Topic_Modeling

Jiun Lee

2022-11-12

Importing Data

```
##Import Data
imdb <- read_csv ("IMDB Dataset.csv",
                   col_names = TRUE,
                   show_col_types = FALSE)
##Remove sentiment column since we don't use it.
imdb <- imdb %>% select(-sentiment)

##Remove duplicates of review, now we have 49582 observations.
imdb <- unique(imdb)

##Sample down for 100 reviews
set.seed(456)
review_index <- 1:dim(imdb)[1]
text_df <- cbind(review_index,imdb)
text_df <- text_df %>% slice_sample(n = 100, replace = FALSE)
rm(review_index)

#After a thorough cleaning, we now have a random sample of 100 observations data frame
```

Cleaning: Tokenizing, Removing stopwords

```
##The imdb dataset has a lot of stopwords and meaningless words. #We will remove stopwords and words un

library(stringr)
library(tidytext)
## Tokenizing, count the number of words within each review.
token <- text_df %>%
  unnest_tokens(word, review) %>%
  count(review_index, word, sort=TRUE) %>%
  rename(count=n)

##There's a lot of stop words. Let's remove them.

##Create a stop word vector
stop <- unlist(stop_words[,1])
##Drop the attribute
```

```r
stop <- StripAttr(stop)
##Restore tokens Data set
check <- token
##Check stop word lists again
remove <- check$word %in% stop
##To make it easier to see, create a data frame
d <- cbind(token,remove)
##Create an index of words(not stopwords)
f <- which(d$remove == FALSE)
##Clean tokens that has no stopwords
clean_token <- d %>% slice(f) %>% select(-remove)

##Let's subset the Clean_Token

##Vector that has meaningless words
strings <- c("br","movie","film", "scene", "character","story","bit","lot","bad","act","hard","awful","g

##Detect numbers of rows that has meaningless words
meaningless <- str_detect(clean_token$word, paste(strings, collapse = "|"))

##Detect numbers of meaningful rows
meaningful <- which(meaningless==F)

##Subset: tokens without meaningless
clean_token <- clean_token %>% slice(meaningful)

##Remove redundant data and values
rm(d,check,f,meaningless,meaningful,strings,stop,remove,token)

##Now we have our new clean_token data with only 3776 observations
```

TF-IDF (Term Frequency - Inverse Document Frequency)

```r
##Let's look tf-idf to see what is the most important words in the whole reviews.
review_tf_idf <- clean_token %>%
  bind_tf_idf(review_index, word, count)
```

Look at terms with high tf-idf in reviews.

```r
review_tf_idf<- review_tf_idf %>%
  arrange(desc(tf_idf))
head(review_tf_idf,20)
```

```
##    review_index       word count tf      idf   tf_idf
## 1         25567      trick     1  1 6.042336 6.042336
## 2          4237     babies     1  1 5.819192 5.819192
## 3          4237 creepiness     1  1 5.819192 5.819192
## 4          4237       pure     1  1 5.819192 5.819192
```

```
## 5          4237 resolution     1   1 5.819192 5.819192
## 6          4237  restraint     1   1 5.819192 5.819192
## 7          4237   sniffing     1   1 5.819192 5.819192
## 8          4237     thomas     1   1 5.819192 5.819192
## 9         18083      empty     1   1 5.819192 5.819192
## 10        18083    georges     1   1 5.819192 5.819192
## 11        18083   retarded     1   1 5.819192 5.819192
## 12        18083     suffer     1   1 5.819192 5.819192
## 13        12248        cry     1   1 5.636871 5.636871
## 14        12248    debased     1   1 5.636871 5.636871
## 15        12248       eats     1   1 5.636871 5.636871
## 16        12248   everbody     1   1 5.636871 5.636871
## 17        12248   everyday     1   1 5.636871 5.636871
## 18        12248    glorify     1   1 5.636871 5.636871
## 19        12248     mained     1   1 5.636871 5.636871
## 20        21756     coster     1   1 5.636871 5.636871
```

High tf-idf's words are identified as words that are important to one document within a collection of documents.

tf-idf algorithm will think those are very important words.

Look at terms with low tf-idf in reviews.

