Topic Modeling G1

Group 1

2022-11-14

```
library(dplyr)
library(stringr)
library(tidyr)
library(tidytext)
library(tidyverse)
library(topicmodels)
library(stopwords)
library(igraph)
library(ggraph)
library(ggplot2)
library(DescTools)
library(widyr)
```

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

```
imdb <- imdb %>% select(-sentiment)
imdb_df <- tibble(review = 1:50000, sentence = imdb[,1])
imdb_df <- imdb_df[1:1000,]
text_df <- imdb_df %>% slice_sample(n = 100, replace = FALSE)
```

Bigram

```
token_bigram <- text_df %>%
  unnest_tokens(bigram,
                sentence,
                token = "ngrams",
                n = 2,
                to_lower=TRUE) %>%
  count(review,bigram,sort = TRUE)%>%
  filter(!is.na(bigram))

## create a stop word vector, drop the attribute, drop the attribute
stop <-  unlist(stop_words[,1])
stop <- StripAttr(stop)
stop <- c(stop, "br")

## split the bigram list into two columns
check <-  token_bigram %>% separate(bigram,
```

```r
                                    sep= " ",
                                    c("w1", "w2"))

## check both words individually agains stop word lists
a <- check$w1 %in% stop
b <- check$w2 %in% stop
## the bigram is included only if neither of the single words is a stop word
remove <- (a|b)
## to make it easier to see create a data frame
d <- cbind(token_bigram, a, b, remove)

d <- d %>% filter(d$a !="br" && d$b != "br")
```

```
## Warning in d$a != "br" && d$b != "br": 'length(x) = 21383 > 1' in coercion to
## 'logical(1)'

## Warning in d$a != "br" && d$b != "br": 'length(x) = 21383 > 1' in coercion to
## 'logical(1)'
```

```r
## create an index of bigram
f <- which(d$remove == FALSE)
## use the index to make a list of bigrams
g <- d$bigram[f]

(review_separated <- token_bigram %>%
  separate(bigram, into = c("word1", "word2"), sep = " ")
)
```

```
## # A tibble: 21,383 x 4
##    review word1 word2     n
##     <int> <chr> <chr> <int>
##  1    557 of    the      13
##  2    425 br    br       10
##  3    679 of    the       9
##  4    178 of    the       8
##  5    274 of    the       8
##  6    557 br    br        8
##  7    718 of    the       8
##  8    992 br    br        8
##  9     30 of    the       7
## 10    187 this  movie     7
## # ... with 21,373 more rows
```

```r
review_united <- review_separated %>%
  filter(!word1 %in% c('br'),
         !word2 %in% c('br')) %>%
  unite(bigram, c(word1, word2), sep = " ")

total_bigram <- review_united %>%
  group_by(review) %>%
  summarize(total = sum(n))

review_bigram <- left_join(review_united, total_bigram)
```

