

Data scraping

Fernanda Sobrino

6/22/2021

Introducción

Qué datos podemos obtener?

- ▶ Cada vez hay más datos disponibles en la web:
 - ▶ discursos, información bibliográfica...
 - ▶ redes sociales, comunicados de prensa, noticias...
 - ▶ información geográfica...

Datos subidos diariamente a la web



4.4M



2M



682M



67M



4Mhrs



4.3M



7500TB

Qué es?

- ▶ La mayoría de estos datos por lo general son **no estructurados**
- ▶ **Web Scraping**: proceso para extraer esta información automáticamente y transformarla en datos **estructurados**

Por qué es una buena idea?

- ▶ Copiar y pegar directamente de una página web:
 - ▶ mucho tiempo
 - ▶ aburrido
 - ▶ fácil equivocarse
 - ▶ inviable
- ▶ Scraping automatizado:
 - ▶ bueno para bases de datos muy grandes
 - ▶ reproducible
 - ▶ técnicas que se pueden adaptar fácilmente a otros contextos
 - ▶ es más fácil detectar errores y corregirlos

Cuándo es buena idea hacer scraping?

1. Trade-off entre el tiempo invertido hoy y el tiempo invertido en el futuro
2. El tiempo de computadora es bárato, el de los humanos no tanto

Tipos de data-scraping

1. Web scraping:

- ▶ funciona como un navegador
- ▶ extrae el html (código base de la página)
- ▶ podemos extraer partes específicas de las páginas

2. Screen scraping:

- ▶ detector visual
- ▶ extrae directamente para extraer directamente de la pantalla
- ▶ podemos obtener información de la interfaz del usuario
- ▶ utiliza OCR (reconocimiento óptico de caracteres)
- ▶ ayuda a recuperar datos de sistemas viejos

Web-Scraping vs APIs

1. APIs (application programming interfaces)
 - ▶ conjunto de protocolos de comunicación y procedimientos que te dan acceso a los datos de una aplicación
 - ▶ acceso directo a la aplicación
 - ▶ el dueño controla que datos y quién puede tener acceso a ellos
 - ▶ no todos los sitios cuentan con una
2. Web scraping:
 - ▶ puedes accesar a casi cualquier página
 - ▶ los datos no están organizados sistemáticamente
 - ▶ no hay limite a la cantidad de búsquedas que puedes hacer

Web Scraping Best Practices

Sugerencias para hacer data-scraping

Regla básica: Se amable

1. Respetar al sitio:
 - ▶ revisar si cuentan con un API y usarlo
 - ▶ citar de donde vienen los datos (cuidado con el copyright)
2. Respetar robots.txt
 - ▶ reglas escritas por el sitio
 - ▶ algunos no dejan hacer scrape (bloquean cualquier bot)
 - ▶ otros te dicen que tan frecuentemente puedes sacar la información

Sugerencias para hacer data-scraping

3. Reducir la velocidad del bot/spyder:
 - ▶ esperar un par de segundos entre búsquedas
 - ▶ extraer solo la información necesaria y si es posible solo una vez
4. Aleatorizar el patrón de búsqueda:
 - ▶ ayuda a evitar ser bloqueado
 - ▶ los bots siguen patrones, las personas no

Sugerencias para hacer data-scraping

5. Usar proxies y rotarlos:

- ▶ el IP es visible siempre
- ▶ el sitio sabrá que datos y cuantos estas tomando
- ▶ crear un pool de IP's y rotarlos

6. Rotar user-agents

- ▶ el user-agent es una herramienta que le dice al servidor que navegador estamos utilizando
- ▶ si usas el mismo todo el rato el bot será detectado
- ▶ la mayoría de las herramientas de scrape no lo tienen incorporado así que hay que agregarlo

Sugerencias para hacer data-scraping

7. Usar herramientas que se parecen más un humano:
 - ▶ especialmente si las páginas utilizan JavaScript
 - ▶ ejemplos: Selenium, Puppeteer, Playwright
8. Revisar si las páginas no son rediseñadas constantemente
 - ▶ algunos sitios tienen la misma estructura en las primeras páginas pero cambian con las siguientes para evitar el scraping

Sugerencias para hacer data-scraping

9. Evitar hacer scape de datos detras de un login
10. Cuidado con los honeypot traps:
 - ▶ herramientas para detectar hackers
 - ▶ links invisibles para un humano pero visibles para el bot
 - ▶ si el bot los extrae o intenta accesarlos te bloquean

Sugerencias para hacer data-scraping

11. Usar servicios que resuelven captchas:
 - ▶ algoritmos capaces de resolver captchas
12. API's:
 - ▶ leer la documentación
 - ▶ límites de busquedas
 - ▶ los datos son publicos

Cómo detectan y bloquean los sitios el web scraping?

