Extracción de la Información de la Web

Máster Online en Ciencia de Datos

UCO ONLINE

Dr. José Raúl Romero

Profesor Titular de la Universidad de Córdoba y Doctor en Ingeniería Informática por la Universidad de Málaga. Sus líneas actuales de trabajo se centran en la democratización de la ciencia de datos (*Automated ML* y *Explainable Artificial Intelligence*), aprendizaje automático evolutivo y analítica de software (aplicación de aprendizaje y optimización a la mejora del proceso de desarrollo de software).

Miembro del Consejo de Administración de la *European Association for Data Science*, e investigador senior del Instituto de Investigación Andaluz de *Data Science and Computational Intelligence*.

Director del **Máster Online en Ciencia de Datos** de la Universidad de Córdoba.

Introducción a la Extracción de Datos de la Web: *Web Scraping*

Introducción a *Web Scraping*

UCO ONLINE

Definición

- *Web scraping* es el proceso de extracción de datos de sitios web
  - Puede hacerse manualmente o con herramientas de terceros que (semi-)automatizan el proceso
  - Es más rápido, eficaz y menos propenso a errores automatizar la tarea
- Algunos datos disponibles en la web se presentan en un formato que facilita su recopilación y uso en análisis de datos (p.ej. CSV, excel, JSON)
- Habitualmente, aunque estén disponibles públicamente, los datos no son fáciles de reutilizar: contenido en PDF, en una tabla, repartidos en varias páginas web, o incrustados en algún elemento multimedia
- La automatización de *web scraping* debe considerar si el proceso debe ejecutarse a intervalos regulares y capturar los cambios en los datos

Diferenciemos…

Scraping: la extracción efectiva de datos / información de un sitio web

Crawling: seguimiento de hipervínculos por la WWW para recorrer múltiples páginas y/o sitios

Búsqueda: uso de motores de búsqueda de terceros (p.ej. Google) de forma automática para encontrar información en la web

- *Web scraping* está relacionado con la indexación web en IR, que hacen los motores de búsqueda cuando analizan la web para construir sus índices
  - A diferencia de la indexación web, que suele analizar todo el contenido de una página web para hacerla consultable, *web scraping* se centra en información específica de las páginas
- Los seres humanos somos buenos categorizando rápidamente y extrayendo datos de interés
- Computacionalmente es más complejo dar sentido a esos datos no estructurados, a menos que les digamos específicamente de qué elementos están hechos los datos (p.ej. mediante etiquetas)
  - Los datos estructurados son elementos individuales están separados y etiquetados
  - Los datos en la web suelen encontrarse en un formato no estructurado o semi-estructurado

Algunos datos se estructuran para facilitar su visualización → se disponen en celdas dentro de una tabla

PERO esto no los convierte en datos estructurados, ya que los diferentes elementos de información pueden no estar claramente etiquetados

```
<div class="ce-mip-mp-tile-container " id="mp-tile-person-id-72029">
    <a class="ce-mip-mp-tile" href="/members/en/dan-albas(72029)">
        <div class="ce-mip-flex-tile">
            <div class="ce-mip-mp-picture-container">
                <img class="ce-mip-mp-picture visible-lg visible-md img-fluid" src="/Content/Parliamentarians/Images/OfficialMPPhotos/44/AlbasDan_CPC.jpg"
                    alt="Photo - Dan Albas - Click to open the Member of Parliament profile">
            </div>
            <div class="ce-mip-tile-text">
                <div class="ce-mip-tile-top">
                    <div class="ce-mip-mp-honourable"></div>
                    <div class="ce-mip-mp-name">Dan Albas</div>
                    <div class="ce-mip-mp-party" style="border-color:#002395;">Conservative</div>
                </div>
                <div class="ce-mip-tile-bottom">
                    <div class="ce-mip-mp-constituency">Central Okanagan&#x2014;Similkameen&#x2014;Nicola</div>
                    <div class="ce-mip-mp-province">British Columbia</div>
                </div>
            </div>
        </div>
        <div>
        </div>
    </a>
</div>
```

```
                <tr role="row" id="mp-list-id-72029">
                    <td data-sort="Albas Dan" class="sorting_1">
                        <a href="/members/en/dan-albas(72029)">
                            Albas, Dan
                        </a>
                    </td>
                    <td data-sort="Conservative">Conservative</td>
                    <td data-sort="Central Okanagan Similkameen Nicola">
                        <a href="/members/en/constituencies/central-okanagan-similkameen-nicola(902)">Central Okanagan&#x2014;Similkameen&#x2014;Nicola</a>
                    </td>
                    <td data-sort="British Columbia">British Columbia</td>
                </tr>
```