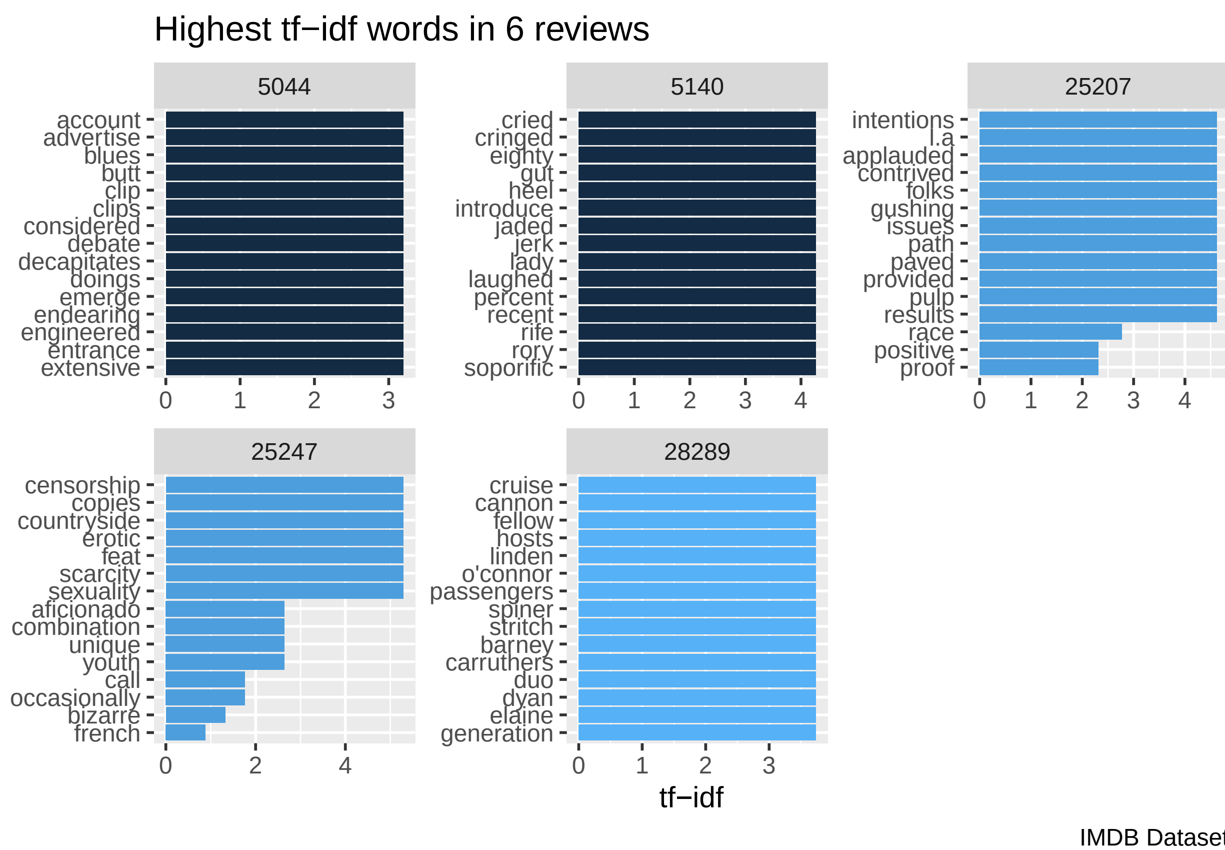
```r
review_tf_idf<- review_tf_idf %>%
  arrange(tf_idf)
head(review_tf_idf,20)
```

```
##    review_index   word count         tf      idf     tf_idf
## 1         37209   love     1 0.03571429 2.608349 0.09315531
## 2         35484   love     1 0.03571429 2.869504 0.10248229
## 3           320   love     1 0.03571429 3.421297 0.12218918
## 4         39350   love     1 0.03571429 3.448949 0.12317674
## 5          8473   love     1 0.03571429 3.599989 0.12857103
## 6         21531   love     1 0.03571429 3.621968 0.12935599
## 7         15595   love     1 0.03571429 3.715058 0.13268065
## 8         45339   love     1 0.03571429 3.791044 0.13539443
## 9         45237   love     1 0.03571429 4.027433 0.14383689
## 10        45274  music     1 0.04545455 3.173018 0.14422807
## 11          320  music     1 0.04545455 3.421297 0.15551350
## 12        27472   love     1 0.03571429 4.408205 0.15743591
## 13        35659   love     1 0.03571429 4.458216 0.15922199
## 14        37209 played     1 0.06250000 2.608349 0.16302179
## 15         7664   love     1 0.03571429 4.720580 0.16859214
## 16        19086 played     1 0.06250000 2.705677 0.16910481
## 17         4008   love     1 0.03571429 4.754482 0.16980291
## 18        11593   love     1 0.03571429 4.789573 0.17105618
## 19        45893  music     1 0.04545455 3.777972 0.17172600
## 20        39511   love     1 0.03571429 4.863681 0.17370289
```

The inverse document frequency is very low almost zero for words occuring in many documents; thus tf_idf is very low too. The word 'love' is common in the documents.

