Aprende Machine Learning

antes de que sea demasiado tarde

GENERAL, PRÁCTICA

Ejemplo Web Scraping en Python: IBEX35® la Bolsa de Madrid

En este artículo aprenderemos a utilizar la librería BeatifulSoap de Python para obtener contenidos de páginas webs de manera automática.

En internet encontramos de todo: artículos, noticias, estadísticas e información útil (¿e inútil?), pero ¿cómo la extraemos? No siempre se encuentra en forma de descarga ó puede haber información repartida en multiples dominios, ó puede que necesitemos información histórica, de webs que cambian con el tiempo.

Para poder generar nuestros propios archivos con los datos que nos interesan y de manera automática es que utilizaremos la técnica de **WebScraping**.

Contenidos:

- Requerimientos para WebScraping
- Lo básico de HTML y CSS que debes saber
- Inspeccionar manualmente una página web
- Al código! **Obtener el valor actual del IBEX35® de la Bolsa de Madrid**
- Exportar a archivo csv (y poder abrir en Excel)
- Otros casos frecuentes de «rascar la web»

> Puedes ver y descargar el código python completo de este artículo desde GitHub haciendo click aquí

Requerimientos

Para poder usar esta técnica hay diversas librerías, pero utilizaremos una muy popular llamada Beautiful Soap. Como siempre, te recomiendo tener instalado el ambiente de desarrollo con Anaconda (se explica cómo instalar en este artículo) que ya trae incluida la librería. Si no, lo puedes instalar a mano, desde línea de comandos con

```
pip install BeautifulSoup4
```

```
pip install BeautifulSoup4
pip install requests
```

Si bien utilizaremos una Jupyter Notebook para el código Python 3, podríamos ejecutar un archivo de texto plano «.py» desde nuestra Terminal.

Conocimientos básicos de HTML y CSS

Daré por sentados conocimientos de html y css. ¿Por qué? **Las páginas webs están hechas con HTML** y deberemos indicarle a nuestro «bot-spider» de qué etiquetas ó campos, deseamos extraer el contenido.

Repaso -MUY- mínimo de HTML es:

```
<html>
<head><title>Titulo de pagina</title>
</head>
<body>
    <p> Soy un parrafo</p>
    <div>Soy un texto en un DIV</div>
    <table><tr><td>soy una celda dentro de una tabla</td></tr>
    </table>
</body>
</html>
```

Aqui Vemos las etiquetas básicas de HTML, es decir las de inicio y cierre y dentro de body el contenido de la página. Como ejemplo vemos un párrafo «p», un «div» y una tabla.

¿Y porqué CSS? en realidad no necesitamos estrictamente saber CSS, pero sí sus selectores, puesto que nos pueden ser de mucha ayuda. Lo básico para comprender selectores, usando este bloque de ejemplo es:

```
<html>
    <head>/head>
    <body>
    <div class="contenedor">
        <div id="123" name="bloque_bienvenida" class="verde">
        Bienvenido a mi web
        </div>
    </div>
    </body>
</html>
```

Para poder seleccionar el texto «*Bienvenido a mi web*«, tenemos diversas formas:

- la más directa será si la etiqueta tiene un **atributo id que es único** en el ejemplo «123»
- Podríamos buscar los nodos de tipo div, pero podría haber muchos y deberemos filtrarlos.
- Podemos filtrar un div con el atributo name = «bloque_bienvenida».
- Podemos **buscar por clase CSS**, en el ejemplo «verde».
- Muchas veces se combinan selectores, por ejemplo: dentro de la clase «contenedor», la clase «verde». O decir: «traer un div con la clase verde»

La librería de Beautiful Soup nos permite buscar dentro de los nodos del árbol de la página web, también conocido como DOM. Al final del artículo veremos como obtener el texto «Bienvenido a mi web» con diversos selectores (y en la Jupyter Notebook de Github)

texto «Bienvenido a mi web» con diversos selectores (y en la jupyter Notebook de Github).

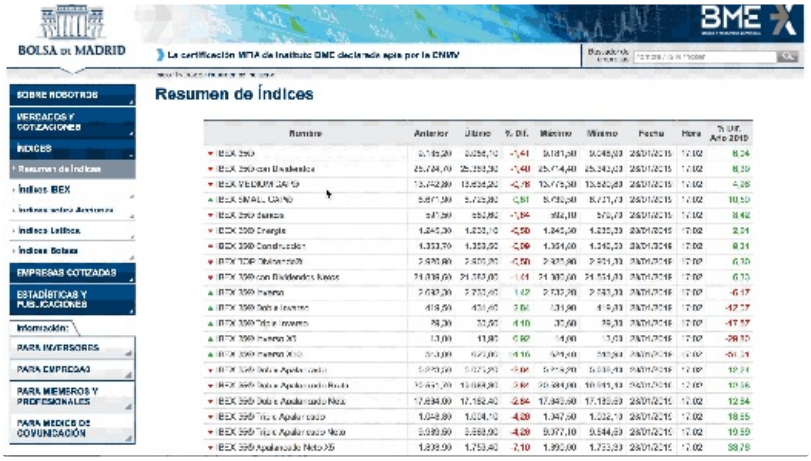
Inspección Manual de la web

Esta es la web de la bolsa de Madrid, en donde nos interesa obtener el Indice del IBEX35®

Para el ejemplo inspeccionaremos la web de la Bolsa de Madrid. **¿Qué es eso de inspeccionar?** Bueno, los navegadores web «modernos» (Safari, Firefox, Chrome) cuentan con una opción que nos permite ver el código html completo de la página que estamos viendo.

