TD : constituer un corpus de critiques de films

Alexandre Roulois (Université Paris Cité, LLF, CNRS)

Table of contents

Interroger les pages Web de chaque film

Objectif : lire le fichier *links.txt*, généré au TD précédent, afin d'extraire les identifiants des films, puis interroger les pages Web concernées.

Besoins :

- ouvrir le fichier en lecture et en récupérer le contenu
- parcourir chaque ligne
- analyser la syntaxe de la ligne
- isoler l'identifiant
- reconstruire les urls sur *Allociné*

Rappel : les identifiants, à la fin de chaque URL, sont composés uniquement de chiffres :

```
/film/fichefilm_gen_cfilm=114782.html
/film/fichefilm_gen_cfilm=143692.html
```

1e étape : ouvrir le fichier en mode lecture et en récupérer le contenu

```
# file descriptor
with open('./data/links.txt') as file:
  # a list of lines
  lines = file.readlines()
```

2e étape : parcourir chaque ligne

```
# for each line
for line in lines:
  # proceed to syntax analysis
  pass
```

3e étape : isoler l'identifiant du film dans la ligne

- importer le module des expressions régulières

```
import re
```

- méthode `search()` du module `re` pour exécuter *regex* :

```
# look up for a pattern in each line
for line in lines:
  pattern = ''
  id_movie = re.search(pattern, line)
```

- motif pour chiffres qui se suivent : `[0-9]+` ou `\d+`

```
for line in lines:
  id_movie = re.search('\d+', line)
```

- résultat de la capture disponible via méthode `group()` :

```
for line in lines:
  id_movie = re.search('\d+', line)
  # for each movie, print the id
  print(id_movie.group(0))
```

4e étape : reconstruire les URLs sur *Allociné*

- critiques spectateurs sur une page où `{id}` est l'identifiant du film :

```
http://www.allocine.fr/film/fichefilm-{id}/critiques/spectateurs/
```

```
for line in lines:
  id_movie = re.search('\d+', line)
```

```
  url = f'http://www.allocine.fr/film/fichefilm-{id_movie.group(0)}/critiques/spectateurs/
```

5e étape : importer le module `utils.py` du package `scrape` et appeler la méthode `get_html_from_url()` pour récupérer le contenu HTML de chaque page.

```
import scrape.utils as scrape

for line in lines:
  id_movie = re.search('\d+', line)
  url = f'http://www.allocine.fr/film/fichefilm-{id_movie.group(0)}/critiques/spectateurs/
  # get HTML code
  html = scrape.get_html_from_url(url)
```

Extraire les critiques

Sur les pages de chaque film, les critiques utilisateurs sont encadrées de marqueurs auxquels sont appliqués la classe CSS `content-txt`.

Grâce à *BeautifulSoup*, on peut facilement sélectionner ces marqueurs et en récupérer le contenu :

```
from bs4 import BeautifulSoup

for line in lines:

  id_movie = re.search('\d+', line)
  url = f'http://www.allocine.fr/film/fichefilm-{id_movie.group(0)}/critiques/spectateurs/
  html = scrape.get_html_from_url(url)

  # extract user reviews
  soup = BeautifulSoup(html, 'html.parser')
  reviews = soup.select('.content-txt')
```

Comme il s'agit d'une procédure déjà utilisée dans le précédent *notebook*, autant définir une fonction à placer dans le module *utils* :

```
from bs4 import BeautifulSoup

def parse_html_by_class(html, selector):
  """Extracts tags from HTML whith CSS selector.
```

```
    Keyword arguments:
    html -- the html page
    selector -- the CSS selector
    """
    soup = BeautifulSoup(html, 'html.parser')
    tags = soup.select(selector)

    return tags
```

Et modifier le code en conséquence :

```
for line in lines:

    id_movie = re.search('\d+', line)
    url = f'http://www.allocine.fr/film/fichefilm-{ id_movie.group(0) }/critiques/spectateur
    html = scrape.get_html_from_url(url)

    # extract user reviews
    reviews = scrape.parse_html_by_class(html, '.content-txt')
```

Pour placer au final toutes les critiques dans une liste :

```
# empty list for collecting movie reviews
movie_reviews = list()

for line in lines:
    id_movie = re.search('\d+', line)
    url = f'http://www.allocine.fr/film/fichefilm-{ id_movie.group(0) }/critiques/spectateur
    html = scrape.get_html_from_url(url)

    # add the user reviews to the list
    movie_reviews.append(
      # A record is a tuple of two elements:
      (
        # 1: id movie
        id_movie.group(0),
        # 2: list of relative reviews
        scrape.parse_html_by_class(html, '.content-txt')
      )
    )
```

Constituer le corpus

Maintenant que nous disposons d'une liste des critiques pour chaque film, nous souhaitons au final constituer un corpus avec les caractéristiques suivantes :

- un fichier par film
- une critique par ligne

Dans les critiques extraites par *BeautifulSoup*, nous souhaitons déjà nous débarrasser des balises HTML :

```
# for each movie…
for id_movie, reviews in movie_reviews:
  # … keep only the textual content of each review
  for review in reviews:
    review = review.get_text()
```

En analysant le retour grâce à la fonction `print()`, on observe que :

- les retours à la ligne utilisateurs sont préservés ;
- il subsiste des espaces superflues avant et après chaque critique.

Python fournit des méthodes sur les chaînes de caractères pour gérer ces effets :

- méthode `.strip()` pour retirer les espaces ;
- méthode `.replace()` pour retirer les retours à la ligne (`'\n'`).

```
for id_movie, reviews in movie_reviews:
  for review in reviews:
    review = review.get_text()
    # delete spaces
    review = review.strip()
    # substitutes new lines by a space
    review = review.replace('\n', ' ')
```

Mieux, il est possible de chaîner les méthodes :

```
for id_movie, reviews in movie_reviews:
  for review in reviews:
    review = review.get_text().strip().replace('\n', ' ')
```

Il ne reste plus qu'à enregistrer les critiques :

```
for id_movie, reviews in movie_reviews:
  for review in reviews:
    review = review.get_text().strip().replace('\n', ' ')
    with open(f'./data/allocine/reviews_{id_movie}.txt', 'a') as file:
      # write the line
      file.write(review)
      file.write('\n')
```