TD2 : analyse morpho-syntaxique

Moteur de recherche

LO17

Fatma Chamekh

2021-2022

Objectifs :

L'objectif de ce TD est d'explorer les principales fonctions de la librairie NLTK pour réaliser une analyse morpho-syntaxique. Il comporte deux parties :

- Partie 1 : analyser le corpus en mettant en place la chaine de traitement morpho-syntaxique vue en coure et le TD précèdent afin de construire
- Partie 2 : Réaliser un moteur de recherche.

Exercice 1 :

En se basant sur le corpus de l'exercice 2 du TD1 et la librairie NLTK, réalisez un programme qui permet de réaliser les tâches suivantes :

> - Lecture d'un fichier de texte en anglais
> - Tokénisation en mots
> - Passage en minuscules
> - Suppression des mots de longueur <= 2
> - Suppression des mots vides
> - Sélection des verbes
> - Lemmatisation
> - Décompte du nombre d'occurrences
> - Affichage des 20 verbes les plus utilisés avec leur nombre d'occurrences

Exercice2 :

En 2000, Reuters Ltd a mis à disposition une vaste collection d'articles de Reuters News à utiliser dans la recherche et le développement de systèmes de traitement du langage naturel, de récupération d'informations et d'apprentissage automatique. Nous allons utiliser ce corpus dans cet exercice. Le dossier reuter-collection est disponible sur Moodle. La liste des textes se trouve dans le fichier **collection.lst**. Un fichier avec une extension ".lst" est un fichier contenant une liste des données. Ce corpus est disponible dans NLTK pour plus de détails consultez le lien suivant https://www.nltk.org/book/ch02.html. Pour réaliser cet exercice vous avez besoin des librairies suivantes : `nltk,codecs,pickle,sys.`

1. Ecrire une fonction DetectTOKENS qui transforme la collection de textes en tokens. Elle a comme paramètre le nom du fichier contenant la collection et retourne une liste de tokens. Il faut penser à écarter les mots vides.

2. Écrivez une fonction Index qui construit l'index de la collection et le sauvegarde dans un fichier. Il prendra comme paramètre le nom du fichier contenant la liste des textes de la collection. L'index doit comporter les informations suivantes pour chaque token l'ensemble des documents dans lequel il apparaît. Cette fonction doit retourner un dictionnaire contenant tous les indexs.
   Indication : utilisez un dictionnaire Python et le module pickle pour la sauvegarde (fonction dump() https://www.codegrepper.com/code-examples/python/pickle+dump).

   exemple : pickle.dump(index,open('index.pkl','w')).

3. Nous souhaitons maintenant implémenter un petit moteur de recherche simple. On suppose qu'une requête est une suite de mots qui doivent appartenir aux documents que l'on cherche. Pour cela, on va écrire un programme python qui va, dans un premier temps, charger l'index (cf. la fonction load() du module pickle) puis traiter les différentes requêtes entrées par l'utilisateur. Votre programme demande à l'utilisateur de rentrer sa requête au clavier (méthode raw_input()). Le programme doit être capable de gérer plusieurs requêtes consécutives indéfiniment. Ce programme comporte les fonctions suivantes :

- Une fonction qui prend comme paramètre la liste des mots (string) et un index d'une collection (l'ndex est le résultat de la fonction Index), et qui renvoie la liste des documents de la collection qui contiennent tous les mots de la requête.
- Une fonction qui prend comme paramètre une liste de mots (string) et un index d'une collection (l'ndex est le résultat de la fonction Index), et qui renvoie la liste des documents contenant au moins un mot de la requête.
- Une fonction qui a comme paramètre la requête saisie par l'utilisateur et le dictionnaire d'index. Cette fonction doit appeler les deux fonctions précédentes afin de prendre en considération les deux modes d'indexation. Pour tester le début de la requête vous pouvez utiliser la fonction suivante : `query.startswith('~').`