UFR SSI, DMIS, Master2 AIDN

Septembre 2020

Projet « RSS-Intelligence »

Sous projet «FEED-collector»

Rappel : point d'éthique : citer les sources que vous exploitez : bibliothèques, licence, etc !

Etape 1 : Effectuer un petit état de l'art à partir du web sur les *reader RSS* « open source » et les systèmes de « découverte/détection » de flux RSS.

Effectuer une étude comparative des *Feed Readers* s'ils existent en open source.

En particulier identifier les mécanismes exploités par les *Feed Readers* pour mettre à jour leurs présentations agrégées lorsqu'une nouvelle information est disponible sur l'un des flux RSS surveillés.

Existe-t-il des méta-moteurs de recherche pour la découverte de flux RSS ? Si oui lesquels ? Produire une synthèse de l'état l'art s'il y a lieu.

Etape 2 : Commencer à développer un « reader » de feeds minimalistes (python3).

Vous pourrez démarrer avec le code HelloFeedParser.py (cf. ENT) qui s'appuie sur la librairie feedparser https://pypi.python.org/pypi/feedparser. Toute autre approche est admissible si celle-ci remplit les objectifs.

Vous renseignerez les champs suivants pour chaque *item RSS* collecté :

- id (identifiant unique) : vous pourrez utiliser un hashing MD5 par exemple.
- url du flux source
- url de la page source (page dont l'item est extrait), si celle-ci existe
- la date
- le titre
- la description
- le résumé
- la langue (récupérer un outil open source de détection de la langue, par exemple https://pypi.python.org/pypi/langdetect?)
- tout autre élément d'information (méta donnée ou contenu) que vous jugeriez utile.

D'autre part, si la page source existe et est accessible via une url, collecter la donnée pointée par cette url, puis convertir son contenu en format ASCII. On exploitera à cette fin une bibliothèque (open source) de fonctions permettant les conversions des formats courrants (PDF, HTM, XML, DOC, RTF, etc, vers TXT (ASCII)).

Par exemple vous pourrez utiliser : Textract https://textract.readthedocs.io/en/stable/

Etape 3 : Prise en compte des points suivants.

Analyse et *post-processing* des données XML/HTML extraites (Parsing)

Toiletter l'information extraite s'il y à lieu en tenant compte de la typographie (UTF-8), de la mise en forme du texte, en utilisant par exemple la bibliothèque Beautifulsoup
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

ou la bibliothèque lxml (import lxml.html.clean Cleaner)
http://lxml.de/

Contrôle des erreurs

Afin que votre code soit robuste, si une erreur survient, en fonction de la gravité de l'erreur, vous serez amené à rejeter soit l'*item RSS* en cours de collecte dans sa globalité, soit certains champs de l'item : par exemple la donnée pointée ou tout autre champ non renseigné. Dans ce cas, indiquer dans votre structure de données que la ou les données posant problème sont manquantes.

Stockage et test

Stocker les item RSS collectés dans des fichiers locaux pour tester le bon fonctionnement de votre application. On utilisera un mécanisme de table de hachage sur disque pour retrouver rapidement à l'aide de la clé « id » tout item RSS collecté. Sous Python, le composant Shelve (https://docs.python.org/3/library/shelve.html) permet de gérer la persistance des données dans des tables hachage sur disque.

Produire des jeux de test montrant le bon fonctionnement des fonctions principales de l'application.

Gestion de flux RSS ou ATOM feeds

Mettre en place un mécanisme de revisite périodique des flux RSS ou ATOM surveillés en adaptant la fréquence de revisite en fonction du type de flux (news (cnn, times, ...), blog, ...

Efficacité

Votre code doit être efficace. Utilisez à cette fin toutes les structures de données et algorithmes adéquats pour accélérer l'exécution chaque fois que possible (Hashtable, structure triée pour la recherche rapide d'éléments dans un ensemble par exemple).

Livrables

Vous devrez fournir :

1) Les diagrammes UML de votre sous-projet, une description des structures de données exploitées
2) Votre code source, avec la documentation (pydoc) et le diagramme des classes
3) Les ressources nécessaires à son exécution
4) Un cas test exécutable
5) Une petite documentation (fichier README) qui décrit ce qu'est censé faire votre programme, comment l'installer, comment l'exécuter et comment le tester.
6) Un exemple documenté présentant les exécutions possible de votre application.
7) La liste exhaustive des codes externes utilisés (librairies, code source, etc.) avec les (c)opyrights précisant les droits d'utilisation