1. Tráfico inusual/alta tasa de descarga (de un mismo IP o usuario)
2. Tareas repetitivas realizadas en el sitio con el mismo patrón de navegación
3. Comprobando si eres un navegador real (ej. JavaScript)
4. Detección usando honeypots

Los detectores de bots se vuelven mejores cada día, utilizan algoritmos de AI y observan las variables, eventos, acciones etc que pueden desenmascarar al bot

Cómo saber si te bloqueo un sitio?

- ▶ CAPTCHAS que no estaban antes
- ▶ Contenido inusual
- ▶ Respuestas HTTP 404, 301 or 50x errors

El arte del web scraping

Flujo de trabajo

1. Aprender de la estructura del sitio
2. Escoger una estrategia
3. Construir el código base: extraer, preparar, validar
4. Generalizar: loops, funciones, etc
5. Limpieza

Escenarios básicos de sitios

- ▶ datos en tablas
- ▶ datos no estructurados
- ▶ datos detrás de formularios

Datos en tablas

| Rank | Year | Movie | Worldwide Box Office | Domestic Box Office | International Box Office |
|------|------|---|----------------------|---------------------|--------------------------|
| 1 | 2009 | Avatar | \$2,845,899,541 | \$760,507,625 | \$2,085,391,916 |
| 2 | 2019 | Avengers: Endgame | \$2,797,800,564 | \$858,373,000 | \$1,939,427,564 |
| 3 | 1997 | Titanic | \$2,207,986,545 | \$659,363,944 | \$1,548,622,601 |
| 4 | 2015 | Star Wars Ep. VII: The Force Awakens | \$2,064,615,817 | \$936,662,225 | \$1,127,953,592 |
| 5 | 2018 | Avengers: Infinity War | \$2,044,540,523 | \$678,815,482 | \$1,365,725,041 |
| 6 | 2015 | Jurassic World | \$1,669,979,967 | \$652,306,625 | \$1,017,673,342 |
| 7 | 2019 | The Lion King | \$1,654,381,934 | \$543,638,043 | \$1,110,743,891 |
| 8 | 2015 | Furious 7 | \$1,516,881,526 | \$353,007,020 | \$1,163,874,506 |
| 9 | 2012 | The Avengers | \$1,515,100,211 | \$623,357,910 | \$891,742,301 |
| 10 | 2019 | Frozen II | \$1,446,925,396 | \$477,373,578 | \$969,551,818 |
| 11 | 2015 | Avengers: Age of Ultron | \$1,395,316,979 | \$459,005,868 | \$936,311,111 |
| 12 | 2018 | Black Panther | \$1,336,494,321 | \$700,059,566 | \$636,434,755 |
| 13 | 2011 | Harry Potter and the Deathly Hallows:... | \$1,334,392,544 | \$381,193,157 | \$953,199,387 |
| 14 | 2017 | Star Wars Ep. VIII: The Last Jedi | \$1,331,635,141 | \$620,181,382 | \$711,453,759 |
| 15 | 2018 | Jurassic World: Fallen Kingdom | \$1,308,334,005 | \$417,719,760 | \$890,614,245 |
| 16 | 2013 | Frozen | \$1,267,483,831 | \$400,953,009 | \$866,530,822 |
| 17 | 2017 | Beauty and the Beast | \$1,255,080,655 | \$504,014,165 | \$751,066,490 |
| 18 | 2018 | Incredibles 2 | \$1,242,805,359 | \$608,581,744 | \$634,223,615 |
| 19 | 2017 | The Fate of the Furious | \$1,236,703,796 | \$225,764,765 | \$1,010,939,031 |
| 20 | 2013 | Iron Man 3 | \$1,215,392,272 | \$408,992,272 | \$806,400,000 |
| 21 | 2015 | Minions | \$1,159,631,140 | \$336,045,770 | \$823,585,370 |
| 22 | 2016 | Captain America: Civil War | \$1,151,918,521 | \$408,084,349 | \$743,834,172 |
| 23 | 2018 | Aquaman | \$1,143,758,700 | \$335,061,807 | \$808,696,893 |
| 24 | 2019 | Spider-Man: Far From Home | \$1,132,183,514 | \$390,532,085 | \$741,651,429 |
| 25 | 2019 | Captain Marvel | \$1,129,727,388 | \$426,829,839 | \$702,897,549 |
| 26 | 2011 | Transformers: Dark of the Moon | \$1,123,794,079 | \$352,390,543 | \$771,403,536 |
| 27 | 2003 | The Lord of the Rings: The Return of ... | \$1,120,214,046 | \$377,845,905 | \$742,368,141 |
| 28 | 2012 | Skyfall | \$1,110,526,981 | \$304,360,277 | \$806,166,704 |
| 29 | 2014 | Transformers: Age of Extinction | \$1,104,054,072 | \$245,439,076 | \$858,614,996 |
| 30 | 2012 | The Dark Knight Rises | \$1,082,228,107 | \$448,139,099 | \$634,089,008 |
| 31 | 2019 | Toy Story 4 | \$1,073,080,329 | \$434,038,008 | \$639,042,321 |
| 32 | 2019 | Star Wars: The Rise of Skywalker | \$1,072,848,487 | \$515,202,542 | \$557,645,945 |
| 33 | 2019 | Joker | \$1,072,507,517 | \$335,451,311 | \$737,056,206 |
| 34 | 2010 | Toy Story 3 | \$1,068,879,522 | \$415,004,880 | \$653,874,642 |

