

Introduction to Applied Data Science

Bas Machielsen
Utrecht University

2023-01-11

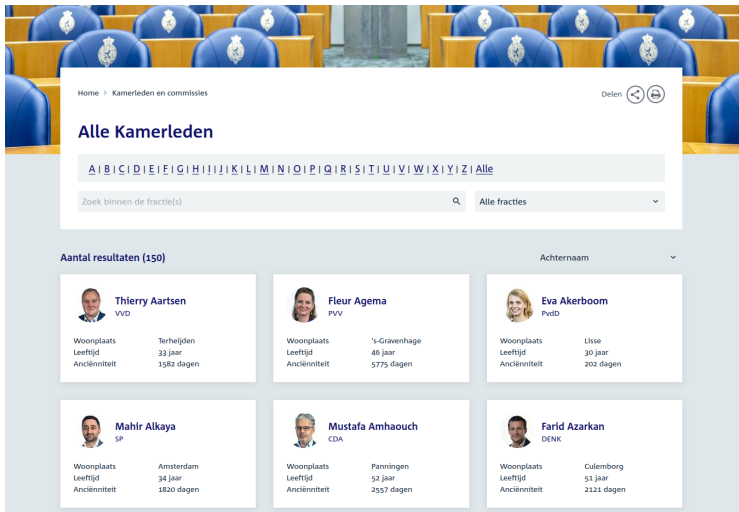
Lecture 4: Web Scraping

Introduction

- Many of you might have heard of ChatGPT:
ChatGPT (Generative Pre-trained Transformer) is a chatbot launched by OpenAI in November 2022. It is built on top of OpenAI's GPT-3 family of large language models, and is fine-tuned (an approach to transfer learning) with both supervised and reinforcement learning techniques.
- What does it do?
While the core function of a chatbot is to mimic a human conversationalist, journalists have also noted ChatGPT's versatility and improvisation skills, including its ability to write and debug computer programs; to compose music, teleplays, fairy tales, and student essays; to answer test questions (sometimes, depending on the test, at a level above the average human test-taker); to write poetry and song lyrics; to emulate a Linux system; to simulate an entire chat room; to play games like tic-tac-toe; and to simulate an ATM.

Suppose we are interested in this data..

- Source



Home > Kamerleden en commissies

Delen

Alle Kamerleden

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z I Alle

Zoek binnen de fractie(s) Alle fracties

Aantal resultaten (150) Achternaam

Naam	Partij	Woonplaats	Leeftijd	anciënniteit
Thierry Aartsen	VVD	Terheijden	33 jaar	1582 dagen
Fleur Agema	PVV	's-Gravenhage	46 jaar	5775 dagen
Eva Akerboom	PvdD	Lisse	30 jaar	202 dagen
Mahir Alkaya	SP	Amsterdam	34 jaar	1820 dagen
Mustafa Amhaouch	CDA	Panningen	52 jaar	2557 dagen
Farid Azarkan	DENK	Culemborg	51 jaar	2121 dagen

What happens if we ask ChatGPT?



Can you write a piece of R code that scrapes the Names, Woonplaats, Ancienniteit, and Leeftijd from https://www.tweedekamer.nl/kamerleden_en_commissies/alle_kamerleden?



Sure, here is an example of how you could scrape the information you've specified from the website you've provided using the `rvest` package in R:



This is the output

```
# Scrape the webpage
url <- "https://www.tweedekamer.nl/kamerleden_en_commissies/alle_kamerleden"
webpage <- read_html(url)

# Extract the table containing the data
table <- html_nodes(webpage, "table") %>% .[[1]]

# Extract the rows of the table
rows <- html_nodes(table, "tr")

# Extract the data from each row
data <- lapply(rows, function(x) {
  name <- html_text(html_nodes(x, ".member-name"))
  woonplaats <- html_text(html_nodes(x, ".member-place-of-residence"))
  ancienniteit <- html_text(html_nodes(x, ".member-seniority"))
  leeftijd <- html_text(html_nodes(x, ".member-age"))
  data.frame(Name=name, Woonplaats=woonplaats, Ancienniteit=ancienniteit, Leeftijd=leeftijd)
})

# Bind the data together
data <- bind_rows(data)
```

- Let us briefly take a look at the intended logic of the code

This is the output

```
data
```

```
## [1] Name          Woonplaats  Ancienniteit Leeftijd  
## <0 rows> (or 0-length row.names)
```

- The code doesn't work!
- But it's a pretty good start
- Let us first correct the code, and then see step by step what happens

