

Projet Python 3I AL

Thème :

Le thème du projet est de réaliser un web crawler en python.

Définition wikipédia :

Un **robot d'indexation** (ou littéralement **araignée** du Web ; en anglais **web crawler** ou **web spider**) est un **logiciel** qui explore automatiquement le **Web**. Il est généralement conçu pour collecter les **ressources** (pages Web, images, vidéos, documents Word, PDF ou PostScript, etc.), afin de permettre à un **moteur de recherche** de les **indexer**.

Notation :

Le projet se décompose en des fonctionnalités, donnant chacune un certain nombre de points.

Les fichiers source de votre projet seront passés à Pylint, le score retourné par chacun d'eux devra être de 10/10, vous obligeant à respecter PEP8.

Tout score inférieur entrainera un malus égal à (10 - moyenne des scores).

En plus des sources de votre projet, vous devrez également fournir un fichier README qui contiendra :

- le nom des deux binomes,
- le descriptif de ce qui a été implémenté,
- les modules à installer pour faire fonctionner le tout
- les façons de faire fonctionner le crawler et les différents outils

Le tout, sources + README, seront zippés dans une archive avec le nom : Python_3IAL_<Nom Binome1>_<Nom Binome2>.zip

Fonctionnalités :

A - Crawler

Toutes les fonctions du crawler seront dans le module **crawler.py**

Le crawler sera exécutable en ligne de commande de la façon suivante :

python crawler.py <url de départ> [options éventuelles]

Jetez un oeil à <https://docs.python.org/3.4/library/getopt.html> pour récupérer les arguments de la ligne de commande.

A1 - Parser de page web (3 points) :

La première fonctionnalité consiste bien entendu à extraire les informations d'une page html dont on connaît l'url.

Ecrire une fonction **extract**, qui prend en paramètre une url, et retourne un dictionnaire contenant les informations suivantes :

- le titre de la page (référéncé dans le dictionnaire par la clé "**title**")
- la description (référéncé par la clé "**description**")
- la liste des mots clés (référéncé par la clé "**keywords**")
- la liste des liens contenus dans la page (référéncé par la clé "**links**")

Il est recommandé d'utiliser BeautifulSoup : <http://www.crummy.com/software/BeautifulSoup/>

A2 - Mapping de sites (8 points) :

Ecrire une fonction **crawl**, qui prend en paramètres :

- une url de départ, **url**, qui sert de point de départ au crawling
- une profondeur maximum, **depth**, par défaut égale à 2, qui indique quand est ce qu'on doit s'arrêter :
 - 0 : on ne parse que l'url de départ
 - 1 : on parse en plus les urls qui sont en lien dans l'url de départ
 - 2 : on parse en plus les urls qui sont en lien dans les urls parsées en 1
 - etc...
- un booléen **go_outside** indiquant si on sort du site ou non, par défaut à **True**. Par exemple, si on part de l'url www.google.fr, et qu'on peut sortir du site, si il y a un lien vers **www.facebook.com**, on ira dessus. Par contre si **go_outside** est à **False**, on ne considerera que les urls contenant google.fr : <http://maps.google.fr/maps>, <http://www.google.fr/intl/fr/options/>, etc...

La fonction fera le parcours des sites en **largeur**, c'est a dire qu'il parsera d'abord l'url de départ, puis tous ses liens, puis tous les liens de ses liens, etc... Il est recommandé pour cela de stocker les liens trouvés dans une file d'attente.

Vous ferez également attention de ne pas parser plusieurs fois la même url.

Plus d'infos : http://fr.wikipedia.org/wiki/Algorithme_de_parcours_en_largeur

La fonction retournera un dictionnaire ou les clés sont les urls parsées, et les valeurs associées les dictionnaires retournés par la fonction **extract**.

Vous devrez gérer des options **--depth** et **--outside** dans votre ligne de commande, pour permettre à l'utilisateur de choisir les valeurs de ces paramètres.

Détail des points :

1. *un crawl qui marche, sans gérer les options : 2points*
2. *gestion de la profondeur : +2 points*

3. *parcours en largeur* : +2 points
4. *gestion de go_outside* : +2 points

A3 - Filtrer les liens a suivre (2 points):

Améliorer la fonction **crawl**, en enlevant les parties des liens derrière des # ou des ?

Exemples :

http://docs.opencv.org/master/modules/core/doc/basic_structures.html#mat-at

devient

http://docs.opencv.org/master/modules/core/doc/basic_structures.html

http://www.google.fr/?gws_rd=ssl devient <http://www.google.fr/>

A4 - Sauvegarde et chargement JSON (3 points) :

Faire des fonctions de sauvegarde et de chargement de résultat de crawling.

La fonction **save** prend en arguments :

- un chemin vers un dossier de sauvegarde, **dir**
- un dictionnaire résultat de la fonction crawl, **datas**

Pour chaque url contenue en tant que clé dans **datas**, la fonction :

1. créera un fichier, dont le nom sera composée uniquement des caractères alphanumériques de l'url, et avec l'extension .json.

Exemple : <http://www.google.com> donne le nom de fichier httpwwwgooglecom.json

2. ajoutera au dictionnaire associé la clé "url" avec comme valeur l'url en question
3. sauvegardera le dictionnaire dans le fichier, au format json

La fonction **load** prend en argument :

- le chemin vers le dossier où se trouvent les fichiers à charger, **dir**

La fonction retourne le dictionnaire contenant l'ensemble des informations sauvegardées dans l'ensemble des fichiers .json contenus dans **dir**.

Vous devrez gérer une option **--output** votre ligne de commande, pour pouvoir spécifier dans quel répertoire sauvegarder les résultats. Si ce paramètre est omis, un répertoire par défaut, **results/** sera utilisé. Si le dossier de sortie n'existe pas, il sera créé par le programme.

B- Outil de recherche par mot clé (4 points):

Faire un module **search.py** qui charge un résultat de crawling grâce a la fonction **crawling.load()**, et permet de chercher parmi elles toutes les urls faisant référence à un mot clé particulier en paramètre.

Pour cela, le module comportera une méthode search qui prend en paramètres :

- un dictionnaire résultat de crawling (ce qui est retourné par **crawling.load()**)
- un mot clé

et retournera la liste des urls contenant ce mot clé, que ce soit dans sa description, son titre, ou ses mots clés.

Le module sera utilisé de la façon suivante :

python search.py --input <repertoire ou se trouve le résultat de crawling> --keyword <mot clé>

Si **--keyword** est omis, l'utilisateur sera invité à le saisir dans la console.

Si **--input** est omis, le répertoire **results/** sera utilisé

Cet appel affichera la liste des urls correspondantes.