Collecte Automatisée de Données Web

Séance 2 de cours/TD

IUT SD Dole

Objectifs:

- Introduction à la collecte de données Web sans API...

Webscraping : Qu'est-ce que c'est ? [Cours]

Le webscraping désigne les techniques d'extraction du contenu des sites internet. C'est une pratique très utile pour toute personne souhaitant travailler sur des informations disponibles en ligne, mais pas nécessairement accessible par une API.
L'idée générale va être de charger le code source de la page web puis d'aller chercher dans celui-ci les informations souhaitées.

Commençons par un rappel rapide de la structure d'un site web : Un site Web est un ensemble de pages codées en HTML qui permet de décrire à la fois le contenu et la forme d'une page Web.

Sur une page web, vous trouverez toujours à coup sûr des éléments comme <head>, <title>, etc. Il s'agit des codes qui vous permettent de structurer le contenu d'une page HTML et qui s'appellent des balises. Citons, par exemple, les balises , <h1>, <h2>, <h3>, ou . Le symbole < > est une balise : il sert à indiquer le début d'une partie. Le symbole </ > indique la fin de cette partie. La plupart des balises vont par paires, avec une balise ouvrante et une balise fermante (par exemple et).

Pour récupérer correctement les informations d'un site internet, il faut pouvoir comprendre sa structure et donc son code HTML. Les fonctions python qui servent au scraping sont principalement construites pour vous permettre de naviguer entre les balises.

Webscraping en python [Cours]

De manière assez similaire à l'utilisation d'API, faire du webscrapping en python commence par faire la requête de la page web.

La différence principale est qu'à ce moment-la, plutôt que de récupérer des données pré-formatées dans un format json, xml ou csv, on récupère le code brut de la page web, qu'il va falloir explorer pour obtenir les données voulues.

Pour cela, on va se servir de la bibliothèque BeautifulSoup. (D'autres existent, notamment Selenium, qui vous permettra de faire du webscrapping sur les parties dynamiques des pages, sur lequel on reviendra plus bas). Son rôle va être de structurer la donnée récupérée afin d'accéder facilement à son contenu en filtrant sur les balises par exemple. Prenons un cas particulier. Nous allons essayer de récupérer le tableau de données de la page wikipédia suivante : https://fr.wikipedia.org/wiki/Liste_des_pays_par_population

On commence par regarder le code source de la page pour repérer le tableau. Vous le trouverez en ligne 481 dans une balise`table` . En y regardant de plus près, on notera que chaque ligne est délimitée par une balise `tr` , et chaque case à l'intérieur de la ligne par une balise `td`

```
import requests
import pandas as pd
import bs4

#On recupere le contenu brut de la page
url_api ="https://fr.wikipedia.org/wiki/Liste_des_pays_par_population"
req = requests.get(url_api)

# On utilise beautifulsoup pour formater le code source
page = bs4.BeautifulSoup(req.text, "html.parser")

#On recupere le tableau
tab = page.find("table")

#Puis on traite chaque ligne
res=[]
for line in tab.find_all("tr"):
    elems = line.find_all("td")
    resline=[]
    for elem in elems:
        resline=resline +[elem.text.strip()]
    res=res+[resline]

print(res)
```

la méthode `get_text` permet de récupérer le contenu textuel d'une balise en enlevant aussi toutes les balises internes.

Pour plus de détails sur les fonctionnalités de la bibliothèques BeautifulSoup, réferrez-vous à sa documentation : https://www.crummy.com/software/BeautifulSoup/bs4/doc/

Les Bonnes Pratiques de Webscraping [Cours]

Le webscraping est une activité qui peut causer un certain nombre de problèmes aux sites ciblés, ce qui en fait quelque chose d'assez mal vu en général. Pour faire du webscraping de manière polie et raisonnable, il y a quelques règles à suivre :

- Comme pour les API, assurez-vous de limiter vos chargements de pages au strict nécessaire.
- Insérez des pauses entre les différents appels pour ne pas surcharger les serveurs (En python, vous pourrez utilisez la méthode `sleep` de la bibliothèque `time`.
- Souvenez-vous qu'il est illégal de webscraper des données considérées comme personnelles, même si elles sont publiées sur un site.
- Respectez les souhaits des webmaster du site concernant ce qui est, ou non, webscrapable sur le site en question.

