TP: Crawl et extraction d'informations Web

Première partie

Pierre Senellart
pierre.senellart@telecom-paristech.fr

23 novembe 2015

Dans ce TP, nous allons nous intéresser au site Politico `http://www.politico.com/`. Le but va être d'extraire le maximum d'informations sur les articles publiés en novembre 2015 sur ce site d'actualité en ligne, en utilisant des techniques de crawl (première partie, ce sujet) et d'extraction d'informations (deuxième partie, demain).

Nous allons exploiter différents outils pour réaliser notre crawl ; du plus simple au plus complexe, `wget`, Scrapy, Selenium.

Installations préliminaires

Le TP peut être réalisé sur tout ordinateur comportant une installation de Python 2.x, Firefox, wget, ainsi que les paquets Python `scrapy` et `selenium` (installables via les commandes `pip install` ou `easy_install`). Sur les machines de la salle de TP, vous pouvez avoir accès à une installation complète en mettant dans un fichier `.zshrc` dans votre répertoire personnel :

```
source /infres/ic2/senellar/virtualenv/bin/activate
export LD_LIBRARY_PATH=/infres/ic2/senellar/lib
```

puis en relançant un nouveau shell.

1 Crawls élémentaires avec wget

`wget` est un outil en ligne de commande permettant d'effectuer des crawls élémentaires. Dans sa forme la plus simple « `wget url` », il permet de récupérer une page Web à l'URL `url`. Vous pouvez consulter la documentation de `wget` avec « `man wget` ».

1. A-t-on le droit de crawler les articles qui nous intéressent sur le site de Politico ? Pourquoi ? Quelles mesures devons-nous prendre ?
2. Choisissez un article de Politico publié en novembre 2015 et téléchargez le avec `wget`. Ouvrez le fichier ainsi téléchargé et comparez-le à la page Web d'origine.

3. Avec les options « `-r --wait=1` », `wget` permet de télécharger de manière récursive un ensemble de pages Web, avec un délai de *politesse* d'une seconde entre deux requêtes vers le même serveur. Essayez cette option pour télécharger l'ensemble des articles de Politico. Que constatez-vous ? Vous pouvez interrompre le crawl avec `CTRL+C`.
4. `wget` dispose d'une option « `-np` » permettant de ne télécharger que les pages Web contenues dans le même répertoire que la page Web initiale (ou dans un sous-répertoire de celui-ci). Essayez de mettre en œuvre cette option pour télécharger les articles de Politico de novembre 2015. Vous devriez parvenir à récupérer une partie d'entre eux, mais il reste des requêtes *inutiles* faites par le crawler.
5. `wget` dispose d'une option « `--reject-regex` » permettant de ne pas télécharger les pages correspondant à un certain motif ; utiliser cette option pour exclure les pages inutiles en indiquant quel(s) caractère(s) les URL que vous souhaitez rejeter comporte(nt) (attention, si vous utilisez des caractères spéciaux pour les expressions rationnelles, il faut les préfixer d'un antislash). Laissez le crawl se terminer : combien avez-vous récupérés d'articles ? Y sont-ils tous ?

2 Crawl systématique avec Scrapy

Nous allons maintenant utiliser le logiciel Scrapy, venant sous la forme d'un module Python et d'un ensemble d'outils externes, pour faire un crawl plus systématique des articles qui nous intéressent. La documentation de Scrapy est disponible à l'URL `http://doc.scrapy.org/en/1.0/`.

1. En expérimentant avec la fonctionnalité de recherche intégrée au site de Scrapy, concevez (sans l'implémenter) une stratégie de crawl exhaustif des articles parus en novembre 2015.
2. Expérimentez avec le code exemple Scrapy fourni sur la page d'accueil `http://scrapy.org/` sans le modifier, et réfléchissez à ce qui y est précisément fait. Par défaut, Scrapy est très verbeux, vous pouvez le rendre plus silencieux en ajoutant « `-L WARN` » à la ligne de commande.
3. Modifiez le code exemple Scrapy en y ajoutant un constructeur pour votre crawler :

   ```
   def __init__(self):
     self.download_delay = 1
   ```

   qui assurera le respect de la politesse de crawl. Testez.
4. Adaptons maintenant le code Scrapy au crawl des articles de Politico de novembre 2015, en suivant la stratégie établie précédemment. Vous aurez besoin des bases de Scrapy suivantes (utilisant en particulier le langage XPath dont nous reparlerons demain) :
   — `response.xpath('//a/@href').extract()` renvoie l'ensemble des liens du document.

— `response.xpath('//a[contains(.,"Toto")]/@href').extract()` renvoie l'ensemble des liens du documents dont le texte contient la chaîne de caractères « `Toto` ».
— Au sein d'une fonction de crawl, `response.url` contient l'URL courante et `response.body` le contenu Web complet téléchargé (le code HTML pour une page Web).

Le but est de sauvegarder dans un répertoire l'ensemble des articles du site. Combien en trouvez-vous ? Comparer avec le crawl par `wget`.

5. Ouvrez l'un de ces articles stockés en local, comparez-le avec la page Web d'origine. Que se passe-t-il si l'on essaie d'accéder aux commentaires ?

3 Crawl d'application Web complexe avec Selenium

Nous allons maintenant utiliser Selenium, un instrumenteur de navigateurs, via son interface par le module Python `selenium`. La documentation de ce module est disponible à l'URL `http://selenium-python.readthedocs.org/`.

1. Suivez le tutoriel de Selenium (2.1 à 2.4 uniquement) présent à l'URL `http://selenium-python.readthedocs.org/getting-started.html`
2. Modifiez votre crawler `scrapy` pour faire appel à Selenium pour récupérer chaque article. Dans Selenium, vous avez accès à l'ensemble du code source d'une page Web (telle que manipulée par le navigateur, pas telle qu'initialement téléchargée) avec `driver.page_source` (sous la forme d'une chaîne d'octets).
3. Demandez à Selenium de cliquer sur le bouton « Show Comments ». Vous pourrez utiliser `driver.find_element_by_id(...)` pour trouver le bouton, et la fonction suivante pour cliquer sur un élément et attendre que la page change :

```
from selenium.webdriver.support.ui import WebDriverWait

def click_n_wait(driver, button, timeout=10):
  source = driver.page_source
  button.click()
  def compare_source(driver):
    try:
      return source != driver.page_source
    except WebDriverException:
      pass
    WebDriverWait(driver,timeout).until(compare_source)
```

4. Ouvrez les pages téléchargées dans un navigateur. Les commentaires sont-ils visibles ? Inspectez la source des pages téléchargées. Les commentaires y sont-ils présents ?

Dans la deuxième partie du TP, nous extrairons le texte principal et les commentaires au moment du crawl, ainsi que d'autres informations plus structurées.