Scraping

Outils et méthodes pour récupérer les données des pages web

Scraping : c'est quoi ?

Le web scraping → technique d'extraction du contenu d'une page (d'un site) web. Les données extraites souvent sont ou deviennent des données structurées (xml, BD, etc.).

Comment fait-on (1) ?

- Analyser le fonctionnement du site à scraper :
  - **le contenu** : Chaque page de recherche affiche les infos principales sur les sites correspondants à nos requêtes, et ensuite il y a chaque URL.
  - **les filtres** : Chaque application d'un filtre (ex: choix logique) modifie l'URL en y ajoutant des variables. Ceci permet à un script d'être paramétrable, en générant les URL en fonction des filtres désirés.
  - **les catégories** : Il vaut mieux ne pas sélectionner plusieurs catégories en même temps. Donc si on veut faire une requête à la fois sur les restos et les cabinets médicaux, il va falloir faire 2 recherches distinctes.
  - **la pagination** : Chaque page affiche 10 résultats, avant de devoir nous rendre sur la page suivante.

Comment fait-on (2) ?

- **Le script :**

  L’algorithme dans sa globalité va se présenter sous la forme d’une double boucle, suivi d’une écriture dans un fichier Xml :

  - Pour chaque page de résultats :

- Générer l’URL de la page à aller scraper
- Ouvrir cette page et passer le résultat HTML au parseur.
- Pour chaque commerce (10 max par page) :

  - Récupèrer chacune des données souhaitées (contenu balise paragraphe, note, etc.)

  - Si c'est un doublon ou une publicité, skipper ce tour de boucle.

  - Nettoyer les données (ex : retirer les trailing spaces)

  - Les placer dans un objet (de classe Objet_Recherche par exemple)

  - Ajouter l’objet créé à une liste d’objets pour le sauver

  - Pour chaque objet créé :

- Ecrire son contenu dans le fichier xml.

Pour un script en Python 3 les pacages suivants sont nécessaires : le module lxml, beautilfulsoup4, resquests