Datos no estructurados

June 24, 2021

[Read Review](#)
[Find tickets](#)

The Ice Road

PG-13 | Adventure, Drama, Thriller | Directed by Jonathan Hensleigh

Liam Neeson fights for traction as a big-rig driver in this mildly entertaining thriller.

By JEANNETTE CATSOULIS



June 24, 2021

[Read Review](#)
[Find tickets](#)

Against the Current

Documentary | Directed by Oskar Pall Sveinsson

The director Oskar Pall Sveinsson follows Veiga Gretarsdottir on a 103-day kayaking journey around Iceland.

By KRISTEN YOONSOO KIM



June 24, 2021

[Read Review](#)
[Find tickets](#)

F9: The Fast Saga

PG-13 | Action, Adventure, Crime, Thriller | Directed by Justin Lin

Vin Diesel returns to lead his fast and furious family on another dangerous and ridiculous mission.

By A.O. SCOTT



June 24, 2021

[Read Review](#)
[Find tickets](#)

Sun Children

Drama | Directed by Majid Majidi

Majid Majidi's social-realist drama about street children in Iran trades narrative complexity for precious visuals.

By DEVINKA GIRISH



Datos detrás de formularios web



A Quiet Place Part II



91%

In Theaters May 28



Return To Gorée
(Retour À Gorée)



100%

In Theaters Jun 11



The Conjuring: The Devil Made Me Do It



56%

In Theaters Jun 4



In The Heights



96%

In Theaters Jun 11



Peter Rabbit 2: The



68%

In Theaters Jun 11



Vivo

In Theaters Jun 4



Queen Bees



48%

In Theaters Jun 11



Queen Of Spades

In Theaters Jun 11

Escenarios básicos de sitios

- ▶ datos en tablas:
 - ▶ extracción automática usando ‘rvest’
- ▶ datos no estructurados:
 - ▶ identificar los elementos utilizando `selectorGadget`
 - ▶ extracción automática utilizando `rvest`
- ▶ datos detrás de formularios:
 - ▶ necesitamos simular a un humano
 - ▶ podemos usar algo como `RSelenium`

Estructura de las páginas web

Estructura de las páginas web

- ▶ los sitios de internet tienen una estructura esperada. Por lo general consiste de dos tipos distintos:
 - ▶ HTML: se centra en la apariencia y el formato de una página web
 - ▶ XML: se centra en la gestión de datos en una página web
- ▶ estos dos tipos de códigos no se parecen tanto entre si

HTML

- ▶ Hypertext Markup Language
- ▶ escondido detrás de la mayoría de los sitios
- ▶ contiene una estructura y formato esperado
- ▶ ayuda a crear sitios web fácil y rápido
- ▶ lo que vemos en el navegador es una interpretación de este

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>
<h1>This is a Heading</h1>
<p>This is a paragraph.</p>
</body>
</html>
```

HTML

- ▶ tags comunes:
 - ▶ elementos del documento: <head>, <body>, <footer>...
 - ▶ componentes del documento: <title>, <h1>,<div>...
 - ▶ estilo del texto: ,<i>,...
 - ▶ hyperlinks: <a>

XML

- ▶ Extensible Markup Language
- ▶ diseñado para guardar y transportar datos
- ▶ diseñado para poder ser leido por los humanos y las computadoras
- ▶ parecido al HTML
- ▶ en este los usurios pueden crear sus propias tags
- ▶ esto lo vuelve más difícil de minar

XML