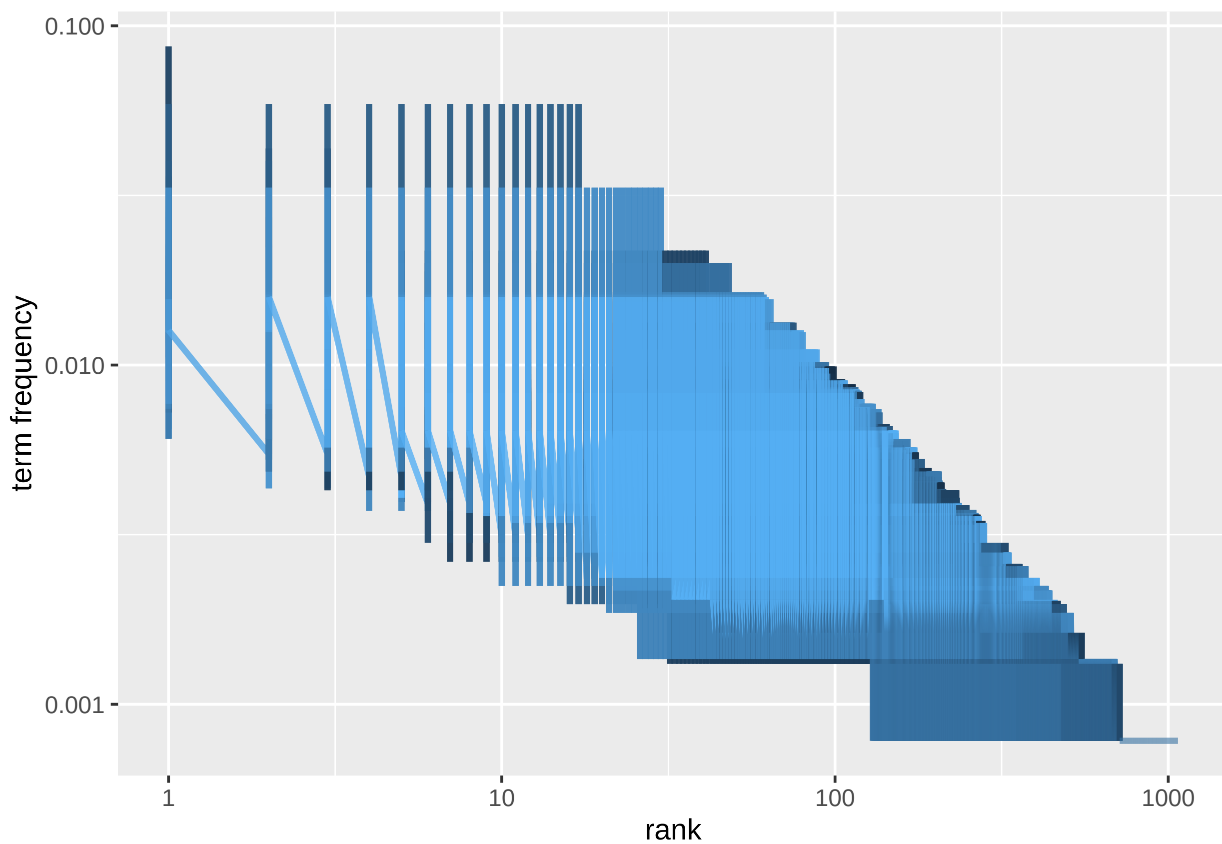
```
## Joining, by = "review"
```

```
rm(token_bigram, review_separated, review_united, total_bigram)
```

frequency

```
freq_by_rank_bi <- review_bigram %>%
  group_by(review) %>%
  mutate(rank = row_number(),
         `term frequency` = n/total) %>%
  ungroup()

freq_by_rank_bi %>%
  ggplot(aes(rank, `term frequency`, color = review)) +
  geom_line(size = 1.1, alpha = 0.8, show.legend = FALSE) +
  scale_x_log10() +
  scale_y_log10()
```



From this graph, we can see that the word frequency has a decrease tendency. By this term frequency graph, we can choose words with the highest frequency and consider them as stop words. The following tf-idf is the method we test bigrams and find stop words.

tf-idf

```
review_tf_idf_bi <- review_bigram %>%
  bind_tf_idf(bigram, review, review)
#look at terms with high tf-idf in reviews.
review_tf_idf_bi <- review_tf_idf_bi %>%
  select(-total) %>%
  arrange(desc(tf-idf))

head(review_tf_idf_bi)
```

```
## # A tibble: 6 x 6
##   review bigram     n     tf   idf  tf_idf
##    <int> <chr>  <int>  <dbl> <dbl>   <dbl>
## 1    538 of the     1 0.0208 0.511 0.0106
## 2    823 of the     1 0.0135 0.511 0.00690
## 3    348 of the     1 0.0133 0.511 0.00681
## 4    858 of the     1 0.0125 0.511 0.00639
## 5    928 of the     1 0.0103 0.511 0.00527
## 6    637 of the     1 0.0102 0.511 0.00521
```

select bigram stop words

```
stopwords <- as.vector(review_tf_idf_bi$bigram)
u1 <- unique(stopwords)
stopwords <- data.frame(u1)
sw <- as.character(stopwords$u1[1:10000])
sw <- tibble(sw)
head(sw)
```

```
## # A tibble: 6 x 1
##   sw
##   <chr>
## 1 of the
## 2 in the
## 3 to the
## 4 is a
## 5 this movie
## 6 and the
```

single-word stop words

```
imdb <- imdb  %>%  mutate(docs = c(1:length(imdb$review)))
data(stop_words)
stop_words <- data.frame(stop_words$word)
stop_words <- rbind(stop_words, "br", "movie", "film", "movies", "films", "scenes", "scene", "character
colnames(stop_words) <- c("word")
```

For the LDA, we choose to use single-word stop words, because they learn beta, the per-topic-per-word probabilities, from the text book.

