Minería de Textos

Introducción a Scraping

Maestría en Data Mining / UBA

Esquema de la Clase

- Cómo obtener datos de internet
- Estructura de un scraper
- Qué es el código html
- Como parsear código html

Esquema de la Clase

- Cómo obtener datos de internet
- Estructura de un scraper
- Qué es el código html
- Como parsear código html

Cómo Obtener Datos de Internet

Antes de pensar cómo vamos a recorrer automáticamente páginas de internet, vamos a ver cómo bajar datos cuando ya sabemos de qué página queremos bajarlos.

Python 3 tiene una amplia gama de librerías para hacer esta tarea, la de más bajo nivel es "`socket`" (en donde uno puede manipular prácticamente todo lo referido al intercambio de datos entre nuestra computadora a través de redes).

Nosotros vamos a usar un librería de un nivel más alto llamada "`urllib`".

Cómo Obtener Datos de Internet

"urllib" es una librería que permite leer páginas de internet a través de Python de manera simple.

A su vez da cierta flexibilidad en lo referido a la información que enviamos al servidor (por ejemplo, modificar headers).

Cómo Obtener Datos de Internet

La forma más simple de bajar datos de una página es usar el método "urlopen". Este permite abrir un "url" del mismo modo que Python abre un archivo.

Para hacer, prueben el siguiente código:

```
import urllib.request

url = "http://www.lanacion.com.ar/1729342"

url_handler = urllib.request.urlopen(url)

url_handler
type(url_handler)
dir(url_handler)
url_handler.geturl()
```

Cómo Obtener Datos de Internet

El objeto devuelto es muy similar a un archivo abierto con "open". Sobre él se pueden usar los métodos "read", "readline" y "readlines" (también es iterable, de modo que se puede usar en un "for"):

¿Qué devuelve "url_handler.read()"?

Generalmente nos va a devolver el código html de una página, pero otras veces puede devolver otras cosas, como por ejemplo un archivo ".csv" o una imagen.

Cómo Obtener Datos de Internet

Vemos la información que había en la página que bajamos:

```
raw = url_handler.read()

print(raw)
print(type(raw))
raw = raw.decode("utf-8")
print(type(raw))
```

Cómo Obtener Datos de Internet

¿Por qué la misma página si la vemos en un celular o en una computadora a veces es diferente?

¿Por qué incluso si la vemos en una computadora difiere si la vemos en Firefox, Chrome o Internet Explorer?

Cómo Obtener Datos de Internet

¿Por qué la misma página si la vemos en un celular o en una computadora a veces es diferente?

¿Por qué incluso si la vemos en una computadora difiere si la vemos en Firefox, Chrome o Internet Explorer?

Cuando un dispositivo hace un pedido a un servidor, antes envía lo que se llama un "*header*" que contiene información del pedido y de quién lo está haciendo.

Uno de los campos de ese pedido es "*user-agent*" que indica qué tipo de navegador es el que está haciendo el pedido.

Cómo Obtener Datos de Internet

Por defecto "urllib" usa como "user-agent" el identificador: "Python-urllib/x.y".

Por eso se suele modificar el "user agent" del header para simular ser un navegador más tradicional. Con "urllib" esto se puede hacer fácilmente. Antes hay que armar un "request" en donde se detalle el "user agent", por ejemplo:

```
user_agent = 'Mozilla/5.0 (Windows NT 6.1; Win64; x64)'
headers = {'User-Agent': user_agent}

req = urllib.request.Request(url, headers=headers)

with urllib.request.urlopen(req) as response:
    raw = response.read()
```

Cómo Obtener Datos de Internet

Los url usan un encoding especial (por ejemplo el especio se suele escribir como "%20"). Para transformar texto a este encoding se usa el método "`quote`" de "`urllib.parse`".

Para hacer, prueben el siguiente código:

```
import urllib.parse

urllib.parse.quote('/El Niño/')

urllib.parse.quote('https://docs.python.org/El Niño/', safe=':/')
```

Esquema de la Clase

- Cómo obtener datos de internet
- Estructura de un scraper
- Qué es el código html
- Como parsear código html

Estructura de un Scraper

Un scraper es un programa cuyo objetivo es extraer automáticamente información de internet, usualmente simulando ser un humano.

