Text Mining and Sentiment Analysis

Prof. Annamaria Bianchi
A.Y. 2024/2025

Lecture 23

27 May 2025

UNIVERSITÀ DEGLI STUDI DI BERGAMO | Dipartimento di Scienze Economiche

Outline

Case study

Construct a function for scraping Amazon reviews

Objective: scrape the reviews about a specific product: the book *The Theory That Would Not Die*

https://www.amazon.co.uk/product-reviews/B0050QB3EQ/

We are interested in all reviews but, then, we will analyse only english-written ones.

Reviews are organized on multiple pages and in this example we can find both “Reviews from UK” and from “Other countries”. The page is dynamic. So we will use RSelenium.

Notice also that, if you look at only at the first page, you will see Reviews from UK only. However, when you try to scrape multiple pages, the function returns an error. This is because the characteristics of the element in the html file are different.

We need first to obtain the webpages using Rselenium and then scrape the pages to obtain the review title, text and stars.

Construct a function for scraping Amazon reviews

Obtain the webpages using Rselenium

```
> rD <- rsDriver(browser = "firefox",
+ verbose = FALSE,
+ port = free_port(),
+ chromever = NULL)
> remDr <- rD[["client"]]

> url <- "https://www.amazon.co.uk/product-reviews/B0050QB3EQ/"
> remDr$navigate(url)
```

Construct a function for scraping Amazon reviews

```
> # identify email field
> field <- remDr$findElement(using = "css", "#ap_email")
> email = "YOUR EMAIL"
> field$sendKeysToElement(list(email))

> # click on the tab
> click <- remDr$findElement(using = "css", "#continue")
> click$clickElement()

> # identify pwd field
> field <- remDr$findElement(using = "css", "#ap_password")
> pwd = "YOUR PASSWORD"
> field$sendKeysToElement(list(pwd))

> # click on the tab
> click <- remDr$findElement(using = "css", "#signInSubmit")
> click$clickElement()
```

Construct a function for scraping Amazon reviews

Next we create a folder in our directory (Amazon) and save the first page Amazon1.html. We click the Next button and save the other pages using a for loop.

```
> folder <- "Amazon/"
> dir.create(folder)
> names <- str_c("Amazon", seq(1, 10, 1), ".html")
> names
[1] "Amazon1.html" "Amazon2.html" "Amazon3.html" "Amazon4.html"
[5] "Amazon5.html" "Amazon6.html" "Amazon7.html" "Amazon8.html"
[9] "Amazon9.html" "Amazon10.html"

> output <- remDr$getPageSource(header = TRUE)
> write(output[[1]], file = str_c(folder,
+ names[1]))
```

Construct a function for scraping Amazon reviews

```
> for (i in 2:length(names)) {
+ if (!file.exists(str_c(folder, names[i]))) {
+ # identify and click the next button
+ button <- remDr$findElement(using = "css", value = ".a-last")
+ button$clickElement()
+ Sys.sleep(3)
+ output <- remDr$getPageSource(header = TRUE)
+ write(output[[1]], file = str_c(folder, + names[i]))
+ Sys.sleep(5)
+ }
+ }

> # close the connection
> remDr$close()
> rD$server$stop() [1] TRUE
```

Construct a function for scraping Amazon reviews

Next, we construct a scraping function. The function will take in input the product id and the page and it will return a tibble with the review title, text and stars.
After parsing the html pages, the function extracts the information required. In real applications, CSS selectors may be not available or difficult to use. In that case we directly identify the class we want. You can try other options: there is not an unique way to scrape!

Construct a function for scraping Amazon reviews

```
> amazon_reviews <- function(id, page){
+
+   file <- str_c(folder, "Amazon", page, ".html")
+   html <- read_html(file, encoding = "utf-8")
+ # Review title (UK and not-UK)
+ title=html |>
+   html_elements("[class='a-size-base a-link-normal review-title a-color-base review-title-content a-text-bold']") |>
+   html_elements("span:nth-child(3)")|>
+   html_text2()
+
+ title=title |> c(html |>
+                  html_elements("[class = 'a-size-base review-title a-color-base review-title-content a-text-bold']")|>
+                  html_text(trim = T))
+
+
+ # Review text (the same for UK and not-UK)
+ text=html |>
+   html_elements("[class='a-size-base review-text review-text-content']") |>
+   html_text(trim = T)
+
+ # Review stars (UK and not-UK)
+ star=html |>
+   html_elements("[data-hook='review-star-rating']") |>
+   html_text2()
+
+ star=star |> c(
+   html |>
+     html_elements("[data-hook='cmps-review-star-rating']") |>
+     html_text2())
+
+ t <- tibble(title,
+        text,
+        star,
+        page = page)
+ return(t)
+
+ }
```

Construct a function for scraping Amazon reviews

Let's try our function! We use the `map_df` function from the `purrr` package in order to iterate the task over multiple pages.

```
> id="B0050QB3EQ"
> page=1:10
> library(purrr)

> data=map_df(page,~amazon_reviews(id, page = .))

> View(data)
```

We also add a `doc_id` and we save the results

```
> data = data |>
+ mutate(id = seq_along(text))

> save("data", file = "data.rds")
```

Data cleaning and preprocessing – detect language

When working with real data, additional data cleaning steps are necessary. As discussed before, we scraped reviews from UK but also from other countries. Thus, some of them can be in an other language. Here, we want to analyze only reviews written in english, but in other applications we may consider the possibility of translating the text or direclty analysing also the review in other languages (with different dictionaries). There are several options in order to detect the language. Here, we use the package `cld2`. This package is built on the Google's Compact Language Detector (in C++). It implements a Naive Bayes clissifier based on n-gram profiles. It is trained on HTML pages, so you can use it in order to find the language of webpages but it also with texts (as in our case). If the language cannot be determined it returns `NA`.

Let's apply the `detect_language` function to the title and to the text respectively.

Data cleaning and preprocessing – detect language

```
> library(cld2)
> data$title_lang=detect_language(data$title)
> data$text_lang=detect_language(data$text)
> table(Text=data$text_lang,Title=data$title_lang,useNA="always")
```

```
      Title
Text   de en es <NA>
  de    1  0  0    0
  en    0 76  0   21
  es    0  0  1    0
  it    0  0  0    1
  <NA>  0  0  0    0
```

```
> data=data |>
+ filter (text_lang=="en")
```

Data cleaning and preprocessing – score

The second step is to extract a numeric score form the `stars` string.

```
> # Extract the star
> data = data |>
+ mutate(score=as.numeric(str_sub(star,1,1)))

> # analyse the score
> data |>
+ summarise(mean(score), min(score), median(score), max(score))

> data |>
count(score) |>
+ mutate(p=round(n/sum(n),2))
```