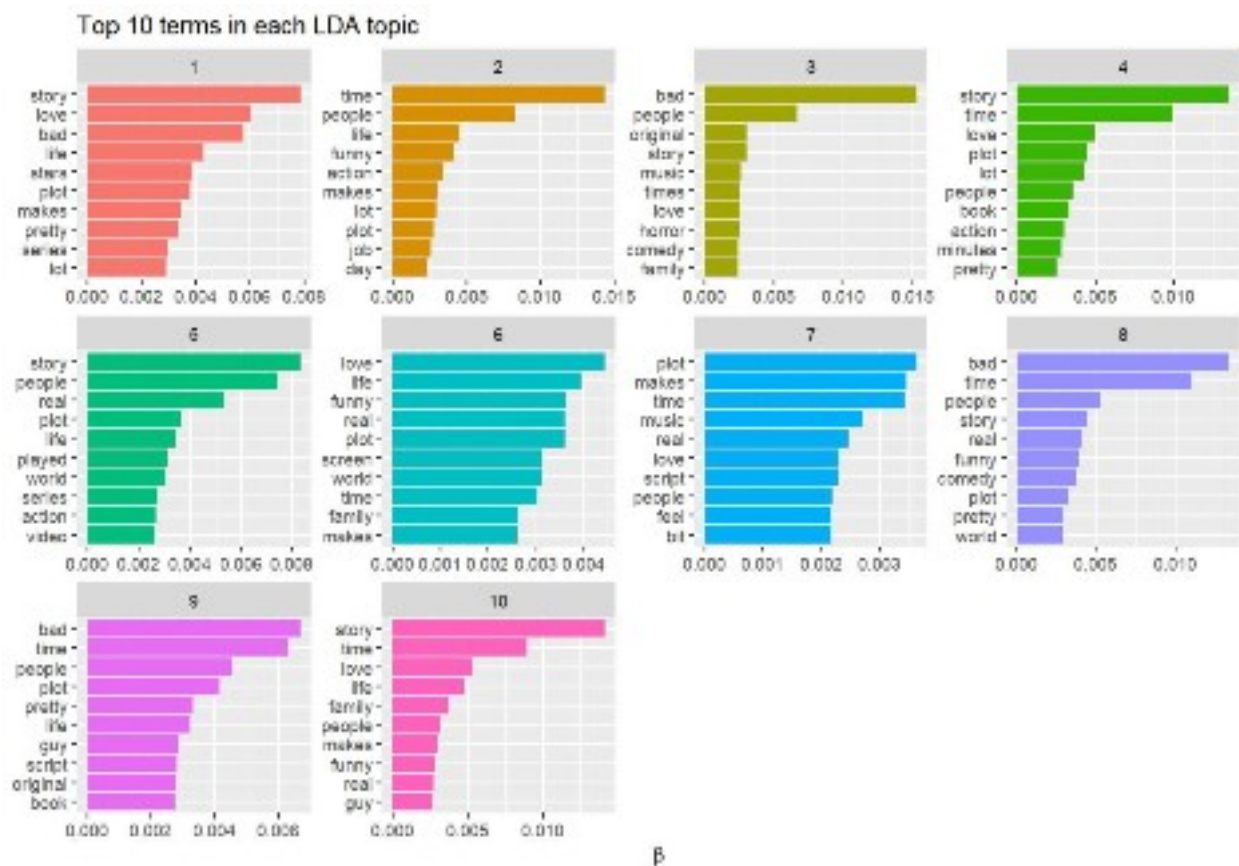

```
## We want to know which topics are associated with each document. We can fin
d this by examining the per-document-per-topic probabilities ("gamma").
lda_gamma <- tidy(ap_lda, matrix = "gamma")

## First we visualize for all topic and then we visualize the per-document-pe
r-topic probability for each topic.
ggplot(lda_gamma, aes(gamma)) +
  geom_histogram(alpha = 0.8, col = 'black', fill = 'light blue') +
  scale_y_log10() +
  labs(title = "Distribution of probabilities for all topics",
       y = "Number of documents", x = expression(gamma))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 13 rows containing missing values (geom_bar).