Este es un muy buen momento para aclarar que con estas técnicas hay que ser precavido y que es muy común que un scraper viole términos de uso de una página.

Estructura de un Scraper

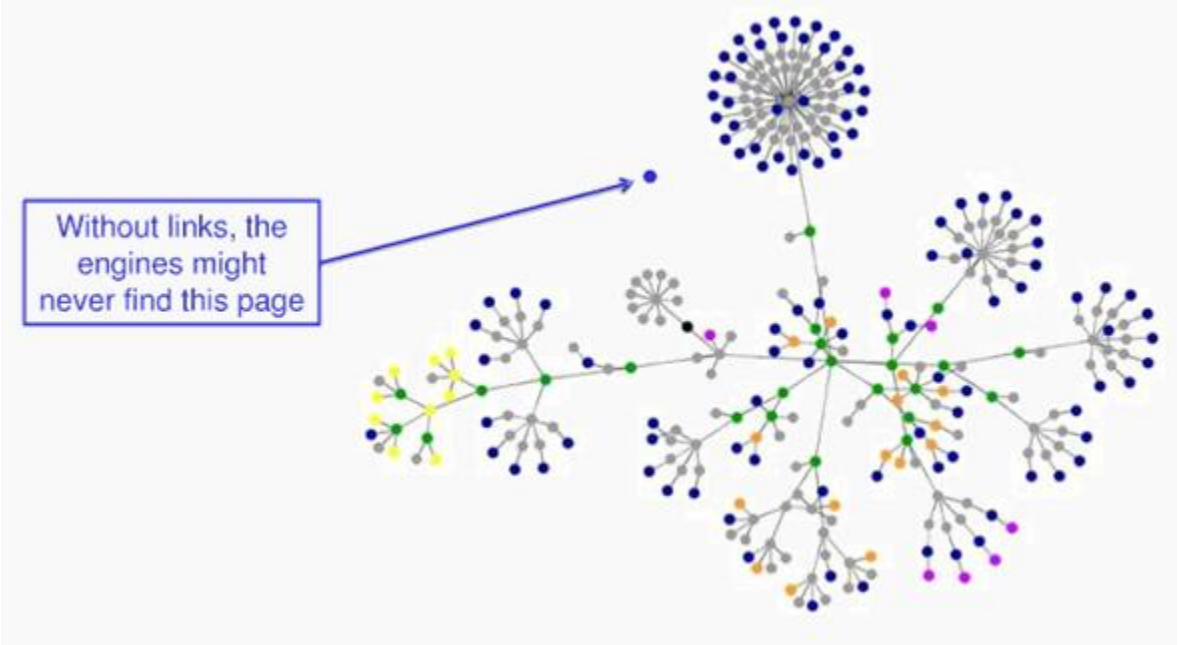
Nos planteemos un objetivo: bajar datos de artículos de http://es.wikipedia.org.

Para eso vamos a diseñar un programa que lee el html de una página de Wikipedia y después hace lo mismo para aquellas páginas para las que hay un link (a La Nación) en el artículo leído.

De este modo vamos aprovechar los links de una página y navegar a través de ellos.

Estructura de un Scraper

En términos más formales vamos "navegar" a través de un grafo dirigido, cuyos "nodos" son páginas de internet (con datos) y cuyos "vértices" son los "links" a otras páginas.