Highest tf-idf words

```
##Let's make the plots with only 6 review_index.
review_tf_idf %>%
  filter(review_index %in% c(25207,5044,5140,28289,25247)) %>%
  arrange(desc(tf_idf)) %>%
  group_by(review_index) %>%
  distinct(word,review_index, .keep_all = TRUE) %>%
  slice_max(tf_idf, n = 15, with_ties = FALSE) %>%
  ungroup() %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  ggplot(aes(tf_idf, word, fill = review_index)) +
  geom_col(show.legend = FALSE) +
  facet_wrap(~review_index, ncol = 3, scales = "free") +
  labs(title = "Highest tf-idf words in 6 reviews",
       caption = "IMDB Dataset",
       x = "tf-idf", y = NULL)
```



On each 6 plot, we can see top 15 words with high tf-idf.

Among them, we can verify some meaningful words for checking their genres.

For example, in review'25247', the words 'censorship','erotic','sexuality' imply that the review is about romance movie.

Latent Dirichelet Allocation model

```
library(topicmodels)

##Convert sample token tibble to DTM(document term matrix) for LDA
clean_token_dmat <- clean_token %>%
  cast_dtm(review_index, word, count)

##Select k= 6 because we have 6 general film genres
imdb_lda <- LDA(clean_token_dmat, k = 6, control = list(seed = 1234))
imdb_lda #topic model with 6 topics.
```

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

```
#extracting the Topic Word Matrix (per-topic-per-word probabilities)
imdb_topics <- tidy(imdb_lda, matrix = "beta")
imdb_topics
```

```
## # A tibble: 20,202 x 3
##    topic term        beta
##    <int> <chr>      <dbl>
##  1     1 timon  1.06e-293
##  2     2 timon  5.60e-296
##  3     3 timon  1.10e-297
##  4     4 timon  3.58e-  2
##  5     5 timon  1.53e-297
##  6     6 timon  7.24e-296
##  7     1 pumbaa 8.40e-294
##  8     2 pumbaa 1.17e-296
##  9     3 pumbaa 4.73e-298
## 10     4 pumbaa 1.86e-  2
## # i 20,192 more rows
```