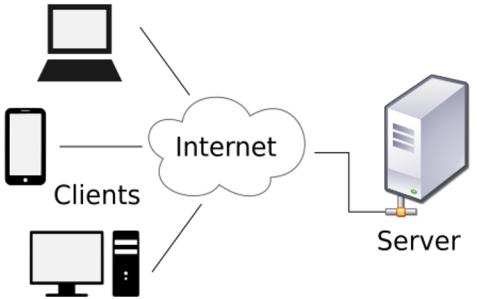
¿Qué debemos conocer para *web scraping?*

- *HTTP:* protocolo de comunicación (*Hyper Text Transfer Protocol*)
- *HTML:* el lenguaje en el que se definen las páginas web
- *JS* (javascript): código que se ejecuta en el cliente
- *CSS:* hojas de estilo, cómo se visualizan las páginas web. No contiene datos.
- *CSV, TXT, JSON, XML:* son datos, ¡interesante!

Otros formatos: tipos multimedia (incluyendo imágenes, video, etc.), PDF, …

HTTP

- Protocolo para el intercambio de información entre máquinas, transportada por Internet, para permitir la compartición de datos hipermedia (texto+multimedia)
- El conjunto de páginas (documentos) enlazadas mediante hiperenlaces se denomina WWW (World Wide Web)
- HTTP define aspectos de autenticación, solicitudes, códigos de estado, conexiones persistentes, solicitud/respuesta cliente/servidor, etc.



Introducción a la Extracción de Datos de la Web: *Web Scraping*

Selección de elementos con XPath

UCO ONLINE

Lenguaje de expresión XPath

- XPath (que significa *XML Path Language*) es un lenguaje de expresión utilizado para especificar partes de un documento XML
- XPath también puede utilizarse en documentos con una estructura similar a XML
- XML y HTML son lenguajes de marcado. Esto significa que utilizan un conjunto de etiquetas o reglas para organizar y proporcionar información sobre los datos que contienen. Esta estructura ayuda a automatizar el tratamiento, la edición, el formateo, la visualización, la impresión, etc. de esa información
- HTML y XML tienen una estructura muy similar, por lo que XPath puede utilizarse casi indistintamente para navegar por documentos HTML y XML. De hecho, HTML5 es un dialecto particular de XML