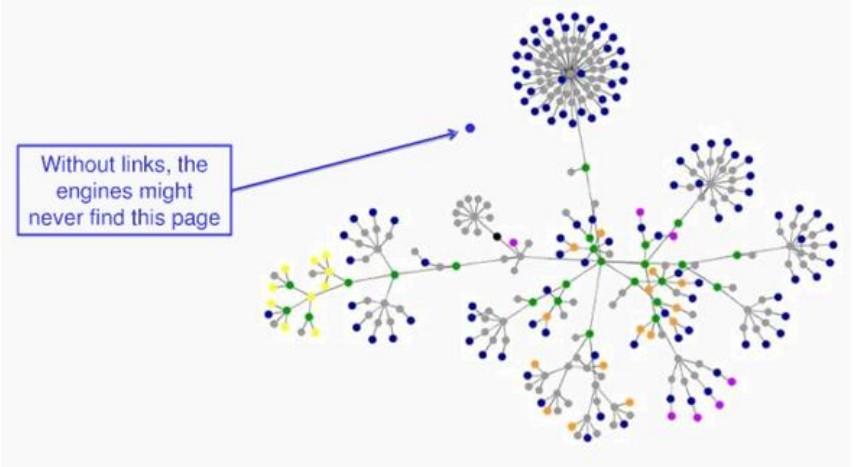


Estructura de un Scraper

¿Por dónde empezamos?

Las páginas con las que se comienza se llaman "seeds" (semillas), y puede ser una o más de una página.

Diferentes semillas pueden dar diferentes resultados. Noten que si ninguna página inicial tiene al menos un link, no vamos a navegar nada.

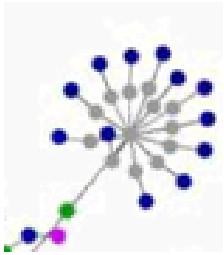


Estructura de un Scraper

¿Cuándo paramos de "scrapear"?

Si no definimos cuando parar, es posible que nuestra programa intente guardar todo internet!

Nosotros vamos a definir un criterio de parada y va a tener que ver con la "profundidad máxima": las páginas seed van a tener profundidad 0, las paginas que surgen de las seeds tendrán profundidad 1, las que surgen de estas 2, y así….

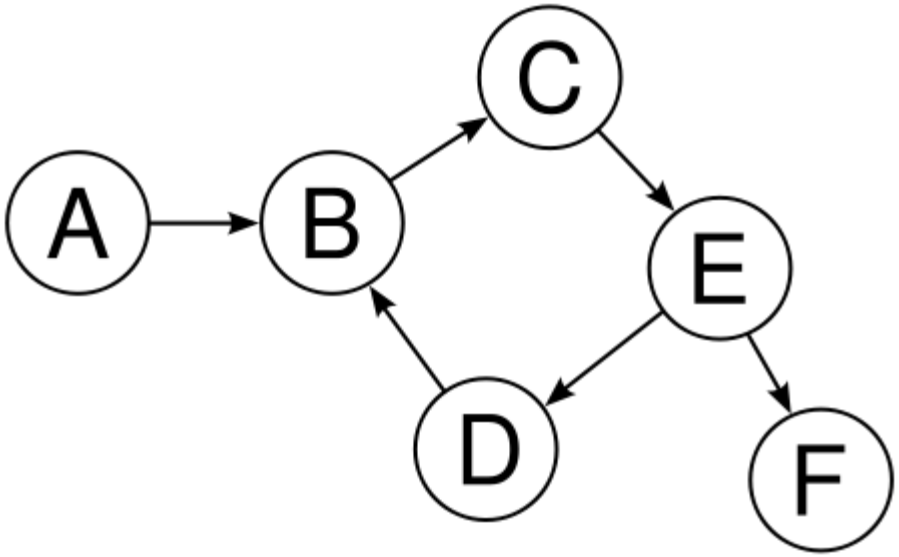


Nosotros vamos a definir de antemano una profundidad a máxima a visitar.

Estructura de un Scraper

¿Qué potencial problema ven?

(por el momento supongamos que no definimos una profundidad máxima)



¿Cómo podemos evitarlo?

Estructura de un Scraper

Para poder implementar estas ideas vamos a usar un diccionario, un conjunto y una lista de tuplas.

- En el diccionario usaremos como clave los url y como valor el código html. Será nuestro resultados. A su vez, las claves nos servirán para mantener registro de qué páginas ya fueron visitadas.

- En la lista vamos a guardar tuplas que indiquen las páginas que nos restan scrapear y su profundidad (cuando comienza el programa esta lista sólo tendrá las páginas seed con un 0).

