Data cleaning and 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 cleaning**.

3 Introduction

The data we have scraped previously is a set of YouTube subtitles. They should contain a number of information which is not necessary to us, such as time stamps, colour, in/out codes etc. Deleting such information from such files is mandatory. In this tutorial we will see two possible approaches, one using a website and another using R and Python. I particularly recommend the second, since we do not get dependent from any external tool, although I think that will mostly depend on how keen on coding you are.

Logically, only the latter will produce an R script. Please note that stopwords, undisired characters, punctuation and other pre-processing will be addressed at the `Processing Topic Modeling and plotting` script.

4 Procedures

4.1 Using a external tool

There are two possible external tools that might help us to clean our data:

- Subtitle Tools
- Happyscribe

I would particularly recommend the first. The process is quite simple, you upload the files you want to convert, the website converts then to text and you might download them all at once. Please, remember: unmark the option *Empty line between cues* to avoid unecessary space between two lines.

4.2 Using R and Python

For this task, we will need to run the following packages:

- abjutils: diacritic removal in Brazilian Portuguese
- tm, tidytext, tidyverse, magrittr: data manipulation and cleaning
- ggridges: graph plotting
- formattable: table formating
- reticulate: interface between R and Python

Please, note that <`your Python instalation`> refers to your Python executable path.

```
library(abjutils)
library(tidytext)
library(reticulate)
reticulate::use_python("<your Python instalation>", required = TRUE)
library(tidyverse)
library(magrittr)
library(stm)
library(tm)
library(ggridges)
library(formattable)
options(scipen = 999)
```

Our next step is to inform the system where our files are. This will help us to import the data into R more easily. We also need to impot a Python script into R, so the tool for cleaning the data (which is in Python) might run inside our environment. I have already written this small script in Python.

```
My.Caption.Files <- dir(Captions.Folder, pattern = '*.vtt', full.names = TRUE)
source_python("caption_to_vector.py") # Importing Python script
```

Our next step will be to write a function to delete unecesssary information

```
Cleaning.Captions <- function(file){ #Defines the variable
  # Imports Python script into the function
  caption_raw <- caption_to_vector(file)
  # Tells us the number of lines
```

```r
  n <- length(caption_raw) #
  # Removes \n from all lines but the last
  caption <- c(stringr::str_remove_all(caption_raw[-n], "[\n].*"),
               caption_raw[n])
  # Removes duplicated lines
  caption <- unique(caption)
  # Removes diacritical marks (remeber that Portuguese has a number of them)
  caption <- rm_accent(caption)
  # Merges them
  caption <- paste0(caption, collapse = "\n")
  caption
}
```

This function will be in our memory. Now we need a new function that extracts metadata from the names of the files.

```r
Metadata.Extract <- function(file, folder = Captions.Folder,
                             fields = basic.fields){
  mat <- str_split(file, "&{3}", simplify = TRUE)
  mat[1,1] <- mat[1,1] %>% str_remove_all(Captions.Folder) %>% str_remove_all("/")
  cols <- fields[1:ncol(mat)]
  colnames(mat) <- cols
  as.tibble(mat)
}
```

This function will be in our memory. Now let us write our last function, which will encapsulate the other two.

```r
caption_to_df <- function(file, ...){
  caption <- Cleaning.Captions(file)
  meta <- Metadata.Extract(file, ...)
  meta <- meta %>% mutate(caption = caption)
  meta
}
```

Finally, it is time to clean our files and make them a single dataframe

```r
df.final <- My.Caption.Files %>%
  map_df(caption_to_df) %>%
  # converte the class of some columns
  mutate(upload_date = lubridate::ymd(upload_date)) %>%
  mutate_at(vars(duration:dislike_count), as.numeric)
```

Please, note that it will be necessary to repeat this code for each channel.