Además existe una opción de «inspección del código» que nos permite ver el HTML, Javascript, CSS y la web al mismo tiempo. Con ello buscaremos la manera de extraer el texto que nos interesa, si buscamos por id, por algún atributo, clase ó nodos.

Por lo general podemos inspeccionar haciendo click con el botón derecho del mouse sobre el área que nos interesa. Veamos cómo hacerlo con un gif animado 🙂



Al hacer clic derecho, aparece la opción de **Inspeccionar Elemento**. Así podemos ver las entrañas de la web en la que estamos navegando y pensar la mejor opción para extraer contenidos.

En nuestro caso nos interesa obtener el valor de la fila con nombre IBEX35® y el valor de la columna «último».

Código Python – Arranquemos!

Veamos en código cómo haremos para acceder a esa porción de texto.

Primero importamos las librerías Python que utilizaremos:

```
import requests
from bs4 import BeautifulSoup
import csv
```

```
from datetime import datetime
```

Indicamos la ruta de la web que deseamos acceder:

```
# indicar la ruta
url_page = 'http://www.bolsamadrid.es/esp/aspx/Indices/Resumen.aspx'
```

Y ahora haremos el request a esa ruta y procesaremos el HTML mediante un objeto de tipo BeautifulSoap:

```
# tarda 480 milisegundos
page = requests.get(url_page).text
soup = BeautifulSoup(page, "lxml")
```

Bien, ahora toca pensar la estrategia para acceder al valor. En nuestro caso nos interesa primero acceder a la tabla, y de allí a sus celdas. Por suerte la tabla tiene un id único!

```
# Obtenemos la tabla por un ID específico
tabla = soup.find('table', attrs={'id': 'ctl00_Contenido_tblÍndices'})
tabla
```

Aqui vemos el id de la tabla marcado en amarillo.
En rojo, se muestra la tercera celda de la primer fila a la que queremos acceder.

Bien, ahora dentro de la tabla y siendo que en este caso no tenemos un acceso directo a las celdas por ids únicos ni por clases, sólo nos queda iterar... Entonces, accederemos a la primer fila y obtendremos de las celdas el nombre del índice y su valor:

NOTA: realmente es la segunda fila, pues hay un encabezado, por eso usamos el índice 1 y no el cero.

```
name=""
price=""
nroFila=0
for fila in tabla.find_all("tr"):
    if nroFila==1:
        nroCelda=0
        for celda in fila.find_all('td'):
            if nroCelda==0:
                name=celda.text
                print("Indice:", name)
            if nroCelda==2:
                price=celda.text
                print("Valor:", price)
            nroCelda=nroCelda+1
    nroFila=nroFila+1
```

Veremos cómo salida:

```
Indice: IBEX 35® <br>Valor: 9.185,20
```

Ya sólo nos queda guardar los datos para usar en el futuro.

Guardar CSV y ver en Excel

Vamos a suponer que ejecutaremos este script una vez al día, entonces lo que haremos es ir escribiendo una nueva línea al final del archivo cada vez.

```
# Abrimos el csv con append para que pueda agregar contenidos al final del archivo
with open('bolsa_ibex35.csv', 'a') as csv_file:
    writer = csv.writer(csv_file)
    writer.writerow([name, price, datetime.now()])
```

Finalmente obtenemos el archivo llamado «bolsa_ibex35.csv» listo para ser usado en nuestro proyecto 🙂

Podemos abrir el archivo csv en Excel, LibreOffice, SpreadSheets ó como archivo de texto plano.

Otros ejemplos útiles de Webscaping:

Veamos otros ejemplos de uso de BeatifulSoap para extraer contenidos con python.

Usemos el bloque **de ejemplo que usé antes** e intentemos extraer el texto «*Bienvenido a mi web*» de diversas maneras:

```
#Obtener por ID:
elTexto = soup.find('div', attrs={'id': '123'}).getText()
print(elTexto)
#Obtener por Clase CSS:
elTexto = soup.find('div', attrs={'class': 'verde'}).getText()
print(elTexto)
#Obtener dentro de otra etiqueta anidado:
elTexto = next(soup.div.children).getText() #con next obtiene primer "hijo"
print(elTexto)
```

Obtener los enlaces de una página web

Otro caso práctico que nos suele ocurrir es querer colectar los enlaces de una página web. Para ello, obtenemos las etiquetas «A» e iteramos obteniendo el atributo HREF que es donde se encuentran las «nuevas rutas», con posibilidad de hacer un nuevo request a cada una y extraer sus contenidos.

```
url_page = 'https://www.lifeder.com/cientificos-famosos/'
page = requests.get(url_page).text
soup = BeautifulSoup(page, "lxml")
contenido = soup.find('div', attrs={'class': 'td-post-content'})
items = contenido.find_all('a')
for item in items:
    print(item['href'])
```

En el archivo Jupyter Notebook de mi cuenta de Github se ven estos ejemplos (y alguno más).