Exemple : scraper *Yelp*

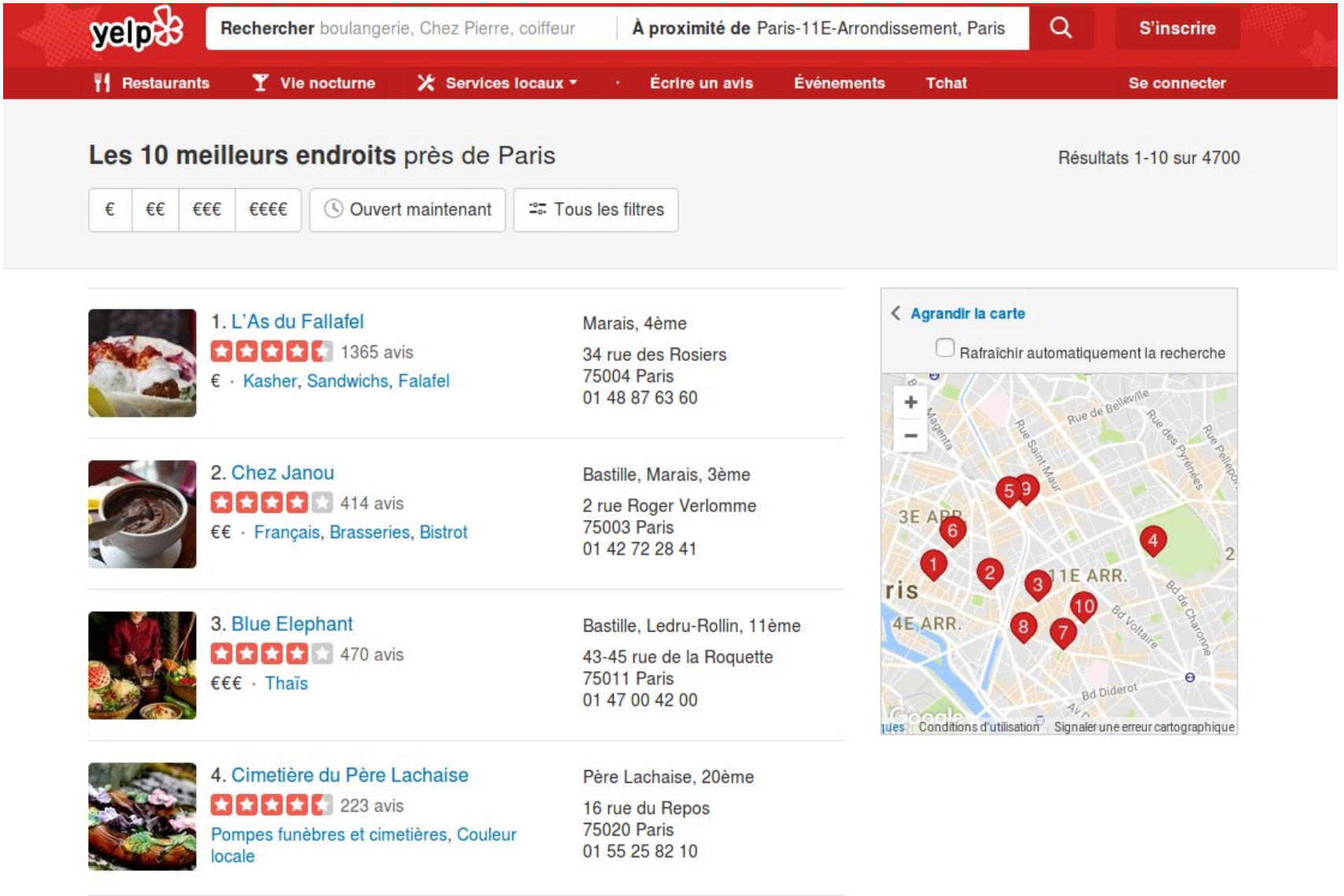
- Le site de Yelp : il permet de parcourir les commerces locaux des grandes villes, en les filtrant (notation des utilisateurs, ville, arrondissement, type de commerce, sous-type de commerce, etc.).
  - Le code de cet exemple se trouve sur Github https://github.com/ddahan/yelp-scraper
  - Ce que l'on va récupérer : '*Tout commerce de telle catégorie dans tels quartiers/arrondissements de Paris*`. Pour chacun des commerces, nous réupérerons le nom, l'adresse, le numéro de téléphone, l'URL, et l'ensemble des catégories auxquelles il appartient (1 commerce appartient à 1->N catégories : il peut vendre des sandwichs, mais aussi des desserts).

Avertissement : les conditions générales d'utilisation de sites comme Yelp ou TripAdvisor, interdisent ce genre de pratique, cet exemple est donc à visée pédagogique uniquement.

Vérifier que ce que nous souhaitons réaliser est possible.

Chaque page de recherche affiche les infos principales sur les commerces.

Ici : application d'un filtre : lorsque je sélectionne la catégorie “Alimentation”, j’obtiens une URL du genre : http://www.yelp.fr/search#cflt=food



Chaque page affiche 10 résultats au maximum, avant de devoir nous rendre sur la page suivante.

En allant en page 2, on se rend compte que l'URL a changé avec une nouvelle variable : `start=10`. On comprend donc facilement que la page 1 contient les résultats de 0 à 9, la page 2 les résultats de 10 à 19, et la page 67 les résultats de 660 à 669.

Expliquer le script (1)

- Génération des URLs à scraper : On a vu qu'il y a un lien direct entre le numéro de la page parcourue, et l'index à écrire en tant que variable dans l'URL, donc une fonction permet de passer de l'un à l'autre :

```
def page_to_index(page_num):
''' Transforms page number into start index to be written in Yelp URL '''
return (page_num – 1)*10
```

Expliquer le script (2)

- Cela nous permet ensuite de construire simplement nos URLs, où vous pouvez voir qu'il s'agit simplement de concatener des variables à notre URL :

```
def build_yelp_url(page, c):
    ''' Builds Yelp URL for the given page and cflt to be parsed according to
    config variables '''

    url = "http://www.yelp.fr/search?&start={0}".format(page_to_index(page))
    if CITY: url += "&find_loc={0}".format(CITY)
    url += "&cflt={0}".format(c) # We assume that CFLTS list is not empty
    if PARIS_DISTRICTS: url += "&l=p:FR-75:Paris::{0}".format(
        build_arglist(PARIS_DISTRICTS))
```

