IMDB topic modeling

Zhi Tu, Tao Guo, Yalong Wang, Tianjian Xie

2022-11-08

Read data

```
imdb <- read.csv("IMDB Dataset.csv")
```

Clean data

```
# Dataset separating
# separate by html format

reviews <- imdb %>%
  mutate(review_number = row_number()) %>%
  separate(review,c("1", "2", "3", "4", "5","6"), sep="<br /><br />", convert = TRUE) %>%
  pivot_longer(c("1", "2", "3", "4","5","6"), names_to = "lines", names_transform = list(lines = as.int
  arrange(review_number, lines) %>%
  relocate(text) %>%
  tibble()

# separate by words

tidy_reviews <- reviews %>%
  unnest_tokens(word, text)
```

Building stop words

```
# custom stop words
data(stop_words)
custom_stop_words <- bind_rows(tibble(word = c("movie", "film", "movies", "time","story","plot","films"
                                      lexicon = c("custom")),
                               stop_words)

# words removed stop words
tidy_reviews <- tidy_reviews %>%
  drop_na %>%
  anti_join(custom_stop_words)
```

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

Checking for top frequent words

```r
tidy_reviews %>%
  count(word, sort = TRUE)
```

```
## # A tibble: 115,967 x 2
##    word       n
##    <chr>  <int>
##  1 bad    17663
##  2 people 17046
##  3 love   12414
##  4 life   12208
##  5 real    8943
##  6 funny   8422
##  7 pretty  6958
##  8 horror  6924
##  9 world   6895
## 10 comedy  6321
## # ... with 115,957 more rows
## # i Use `print(n = ...)` to see more rows
```

Build tf-idf tibble

```r
## Build tf-idf tibble
tidy_reviews_origin <- reviews %>% unnest_tokens(word, text)%>%
  count(review_number, word, sort = TRUE)
total_words_origin <- tidy_reviews_origin %>% group_by(review_number) %>%
  summarize(total = sum(n))

reviews_words_origin <- left_join(tidy_reviews_origin, total_words_origin)
```

```
## Joining, by = "review_number"
```

```r
review_tf_idf_origin <- reviews_words_origin %>%
  bind_tf_idf(word, review_number, n)
review_tf_idf_origin %>%
  select(-total) %>%
  arrange(desc(tf_idf))
```

```
## # A tibble: 6,787,230 x 6
##    review_number word                                    n     tf   idf tf_idf
##            <int> <chr>                               <int>  <dbl> <dbl>  <dbl>
##  1         10012 tenchi                                  1 0.25   10.1   2.53
##  2         18694 trivialboring                          26 0.220  10.8   2.38
##  3         45316 blahblahblahblahblahblahblahblah~       6 0.12   10.8   1.30
##  4         36845 stop.oz                                23 0.117  10.8   1.26
##  5         39183 smallville                              3 0.15    7.82  1.17
##  6         48928 smallville                              3 0.15    7.82  1.17
##  7         34098 kapoor                                  3 0.158   6.48  1.02
##  8         28921 primary                                 2 0.182   5.60  1.02
##  9         10012 spoilers                                1 0.25    3.91  0.978
## 10         22815 cognac                                 12 0.0896  9.43  0.845
## # ... with 6,787,220 more rows
## # i Use `print(n = ...)` to see more rows
```