https://www.scrapingbee.com/blog/practical-xpath-for-web-scraping/

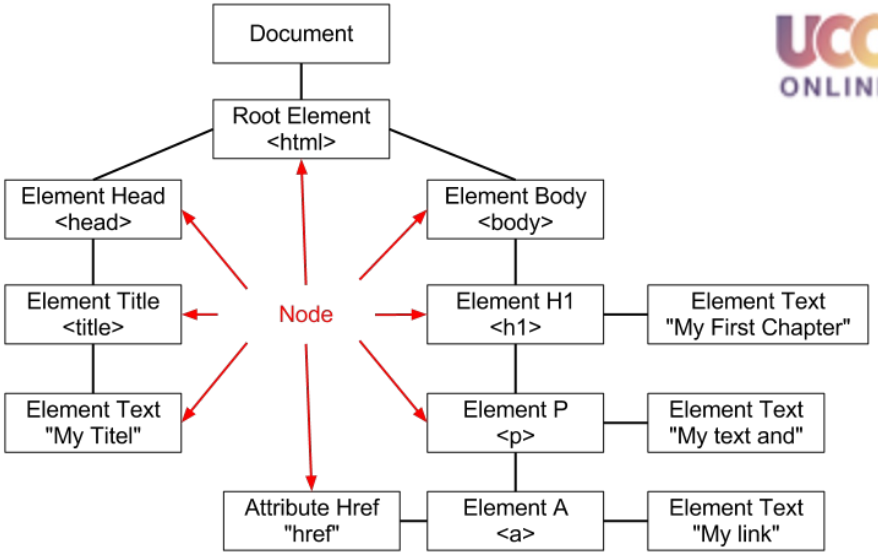
Lenguaje de expresión XPath

- El documento XML sigue unas reglas sintácticas básicas:
  - Un documento XML se estructura mediante nodos, que incluyen nodos de elementos, nodos de atributos y nodos de texto
  - Los nodos de elementos XML deben tener una etiqueta de apertura y otra de cierre, por ejemplo, <archivo> etiqueta de apertura y </archivo> etiqueta de cierre
  - Las etiquetas XML distinguen entre mayúsculas y minúsculas; por ejemplo, <archivo> no es igual a <archiVo>.
  - Los elementos XML deben estar correctamente anidados
  - Los nodos de texto (datos) están contenidos dentro de las etiquetas de apertura y cierre
  - Los nodos de atributos XML contienen valores que deben citarse, por ejemplo, <archivo type=”CSV"></archivo>

Expresiones XPath

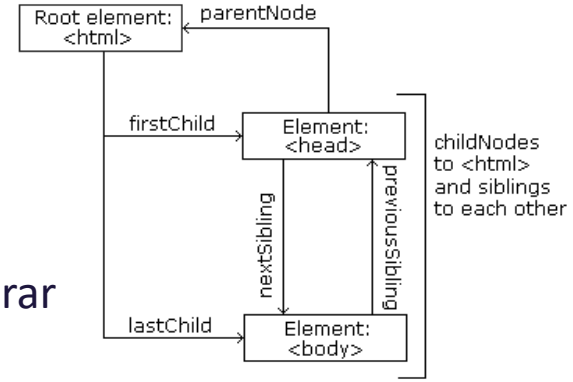
- XPath se escribe utilizando expresiones que consisten en valores y operadores, que devolverán un único valor
  - 45+34 es un ejemplo de expresión que se reducirá al valor 79
- En la terminología de programación, esto se denomina evaluar, lo que significa reducir a un único valor sin operadores
  - Un valor es una expresión que se reduce a sí mismo
- Con XPath no se requiere saber de antemano cómo son los datos, a diferencia de las expresiones regulares, que demandan conocer su patrón
- Dado que los documentos XML están estructurados en nodos, XPath hace uso de esa estructura para navegar por los nodos y seleccionar los datos necesarios
  - Las expresiones XPath sobre documentos XML devuelven objetos conteniendo los nodos seleccionados

Expresiones XPath



- En un documento HTML, todo es un nodo:
  - Todo el documento es un nodo del documento
  - Cada elemento HTML es un nodo de elemento
  - El texto dentro de los elementos HTML son nodos de texto
- Los nodos de un árbol de este tipo tienen una relación jerárquica entre sí. Utilizamos los términos padre (*parent*), hijo (*child*) y hermano (*sibling*) para describir estas relaciones:
  - En un árbol de nodos, el nodo superior se llama *root* (o nodo raíz)
  - Cada nodo tiene exactamente un padre, excepto la raíz
  - Un nodo puede tener cero, uno o varios hijos
  - Los hermanos son nodos con el mismo padre
  - La secuencia de conexiones de nodo a nodo se llama ***path*** (ruta)

Expresiones XPath: rutas



- Las **rutas en XPath** se definen utilizando barras / para separar los pasos de una secuencia de conexión de nodos
- Todas las expresiones se evalúan conforme a un **nodo de contexto**
  - El nodo de contexto es el nodo en el que comienza una ruta
  - El contexto por defecto es el nodo raíz, indicado por una sola barra /

Expresión	Descripción
nombre	Seleccionar todos los nodos con “nombre”
/	Al principio de la expression indica una selección desde el nodo raíz, las barras subsiguientes indican la selección de un nodo hijo desde el nodo actual
//	Seleccionar los nodos hijos directos e indirectos (es posible saltar niveles) del documento a partir del nodo actual
.	Seleccionar el nodo de contexto actual
..	Seleccionar el padre del nodo de contexto
@	Seleccionar los atributos del nodo de contexto
[@attribute = 'value']	Seleccionar nodos con un valor de atributo determinado
text()	Seleccionar el contenido de texto de un nodo
|	Encadenar expresiones y devolver los resultados de cualquiera de ellas (unión de conjuntos)