Expliquer le script (3)

- **Parsing HTML**

  Nous utilisons la lib BeautifulSoup qui permet de faire du parsing HTML très simplement. Chaque commerce issu d'une page de résultat de recherche, est contenu dans une `<div class="search-result">`. On regarde le code source de Yelp. Pour récupérer chacune des portions HTML liées, il suffit donc de créer une liste :

  ```
  search_results = soup.find_all('div', attrs={"class":u"search-result"}):
  ```

  Au sein de chacune de ces portions de HTML récupérées, récupérer le nom est très similaire, à la différence ici que le nom du commerce est situé dans un lien html :

  ```
  for sr in search_results:
    shop_name = sr.find('a', attrs={"class":u"biz-name"}).get_text()
  ```

  Le mécanisme est du même genre pour les autres informations à récupérer (adresse, téléphone, url, catégories, etc.)

Expliquer le script (4)

- Certaines pages contiennent des annonces publicitaires qui viennent polluer nos résultats, puisque ces dernières ne correspondent en rien à nos filtres de recherche. Pour les supprimer, il suffit d'observer la classe qui leur est associée, et d'écrire une fonction associée :

```
def is_advertisement(search_result):
    ''' Return True is the search result is an add
'''
   if search_result.find('span',
attrs={"class":u"yloca-tip"}):
        return True
  return False
```

Expliquer le script (5)

- Un point sur le contournement de la sécurité

  Scraper des sites web n’est généralement pas apprécié par les sites en questions (ce qui est tout à fait compréhensible). Partez donc du principe que ces derniers feront tout leur possible pour vous compliquer la vie. Voici deux mesures très simples qui vont réduire les chances d’être identifié comme un “robot” :

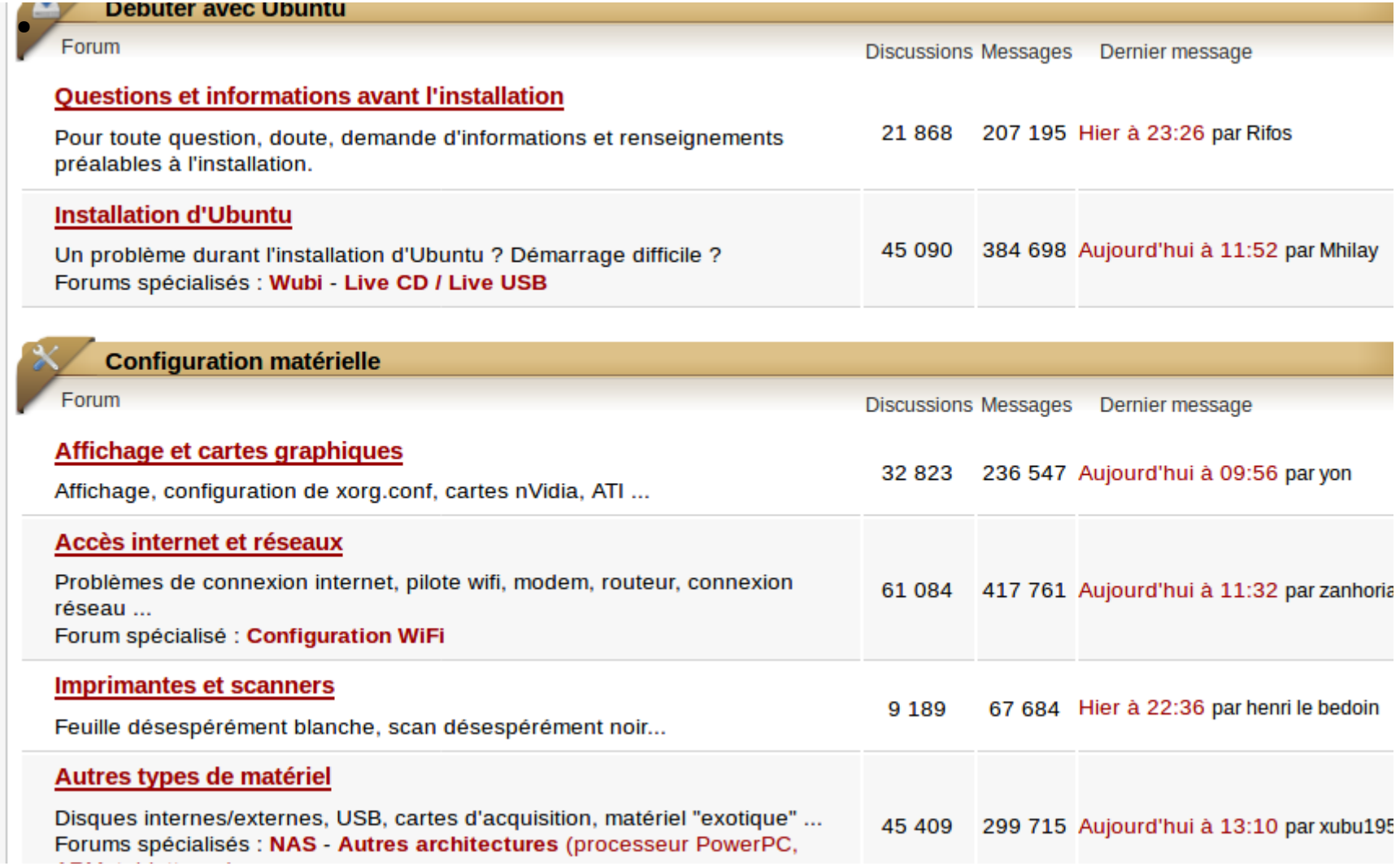
  Durée d’attente aléatoire entre chaque ouverture de page

