Economía y Ciencia de los Datos

Web Scraping

Carlos Alvarado & Pablo González

29 de agosto de 2022

Bases necesarias y motivación

¿Qué es una Web Scraping?

El término web scraping se refiere al proceso de extracción de información en sitios web. Esto puede ser realizado via protocolo HTTP manualmente o via aplicaciones que manejen a un navegador (por ejemplo, mediante Selenium).

Aspectos generales:

- Datos extraídos "personalmente".
  (No son facilitados por el proveedor)
- Aplicación generada busca emular a usuario del sitio.
- Sitio puede cambiar sin previo aviso.
  Se recomienda registro de logs
- Cada sitio puede requerir una estrategia diferente.

¿Por qué es relevante?

Los datos que requerimos para efectuar un estudio no siempre se encuentran inmediatamente disponibles y/o completos. Además, si bien prácticas, tampoco podemos contar con la existencia de APIs para todo tipo de aplicaciones.

En ocasiones es necesario recolectar datos por nuestra cuenta y aprender técnicas que permitan hacer esto expandirá el tipo de fuentes de información que podrán agregar a sus análisis.

Algunos casos de uso:

- Búsqueda y extracción de noticias
- Evaluar productos en supermercados:
  (Cambios de precios y/o entrada y salida de productos)
- Análisis de redes sociales

(*) Procesos de extracción única o serie de procesos planificados suelen conllevar diferente nivel de cuidado al momento de generar la aplicación.

Acerca de la web - Conceptos Fundamentales

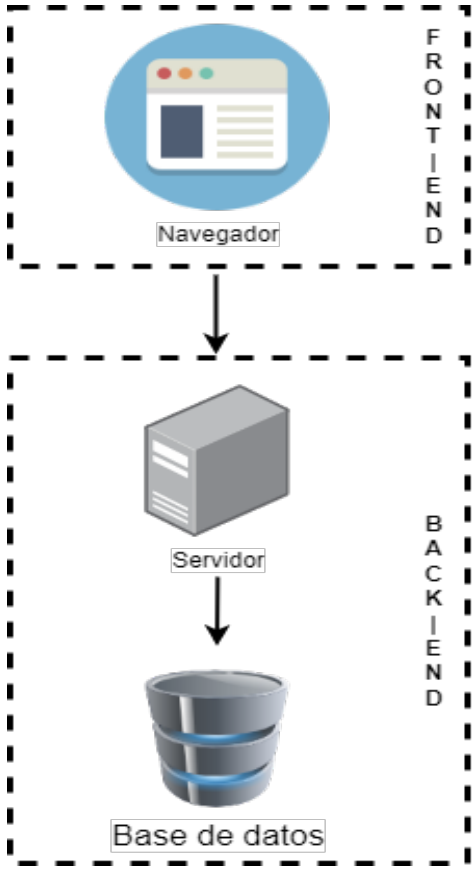
Funcionamiento de un sitio web

Desde una perspectiva general, el desarrollo web se puede separar en dos áreas principales:

- Back-end: Se encarga de aquellas partes con las que no interactuamos (conexión con las bases de datos y procesamiento, etc.). Utiliza lenguajes tales como SQL, PHP, Python, Perl y Javascript.
- Front-end: Se encarga de la presentación e interacción con el sitio web. Utiliza lenguajes tales como HTML, CSS y Javascript.