Expresiones XPath: accesos

- Para seleccionar todos los *header* de un documento podría bastar con "//header"
  - Esto retorna un array de objetos, que pueden accederse con la sintaxis ***elem*[1]** (*1-indexed*)

```
$x("//blockquote[@class=retos']")[1]
```

- En HTML, los elementos se categorizan con **class** e **id**

```
//blockquote[@class=retos']
```

```
//blockquote[@id='micitafavorita']
```

- Se pueden considerar los **operadores lógicos** en las expresiones: **= != > >= < <= or and not**

```
//header/@id!='books' and @id!='conferences'
```

Expresiones XPath: predicados

- Se utilizan para encontrar un nodo específico o que contiene un valor específico
- Los predicados se encierran en [ ] y están pensados para filtrado de resultados
- Permiten operadores y funciones:

Predicado	Descripción
[1]	Seleccionar el primer nodo
[last()]	Seleccionar el último nodo
[last()-1]	Seleccionar el penúltimo nodo
[position()<4]	Seleccionar los tres primeros nodos (1-indexed)
[@lang]	Seleccionar los nodos que tengan el atributo “lang”
[@lang=‘es']	Seleccionar los nodos que tengan el atributo “lang” con valor “es”
[salario>3500.00]	Seleccionar los nodos que tengan un nodo ”salario” con un valor superior a 3500.00
//h1[2]	Seleccionar el segundo de los nodos H1

Expresiones XPath: comodines

- Se utilizan para seleccionar nodos XML/HTML5 desconocidos
- Los predicados se encierran en [ ] y están pensados para filtrado de resultados
- Permiten operadores y funciones:

Predicado	Descripción
*	Cualquier nodo de elemento
@*	Cualquier nodo de atributo
node()	Cualquier nodo de cualquier tipo

- Ejemplo: seleccionar todos los nodos de clase “journal”, sea el tipo que sea

```
//*[@class='journal']
```

Expresiones XPath: búsquedas en texto

- Xpath permite realizar búsqueda en cadenas de texto, incluyendo el uso de **REGEX** y su función `matches()`
- Discrimina entre minúsculas y mayúsculas:

