Processing Topic Modelling YouTube Channels

Rodrigo Esteves de Lima-Lopes
University of Campinas
rll307@unicamp.br

Contents

1 Introduction

This script was developed for the analysis of Portuguese. I hope it helps colleagues in the LC area and popularize the use of R. It is part of our research project developed with CNPQ. Please drop me a line if you have any doubts or need any help.

2 Purpose of this repository

This git brings the scripts for my article:

- Lima-Lopes R.E. (forthcoming). Beyond the Binary: Trans Women's Video Activism on YouTube. Accepted for publication at *Digital Scholarship in the Humanities*.

This script is specifically about **data processing**.

3 Packages

For data processing we are going to need some packages, each has a different function

- abjutils: diacritic removal in Brazilian Portuguese
- tm, tidytext, tidyverse, magrittr: data manipulation and cleaning
- ggridges: graph plotting

- formattable: table formating

```
library(abjutils)
library(tidytext)
library(tidyverse)
library(magrittr)
library(stm)
library(tm)
library(ggridges)
library(formattable)
```

4 Topic modelling

4.1 Preparing a general data frame

If you are processing more than one channel like I did, it will be necessary first to merge all in a single data frame. I am assuming your corpus has 140 videos of each channel, as mine. So the frist step is to identify each video before the dfs are merged. It will help a lot of confusion.

```
# we use a similar code for each channel (from 1 to 4)
channel1 <- rep("channel1", 140)
df.channel1 <-cbind(df.channel1,channel1)
colnames(df.channel1)[colnames(df.channel1)=="channel1"] <- "Channel"
df.final.all <-rbind(df.channel1,df.channel2,df.channel3,df.channel4)
```

The next step was to create a data frame with the words. Although it is not necessary to the actual topic modelling analysis, it is nice for us to have a look at our corpus' lexis.

```
df.final.all  %>%
  unnest_tokens(word, caption) %>%
  # filtering stopwords
  filter(!word %in% sw_pt_tm)%>%
  # Counting
  count(word) %>%
  arrange(desc(n)) %>%
  formattable()
```

4.2 Processing

Our first step will be preparing a stopwords list to apply to our corpus. Please notice that some words, like proper names, names of cities or locations that can identify the YouTubers may pop up during the analysis. In this case, the best is to add those words to the list and rerun the script.

In this paper, I got a wordlist from `tm` package. Note that there is a command to clean the diacritic symbols

```
sw_pt_tm <- tm::stopwords("pt")
sw_pt_tm <- rm_accent(sw_pt_tm)
```

Now let us process the corpus itself

```
proc <- stm::textProcessor(df.final.all$caption, metadata = df.final.all,
                           language = "portuguese",
                           customstopwords = sw_pt_tm)
out <- stm::prepDocuments(proc$documents, proc$vocab, proc$meta,
                          lower.thresh = 10)

storage <- stm::searchK(out$documents, out$vocab, K = c(3:15),
                          data = out$meta)

fit <- stm::stm(
  documents = out$documents, vocab = out$vocab, data = out$meta,  K = 4,
  max.em.its = 75, init.type = "Spectral", verbose = FALSE
)
```

This will give us the four topics, like in the final paper. Depending on the size of your corpus, the `k=n` might change. If you do so, the number of topics is going to change. My advice is keep doing this until the optimal number of topics is subjectively reached.

In order to check the results and make some ajustments, the following block of code might help:

```
## More common words in each topic
stm::labelTopics(fit)
#plot(fit, "summary")
plot(fit$theta, type = "p", col="blue")
head(fit$theta)
view(fit$theta)
```

After we checked everything, it is time to name the topics, four in our case

```
Topic.Names <- c("Relationships",
                 "Gender",
                 "Beauty",
                 "Transition")
```

Now, we are going to extract the best possibility of a video fit into a topic:

```
prob <- apply(fit$theta, 1, max)
```

Now we are going to associate such probality to a topic name

```
Videos.Topic <- Topic.Names[apply(fit$theta, 1, which.max)]
```

Now it is time to join all these pieces of information in a data frame

```
df_topics <- df.final.all %>%
  mutate(best_prob = prob,
         topic = Videos.Topic)
```