TD indexing : Document explicatif

MENDY Elie

22 mars 2020

Tables des matières

1 Preambule

Ce dossier est organisé de la manière suivante :

- *un readme*
- *le fichier notebook traitant des parties 1 et 2*
- *le sous-dossier partie 1*
  - *le texte 'la cigale et la fourmi' à indexer*
  - *le script* ***partie_1.py***
- *le sous-dossier partie 2*
  - *le sous-dossier test contenant un dictionnaire renvoyé après 25 d'indexations*
  - *le fichier* ***stop_caractere.txt*** *contenant la liste des caractères à nettoyer dans un mot*
  - *le fichier* ***stoplist.txt*** *contenant la liste des mot à exclure de l'indexation*
  - *le script* ***TD_indexing_partie_2.py***

2 Partie 1 - Indexation de texte

2.1 Methodologie pour créer une stoplist

- Cette proposition de stop liste est basée sur trois sources :
  - le cours de Jym Feat
  - l'introduction du cours de logique
  - un document traitant du tri des mots de lalangue française en classe gramaticales (provenance : université de montpellier)
- La stopliste proposée dans le notebook en partie 1 n'est pas exhaustive ...
- (dans un soucis de rendre le code évolutif) Je propose dans la partie 2 de stocker l'integralité des mots à exclure dans un fichier .txt qui sera lu pour extraire la liste des mots
- Cette solution ouvrira la possibilité d'ajouter/supprimer/modifier des mots de la stoplist

2.1.1 cours de logique

L'introduction du cours aborde le language et les mots qui composent un énnoncé ; Il y a deux types de mots dans une proposition - ceux qui ont une signification propre et peuvent etre sujets ou predicats (noms et verbes) ils sont la matière, le contenu du discours - ceux qui sont co-signifiants et ont pour fonction de déterminer les autres (connecteurs, quantificateurs).

2.1.2 cours de jym feat

la stoplist du cours de Jym feat est la suivante :

stoplist = 'ce de du en le la mais on ou par pas pour qui un une'.split()

Il parait évident que ce qui rend un mot "interessant" dans un index, c'est qu'il constitue 'la matière'/'le contenu' du discours.
L'objectif sera donc le suivant pour completer la stoplist :

- Garder les mot qui ont une "signification propre" , 'la matière du texte' (nom, adjectif, verbes, adverbes).
- Exclure les connecteurs et quantifieurs (déterminants, pronoms, prépositions, conjonctions).

2.1.3 ressources

(pour ce faire, je propose de s'inspirer du document indiqué plus haut dans les sources) :

http ://www4.ac-nancy-metz.fr/ia54-circos/ienjarny/sites/ienjarny/IMG/pdf/memento$_l es_c lasses_d e_m ots.pdf$

2.1.4 définition de la stoplist provisoire

stoplist = "le la les un une des du de ce cet cette ces mon ton son ma ta sa notre votre leur mes tes ses nos vos en au leurs quel quelle quelles quels je tu il lui nous vous ils elle elles me te se qui que quoi ne mais ou est donc or ni car et eh".split()

3 Partie 2 - Indexation de pages web

3.1 Adaptation du code

3.1.1 écriture du dictionnaire dans un fichier

Une série d'indexation sur plus de 5 pages Web rend déjà compliqué l'exploitation du dictionnaire sur un seul terminal. En effet, de par sa taille au vu du nombre de mots indexés et de par la taille des urls, il devient fastidieux de traiter cette donnée. De plus, exploiter un dictionnaire de cette taille s'avère etre gourmand en mémoire tampon.

Je propose donc dans cette partie d'écrire le dictionnaire dans le fichier **dictionnaire.py** à la fin de l'execution pour l'exploiter par la suite.

3.1.2 définition de la stoplist

La stopliste elle aussi devient plus exploitable en étant lu à partir du fichier **stoplist.txt**
On peut de ce fait, ajouter / supprimer des mots dans cette liste.

3.2 Finalisation du robot

3.2.1 fixer une limite de pages à indexer

Dans cette partie, le parcours du web se fait au moyen de la fonction parcourir().

Je laisse le choix à l'uilisateur de fixer la limite du nombre de pages à parcourir au moyen d'une saisie en début de programme *(variable nb_ indexage)*.

3.2.2 améliorations apportées

Dans le soucis d'exploiter les données relative à l'execution du programme je propose trois options qui ne sont pas demandées dans l'ennoncé.

visualisation de l'indexage des pages
Lors du dévelloppement de ce programme, j'ai remarqué que, pour une grande série d'indexation (200 pages et plus par exemple...), l'execution pouvait prendre un certains temps. C'est pourquoi à la ligne 26, dans la fonction parcourir(), j'ai intégrer la possibilité d'afficher les pages indexées au cours de l'execution.
Cette fonctionnalitée permet selon moi à l'utilisateur de s'assurer que le programme ne tourne pas dans le vide, et lui permet d'avoir une visibilité sur les actions du programme. (je propose de passer cette ligne en commentaire si cette fonctionnalitée n'est pas souhaitée).

récupération des données dans des fichiers externes
Je laisse la possibilité de récuperer la liste des pages web parcourues ainsi que la liste des liens extraits sur ces pages par le programme en supprimant les *"#"* dans la partie ***"OPTIONS"***.

3.2.3 évolutivité du programme

Enfin, je propose plusieurs axes d'améliorations de ce programme s'il devait évoluer.

- La saisie par l'utilisateur du lien de départ.
  - Le lien de départ.
  - Laisser le choix de visualiser les pages indexée ou non.
  - Laisser le choix de récuperer les données dans des fichiers externes en fin d'execution ou non
- Gerer les erreurs potentielle de saisie utilisateur
- Optimiser l'utilisation de mémoire tampon en mettant a jour le dictionnaire à la fin de chaque pages indexé et non en l'ecrivant à la fin de l'execution du programme.

- Utiliser les expression regulières pour :
  - Extraire les liens sur les pages web
  - Supprimer les balises du contenu de la page :
  - Nettoyer les blocs script et style.
  - Ne selectionner que les caractères alphabetique pour le nettoyage d'un mot.