```
<note>
  <to>Keith</to>
  <from>Steve</from>
  <heading>Kudos</heading>
  <body>Awesome work, dude!</body>
</note>
```

Cómo ver los html o xml

- ▶ Antes de extraer los datos de la web es importante:
 - ▶ explorar el código base
 - ▶ comprender como se relaciona el código con lo que vemos en la página
 - ▶ right click + inspect

Ejemplo

The screenshot shows a web browser window with the following details:

- Header:** Control, Finally, take control of benefits... administer yourself and reduce cost., Zest.
- Main Content:** A chart page listing songs:
 - 1 Old Town Road (Lil Nas X Featuring Billy Ray Cyrus)
 - 2 Bad Guy (Billie Eilish)
 - 3 I Don't Care (Ed Sheeran & Justin Bieber)
 - 4 Senorita (Shawn Mendes & Camila Cabello)
 - 5 Talk (Khalid)
 - 6 Truth Hurts (Lizzo)
- Floating Overlay:** A modal window titled "Choose the speed you..." containing a 3D model of a delivery truck.
- DevTools:** The browser's developer tools are open, showing the DOM structure and a CSS selector tool. The selector tool highlights the "header" element with the class "header".

Estructura del código

- ▶ por lo general la estructura sigue la siguiente jerarquía
 - ▶ hasta arriba <html>
 - ▶ en el segundo nivel <head> y <body>
 - ▶ dentro del <body> elementos diferentes como <div>
 - ▶ dentro de cada una de las divisiones hay otros tipos de tags
- ▶ esto es bueno porque podemos minar solo una parte del html y no el documento completo

CSS

- ▶ Cascading Style Sheets
- ▶ lenguaje utilizado para estilizar código HTML
- ▶ describe como se deben mostrar los elementos del HTML
- ▶ simplifica el uso de HTML
- ▶ puedes utilizar selectores CSS para extraer cualquier elemento de un HTML

JavaScript

- ▶ uno de los lenguajes de programación mas populares
- ▶ agrega funcionalidades a un sitio
- ▶ dinámico
- ▶ por ejemplo: cambiar la estructura del sitio después de que se cargo
- ▶ necesitamos otras técnicas de scraping para estas páginas

Analizando el HTML

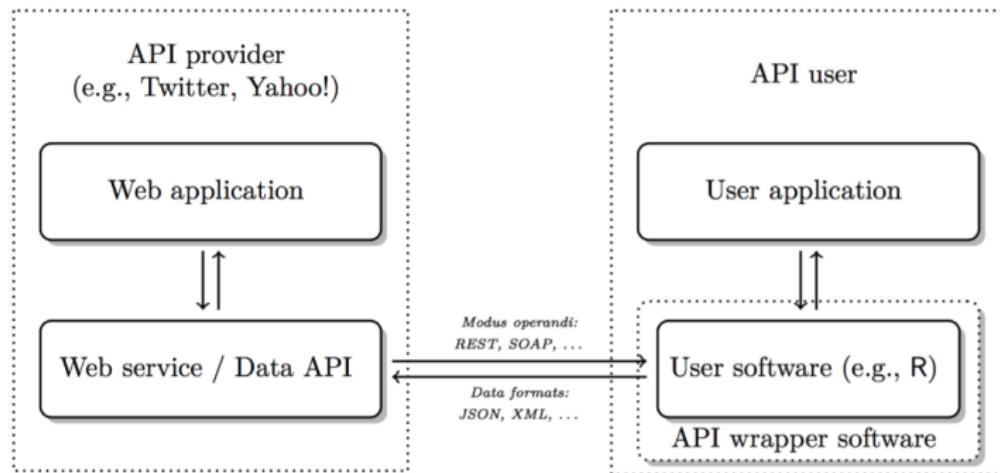
- ▶ leer el HTML en R
- ▶ entender su estructura
- ▶ cómo? rvest
 - ▶ `read_html` lee el HTML a R
 - ▶ `html_text` extrae el text del HTML
 - ▶ `html_table` extrae tablas del HTML
 - ▶ `html_nodes` extrae componentes usando el CSS selector
 - ▶ `htmlAttrs` extrae atributos de los nodos
- ▶ cómo identificamos CSS selector relevantes?
*selectorGadget extensión de Chrome y Firefox

APIs

Qué son?

- ▶ API (Application Programming Interface)
 - ▶ proveen una capa de abstracción al usuario
 - ▶ esconde todo lo que no es relevante para el usuario haciéndola fácil de usar
- ▶ HTTP (Hypertext Transfer Protocol)
 - ▶ como se comunican los navegadores se comunican a los servidores

Qué son?



Source: Munzert et al, 2014, Figure 9.8

Tipos de APIs

1. RESTful APIs: consulta información estática al momento (perfiles de redes sociales, posts, . . .)
2. Streaming APIs: cambian en tiempo real (nuevos tweets, el clima, etc. . .)

Características de las APIs

- ▶ Documentación extensiva
 - ▶ escrita para los usuarios (en lenguaje sencillo de entender)
 - ▶ contienen información de:
 - ▶ parámetros
 - ▶ endpoints: las urls que reciben o regresan información de un WebAPI
- ▶ La mayoría de los APIs tienen límites pre-establecidos
 - ▶ restricciones en el numero de API calls que puede hacer cada usuario en cierto periodo de tiempo
 - ▶ algunas empresas cobran por el uso de sus APIs

Conexión con una API

- ▶ Llamada a una REST API:
 - ▶ url base (endpoint) :
`https://maps.googleapis.com/maps/api/geocode/json`
 - ▶ parámetros: ?address = bogota
 - ▶ autenticación(opcional): &key=XXXXX
- ▶ En R podemos usar paquetes como httr para obtener la request