```
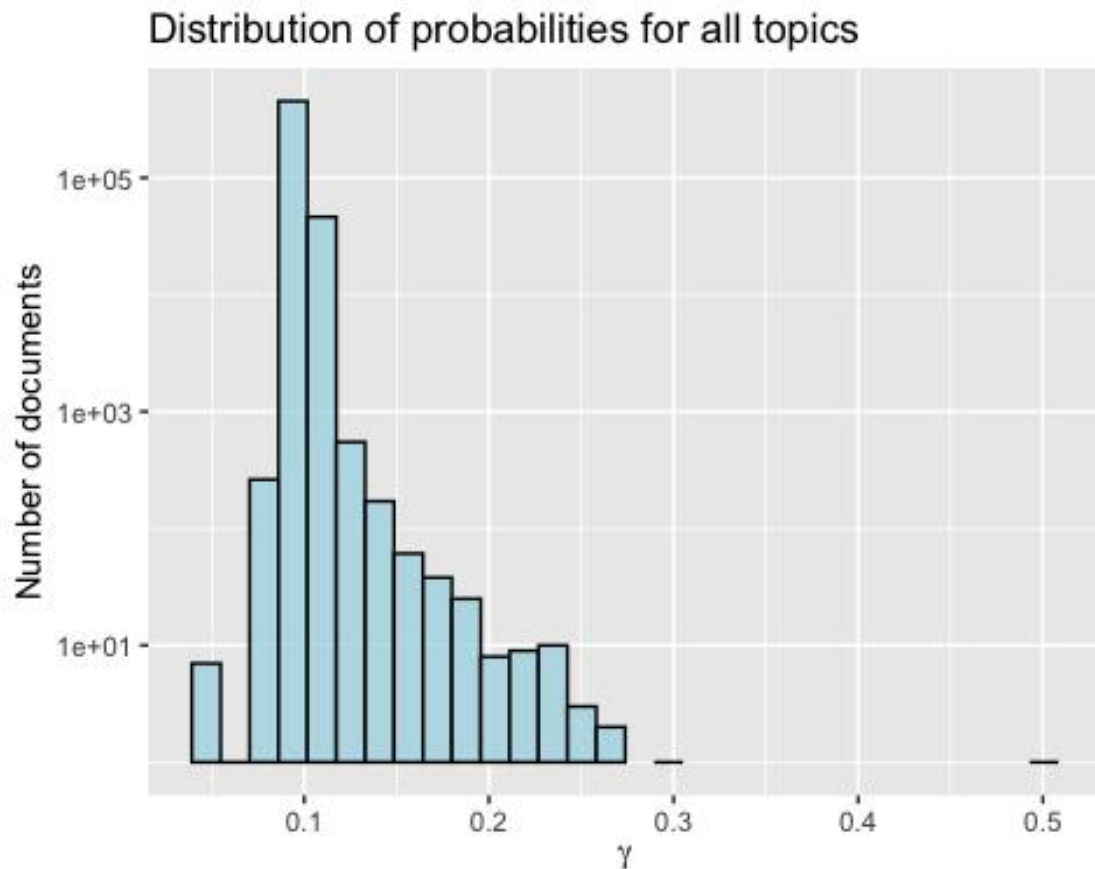
Distribution of probabilities for all topics


```
ggplot(lda_gamma, aes(gamma, fill = as.factor(topic))) +
  geom_histogram(alpha = 0.8, show.legend = FALSE) +
  facet_wrap(~ topic, ncol = 4) +
  scale_y_log10() +
  labs(title = "Distribution of probability for each topic",
       y = "Number of documents", x = expression(gamma))

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Transformation introduced infinite values in continuous y-axis