This is the corrected output

- This is the corrected code

```
library(rvest, quietly = TRUE)

# Scrape the webpage
url <- "https://www.tweedekamer.nl/kamerleden_en_commissies/alle_kamerleden"
webpage <- read_html(url)

# Extract the table containing the data pieces
table <- html_nodes(webpage, "div.m-card__content")

# Extract the data from each member
data <- lapply(table, function(x) {
  name <- html_text(html_nodes(x, "a.m-member__name"))
  party <- html_text(html_nodes(x, "span.m-member__subtitle"))
  wal <- html_text(html_nodes(x, "td"))
  if(length(wal) == 2) {
    woonplaats <- NA
    leeftijd <- wal[1]
    ancienniteit <- wal[2]
  } else {
    woonplaats <- wal[1]
    leeftijd <- wal[2]
    ancienniteit <- wal[3]
  }

  data.frame(Name=name, Partij=party, Woonplaats=woonplaats,
             Ancienniteit=ancienniteit, Leeftijd=leeftijd)
})

# Bind the data together
data <- bind_rows(data)
```


Correct Code

- In comparison to the incorrect code offered to us by ChatGPT, we changed a couple of things:
- We changed the table object from:

```
table <- html_nodes(webpage, "table") %>% .[[1]]
```

to:

```
table <- html_nodes(webpage, "div.m-card__content")
```

- And we changed the function to pick up Woonplaats, Ancienniteit and Leeftijd more carefully.

Results

- This code gives you the following dataset:

```
data %>% head(10)
```

##	Name	Partij	Woonplaats	Ancienniteit	Leeftijd
## 1	Thierry Aartsen	VVD	Terheijden	1582 dagen	33 jaar
## 2	Fleur Agema	PVV	's-Gravenhage	5775 dagen	46 jaar
## 3	Eva Akerboom	PvdD	Lisse	202 dagen	30 jaar
## 4	Mahir Alkaya	SP	Amsterdam	1820 dagen	34 jaar
## 5	Mustafa Amhaouch	CDA	Panningen	2557 dagen	52 jaar
## 6	Farid Azarkan	DENK	Culemborg	2121 dagen	51 jaar
## 7	Stephan van Baarle	DENK	Rotterdam	652 dagen	31 jaar
## 8	Thierry Baudet	FVD	Amsterdam	2121 dagen	39 jaar
## 9	Bente Becker	VVD	Wassenaar	1897 dagen	37 jaar
## 10	Sandra Beckerman	SP	Groningen	2121 dagen	39 jaar

- In this lecture, we'll learn how to scrape data from websites using the `rvest` package.
 - And we can always use ChatGPT to give us a starting point

Demonstration

- Let's see if we can do something nice with this newly obtained dataset.

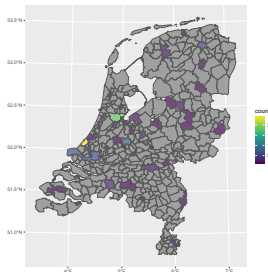
```
library(tmap, quietly = TRUE); library(sf, quietly = TRUE)
```

```
## Linking to GEOS 3.11.1, GDAL 3.6.2, PROJ 9.1.1; sf_use_s2() is TRUE
```

```
data("NLD_muni")  
freq <- data %>% group_by(Woonplaats) %>% summarize(count = n())  
NLD_muni <- NLD_muni %>% left_join(freq, by = c("name" = "Woonplaats"))
```

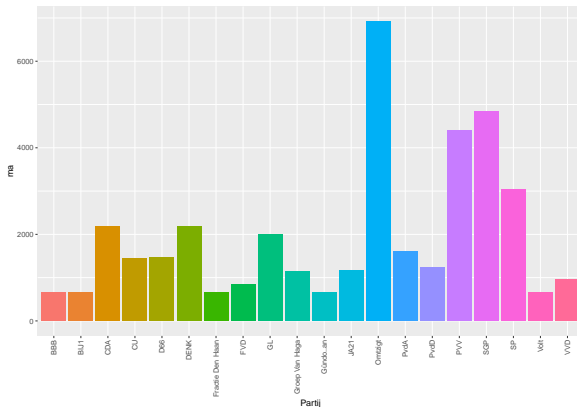
```
## old-style crs object detected; please recreate object with a recent sf::st_crs()
```

```
NLD_muni %>% ggplot() + geom_sf(alpha=0.7,aes(fill=count)) + scale_fill_continuous(type='viridis')
```



Demonstration

```
data %>%  
  group_by(Partij) %>%  
  summarize(ma = mean(parse_number(Ancienniteit))) %>%  
  ggplot(aes(x = Partij, y = ma)) + geom_col(aes(fill = Partij)) +  
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1),  
        legend.position="none")
```



Introduction to Web Scraping

Introduction to Web-Scraping

- To do web scraping, you need to know some basic HTML.
- HTML stands for “Hyper Text Markup Language”. HTML page consists of series of elements which browsers use to interpret how to display the content. HTML tags are names of the elements surrounded by angle brackets like so: `<tagname>` content goes here... `</tagname>`.
- Most HTML tags come in pairs and consist of opening and a closing tag, known as start tag and end tag, where the end tag name is preceded by the forward slash /.