Conclusiones, repaso y código

Ahora sabemos cómo afrontar el proceso de obtener información de cualquier página web. Resumiendo el procedimiento básico que seguimos es:

1. Cargar la página en el navegador
2. Inspeccionar e investigar el HTML
3. En Python: importar las librerías

4. Obtener la página, parsear el contenido con BeautifulSoap
5. Obtner el «trozo» de contenido que buscamos
    - Mediante ID
    - Mediante Etiqueta
    - Mediante Clases CSS
    - Otros Atributos
6. Guardamos los datos en csv

Repasando, cuando ya tenemos el contenido en un objeto «soap», solemos utilizar los métodos find() ó para múltiples etiquetas el find_all().

Si combinamos un script para webscraping, como en el ejemplo para capturar valores de la bolsa con el cron del sistema (ó con algún tipo de «repetidor de tareas del sistema») que nos permita ejecutar nuestro código cada «x» tiempo, podremos generar un valioso archivo de información muy a medida de lo que necesitamos.

Otro ejemplo «clásico» es el de la obtención automática de los resultados de partidos de fútbol y en el código de este ejemplo en Github, encontrarás cómo hacerlo.

> Obtener el Jupyter Notebook con código Python con este y más ejemplos de WebScaping

Si bien este artículo no es estrictamente sobre Machine Learning, me pareció bueno comentarlo pues he utilizado técnicas de Webscraping en ejercicios anteriores (como en el de Procesamiento del Lenguaje Natural) pasando por alto la explicación de esta porción del código. Además es un recurso utilizado frecuentemente para trabajar y para hacer pequeños experimentos con datos.

> Ya estás listo para la Fase EDA!!!

Suscripción al Blog

Recibe los próximos artículos sobre Redes Neuronales y Deep Learning, Herramientas para Big Data y Data Science y ejercicios en código Python cada 15 días en tu bandeja de entrada.

Email: Your email address here

ENVIAR

Comparte el artículo:

Relacionado

Instalar ambiente de Desarrollo Python Anaconda para Aprendizaje Automático

Principales Algoritmos usados en Machine Learning

¿Machine Learning en la Nube? Google Colaboratory con GPU!

Big Data código Data Science ejemplo Ejercicio Python

NLP: ANALIZAMOS LOS CUENTOS DE HERNAN CASCIARI

¿MACHINE LEARNING EN LA NUBE? GOOGLE COLABORATORY CON GPU!

11 comments

Oscar Riojas · enero 29

Muchas gracias buen ejemplo y muy útil

Responder

Na8 · enero 29

Hola Oscar Riojas!, gracias por participar en el blog! seguimos en contacto, saludos

Responder

Ignacio Álvarez · febrero 3

Realmente muy interesante el artículo.
Un saludo

Responder

Na8 · febrero 3

Hola Ignacio, muchas gracias por el comentario!
Saludos

Responder

Elias · marzo 14

En Chrome Safari y imagino otros, puedes dar ya dentro del inspector de elementos, copy PATH SELECTOR, eso te arma la ruta completa de un selector como el que usaría una libreria de DOM como jQuery o cheerio (NodeJs)

Llegue aquí por tema de ML, gran blog,

Responder

Na8 · marzo 14

Muy buen consejo Elías! Gracias por colaborar! Saludos

Responder

pablo rosa · agosto 24

Muchas gracias por tus aportes! realmente muy valiosos y concretos. Saludos desde córdoba!

Responder

Na8 · agosto 27

Hola Pablo, gracias por escribir! me alegra saber que ayudan! Saludos a Córdoba 🙂

Responder

Gordon Freeman · octubre 8

Mucha gracias , esto me fue de gran ayuda

Responder

Jimmy · octubre 24

me sale este error:

AttributeError Traceback (most recent call last)
in ()
3 soup = BeautifulSoup(page, “lxml”)
4 contenido = soup.find(‘div’, attrs={‘class’: ‘td-post-content’})
—-> 5 items = contenido.find_all(‘a’)
6 for item in items:
7 print(item[‘href’])

AttributeError: ‘NoneType’ object has no attribute ‘find_all’

Responder

Miguel · 2 Hours Ago

Muchas gracias por esta maravillosa página, es muy útil y estoy aprendiendo mucho. Ánimo y te deseo todo lo mejor.

Responder

Deja un comentario

Deja un comentario

Introduce aquí tu comentario...

Visita nuestra Guía de Aprendizaje

Buscar

Search ...

Contacto

Suscripción

Recibe los artículos de Aprende Machine Learning en tu casilla de correo. Cada 15 días y sin Spam!

Email: Your email address here

ENVIAR