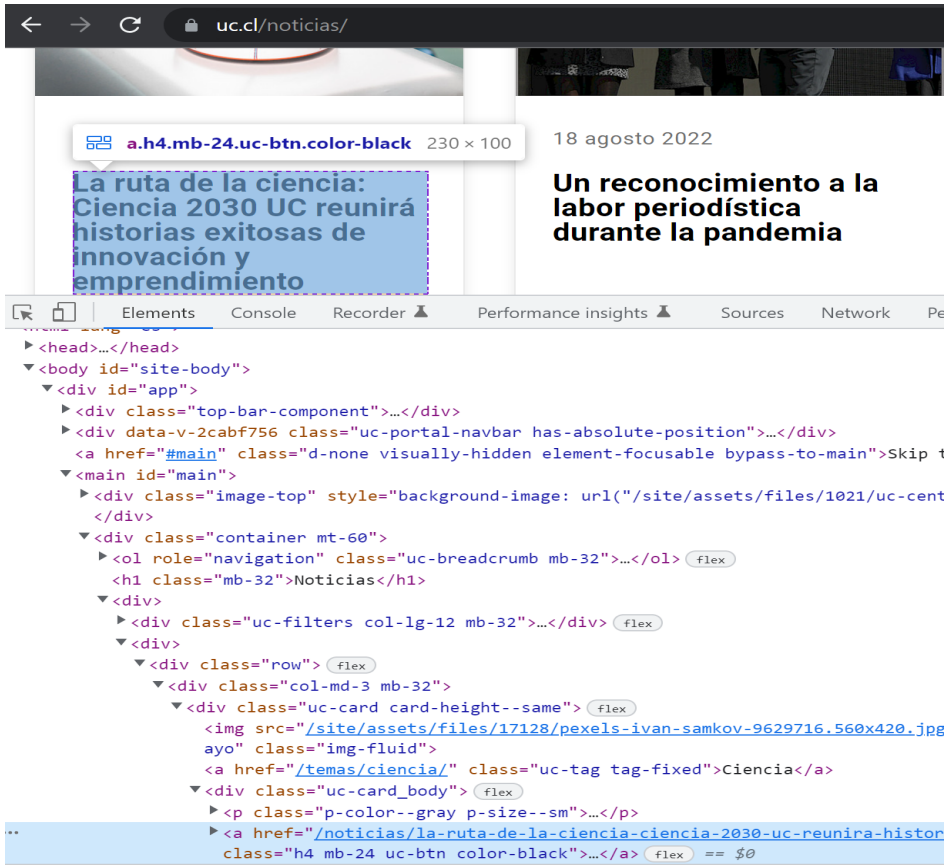
(*) Javascript es un caso especial: Hoy en día permite, incluso, ejecutar Python.

Para extraer datos de la web, primero requerimos saber algo del front-end. En particular, cómo están etiquetados (HTML) los datos que queremos extraer.



Estructura del sitio web → *HTML*

- El esqueleto de un sitio web (su estructura) se basa en como se encuentra etiquetado. Esto se realiza mediante HTML.
- Tanto el diseño (CSS) como las funciones (Javascript) se encuentran en el encabezado del código (<header>...</header)
- El contenido se encuentra etiquetado dentro del cuerpo, en el DOM (<body>...</body)



(*) Basados en la id de un elemento (directo) o en la clase o XPATH (patrones), buscamos automatizar la extracción de información consistentemente presente en un sitio web.

Pongámoslo en práctica

Analisemos el etiquetado de un sitio web

Técnicas de Web Scraping

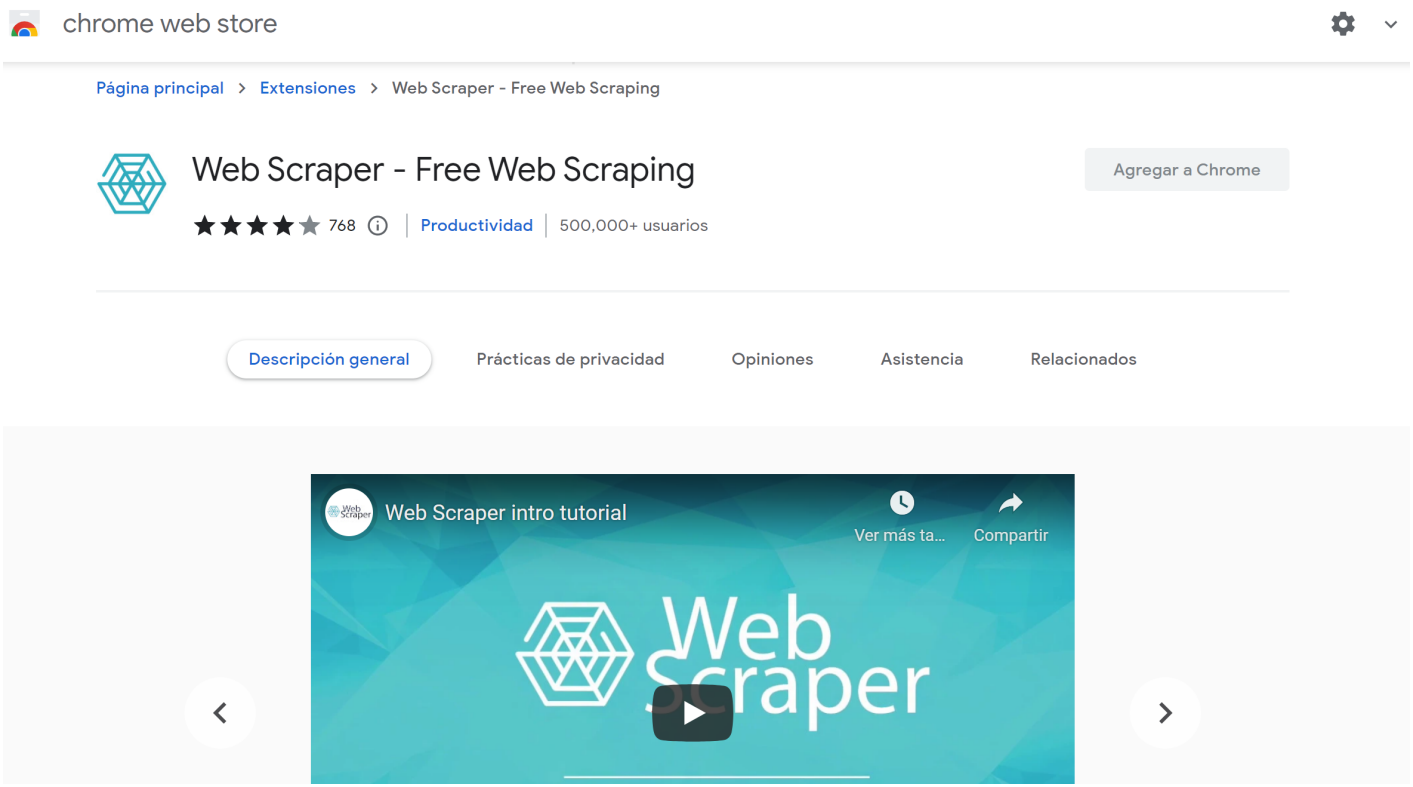
¿De qué forma podemos extraer datos desde la web?

