TP: Crawler

Jean-Claude Moissinac
jean-claude.moissinac@telecom-paristech.fr

inspiré d'un TP de Pierre Sennelart

10 mai 2017

Dans ce TP, nous allons nous intéresser au site FPO http://www.freepatentsonline.com/. Le but va être d'extraire des informations sur des brevets sur ce site de description de brevets, en utilisant des techniques de crawl (première partie, ce sujet) et d'extraction d'informations (deuxième partie).

Nous allons exploiter différents outils pour réaliser notre crawl ; du plus simple au plus complexe, wget, Tidy, Html5Lib...

A la fin du TP, ou, au plus tard dimanche 14/5, vous m'enverrez par email une archive portant votre nom. Elle comportera les sources de vos programmes nommés comme spécifié dans le sujet et un fichier README.txt où vous expliquerez l'organisation des fichiers fournis, les choix d'informations que vous avez cherché à obtenir et des indications sur les informations réellement obtenues.

ATTENTION : lors des appels répétés à un même site et compte tenu du nombre que vous êtes, respectez un délai d'au moins 5 secondes entre 2 appels ; le non-respect de cette règle risque d'entrainer le blocage de nos accès

(des outils comme lxml, Scrapy, Selenium, BeautifulSoup pourront être utilisés optionnellement)

Installations préliminaires

Le TP peut être réalisé sur tout ordinateur comportant une installation de Python 2.x, Firefox, wget, ainsi que les paquets Python tidylib et html5lib (installables via les commandes pip install ou easy_install). Il est vivement recommandé de créer un environnement virtuel avant de faire l'installation des outils:

```
python3 –m venv ./nomDossierPourVotreEnvironnement
source nomDossierPourVotreEnvironnement/bib/activate
```

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. Choisissez une description de brevet, sur un sujet qui vous intéresse, dans FPO et téléchargez la 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 ré- cursive 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 un en- semble de descriptions de brevets depuis FPO. 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 télé- charger des articles de FPO. Vous devriez parvenir à récupérer une partie d'entre eux, mais il reste des requêtes *inutiles* faites par lecrawler.
5. wget dispose d'une option « --reject-regex » permettant de ne pas téléchar- ger les pages correspondant à un certain motif ; utiliser cette option pour exclure des pages qui vous semblent inutiles en indiquant quel(s) motifs de 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 travailler un moment : combien avez-vous récupérés de description de brevets?

2 Crawl systématique par programme Python

Nous allons maintenant utiliser Python3 pour faire un crawl plus systématique des articles qui nous intéressent. Téléchargez le fichier tasktimer.py et un exemple d'utilisation dans etape0.py à ces adresses :

https://perso.telecom-paristech.fr/moissina/TPCRAWL/etape0.py

https://perso.telecom-paristech.fr/moissina/TPCRAWL/modules/tasktimer.py

1. Commencez par faire un programme minimal qui fait une requête sur le site FPO pour charger et sauver dans un fichier la réponse du serveur FPO à cette requête. Appelez le programme etape1.py
2. Faire un programme –inspiré des programmes etape0 et etape1- qui charge une liste de pages. Appelez ce programme etape2
   Maintenant, au lieu de sauver le contenu reçu, vous allez commencer à analyser le contenu. Pour commencer vous allez utiliser pytidylib pour corriger les défauts éventuels des pages chargées (optionnellement, si votre compte permet d'utiliser BeautifulSoup, vous pouvez faire les choses avec cet outil). Ensuite, vous allez utiliser html5lib pour accéder à des parties de la page. Avec le XPATH './/xh :a', récupérez tous

les tags a présents dans la page (xh est le préfixe associé au namespace http://www.w3.org/1999/xhtml)
Combien y a-t-il de lien dans chacune des pages que vous visitez ?
Ajoutez chaque lien à la liste des liens à charger.
Ajoutez une condition d'arrêt lorsque N descriptions de brevets ont été chargées.
A ce moment-là, combien de pages ont été visitées ? Combien de liens sont dans la file d'attente des liens à traiter ?
Donnez le nom etape3.py à ce programme.

3. Ajouter au programme précédent du code pour éviter de charger plusieurs fois la même URL. Programme etape4.py
4. Ajouter au programme précédent du code
   - pour déterminer si une page est une page de description d'un brevet (+> une page qui nous intéresse)
   - pour estimer empiriquement si un lien mérite d'être suivi

   Programme etape5.py
5. A l'aide de html5lib et d'expressions régulières sélectionnez certaines URL qui respectent certains modèles (vous pourrez mettre au point vos expressions régulières à l'aide de https://regex101.com/). Programme etape6.py
6. Choisissez d'autres informations dans ces pages dont vous allez chercher à récupérer la valeur (nom d'auteur de brevet ? entreprise ?...). Sauvez les données obtenues dans une structure json. Programme etape7.py

Pour aller plus loin sur ce sujet, pour développer un crawler en Python, nous vous recommandons scrapy.