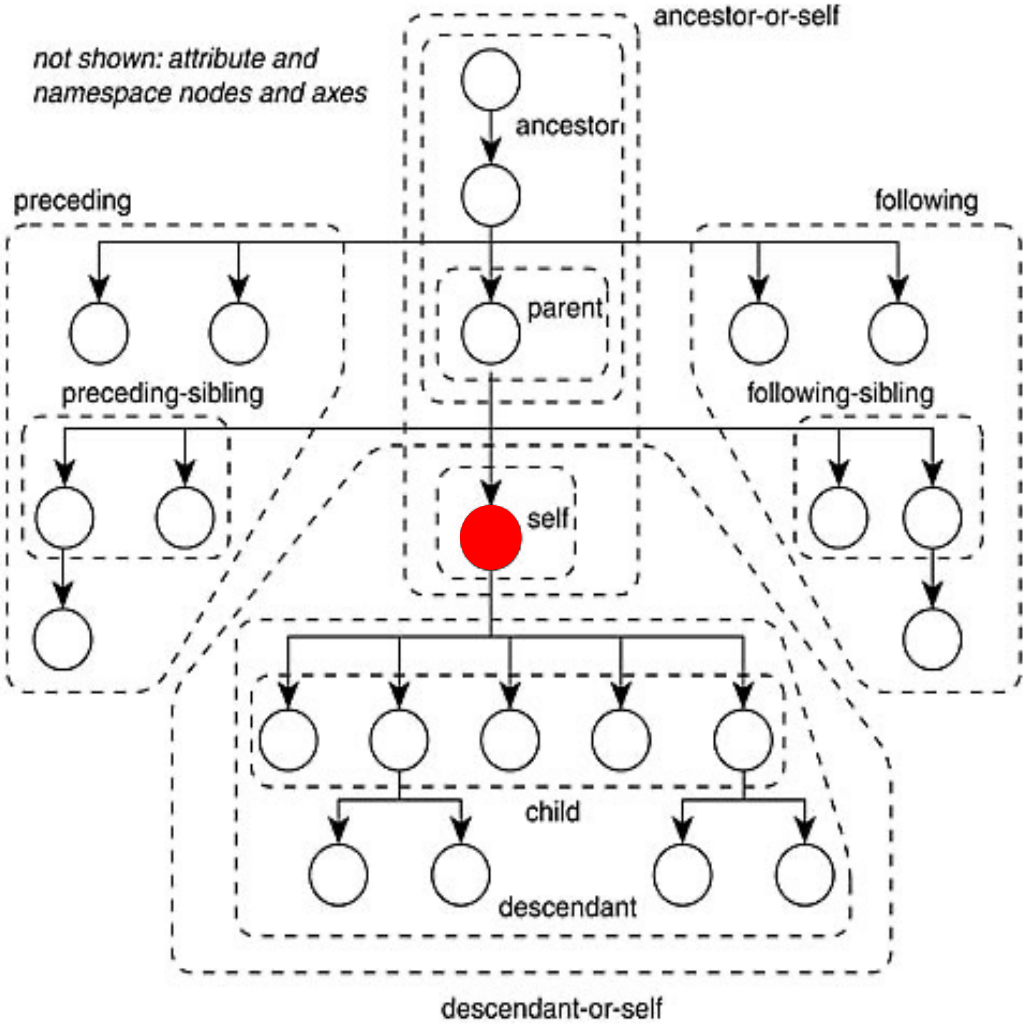
Función	Descripción	Ejemplo
contains	Encuentra todos los nodos que contienen la cadena indicada	//*[contains(.,"Journal")]
starts-with	Encuentra todos los nodos que empiezan por la cadena indicada	//*[starts-with(.,"J")]
ends-with	Encuentra todos los nodos que terminan por la cadena indicada	//*[ends-with(.,"Proceedings")]
matches	Encuentra todos los nodos que cumplen con la expresión regular	//*[matches(.,"J.R.*")]

*El uso de comodín * es un mero ejemplo, pueden utilizarse otras expresiones*

Expresiones XPath: XPath Axes

- XPath Axes ofrece todo un conjunto de mecanismos para especificar una ruta en base a las relaciones entre elementos y sus conexiones
- Utiliza **13 ejes** (*Axes*)
- **`self`** es el nodo de contexto

```
self  child  descendant  parent  ancestor
  descendant-or-self  following-sibling
preceding-sibling  following  preceding
```



Expresiones XPath: XPath Axes

- **Ejemplos**: Todos los hermanos siguientes del H1 “publica” en el **body** del HTML5

