Topic Modelling In Class

Vajinder

2024-11-01

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
##
## Loading required package: RColorBrewer
```

Including Plots

You can also embed plots, for example:

```
movie <- read.csv("movie_plots.csv")
movies_genre <- read.csv("movie_plots_with_genres.csv")
```

```
plots <- movies_genre |> unnest_tokens(word,Plot)
plot_word_counts <- plots |>
  anti_join(stop_words) |>
  count(Movie.Name, word, sort=TRUE)
```

```
## Joining with `by = join_by(word)`
```

```
data("freq_first_names")

first_names <- tolower(freq_first_names$Name)
plot_word_counts <- plot_word_counts |>
  filter(!(word %in% first_names))
```

```
dtm <- plot_word_counts |>
  cast_dtm(Movie.Name, word, n)

dtm
```

```
## <<DocumentTermMatrix (documents: 1063, terms: 13394)>>
## Non-/sparse entries: 44143/14193679
## Sparsity           : 100%
## Maximal term length: 17
## Weighting          : term frequency (tf)
```

```
lda <- LDA(dtm, k = 20, control = list(seed = 1234))

top_terms <- terms(lda, 10)
```

LDA with 30 topics plot_lda <- LDA(plot_dtm, k = 30, control = list(seed = 1066)) beta - words per topics #retrieving gammas gamma - topics per documents Notes taken during the class : (plot_ga„a <- tidy(plots_lda, matrix = "gamma") cluster (how similar movies are based on the genre) multinomial extension of beta distribution - drichlet distribution)

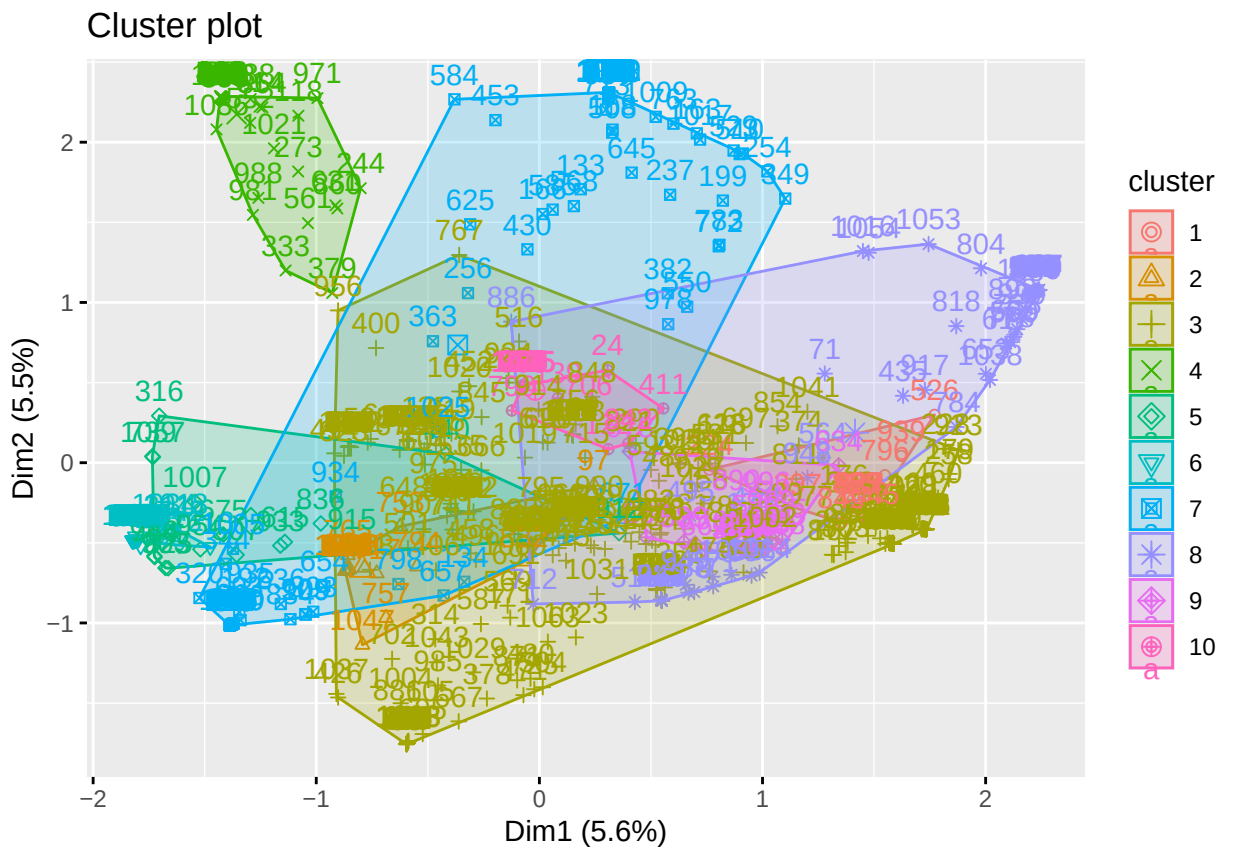
```
betas <- tidy(lda, matrix = "beta")
gammas <- tidy(lda, matrix = "gamma")
gammas <- gammas |>  pivot_wider(names_from = topic, values_from = gamma)
```

Take highest gamma for each movie to get the top movie by topic

