Topic_Modeling

Group 10: Bill Gao, Jiun Lee, Priam Vyas, Danya Zhang

2022-11-16

##Import Data from kaggle

```
##Import Data
imdb <- read_csv ("/Users/jiunlee/MSSP22/Fidelity/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,tf-idf

```
##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
stop <- StripAttr(stop)
##Restore tokens Data set
check <- token
```

```r
##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
```

##Create Latent Dirichelet Allocation model

```r
library(topicmodels)

##Convert sample token tibble to 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 #A LDA_VEM topic model with 6 topics.
```

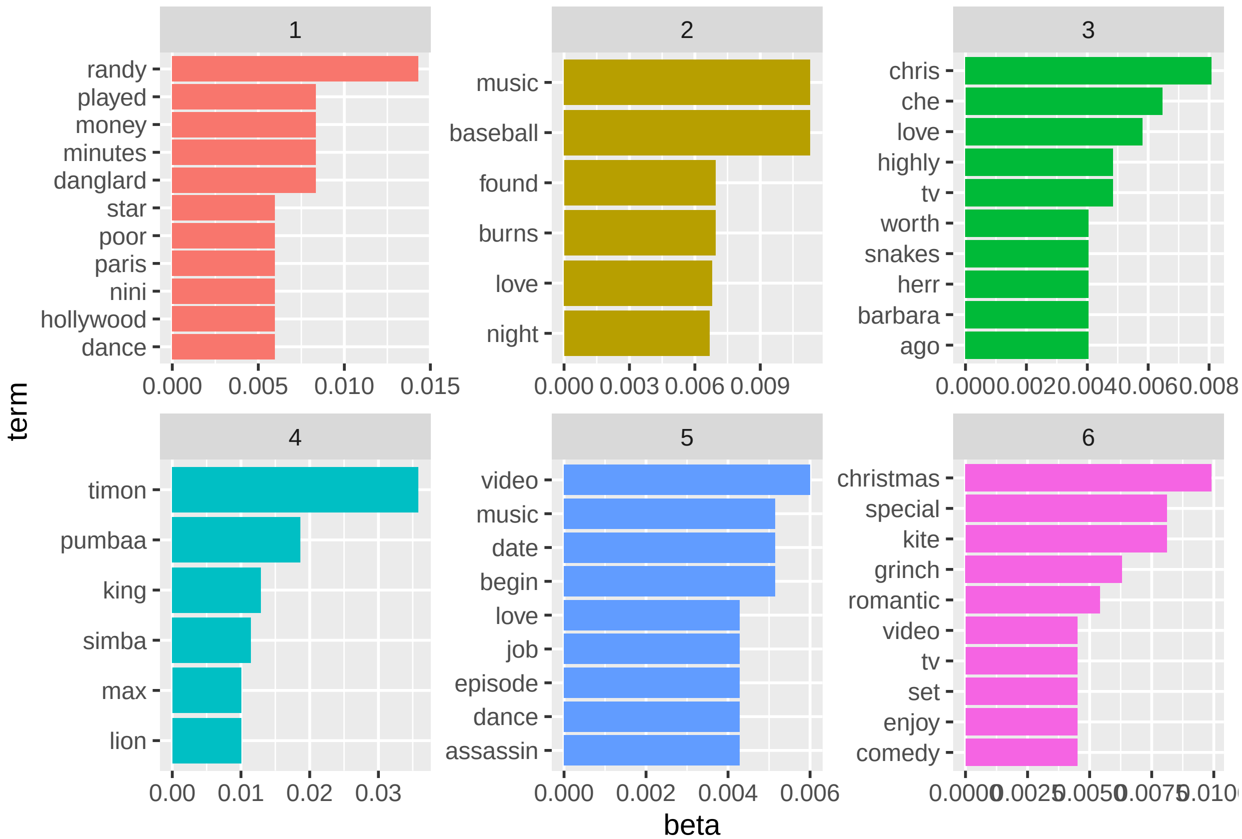
```
## A LDA_VEM topic model with 6 topics.
```

```r
imdb_topics <- tidy(imdb_lda, matrix = "beta")
```

##Now we create our plots

```r
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 above shows the top words for each topic in our LDA.
##We've split up our LDA into 6 genres, which represents the number of topics we have.
```

##Now let's look at Document-topic probabilities

```
##Gamma: per-document-per-topic probabilities
imdb_documents <- tidy(imdb_lda, matrix = "gamma")

#Most common words in document
tidy(clean_token_dmat) %>%
  filter(document == 6) %>%
  arrange(desc(count))
```