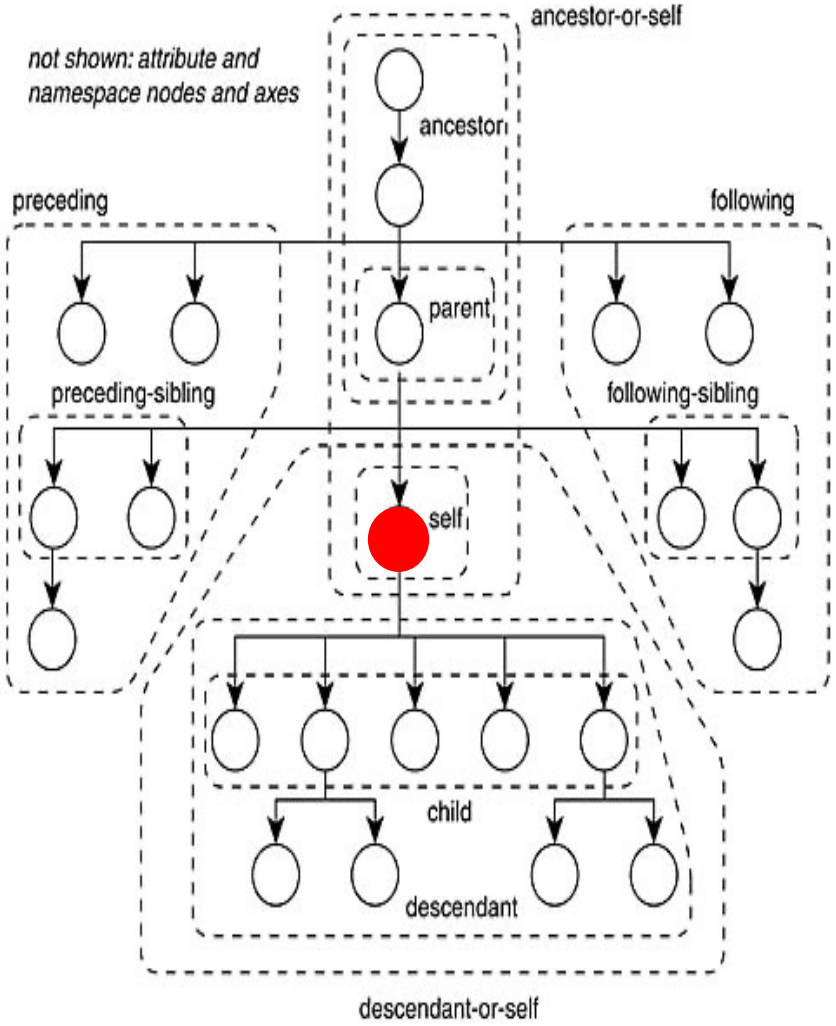
```
/html/body/h1[@id='publica']/following-sibling::h1
```

- Todos los H1 hermanos siguientes del H1 “publica”

```
//h1[@id='publica']/following-sibling::*
```

- Tomando la sección con **header** “Journals”, se busca la imagen contenida en el segundo párrafo de la siguiente sección

```
//header[@id='Journals']/following-sibling::p[2]/img
```



Introducción a la Extracción de Datos de la Web: *Web Scraping*

Código de conducta

UCO ONLINE

Ataque DoS (*Denial of Service*)

- *Scraping* implica consultar un sitio web repetidamente y/o acceder a un número potencialmente grande de páginas
  - Implica solicitudes al servidor web que aloja el sitio, que tendrá que procesar para enviar una respuesta
  - Cada petición consume recursos del servidor, durante los cuales no estará haciendo otra cosa
  - Si enviamos demasiadas peticiones en un corto espacio de tiempo, podemos impedir el funcionamiento "normal", o incluso hacer que el servidor se quede sin recursos y se caiga
- De hecho, esta es una forma tan eficaz de interrumpir un sitio web que los hackers suelen hacerlo a propósito [ataque de denegación de servicio (DoS)]
  - Los servidores web modernos incluyen medidas para evitar ese uso ilegítimo de sus recursos
  - Un "web scraper", incluso uno con fines legítimos, puede mostrar un comportamiento similar
  - Podemos ser bloqueados (baneados)

Ataque DoS (*Denial of Service*)

- Hay frameworks, como Scrapy, que incluyen medidas para evitar que el código parezca lanzar un ataque DoS a un sitio web:
  - Insertan retardos aleatorio entre las solicitudes individuales (por defecto en Scrapy)
  - Uso de proxies
- Es una buena práctica limitar el número de páginas que estamos rastreando durante la codificación y pruebas del código (p.ej. 5) – a poder ser, de servidores conocidos
- Es recomendable limitar las peticiones a un dominio en particular, utilizando la propiedad **`allowed_domains`** de Scrapy
- Los servidores web tienden a protegerse de ataques DoS, por lo que es necesario tomar medidas, limitando los riesgos de causar