```
top_movies <- gammas |>
  pivot_longer(cols = `1`:`20`, names_to = "topic", values_to = "gamma") |>
  group_by(document) |>
  slice_max(gamma, n = 1) |>
  ungroup() |>
  select(document, topic, gamma)
```

But do get a more detailed look, we need to cluster the movies into 10 clusters by topic

```
library(dplyr)
cluster <- kmeans(gammas |>  select(-document),10)
fviz_cluster(cluster, data = gammas |> select(-document))
```




```
movies_genre <- movies_genre |>
  distinct(Movie.Name, .keep_all = TRUE)
clusters <- cluster[["cluster"]]
cluster$cluster <- clusters
movies_genre$cluster <- clusters
```

Creating clusters

```
gammas <- gammas |>
  left_join(movies_genre |> select(Movie.Name, cluster), by = c("document" = "Movie.Name"))
cluster_1 <- gammas |> filter(cluster == 1)
cluster_2 <- gammas |> filter(cluster == 2)
cluster_3 <- gammas |> filter(cluster == 3)
cluster_4 <- gammas |> filter(cluster == 4)
cluster_5 <- gammas |> filter(cluster == 5)
cluster_6 <- gammas |> filter(cluster == 6)
cluster_7 <- gammas |> filter(cluster == 7)
cluster_8 <- gammas |> filter(cluster == 8)
cluster_9 <- gammas |> filter(cluster == 9)
cluster_10 <- gammas |> filter(cluster == 10)
```

Find which topic is most associated with each cluster

```r
col_avg <- function(df) {
  selected_columns <- df |>
    select(`1`:`20`)

  column_averages <- colMeans(selected_columns, na.rm = TRUE)

  return(column_averages)
}
averages_cluster_5 <- col_avg(cluster_5)
print(averages_cluster_5)
```

```
##            1            2            3            4            5            6
## 0.0003297212 0.0588545637 0.0585353701 0.0003297212 0.0774774213 0.0589245697
##            7            8            9           10           11           12
## 0.0003297212 0.0003297212 0.0003297212 0.0583429378 0.1173619576 0.1174100291
##           13           14           15           16           17           18
## 0.0590760699 0.0003297212 0.1082028475 0.0131016094 0.1584518308 0.0173634814
##           19           20
## 0.0003297212 0.0945892633
```

All of these probablities are pretty small numbers except 3,6, 7, 8 and 20 indicating cluster 5 is most associated with these three topics.

Make word clouds

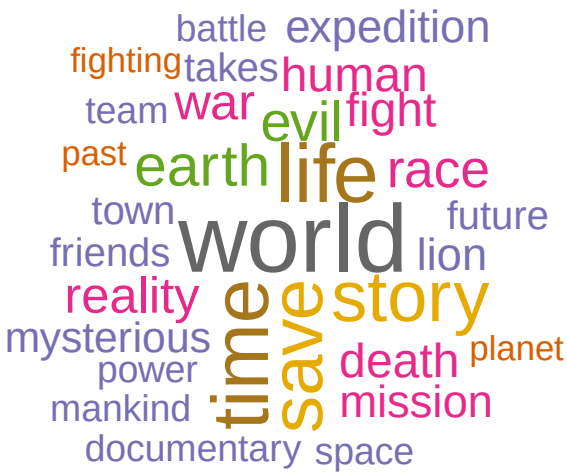
```r
wordcloud <- function(topic_number) {
  top_words <- betas |>
    filter(topic == topic_number) |>
    top_n(30, beta) |>
    arrange(desc(beta))

  wordcloud::wordcloud(words = top_words$term,
            freq = top_words$beta,
            min.freq = 0,
            scale = c(3, 0.5),
            random.order = FALSE,
            colors = brewer.pal(8, "Dark2"))
}
top_terms <- terms(lda, 10)
avg_9 <- col_avg(cluster_9)
print(avg_9)
```

```
##          1          2          3          4          5          6          7
## 0.03901539 0.09060719 0.03455287 0.05906301 0.04755690 0.04785692 0.05837207
##          8          9         10         11         12         13         14
## 0.09714407 0.01224276 0.01980186 0.05773200 0.09472530 0.03601815 0.02820494
##         15         16         17         18         19         20
## 0.04589134 0.03885582 0.03290827 0.08492292 0.04322620 0.03130204
```

Lets make some fun word clouds

```
wordcloud(1)
```


```
wordcloud(2)
```


battle
expedition
fighting
takes
human
team
war
evil
fight
past
earth
life
race
town
world
future
friends
lion
reality
time
save
story
mysterious
death
planet
power
mankind
mission
documentary
space

```
wordcloud(3)
```

I got help from Jon and jordan for the probabilities and cloud part. I hadn't added cloud in the original file.