LDA

```
imdb_dtm <- imdb %>%
  unnest_tokens(word, review) %>%
  anti_join(stop_words)%>%
  count(docs, word) %>%
  cast_dtm(docs, word, n)
```

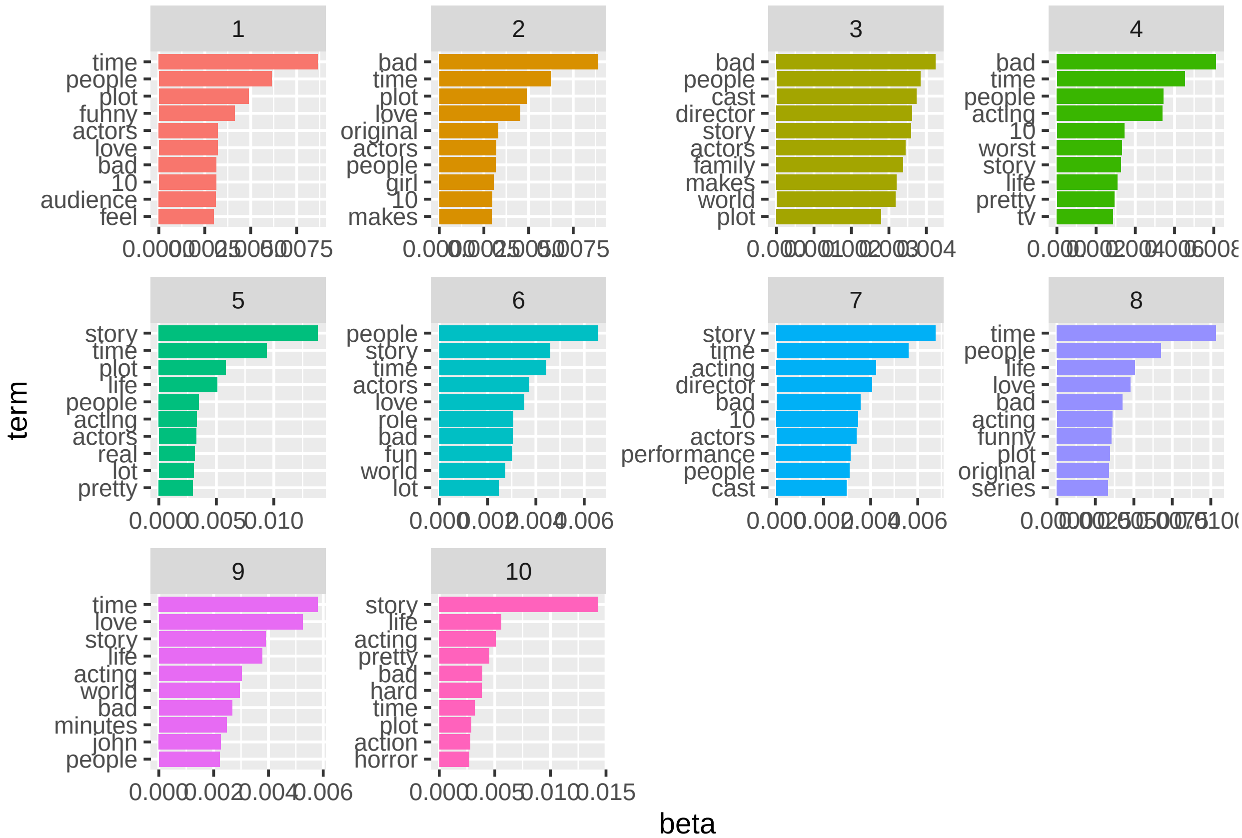
```
## Joining, by = "word"
```

```
imdb_lda <- LDA(imdb_dtm, k = 10, control = list(seed = 2022))
imdb_topics <- tidy(imdb_lda, matrix = "beta")
imdb_topics
```

```
## # A tibble: 1,189,320 x 3
##    topic term       beta
##    <int> <chr>     <dbl>
##  1     1 1     0.000430
##  2     2 1     0.000636
##  3     3 1     0.00177
##  4     4 1     0.00198
##  5     5 1     0.000277
##  6     6 1     0.000330
##  7     7 1     0.0000457
##  8     8 1     0.00166
##  9     9 1     0.000552
## 10    10 1     0.00181
## # ... with 1,189,310 more rows
```

```
imdb_top_terms <- imdb_topics %>%
  group_by(topic) %>%
  slice_max(beta, n = 10) %>%
  ungroup() %>%
  arrange(topic, -beta)

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()
```




```r
beta_wide <- imdb_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))
```

We seperate the data to 10 topic, according to this, there are meaningful differences between this words, we can get the label of the films, “horror”, “love”, “family” etc.