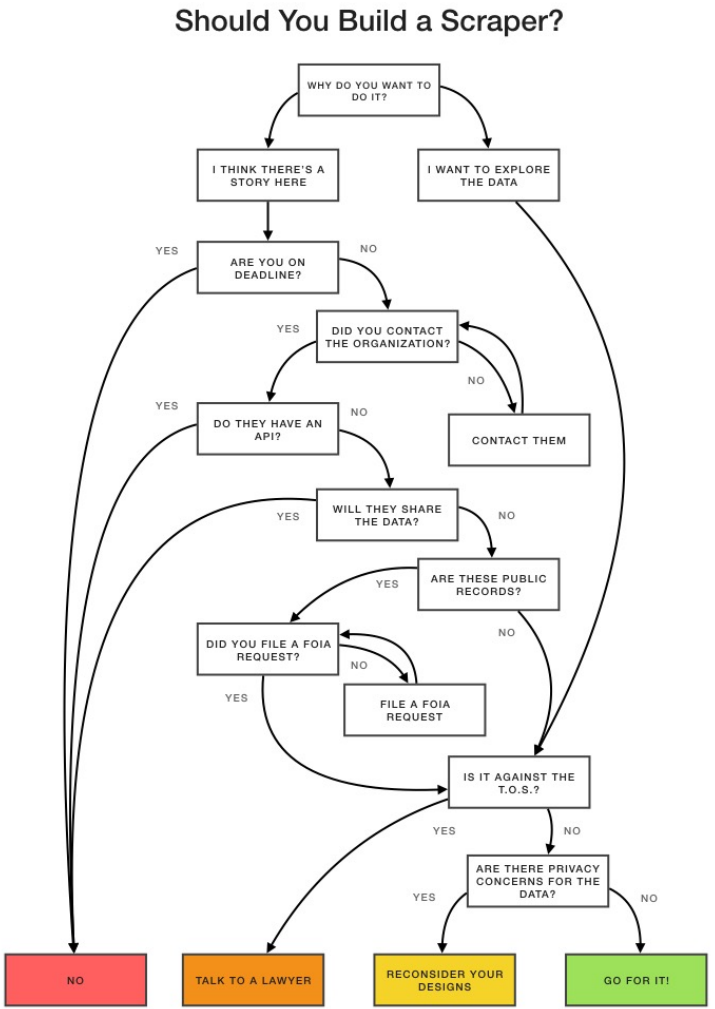
Ataque DoS (*Denial of Service*)

- En determinadas circunstancias, *web scraping* puede ser ilegal
  - Hay que estudiar los términos y condiciones del sitio web para comprobar si prohíben específicamente descargar y copiar su contenido
- En la práctica, el *web scraping* es una práctica tolerada, siempre que no se interrumpa el uso "regular" del servidor
  - En cierto sentido, el *web scraping* no difiere del uso de un navegador para visitar una página web, ya que equivale a utilizar un programa informático para acceder a datos disponibles públicamente
- En general, si los datos están disponibles públicamente, entonces está bien el *scraping* siempre que no se publiquen posteriormente violando los derechos de autor
  - La mayoría de legislaciones sobre derechos de autor reconocen casos en los que la reutilización de alguna información protegida por derechos de autor, en un formato agregado o derivado se considera "uso justo" (p.ej. uso privado no compartido)

Código de conducta

1. **Preguntar amablemente**. Si el proyecto requiere datos de una organización concreta, se puede preguntar directamente si pueden proporcionarte lo que buscas. Con suerte, los tendrán en formato estructurado
2. **No descargar copias de documentos que no son públicos**, p.ej. descargar el PDF de artículos de la editorial Elsevier aprovechando la VPN de la UCO
3. **Comprobar la legislación local**. Por ejemplo, algunos países tienen leyes que protegen la información personal (email, teléfono), incluso de sitios web de acceso público (p.ej. Australia)
4. **No compartir contenidos descargados de forma ilegal**. El *scraping* con fines personales suele ser aceptable ("uso justo") pero compartir los datos sin tener derecho a hacerlo es ilegal
5. **Compartir lo que se pueda**. Si los datos obtenidos son de dominio público o se tiene permiso para compartirlos, publicarlos para reutilizarlos es legal (p.ej. Github). Citar las fuentes
6. **No romper los servidores**. Precaución con los *scrapers* recursivos, es decir, los que siguen hipervínculos → ajustar la configuración de forma conservadora
7. **Publicar los datos propios de forma reutilizable**. No obligues a otros a escribir sus propios *scrapers*

¿Debemos desarrollar un scraper?



https://www.storybench.org/to-scrape-or-not-to-scrape-the-technical-and-ethical-challenges-of-collecting-data-off-the-web/

¡Gracias!

UCO ONLINE