LDA

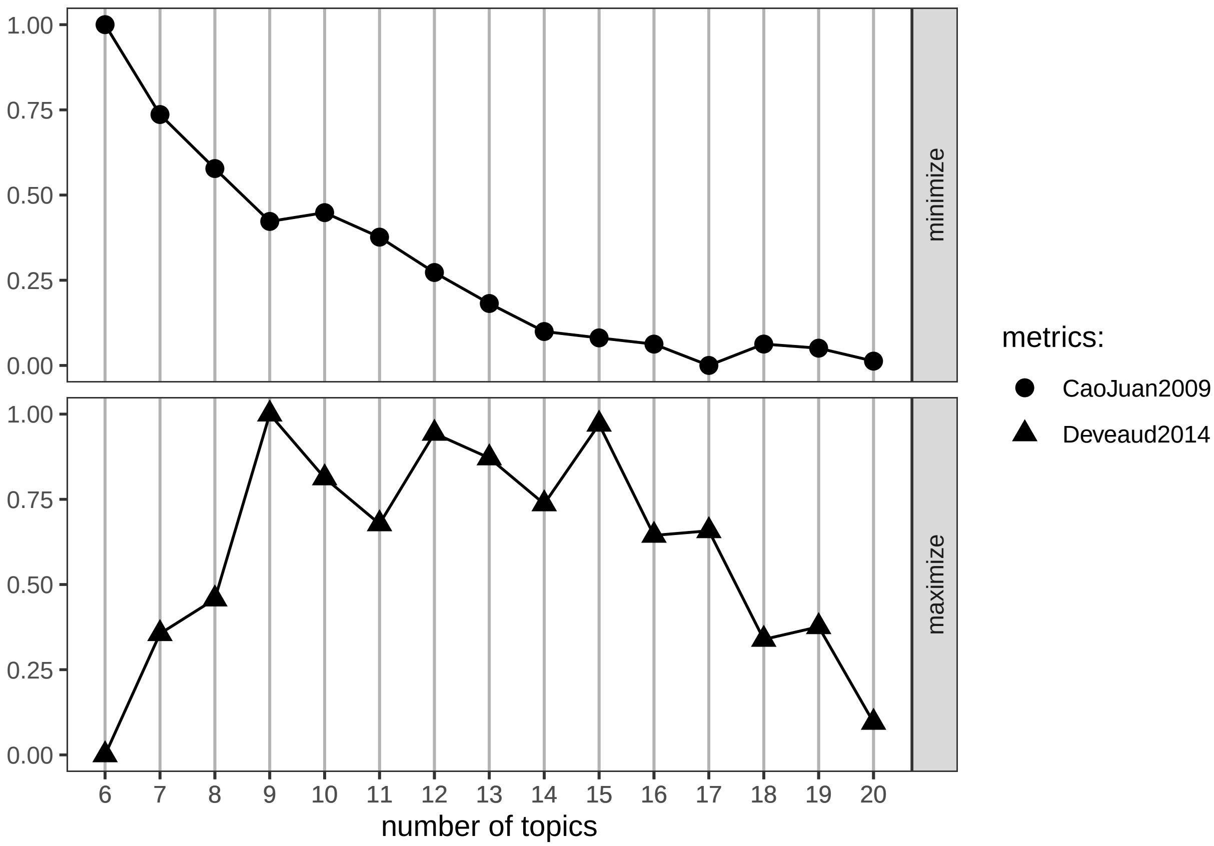
```
# transfer tidy words to dtm form
review_dtm <- tidy_reviews %>% select(-sentiment,-lines) %>%
  group_by(review_number) %>% count(word) %>% arrange(review_number,desc(n)) %>%
  cast_dtm(review_number, word,n)
```

```
# find proper number of topics
result <- ldatuning::FindTopicsNumber(
  review_dtm,
  topics = seq(from = 6, 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.
```

```
# plot the graph of performance of different numbers of topics
ldatuning::FindTopicsNumber_plot(result)
```

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



```r
# LDA to separate to 15 topics
review_lda <- LDA(review_dtm, k=15,method = "Gibbs", control = list(seed = 1234))
review_lda
```

```
## A LDA_Gibbs topic model with 15 topics.
```

```r
# result of posterior distributions
tmResult <- posterior(review_lda)
# show the probability of each review on all 15 topics
theta <- tmResult$topics
# show the probability of each words on all 15 topics
beta <- tmResult$terms
```

```r
# rank top topic terms for topic names
topicNames <- apply(lda::top.topic.words(beta, 5, by.score = T), 2, paste, collapse = " ")

# most probable topics in the collection
topicProportions <- colSums(theta) / nDocs(review_dtm)  # mean probabilities over all paragraphs
names(topicProportions) <- topicNames     # assign the topic names we created before
sort(topicProportions, decreasing = TRUE) # show summed proportions in decreased order
```