Document Classification

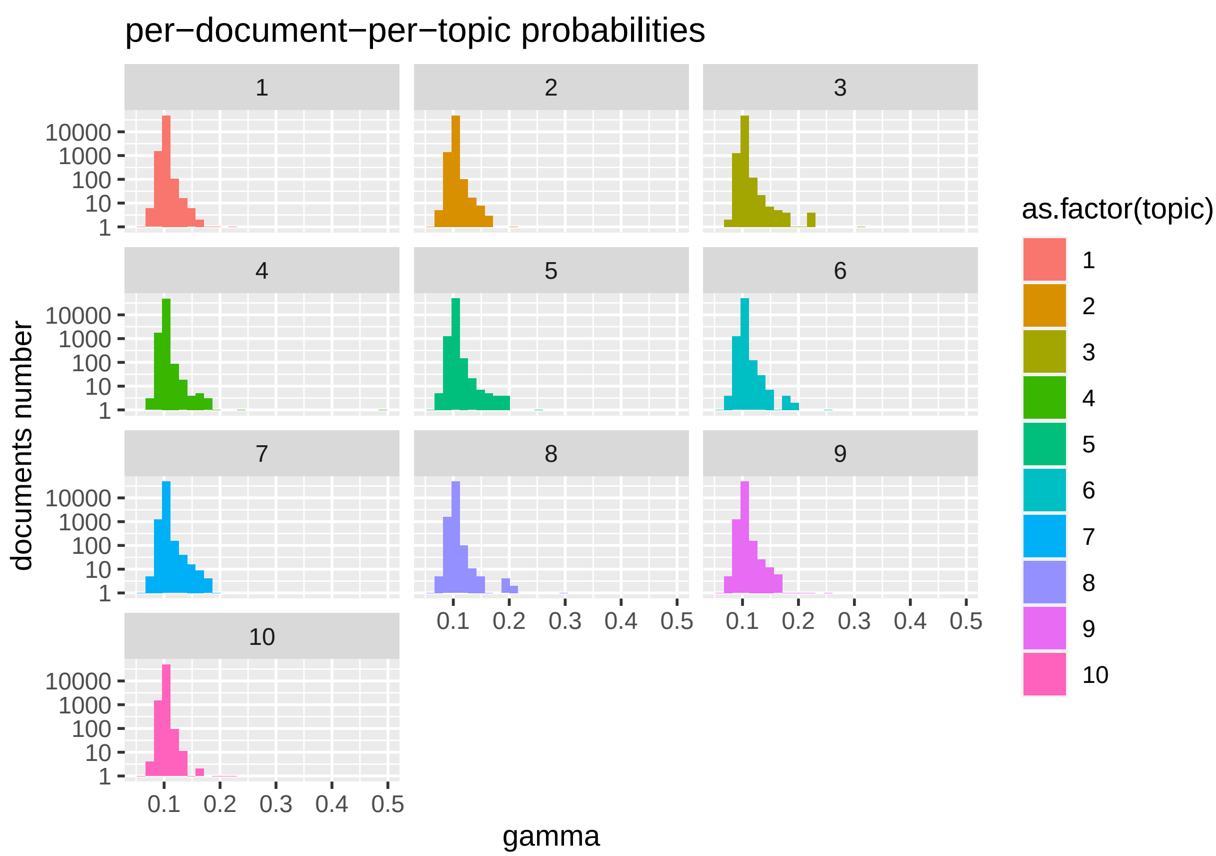
```r
imdb_documents <- tidy(imdb_lda, matrix = "gamma")
# check the per-document-per-topic probabilities using gamma
imdb_documents <- imdb_documents %>%
  separate(document, c("title"),sep = "_", convert = TRUE)
head(imdb_documents)
```

```
## # A tibble: 6 x 3
##   title topic  gamma
##   <int> <int>  <dbl>
## 1     1     1 0.102
## 2     2     1 0.0981
```

```
## 3      3      1 0.102
## 4      4      1 0.101
## 5      5      1 0.101
## 6      6      1 0.0997
```