```
library(httr)
r <- GET("https://maps.googleapis.com/maps/api/geocode/json"
         query = list(address = "bogota"))
```

Posibles resultados de la conexión

- ▶ 200 → conexión exitosa
- ▶ 301 → movida temporalmente
- ▶ Problemas:
 - ▶ 401 → no autorizado
 - ▶ 403 → prohibido
 - ▶ 404 → no encontrada
 - ▶ 408 → tiempo agotado
 - ▶ 429 → demasiadas requests
 - ▶ 501 → servicio no disponible

Qué nos regresó la request anterior?

```
Response [https://maps.googleapis.com/maps/api/geocode/json]
Date: 2021-06-27 18:32
Status: 200
Content-Type: application/json; charset=UTF-8
Size: 1.55 kB

{
  "results" : [
    {
      "address_components" : [
        {
          "long_name" : "Bogotá",
          "short_name" : "Bogotá",
          "types" : [ "locality", "political" ]
        },
        {
          ...
        }
      ],
      "formatted_address" : "Bogotá, Colombia"
    }
  ]
}
```

JSON

- ▶ la mayoría de las APIs van a regresar los resultados en formato .json
- ▶ JavaScript Object Notation
- ▶ content(r,"text")

```
{\n    \"results\" : [\n        {\n            \"address_componen\n}\n}
```

JSON

- ▶ los datos están guardados en un diccionario: parejas de llave-valor
- ▶ esto es ligero (pocos bytes)
- ▶ flexible
- ▶ {}: para cada uno de los objetos
- ▶ []: matrices/vectores de características del objeto
- ▶ fromJSON de la librería jsonlite permite leer datos en formato JSON a R
- ▶ existen más paquetes y funciones capaces de esto

JSON

- ▶ el paquete httr tiene content(r, "parsed")

```
$results
$results[[1]]
$results[[1]]$address_components
$results[[1]]$address_components[[1]]
$results[[1]]$address_components[[1]]$long_name
[1] "Bogotá"
```

R packages

- ▶ existen paquetes específicos para muchas APIs
- ▶ antes de empezar un proyecto conviene buscarlos
 - ▶ CRAN Web Technologies Task View
 - ▶ GitHub
 - ▶ rOpenSci Consortium

Por qué usar APIs?

Ventajas

- ▶ evitas problemas como:
 - ▶ bloqueos
 - ▶ HTML con formatos incorrectos
 - ▶ eres amable con la empresa que provee los datos
- ▶ formato estándar en los datos
 - ▶ transparente
 - ▶ 'replicable'
- ▶ robusto
 - ▶ beneficio del trabajo de otras personas

Por qué usar APIs?

Problemas

- ▶ no todos los sitios tienen uno
- ▶ depende del proveedor
- ▶ no hay una conexión natural con R
- ▶ límites a lo que puedes extraer, cuando y cuanto