TP: Crawl et extraction d'informations Web

Deuxième partie

Pierre Senellart
pierre.senellart@telecom-paristech.fr

24 novembe 2015

Nous continuons le TP de crawl et extraction d'information du site Politico, au travers des trois étapes suivantes : récupération avec Selenium et XPath des commentaires sur l'article ; extraction à l'aide de BeautifulSoup4 et les sélecteurs CSS de contenu structuré sur les articles et sur les commentaires ; utilisation de NLTK pour extraire des informations depuis le texte des articles.

Installations préliminaires

Le TP peut être réalisé sur tout ordinateur comportant une installation de Python 2.x, Firefox, ainsi que les paquets Python `scrapy`, `selenium`, `nltk` (installables via les commandes `pip install` ou `easy_install`), et les modules NLTK `averaged_perceptron_tagger`, `maxent_ne_chunker` `punkt` et `words` (récupérables avec `nltk.download()`). Sur les machines de la salle de TP, vous pouvez avoir accès à une installation complète si vous avez suivi les instructions du TP précédent.

1 Récupération des commentaires avec Selenium

Nous allons maintenant utiliser Selenium pour récupérer l'ensemble des commentaires postés sous un article et les stocker dans un fichier.

1. Écrire une expression XPath qui renvoie l'élément `<iframe>` contenant la souspage des commentaires d'un article de Politico après que le bouton a été activé. Vous pouvez utiliser la console de développement de Firefox, Chromium ou Chrome pour tester une expression XPath en tapant : `$x("//div")` où `//div` est l'expression XPath à tester.
2. Remplacer la fonction `click_n_wait` par une fonction testant spécifiquement la présence de l'iframe à l'aide de la méthode Selenium `find_element_by_xpath` d'un objet `webdriver`. Attention cette méthode renvoie une `selenium.common.exceptions.NoSuchElementException` si aucun résultat n'est trouvé.

3. Une fois l'iframe chargé, demander à Selenium de basculer dans le contexte de l'iframe en appelant `.switchto.frame` sur l'objet `webdriver` avec comme argument l'iframe en question. Puis, tant qu'un bouton « Load ... more comments », cliquer dessus et attendre que le résultat se charge ; il faudra pour ça écrire une nouvelle variante de la méthode `click_n_wait`. On pourra par exemple vérifier que le nombre de balises `<img>` (correspondant aux images de profil) augmente. On ignorera pour simplifier les liens « Show ... more replies in this thread ».
4. Une fois l'ensemble des commentaires chargés (sauf éventuellement certaines réponses à des commentaires), les sauvegarder dans un fichier distinct du fichier contenant l'article. *Il est possible que les scripts JavaScript présents dans la page posent problème dans la page sauvegardée. Si c'est le cas, remplacer l'appel à* `.page_source` *par* `.find_element_by_tag_name("div").get_attribute("innerHTML")` *qui se contente de récupérer une sérialisation du contenu du premier* `<div>` *du document, ce qui exclue les scripts JavaScript.*
5. Crawler une dizaine de pages de cette manière (pas la peine d'en récupérer plus), et vérifier leur contenu sauvegardé et les commentaires sauvegardés.

2 Extraction structurée avec BeautifulSoup4

On utilise dans cette partie le module Python BeautifulSoup4, dont la documentation est disponible ici : `http://www.crummy.com/software/BeautifulSoup/bs4/doc/`. Vous pouvez vous référer à cette page Web pour découvrir les commandes à utiliser dans cette partie. Nous nous appuierons sur les résultats téléchargés dans les étapes précédentes du TP. C'est généralement une bonne pratique de séparer le crawl des étapes d'extraction proprement dites.

1. Écrire un script Python qui récupère l'ensemble des articles Politico sauvegardés d'un répertoire donné et en extrait :
   — l'identifiant (provenant du nom du fichier) ;
   — le titre ;
   — les auteurs, séparés par une virgule ;
   — une estampille temporelle au format ISO8601 ;
   — le texte complet de l'article, les différents paragraphes étant séparés par des espaces.

   Stocker le résultat dans un fichier TSV (tab-separated values), une ligne par article et vérifier soigneusement son contenu.
2. Écrire un script Python qui récupère l'ensemble des commentaires sur les articles Politico sauvegardés d'un répertoire donné et en extrait :
   — l'identifiant de l'article (provenant du nom du fichier) ;
   — le nom d'auteur ;
   — l'emplacement ou autre description de l'auteur ;
   — le texte du commentaire.

   Stocker le résultat dans un fichier TSV (tab-separated values), une ligne par article, et vérifier soigneusement son contenu.

3 Extraction d'informations à partir du texte

On utilise maintenant le module Python NLTK. On pourra se référer à la documentation disponible page 3 de `http://www.crummy.com/software/BeautifulSoup/bs4/doc/`.

1. Utiliser NLTK pour extraire du texte des articles les entités nommées qui y sont contenues. On pourra par exemple ajouter au fichier TSV une colonne avec la liste des personnes trouvées dans l'article, ou la liste des villes ou pays (classifiés par NLTK comme GPE pour geopolitical entity).
2. (plus ouvert) Réfléchir à des motifs de Hearst ou d'extraction de faits pouvant être mis en œuvre pour de l'extraction structurée depuis le texte annoté par NLTK. On pourra par exemple tenter d'associer à chaque personne pour lequel c'est possible leur fonction ou profession indiquée dans l'article. D'autres applications sont possibles. Tester soigneusement la qualité des résultats (qui ne seront probablement pas parfaits).