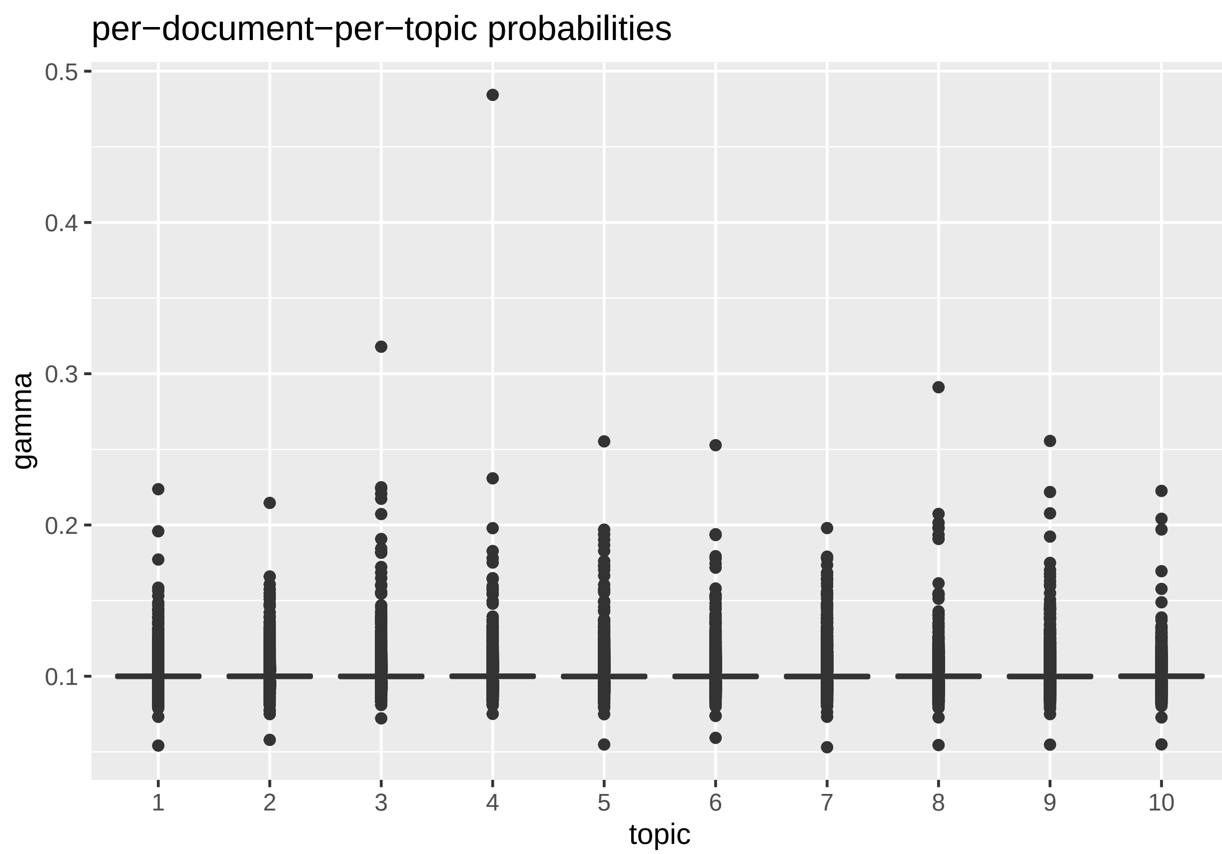
According to imdb_document, each of these values is an estimated proportion of words in the document that were generated from that topic. We check the per-document-per-topic probabilities using gamma.

```
ggplot(imdb_documents, aes(x = gamma , fill = as.factor(topic))) +
  geom_histogram()+
  facet_wrap(~topic, ncol = 3) +
  scale_y_log10() +
  labs(title = "per-document-per-topic probabilities",
       y = "documents number", x= "gamma")
```



The plots above demonstrates the per document per topic probabilities of the words.the x-axis illustrates the per-document-per-topic probabilities, y-axis is the document number and different color represents different topics.

```
ggplot(imdb_documents, aes(factor(topic),gamma )) +
  geom_boxplot() +
  labs(title = "per-document-per-topic probabilities",
       y = "gamma", x= "topic")
```



The box plot above demonstrates the gamma probabilities for each chapter within each book.