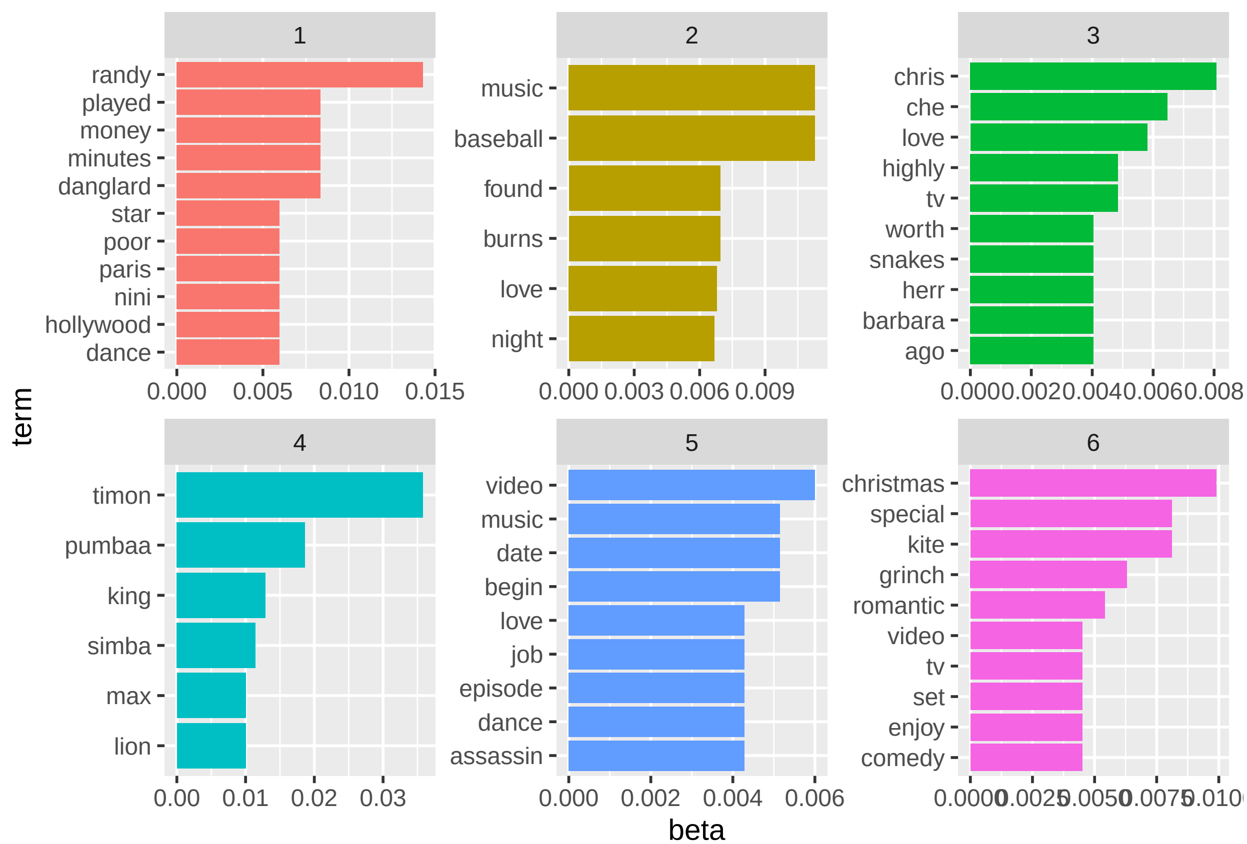
Probability of that term being generated from that topic.

For example, the term "timon" has an almost zero probability of being generated from topics 1, 2, or 3, but it makes up 3% of topic 4.

Visualization: the most common words within each topic.

```
library(ggplot2)
##Get top used terms and arrange them
imdb_top_terms <- imdb_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 6) %>%
  ungroup() %>%
  arrange(topic, -beta)

##Create the plot
imdb_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") +
  scale_y_reordered()
```



The plot shows the most common words within each topic.

We've split up our LDA into 6 genres, which represents the number of topics we have.

Topic 4 has common words such as 'timon', 'pumbaa', 'king', and 'lion'. It represents it's an animation movie 'Lion King".

Now let's look at Document-topic probabilities.

```r
##Document Topic Matrix (per-document-per-topic probabilities)
imdb_documents <- tidy(imdb_lda, matrix = "gamma")
imdb_documents %>%
  arrange(desc(gamma)) ##descending sort
```

```
## # A tibble: 600 x 3
##    document topic gamma
##    <chr>    <int> <dbl>
##  1 35484        4 1.00
##  2 37209        1 1.00
##  3 19086        6 1.00
##  4 15420        5 1.00
##  5 16613        4 1.00
##  6 5044         2 1.00
##  7 45274        2 1.00
##  8 29214        1 0.999
##  9 320          2 0.999
## 10 6706         5 0.999
## # i 590 more rows
```

Each document as a mixture of topics.

Each of these values is an estimated proportion of words from that document that are generated from that topic. The model estimates that 99% of the words in document 35484 were generated from topic 4.

Word Assignment: assigning each word in each document to a topic

```r
assignments <- augment(imdb_lda, data = clean_token_dmat)
assignments
```

```
## # A tibble: 5,286 x 4
##    document term      count .topic
##    <chr>    <chr>     <dbl>  <dbl>
##  1 35484    timon        25      4
##  2 35484    pumbaa       13      4
##  3 29214    randy        12      1
##  4 12301    christmas    11      6
##  5 320      baseball     10      2
##  6 38255    baseball      3      2
##  7 19086    kite          9      6
##  8 320      burns         8      2
##  9 8473     chris         8      3
## 10 14281    chris         2      3
## # i 5,276 more rows
```

```
##The assignments tibble above count up the words for each topic.
```

The document 35484 - term 'timon' pair was assigned to topic 4.