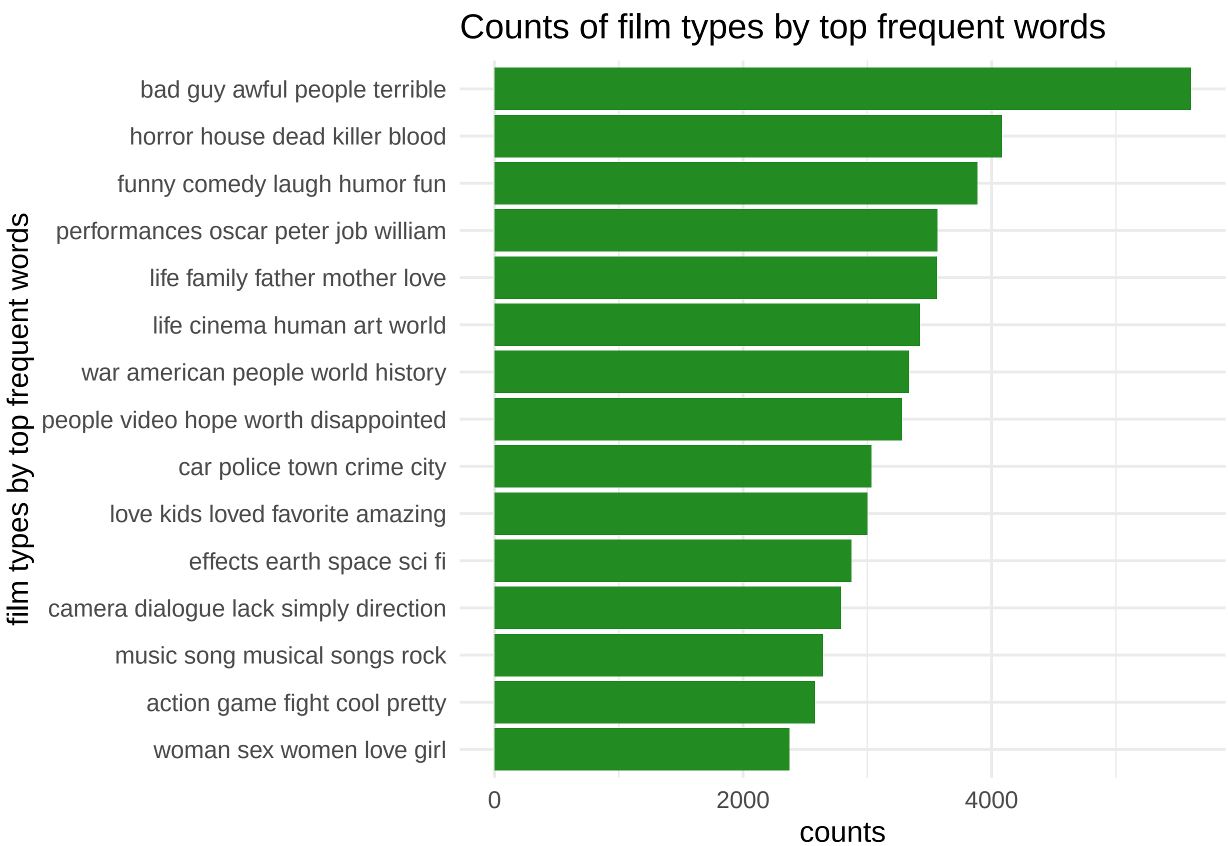
```
##         bad guy awful people terrible           life cinema human art world
##                            0.07128917                            0.07028111
##   people video hope worth disappointed camera dialogue lack simply direction
##                            0.06939899                            0.06892026
##        life family father mother love        horror house dead killer blood
##                            0.06767284                            0.06736978
##     war american people world history          funny comedy laugh humor fun
##                            0.06662282                            0.06598105
##   performances oscar peter job william           car police town crime city
##                            0.06591672                            0.06517982
##      love kids loved favorite amazing           effects earth space sci fi
##                            0.06471319                            0.06466635
##         music song musical songs rock            woman sex women love girl
##                            0.06438569                            0.06402828
##         action game fight cool pretty
##                            0.06357393
```

```r
countsOfPrimaryTopics <- rep(0, 15)
names(countsOfPrimaryTopics) <- topicNames
for (i in 1:nDocs(review_dtm)) {
  topicsPerDoc <- theta[i, ] # select topic distribution for document i
  # get first element position from ordered list
  primaryTopic <- order(topicsPerDoc, decreasing = TRUE)[1]
  countsOfPrimaryTopics[primaryTopic] <- countsOfPrimaryTopics[primaryTopic] + 1
}

counts <- data.frame(names(countsOfPrimaryTopics),countsOfPrimaryTopics)

# graph of most popular topics
counts <- counts %>% rename(class=names.countsOfPrimaryTopics.,
                            count=countsOfPrimaryTopics) %>%
  arrange(desc(count))
ggplot(counts) +
  aes(x = reorder(class,count), y = count) +
```

```
  geom_col(fill = "#228B22") +
  labs(x = "film types by top frequent words", y = "counts", title="Counts of film types by top frequent
  theme_minimal() +
  coord_flip()
```




```
# assign each document of topic with the top frequent words and assign film types to each review
imdb_classed <- imdb
New_topicNames <- c('Family','Villian','Crime','Kids','Comedy','Horror','Action','Oscar','History','Fema
for (i in 1:nDocs(review_dtm)) {
  max_index <- which.max(theta[i, ])[[1]] # select topic distribution for document i
  # get first element position from ordered list
  imdb_classed$top_words[i] <- topicNames[max_index]
  imdb_classed$class[i] <- New_topicNames[max_index]
}
head(imdb_classed)
```

```
##
## 1 One of the other reviewers has mentioned that after watching just 1 Oz episode you'll be hooked. Th
## 2
## 3
## 4
## 5
## 6
##   sentiment                              top_words    class
## 1  positive             car police town crime city    Crime
## 2  positive camera dialogue lack simply direction Dialogue
## 3  positive           funny comedy laugh humor fun   Comedy
```

```
## 4  negative        life family father mother love   Family
## 5  positive           life cinema human art world     Arts
## 6  positive        life family father mother love   Family
```