## Warning: Removed 194 rows containing missing values (geom_bar).
```

Distribution of probability for each topic
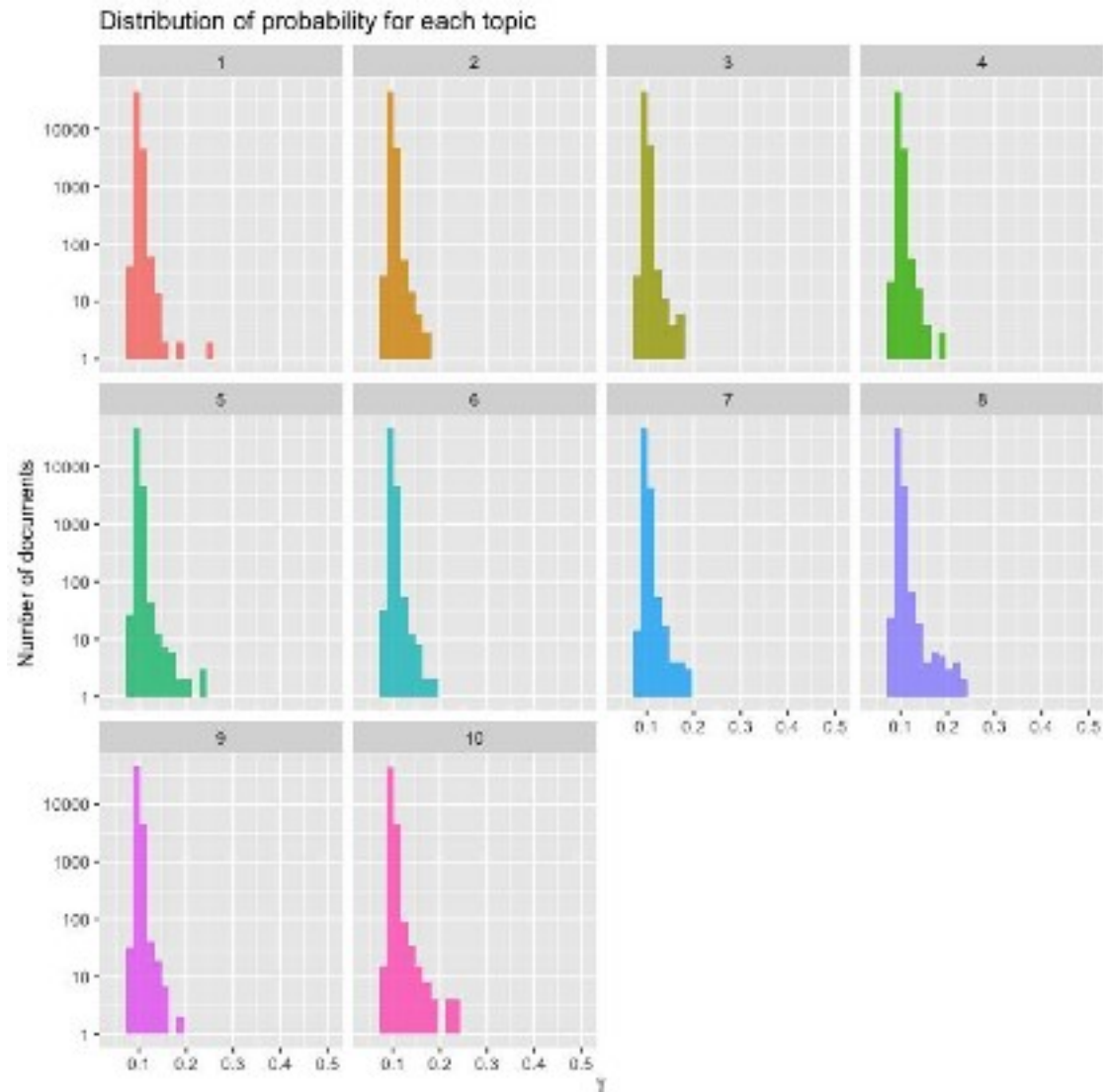
From the each topic visulization plots, we know that in each topic, there are similar distribution.