```
## # A tibble: 0 x 3
```

```
## # ... with 3 variables: document <chr>, term <chr>, count <dbl>
## # i Use `colnames()` to see all variable names
```

```
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
## # ... with 5,276 more rows
## # i Use `print(n = ...)` to see more rows
```

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

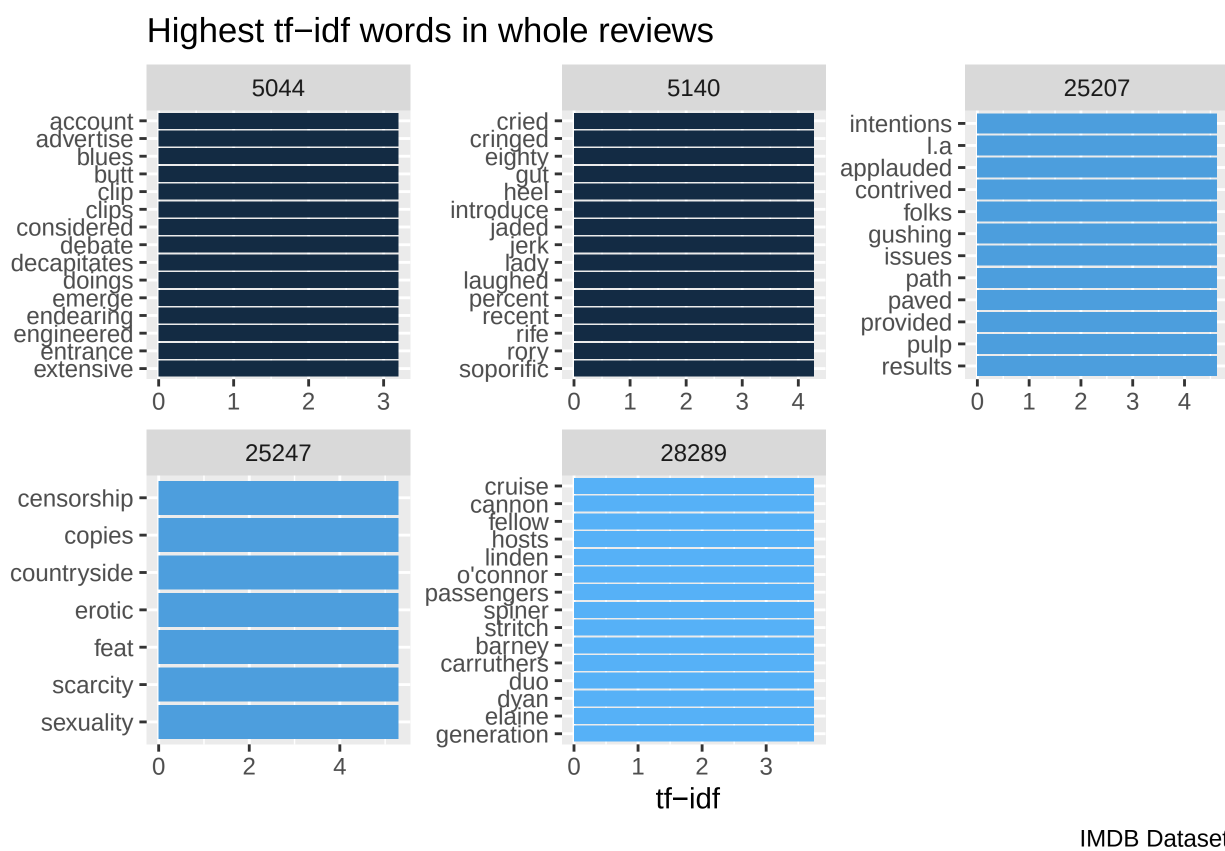
##Term Frequency - Inverse Document Frequency

```
##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.
review_tf_idf<- review_tf_idf %>%
  arrange(desc(tf_idf))

##It looks like the high tf-idf's tf are mostly 1.
##For words that tf=1, it means those words are only contained in one review, and the tf-idf algorithm

##So, let's remove all tf = 1.
tf_1 <- which(review_tf_idf$tf==1)
tf_idf_high <- review_tf_idf %>% slice(tf_1) %>% select(-count,-tf,-idf) #remove column 'count','tf','i
rm(review_tf_idf)
##Now we only have 1993 observations

##Review_index numbers in tf_idf_high
index <- unique(tf_idf_high$review_index)
```

##tf-idf plot

```
##Let's make the plots with only 6 review_index.
tf_idf_high %>%
  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 whole 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.
```