Université de Paris

Interrogation de WIKTIONARY en ligne de commande

Guillaume Wisniewski
guillaume.wisniewski@linguist.univ-paris-diderot.fr

5 avril 2020

1 Programme à réaliser

Le programme que vous réaliserez devra fournir une interface en ligne au dictionnaire WIKTIONARY [1]. En plus des définitions, ce dictionnaire contient une véritable base de données lexicales en indiquant souvent les antonymes, synonymes, traductions, ... L'interface que vous développerez permettra à un utilisateur d'afficher les informations suivantes (lorsqu'elles sont disponibles) sur un mot français (les mots dans d'autres langues contenus dans WIKTIONARY sont à ignorer) :

— sa prononciation ;
— ses catégories morpho-syntaxiques ;
— ses traductions ;
— ses synonymes ;
— ses antonymes.

La figure 1 donne quelques exemples de requêtes que vous devez pouvoir effectuer avec votre programme (le nom des options et la manière dont les résultats sont présentés ne sont que des suggestions).

Pour répondre aux requêtes de l'utilisateur, votre programme devra être capable d'analyse les pages de WIKTIONARY [2] pour en extraire les informations pertinentes.

2 Organisation du travail

On peut distinguer 4 parties dans le programme vous devez réaliser :

1. fr.wiktionary.org
2. Comme pour tous les projets de la WIKIMEDIA FOUNDATION la totalité des pages de WIKTIONARY sont téléchargeables ; une version simplifiée de ce *dump* est disponible sur le site du cours.

```
➜  projet git:(master) ✗ ./wiktionary_query --what pos --for rigolo
['ADJ', 'NOUN']
➜  projet git:(master) ✗ ./wiktionary_query --what pronunciation --for rigolo
ʁi.gɔ.lo
➜  projet git:(master) ✗ ./wiktionary_query --what translations --for rigolo --into br
['farsus']
➜  projet git:(master) ✗ ./wiktionary_query --what translations --for rigolo
{'de': ['witzig'], 'en': ['funny'], 'br': ['farsus'], 'zh': ['滑稽的'], 'el': ['αστείος'], 'sv': ['skojig', 'rolig']}
➜  projet git:(master) ✗ ./wiktionary_query --what antonyms --for rigolo
[]
```

FIGURE 1 – Exemple d'utilisation du programme à réaliser.

1. l'analyse des informations passées en ligne de commande ;
2. l'analyse du *dump* de WIKTIONARY pour extraire les informations pertinentes ;
3. la conception d'une classe (ou de plusieurs) pour stocker les informations extraites ;
4. l'« interrogation » des informations extraites pour afficher la réponse demandée.

Les deux premières parties peuvent être réalisées dès à présent. Vous trouverez dans la suite quelques conseils sur la manière dont vous pouvez traiter ces deux parties.

Analyse des arguments L'analyse des arguments peut se faire de manière très simple en utilisant une bibliothèque adaptée, comme, par exemple, la bibliothèque Common CLI. [3]

Le travail à faire dans cette partie peut être décomposé en 3 étapes :

— installer la bibliothèque et se familiariser avec les différentes possibilités offerte par celle-ci (par exemple en lisant la présentation de celle-ci ou un tutorial) ;
— réfléchir aux différents arguments que devra prendre votre programme et à leurs propriétés (typiquement est-ce qu'ils sont obligatoires ou non) ;
— coder l'analyseur.

On pourra, dans un premier temps, se contenter d'afficher la requête de l'utilisateur en attendant que les autres parties aient été développées.

Analyse du *dump* WIKTIONARY L'extraction d'une page de WIKTIONARY soulève plusieurs difficultés :

— WIKTIONARY est un dictionnaire multilingue : il comporte des entrées pour des mots en langue étrangère. Par exemple la page décrivant le mot cat contient, en plus de la définition du mot français, la définition de ce mot dans une douzaine de langues (y compris le breton). Il faudra donc arriver à identifier uniquement la partie décrivant le mot en français sur la page ;

3. Le support de cours contient une introduction rapide à l'utilisation des librairies en java.

— WIKTIONARY est une base semi-structurée : contrairement aux bases de données classiques dans lesquelles les informations sont clairement identifiées et facile d'accès, les pages du dictionnaires sont construites automatiquement à partir de documents textuels écrit dans un format particulier, le WIKITEXT. Ce format décrit à la fois la mise en page (il y a, par exemple, des commandes pour mettre le texte en gras ou en italique) et permet d'identifier certains éléments sémantique à l'aide de *templates* indiqué à l'aide de la syntaxe {{...}}. Par exemple la prononciation du mot accueil est donnée par le template {{pron|a.kœj|fr}}

Vous trouverez sur le site du cours :

— un dump simplifié de la version française de WIKTIONARY (une version complète [4] ainsi qu'une version ne comportant que les 100 premières pages que vous pouvez utiliser pour développer votre code) [5] ;
— une bibliothèque permettant de lire ce dump et d'itérer sur les pages (au format Wikitext) qu'il contient.

Le travail à a faire dans cette partie nécessite de :

— comprendre comment les informations que vous devez extraire sont représentées dans le code Wikitext (vous pouvez partir d'une page du WIKTIONARY et demander à afficher le code source de la partie qui vous intéresse) ;
— écrire le code permettant d'extraire les informations d'une page et affichez celles-ci dans le terminal (nous verrons dans la partie suivante comment les stocker).

3 Travail à réaliser

Le projet est à faire en binômes. Vous devrez m'envoyer votre code ainsi qu'un court rapport (≃ 10 pages) durant le mois de mai (la date de rendu exacte sera fixée dès que les dates de jury seront connues). Votre rapport devra mettre en avant les difficultés que vous avez rencontrées et les solutions que vous avez trouvées pour résoudre celles-ci. Vous devrez également quantifier la taille du dictionnaire que vous avez réussi à extraire (nombre de mots, nombre de mots avec au moins une traduction, ...)

4. Le dump complet fait 209 Mo, faites attention avant de le télécharger
5. Il est également possible de sauvegarder une page donnée à l'aide de la méthode toXML pour ne travailler ensuite qu'avec celle-ci