En general, podemos clasificar las metodologías de extracción de datos de la siguiente forma:

- **Servicios profesionales**: Existen empresas, con softwares generales, que venden servicios de extracción de datos (potencialmente costo-eficiente en términos de tiempo).
- **Alternativas No-Code**: Extensiones que permiten realizar estos procesos desde el navegador. Sin embargo, cabe destacar que el control que tendrías sobre este sería menor.
- **Solo código**: Solicita código HTML, el que lee mediante un parser. Suele requerir de mayor familiaridad con proxies.
- **Controlando el navegador**: Controlamos un navegador (por ejemplo: Google Chrome) mediante Python. Nos permite, fácilmente, emular a un usuario e interactuar con sitios de mayor complejidad (por ejemplo, dinámicos).

(*) Para efectos del curso, revisaremos las últimas dos alternativas y nos centraremos en la última.

Ejemplo: Extensión Web Scraper (No-Code)



Algunos aspectos relevantes a mencionar son:

- Facilidad de uso. (No requiere saber programar)
- Menor libertad en procesos de extracción y difícil de emplear en procesos regulares.

Ejemplo: Solo código (parser: BeautifulSoup)

- Rápido e ideal para sitios estáticos.
- No requiere de dependencias adicionales.
- Proceso puede complejizarse en caso de sitios dinámicos.

(Nota: Este ejemplo extrae noticias del sitio de la universidad) →

```python
import requests
import pandas as pd
from bs4 import BeautifulSoup

base_url, noticias = 'http://www.uc.cl', {}

# Extraemos noticias (primeras 5 páginas)
for i in range(1,6):
    r = requests.get(f'{base_url}/noticias/pagina{i}')
    soup = BeautifulSoup(r.text, 'html.parser')
    elementos = soup.find_all('div', class_='uc-card_body')
    for i in elementos:
        noticias[i.a.get('href')] = i.a.text

# A dataframe para luego agregar datos
df = {i: [noticias[i]] for i in noticias} # para pandas
df = pd.DataFrame.from_dict(data = df , orient = 'index')\
                 .reset_index(drop = False)\
                 .rename(columns = {'index':'url',0:'titulo'})

def get_text(url):
    r = requests.get(f'{base_url}/{url}')
    soup = BeautifulSoup(r.text, 'html.parser')
    elementos = soup.find_all('div', class_='body-article')
    text = ' '.join([i.text for i in elementos])
    text = text.replace('\n','').replace('\xa0','')
    return text

df['contenido'] = df.url.apply( get_text )
```

Ejemplo: Selenium

```
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
from utils import get_data

if __name__ == '__main__':

    service = webdriver.chrome.service.Service(executable_path = ChromeDriverManager().install() )
    options = webdriver.ChromeOptions()
    options.add_experimental_option('prefs', {'profile.managed_default_content_settings.images':2,
                                    'profile.managed_default_content_settings.javascript':2})

    # Abrimos Chrome con las configuraciones deseadas
    browser = webdriver.Chrome(service = service, options = options)

    df = get_data( browser , 'url')
    df.to_parquet('nombre_archivo', index = False)
```

- Ejemplo inicializa navegador controlado mediante objeto "browser".
- Para acelerar navegación, se omite carga de imagenes y ejecución de códigos javascript.
- Objeto se puede pasar a una función que realice el proceso.

Pongámoslo en práctica

Ir a Notebook