  Si votre script ouvre 30 pages en 5 secondes, vous n'agissez pas comme un humain. L’idée est donc de générer un sleep aléatoire à la fin de chaque tour de la boucle principale :

```
def r_sleep():
'' generates a random sleep between 2.000 and 10.000 seconds '''
   length = float(randint(2000, 10000)) / 1000
   mylog("Safety Random Sleep has started for {0} sec".format(length))
   sleep(length)
   mylog("Safety Random Sleep is over")
```

Scraping : Ce qu'il fait

Le fait de “scraper” une page web signifie à la base que l'on transforme des infos présentes non structurées en données structurées exploitables.



Débuter avec Ubuntu

Forum	Discussions	Messages	Dernier message
Questions et informations avant l'installation Pour toute question, doute, demande d'informations et renseignements préalables à l'installation.	21 868	207 195	Hier à 23:26 par Rifos
Installation d'Ubuntu Un problème durant l'installation d'Ubuntu ? Démarrage difficile ? Forums spécialisés : **Wubi** - **Live CD / Live USB**	45 090	384 698	Aujourd'hui à 11:52 par Mhilay

Configuration matérielle

Forum	Discussions	Messages	Dernier message
Affichage et cartes graphiques Affichage, configuration de xorg.conf, cartes nVidia, ATI ...	32 823	236 547	Aujourd'hui à 09:56 par yon
Accès internet et réseaux Problèmes de connexion internet, pilote wifi, modem, routeur, connexion réseau ... Forum spécialisé : **Configuration WiFi**	61 084	417 761	Aujourd'hui à 11:32 par zanhoria
Imprimantes et scanners Feuille désespérément blanche, scan désespérément noir...	9 189	67 684	Hier à 22:36 par henri le bedoin
Autres types de matériel Disques internes/externes, USB, cartes d'acquisition, matériel "exotique" ... Forums spécialisés : **NAS** - **Autres architectures** (processeur PowerPC,	45 409	299 715	Aujourd'hui à 13:10 par xubu195

L'exemple Ubuntu forum

- Dans l'exemple ci-dessus nous avons des liens vers les discussions des internautes suivant les thématiques présentes :

  par ex :

  - “Débuter avec Ubuntu”, et

  - “Configuration matérielle”

- Dans chaque thématique nous trouvons des liens vers des discussions spécifiques selon un sujet correspondant :

  par ex :

  - “Questions et informations avant l'installation” et

  - “installation d'Ubuntu”

  Allons voir la compostion html de cette page …