HTML Structure

- Below is a visualization of a simple HTML page structure:

```
<html>
  <head>
    <title> Page title </title>
  </head>
  <body>
    <h1> Page title </h1>
    <p> This is a paragraph. </p>
    <p> This is another paragraph </p>
  </body>
</html>
```

- Pretty much all webpages look like this
 - But there are exceptions

HTML Structure

- It is possible to define HTML attributes inside HTML tags.
- Attributes provide additional information about HTML elements, such as hyperlinks for text, and width and height for images.
- Attributes are always defined in the start tag and come in name="value" pairs, like so: `This is a link`
 - Here href is the attribute and it's value is "https://www.example.com"

CSS Selectors

- CSS is a language that describes how HTML elements should be displayed.
- One of the ways to define useful shortcuts for selecting HTML elements to style is CSS selectors.
- CSS selectors represent patterns for locating HTML elements.
- This is what we use to find particular attributes in a HTML page, and extract them or their text into R

Example

- If we focus once again the the Dutch Lower House website, right-click with your mouse and then select `Inspect` (Q)
- A subscreen will pop-up with the html code underlying the website
- Your job is then to find the elements you want to scrape, and find the pattern.

Example

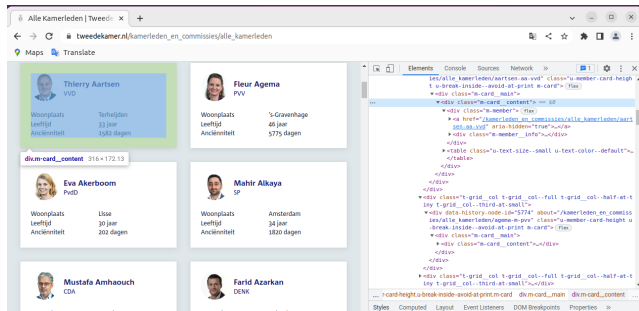
- Let us revisit some excerpts from the (correct) code from above:

```
# Extract the table containing the data pieces
table <- html_nodes(webpage, "div.m-card__content")

# Extract the data from each member
data <- lapply(table, function(x) {
  name <- html_text(html_nodes(x, "a.m-member__name"))
  party <- html_text(html_nodes(x, "span.m-member__subtitle"))
  wal <- html_text(html_nodes(x, "td")) } )
```

Example

- By inspecting the website, we can see that each politician is featured inside a `<div>` with the attribute `m-card__content`:



Example

The screenshot shows a web browser displaying a grid of member cards on the website `tweedekamer.nl/kamerleden_en_commissies/alle_kamerleden`. The grid contains six cards for different members, each with a profile picture, name, party affiliation, and a table of statistics.

Member	Party	Woonplaats	Leeftijd	anciënniteit
Thierry Aartsen	VVD	Terheijden	33 jaar	1582 dagen
Fleur Agema	PVV	's-Gravenhage	46 jaar	5775 dagen
Eva Akerboom	PvdD	Lisse	30 jaar	202 dagen
SP		Amsterdam	34 jaar	1820 dagen
Mustafa Amhaouch	CDA			
Farid Azarkan	DENK			

The developer tools on the right show the HTML structure of the grid. The grid is implemented using a combination of `div` and `table` elements. The `div` elements use a grid system with classes like `div-t-grid_col-t-grid_col--full-t-grid_col--half-at-small` and `div-m-card--main` and `div-m-card--content`. The `table` element is used for the statistics, with a class `u-text-size--small u-text-color--default`.

Example

- Hence, this part of the code:

```
# Extract the table containing the data pieces
table <- html_nodes(webpage, "div.m-card__content")
```

- is finding *all the nodes* which contain the information about the politician.
- This part of the code:

```
data <- lapply(table, function(x) {
  name <- html_text(html_nodes(x, "a.m-member__name"))
  party <- html_text(html_nodes(x, "span.m-member__subtitle"))
  wal <- html_text(html_nodes(x, "td")) } )
```

- is looping through all the 150 unique politicians, and getting the text from *all* nodes that are ``, `` and `<td>`.
- In the first two cases, there is only 1 element each. In the case of `<td>`, there are more. Hence the rest of the code.

Another Example: Wikipedia

Wikipedia Example