Estructura de un Scraper

La notebook "crawler_wiki.ipynb" tiene tres funciones implementadas:

- gets_html_wikipedia: toma un url y baja el código html de la página.
- gets_n_wiki_links: toma una página de Wikipedia bajada y devuelve los primeros "n_links" links a otras páginas de Wikipedia.
- scraper: implementa el scraper.

Veamos qué hace y cómo funciona.

Esquema de la Clase

- Cómo obtener datos de internet
- Estructura de un scraper
- Qué es el código html
- Como parsear código html

Qué es el Código HTML

Hasta ahora usamos "`urllib`" para hacer algún pedido a un servidor y recibir una respuesta.

Mencionamos que el contenido que devuelve generalmente es código html, pero no ahondamos bien en qué es esto.

Ahora vamos a ver mejor cómo trabajar con código html.

Qué es el Código HTML

El código html (por HyperText Markup Language) es un lenguaje con etiquetas que es usado para crear y renderizar páginas web.

El código html contiene etiquetas (tags) que se encuentran rodeadas por los signos "<" y ">" (por ejemplo " ") y generalmente, aunque no necesariamente, están de a pares para indicar comienzo y final (por ejemplo " " y " "), aunque el cierre no es obligatorio.

Un navegador puede "entender" estas etiquetas y transformaras en representaciones visuales que pasan a dar forma a una página web.

Qué es el Código HTML

En términos generales las etiquetas suelen tener la siguiente estructura:

```
<tag attribute1="value1" attribute2="value2" … >content</tag>
```

Dentro de una etiqueta puede haber otras etiquetas, de modo que las páginas web suelen tener una estructura de árbol de etiquetas.

Nosotros vamos a poder aprovechar para extraer información de ellas.

Qué es el Código HTML

Copien el siguiente código en un archivo de texto, guárdenlo con extensión "html" y abránlo con algún navegador.

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8"/>
    <title>Este es el título de la página</title>
  </head>
  <body>
    <p>Hola mundo!</p>
  </body>
</html>
```

Qué es el Código HTML

Tags comunes que vamos a ver son:

"<a>": crea texto que contiene un link.

" ": define divisiones de la página.

"<p>": crea un párrafo

"<img>": inserta una imagen en el documento.

"<meta>": contiene meta información sobre la página.

"<li>": define un ítem de una lista.

Qué es el Código HTML

Los atributos asociados a los tags son muy útiles. Generalmente están puestos por cuestiones de formato y organización de la página. Nosotros los vamos a aprovechar para extraer información.

Por ejemplo vean el siguiente tag "li" de una página de https://sfbay.craigslist.org/d/all-housing-wanted/search/hsw:

```
<li class="result-row" data-pid="6371849895">
   ...
</li>
```

El mismo tiene un atributo que se llama "`data-pid`", que es el número que Craiglist asigna internamente a un post.

Qué es el Código HTML

Ustedes pueden ver tanto en Chrome como en Firefox el código html de una página. Para ello deben apretar "ctrl+u" o pueden inspeccionarlo interactivamente con la herramienta "inspect element" disponible tanto en Chrome como en Firefox.

¡Nosotros no vamos a hacer un curso de html! Pero tenemos que entender que hay una estructura en las páginas y que no tiene sentido no aprovecharla (por ejemplo: trabajarla como si fuese texto ordinario).

Esquema de la Clase

- Cómo obtener datos de internet
- Estructura de un scraper
- Qué es el código html
- Como parsear código html

Como Parsear código HTML

Hacer que una computadora interprete la estructura inherente a un texto se le llama hacer "parsing".

Python trae muchas librerías que hacen parsing automático de código html y luego permiten hacer búsquedas en el código aprovechando la estructura del mismo. Nosotros vamos a ver una librería que se llama "bs4" (BeautifulSoup versión 4).

Por convención la vamos a importar ejecutando:

```
from bs4 import BeautifulSoup
```

Como Parsear código HTML

BeautifulSoup va a hacer todo el trabajo de "entender" la estructura del código html por nosotros, pero no va a abrir la página por nosotros.

Para obtener el código html vamos a usar `urllib` y vamos a pasar el código html a BeautifulSoup para que lo "entienda" (haga el parsing).

Para hacer, prueben el siguiente código:

```
soup = BeautifulSoup(raw, "lxml")
print(soup)
print(type(soup))
print(dir(soup))
```

Como Parsear código HTML

Una vez que ya tenemos el código html parseado, nuestro objetivo va a ser realizar búsquedas sobre el mismo para poder de esta manera extraer información.

La forma más simple de extraer información es usar el nombre del tag que queremos.

Prueben el siguiente código

```
print(soup.title)
print(soup.a)
print(soup.figure)
print(type(soup.title))
print(dir(soup.title))
```

Como Parsear código HTML

Sin embargo esto nos devuelve la primera aparición de cada uno de los tags, nosotros podemos estar interesados en obtener todas las apariciones de un tag, para eso existe el método "`find_all`" (ya vamos a ver con más detalle este método).

Prueben el siguiente código.

```
print(soup.find_all('a'))
print(soup.find_all('img'))
```

Como Parsear código HTML

Una vez que uno encuentra un elemento, uno le puede pedir diferentes propiedades.

Prueben el siguiente código:

```
elemento = soup.find_all('a')[70]
print(type(elemento))
print(elemento.name)
print(elemento.string)
print(elemento.attrs)
print(type(elemento.attrs))
print(elemento["href"])
```

Como Parsear código HTML

Uno puede pedir a "find_all" que devuelva tags que tienen un atributo con determinado valor.

Para eso se tiene que pasar el atributo a analizar y el valor que se busca (el valor, entre otras cosas, puede ser una expresión regular compilada). Si el atributo que se busca es "class" se debe escribir "class_" (porque class es una palabra reservada en Python)

Para hacer, prueben el siguiente código:

```
print(soup.find_all(id="pie"))
print(soup.find_all(id="pie-facebook", class_="icon-facebook"))
print(soup.find_all(class_="icon-facebook"))
print(soup.find_all("a", class_="icon-facebook"))
print(soup.find_all("button", class_="icon-facebook"))
```

Como Parsear código HTML

Para ahondar más sobre find_all pueden leer la documentación de bs4 referida a este método:

https://www.crummy.com/software/BeautifulSoup/bs4/doc/#searching-the-tree

BeautifulSoup es una herramienta muuuy potente. De hecho tiene muchas más potencialidades que las que vimos hasta ahora, sin embargo con lo que ya vimos podemos hacer mucho.

Como Parsear código HTML

Probemos sacar todos los links que se encuentren en el texto del articulo de La Nación. Inspeccionando el código html, podemos ver que el texto principal del artículo se encuentra dentro de un tag "section" que tiene un atributo "itemprop" con valor "articleBody". Usemos es patrón:

```
cuerpo = soup.find_all("section", id="cuerpo")[0]
```

Luego podemos ver que el texto del artículo se encuentra dentro de tags "p". Usemos ese patrón:

```
texto_art = cuerpo.find_all("p")
```

Después dentro de cada párrafo podemos buscar los links que estos tienen:

```
for p in texto_art:
    print(p.find_all("a", class_="link"))
```

Como Parsear código HTML

Lo óptimo sería juntar todas estas ideas en una función y hacerla un poco más eficiente, como la siguiente.

```
def quita_links_lanacion(soup):
    cuerpo = soup.find_all("section", id="cuerpo")[0]
    texto_art = cuerpo.find_all("p")
    out = []
    for p in texto_art:
        for l in p.find_all("a", class_="link"):
            # Sólo links de la nación
            if l["href"].startswith("http://www.lanacion.com.ar/"):
                out.append(l["href"])
    return(out)

quita_links_lanacion(soup)
```

Como Parsear código HTML

Scrapear es un arte. Nosotros acá estamos viendo la introducción, pero en base a lo que uno deba bajar, la tarea se puede complejizar mucho.

Como Parsear código HTML

Para hacer:

Modifiquen el scraper de Wikipedia de modo que se transforme en un scraper de La Nación.