Web scraping

Alexandre Roulois (Université Paris Cité, LLF, CNRS)

Table of contents

Définition

Le *Web Scraping* est une technique d'exploration du Web visant à en extraire du contenu. Comme les pages Web sont un minimum structurées grâce à un langage de balisage, le HTML, il est possible de repérer dans le code source des segments de texte placés entre des marqueurs :

```
<a href="http://www.lemonde.fr">Le Monde.fr</a>
```

Dans l'exemple ci-dessus, les balises `<a>` et `</a>` encadrent le texte *Le Monde.fr*. Si notre ambition est d'étudier la sémantique des liens hypertextes, on peut facilement adresser une requête à un outil d'analyse du code HTML de la forme :

> Récupèrer tous les segments encadrés par les marqueurs de l'élément HTML `a`.

Pré-requis

Cette facilité est mise en œuvre par deux modules de Python qu'il est nécessaire d'importer :

- *urllib.request*
- *BeautifulSoup*

```
#| code-fold: true
import urllib.request
from bs4 import BeautifulSoup
```

Extraire du code HTML

Le premier module, *urllib.request*, est chargé de récupérer le document HTML à analyser. Il a juste besoin d'une adresse URL pour fonctionner, mais nous lui fournissons en prime, par politesse, un *User-agent*. Comme la visite du programme que nous construisons aura un impact sur la performance du site Web visité, il est de bon ton d'expliquer rapidement notre volonté et de laisser une adresse mail pour nous contacter au besoin. Cette convenance nous évitera peut-être un fichage sur un fichier `robots.txt`.

```
# URL
url = 'http://www.llf.cnrs.fr'

# additional headers
headers = { 'User-agent' : 'HyperText extractor (Alexandre Roulois)' }

# HTTP request
request = urllib.request.Request(url, headers=headers)

# load HTML document
with urllib.request.urlopen(request) as webpage:
    # get the html content
    html = webpage.read()
```

Un affichage du contenu de la variable `html` renverra la page au format `bytes`.

Analyser du code HTML

Le module *BeautifulSoup* permet d'extraire des données depuis un document structuré aux formats XML ou HTML. Il met à disposition un ensemble de méthodes et de stratégies pour les exploiter simplement.

La méthode `.find_all()` par exemple permet de rechercher toutes les occurrences d'une balise dans le document. Et, pour une occurrence donnée, la méthode `.get_text()` en affiche le contenu textuel.

Prenons une portion de code HTML issue d'un menu de navigation classique :

```
<nav>
  <a href="home.html" class="nav-link">Accueil</a>
  <a href="tools.html" class="nav-link">Outils</a>
</nav>
```

Plaçons ce menu dans une variable `html` :

```
html = b'<nav>\
  <a href="home.html" class="nav-link">Accueil</a>\
  <a href="tools.html" class="nav-link">Outils</a>\
</nav>'
```

Puis, pour ne récupérer que le contenu textuel des ancres (balises `<a>`), à savoir les textes « Accueil » et « Outils » :

```
# new instance of BeautifulSoup
soup = BeautifulSoup(html, 'html.parser')

# find all anchors in the HTML page
anchors = soup.find_all('a')

# for each anchor found…
for anchor in anchors:
  # … print the textual content
  print(anchor.get_text())
```

Constituer un corpus à partir du Web

Grâce à ces techniques, il devient très abordable avec Python de constituer son propre corpus à partir de données sur le Web. Dans notre démonstration, nous souhaitons récupérer les commentaires des spectateurs de films de science-fiction sur le site *Allociné*. Du point de vue légal, les droits demeurent ceux de la plateforme et même si au final nous concevons un nouveau produit intellectuel (un corpus de données), il nous est interdit de diffuser ces données.

Méthodologie

Pour réaliser notre objectif, nous mettons au point une méthodologie en sept étapes, représentée par le schéma ci-dessous :

1. cibler une page du site *Allociné* qui recense les films de SF de la décennie 2010-2019 ;

2. analyser le code HTML afin de récupérer, dans la liste des films, leurs identifiants uniques ;
3. enregistrer les identifiants dans un fichier plat ;
4. reconstruire les URLs de chaque film à partir des identifiants ;
5. interroger les pages Web de chaque film ;
6. identifier la section qui recueille les commentaires ;
7. extraire les commentaires et les enregistrer dans un fichier texte.

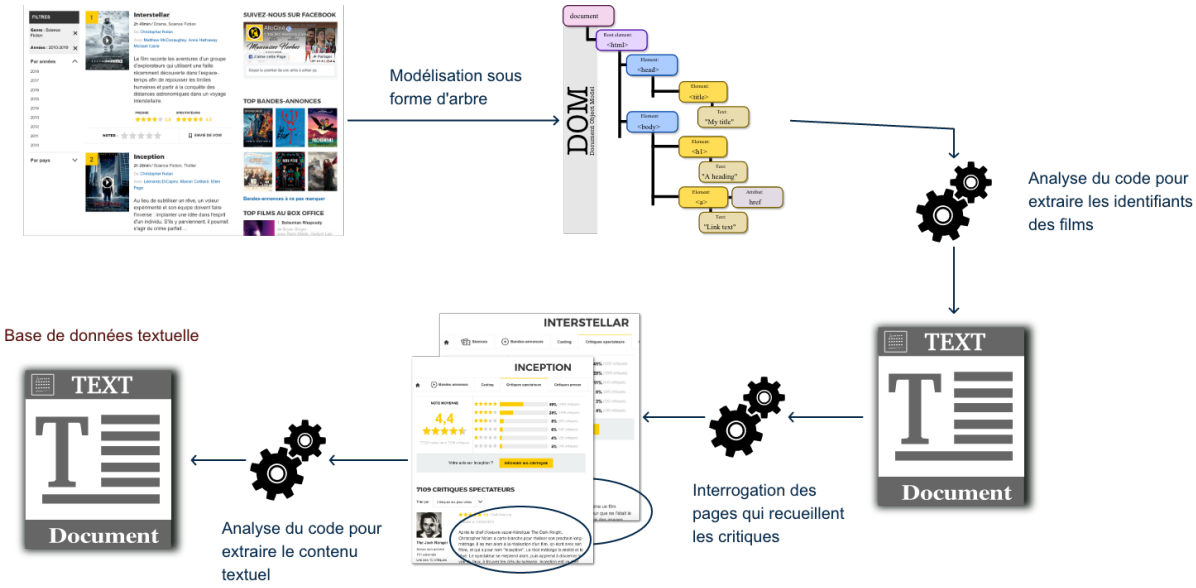

Figure 1: Méthode pour extraire des critiques de spectateurs