DESGRANGES Cédric

Rapport de Projet MAAIN

Le projet se trouve ici :
https://github.com/Kalaww/search_engine
La version 3 correspond à la version finale :
https://github.com/kalaww/search_engine/releases/tag/3.0

Pour savoir comment exécuter le programme, voir le README à la racine du projet.
Il est plus agréable de lire le README bien formaté depuis GitHub :
https://github.com/Kalaww/search_engine

PAGE RANK

Le page rank respecte les contrainte du sujet :

- utilisation d'une matrice au format CLI pour la structure du graphe
- multiplication matrice x vecteur selon le produit transposé en une seule passe des tableaux
- possibilité de choisir les paramètres zap et epsilon

On peut également choisir comment initialiser le graphe pour le calcul du page rank : soit en donnant une valeur 1/N (N étant le nombre de sommet), soit en mettant 1 à un sommet, 0 aux autres.

COLLECTOR

Dictionnaire

J'ai commencé par récupérer les mots imposés par le sujet de la langue française.
Pour voir la démarche que j'ai suivi, un notebook retrace les étapes à ce lien : https://github.com/Kalaww/search_engine/blob/master/generate_dictionary.ipynb

J'ai retiré les accents des mots, les majuscules, et une liste de mots commun de la langue française qui va par la suite alléger les structures de données.
Le fichier contenant ces mots communs se trouve dans data/stopwords.fr.txt depuis la racine du projet ou ici :
https://github.com/Kalaww/search_engine/blob/master/data/stopwords.fr.txt

Le fichier ainsi obtenu se situe dans data/dictionary.fr.csv depuis la racine depuis la racine du projet ou bien ici :
https://github.com/Kalaww/search_engine/blob/master/data/dictionary.fr.csv
Chaque mot possède un identifiant unique ID associé.

Exploration

Le collector se déroule en 2 étapes :

Etape 1

La première étape récupère les titres des pages dans le fichier XML. J'attribu alors un identifiant unique à chaque page. Je ne prend pas l'ID des pages du XML car les numéro ne sont pas continue.
Je sauvegarde ensuite cette liste d'ID = titre de page dans un fichier "pageID_to_title.txt".

Etape 2

Ensuite je parcours le fichier une seconde fois en m'intéressant au contenu de chaque page. Je commence par rechercher les liens vers d'autres pages dans le contenu. Grâce à l'étape 1, je peux associer chaque lien à un ID et ainsi obtenir pour chaque page une liste d'ID

correspondant aux liens contenus dans la page. Au fur et à mesure du parcours des pages, je sauvegarde dans un fichier “page_links.txt” la relation “ID page -> liste des ID des liens”.
En même temps, je transforme le contenu de mots en liste d’ID (ID correspondant aux mots) avec leur fréquence d’apparition dans le contenu. Si le mot est dans le titre de la page, j’ajoute 1.0 à la fréquence de ce mot afin de lui donner un poids très élevé dans les résultats.
Pour chaque mot, je tiens au fur et à mesure du parcours des pages, une liste des N ID de page ayant la plus grande fréquence pour ce mot. (N pouvant être spécifié avec l’option “-p” et est par défaut 10).
Puis à la fin je sauvegarde cette liste de relation “ID mot -> liste de N ID de page par ordre décroissant” dans un fichier “words_appearance.txt”.

Le garder que N page par mot permet d’éviter d’obtenir un fichier “words_appearance.txt” trop volumineux et évite de stocker trop de relation “ID mot -> ID page” par ID mot. Lors de la recherche, ce qui intéresse ce sont les résultats les plus pertinents, pas d’obtenir tous les résultats possible pour une recherche.

On obtient ainsi à la fin du collector 3 fichiers :

- pageID_to_title.txt : relation pageID -> page title
- page_links.txt : relation pageID -> links page id
- words_appearance.txt : relation word id -> list(pageID)

SEARCH

Requête

La requête se compose d’une suite de mots (séparé par un espace).
Lors d’une recherche, les mots de la requête sont traduit en ID correspondant.
On récupère ensuite, via le fichier “words_appearance.txt”, les N ID de page correspondant à ce mot (ou bien l’union des X ensembles d’ID de page, X correspondant aux X mots de la requête). Si un mot n’est pas dans le dictionnaire “data/dictionary.fr.csv”, il est ignoré.
On utilise ensuite le résultat du page rank, qui a été sauvegardé dans un fichier, pour ordonné ces N ID de page par ordre décroissant de page score.
On finit par traduire ces N ID de page par leur titre correspondant. Ici, un simple parcours du fichier “pageID_to_title.txt” permet de traduire. Pas besoin de stocker en mémoire la liste des relations “pageID -> titre”.

Interface

Le résultat obtenu est affiché sur la sortie standard et dans un fichier nommé selon les mots utilisés de la requête.
Une fois une recherche effectuée, le programme propose d’en faire une suivante.

Améliorations possibles

Je n’ai pas stocker la fréquence d’apparition des mots dans les pages, la fréquence m’a juste permis d'ordonner de façon décroissante les ID de page et de récupérer les N premiers.
On peut améliorer la recherche en prenant les mots dérivés pour chaque mot de la requête : la conjugaison des verbes, le pluriel, pourquoi pas des synonymes.

Déploiement

Toute les commandes pour installer et exécuter le programme sont précisées dans le README.
Une commande “all” permet d’exécuter les 3 parties (page rank, collector et search) en une seule fois avec seulement le fichier XML wikipedia en entré (utilisation détaillée dans le README).