Pour ce dernier point, vous pourrez par exemple vérifier ce qui est indiqué sur le robots.txt du site web.

robots.txt est un fichier disponible normalement sur tout site web (à l'adresse http ://www.adressedusite.com/robots.txt) et indiquant ce qu'il est possible de faire sur le site. Initialement créé pour les robots de moteur de recherche, il est aussi utilisé pour indiquer les limites du webscraping.

Par exemple, l'extrait du fichier robots suivant :

```
User-agent: *
Disallow: /mandataire/*
Disallow: /coach/*
Disallow: /notreequipe/*

User-agent: bingbot
Crawl-delay: 1
```

indique que les pages du site dans le sous-dossier mandataire sont interdites à tout robot (et donc à tout webscraping), et que les robots de type bingbot doivent respecter un délai d'une seconde entre deux pages chargées.

Il est fortement conseillé de suivre ces recommandations, pour éviter d'être banni des sites en question.

Exercice 1 (,)
Afficher la liste des liens présents dans la page wikipedia proposée dans l'exemple. □

Exercice 2 (,)

En allant récupérer des données sur wikipedia, produisez un fichier contenant la liste des rois de france avec leur date de règne.

□

Exercice 3 (,)

En allant récupérer des données sur des sites de votre choix, proposez un programme permettant de vérifier automatiquement si deux acteurs donnés ont déjà joué ensemble.

□

Exercice 4 (,)

En allant récupérer des données sur trois sites d'info de votre choix, proposez un programme permettant de générer un fichier csv affichant les titres et les liens vers les infos du jour.

□

Exercice 5 (,)

Ecrire un programme qui fonctionne comme un dictionnaire (l'utilisateur entre un mot et le programme affiche les définitions du mot concerné). □

Webscraper des pages dynamiques [Cours]

Nous avons précédemment vu comment récupérer les données de pages statiques (où l'ensemble des données est présent dans le code html de la page à étudier). Nous allons maintenant voir comment récupérer les données pages dynamiques, contenant notamment du javascript. Nous allons pour cela utiliser la bibliothèque Selenium de python.

Cette bibliothèque permet de simuler une connection depuis un navigateur web, et de manipuler la page comme pourrait le faire un utilisateur. En particulier, il est possible pour le programme python de scroller pour charger plus de la page, de cliquer sur des boutons, ou de remplir des formulaires, notamment d'authentification.

Vous aurez besoin d'installer la bibliothèque selenium, mais aussi la bibliothèque webdriver_manager pour simplifier les choses.

Prenons un exemple :

```
# On commence par charger les bibliothÃ¨ques

from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
from selenium.webdriver.common.keys import Keys
from selenium.webdriver.common.by import By
from time import sleep


# On crÃ©er un simulateur de navigateur web, ici chrome
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install
    ()))


# On connecte le simulateur Ã  google
driver.get("https://www.google.com/")


# On detecte le bouton "accepter les cookies"
cookiebutton = driver.find_element(by=By.ID,value="L2AGLb")

# on clique sur ce bouton
cookiebutton.click()

# On patiente quelques secondes que la page se charge entiÃ¨rement
sleep(3)

# On detecte la barre de recherche de google
search = driver.find_element(by=By.NAME,value="q")


# on y entre le texte selenium, puis on appuie sur entree
search.send_keys("Selenium")
search.send_keys(Keys.ENTER)
sleep(3)

# on scrolle tout en bas de la page, pour charger la deuxiÃ¨me page de rÃ©sultats
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);"
    )
sleep(3)

# on detecte tous les elements html de balise h3 ( qui contiennent les titres des
    rÃ©sultats de recherche google
links2 = driver.find_elements(by=By.TAG_NAME, value="h3")

# on affiche les textes de ces resultats.
for x in (links2):
    print(x.text)
```

on a ainsi récupéré les résultats des deux premières pages google pour la recherche "selenium".

Exercice 6 (,)

Récupérez sur le site www.la-maronne-immobiliere.com l'ensemble des informations standard sur les biens immobiliers situés à moins de 50 km d'Aurillac. □