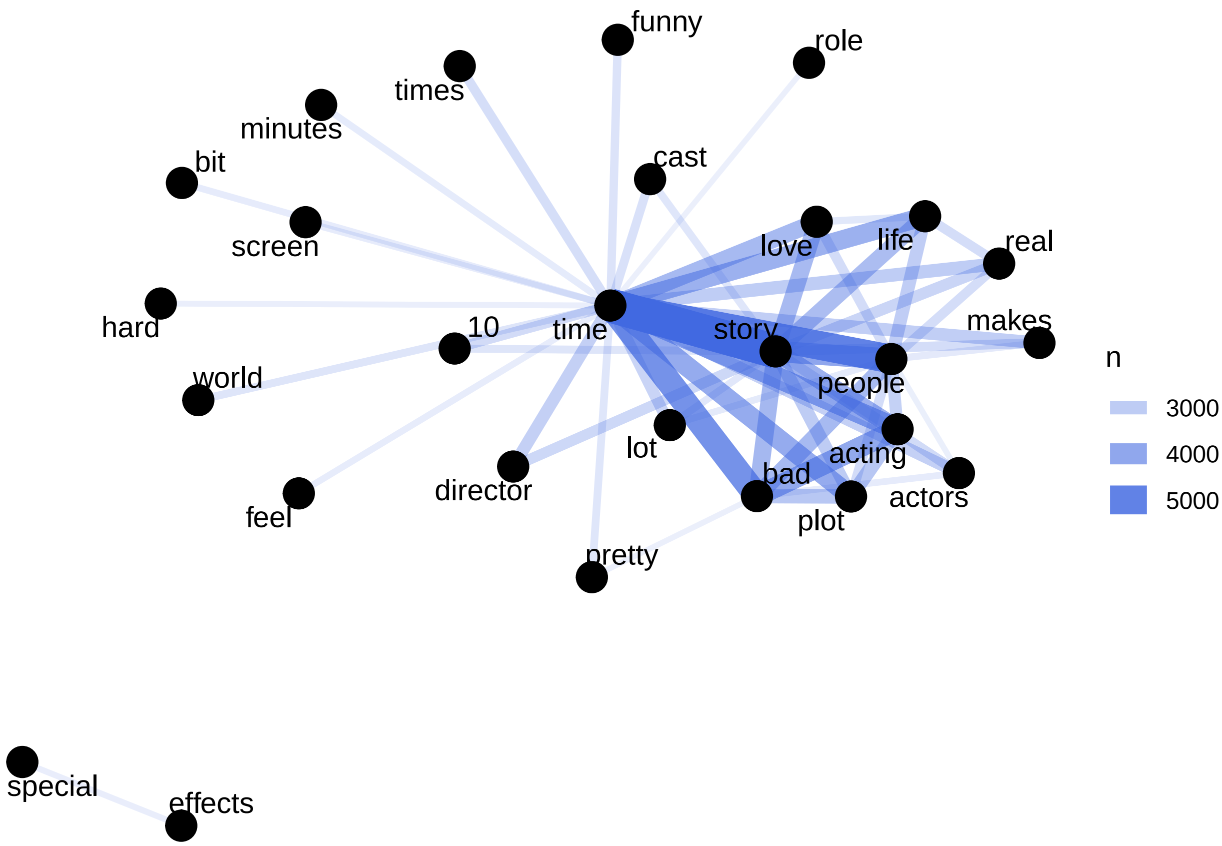
```
imdb_title <- imdb %>%
  unnest_tokens(word, review) %>%
  anti_join(stop_words) %>%
  count(docs, word, sort = TRUE)

imdb_total <-imdb_title %>%
  group_by(docs) %>%
  summarize(total = sum(n))

imdb_title <- left_join(imdb_title,imdb_total)

imdb_title_pair <- imdb_title %>%
  pairwise_count(word, docs, sort = TRUE, upper = FALSE)

imdb_title_pair %>%
  filter(n >= 2000) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = n, edge_width = n), edge_colour = "royalblue") +
  geom_node_point(size = 5) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```

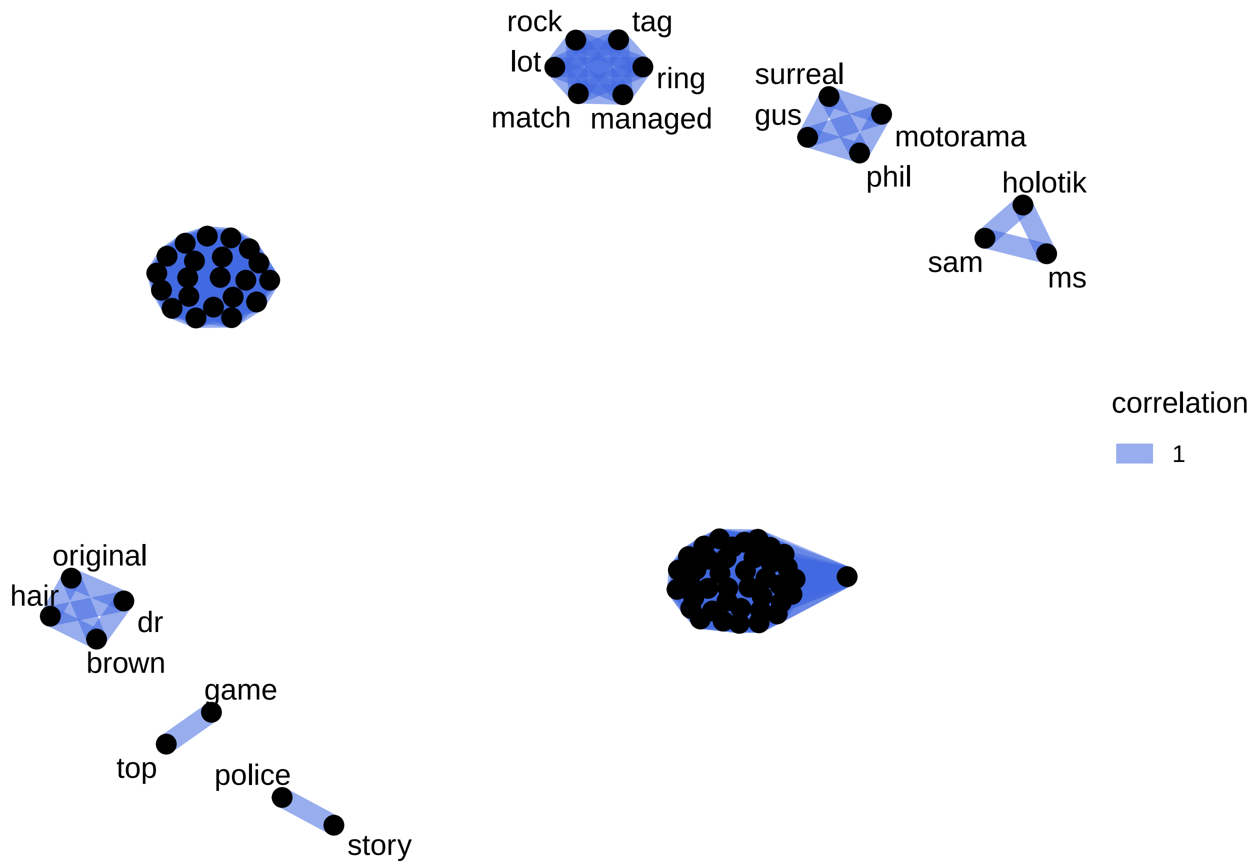


We use pair of words in the imdb dataset that occur together most often in the fields, in this graph we can see that the words are organized in to a large family, and in the middle of the graph, “time” has strong connection with the words around it.

```
imdb_title_cor <- imdb_title %>%
  group_by(docs) %>%
  filter(n() >= 500) %>%
  pairwise_cor(word, docs, sort = TRUE, upper = FALSE)

imdb_cor <- imdb_title_cor[1:1000,]

set.seed(2022)
imdb_cor %>%
  filter(correlation > .9) %>%
  graph_from_data_frame() %>%
  ggraph(layout = "fr") +
  geom_edge_link(aes(edge_alpha = correlation, edge_width = correlation), edge_colour = "royalblue") +
  geom_node_point(size = 3) +
  geom_node_text(aes(label = name), repel = TRUE,
                 point.padding = unit(0.2, "lines")) +
  theme_void()
```



This network shows the correlation of the keywords which occur more often together than with other keywords.