Recherche d'informations

FELIN Rémi
VIGHI Alexis
M2 MIAGE IA2

Pour décrire notre algorithme de recherche d'information, nous allons fonctionner par fonction. En effet, nous allons vous décrire les fonctions créés une à une, ensuite, nous pourrons tout mettre ensemble afin d'expliquer clairement le processus.

Nous avons différents types de fonctions :

1) **Des fonctions de preprocessing (preproc, tfidf)**

2) **Des fonctions annexes (le traitement des requêtes)**

3) **Les fonctions principales du main, qui mettent en relation les fonctions précédemment citées.**

Nous allons commencer par nos fonctions de preprocessing.
La première fonction, au nom de « preproc », nous permet de traiter les documents en entrées.
Nous allons commencer par :

1) *utiliser un tokenizer.*
2) *Ensuite, si nous voulons traiter les stopword, on définit le dictionnaire de stopword, et on va appliquer pour tous les mots (token) qui ne sont pas des stopword un lemmatizer, afin de garder seulement le lemme.*
3) *On peut enfin retourner notre liste de mots preprocessés.*

La seconde fonction est la fonction tfidf.
Cette fonction permet de discriminer les mots afin d'accorder plus d'importance aux termes les plus fréquents. Elle prend en entrée un dataframe comprenant le résultat de la requête avec la fréquence de chaque mot trouvé dans chaque document.

1) *Nous allons appliquer la fonction Tfidf de Sklearn, afin de classer la pertinence des documents suivant les mots de la requête*

On va pouvoir passer aux fonctions annexes.
Dans ce script python, nous allons retrouver les différentes types de requêtes.
En premier lieu, nous allons tester la requête, si on trouve un mot clé de tel ou tel requête, on va appeler la fonction dédiée.

1) *Ces fonctions dédiées sont :*

a. ***Boolean request****, qui prend en entrée la requête et*
b. ***Complex_request****, qui prend en entrée la requête.*

_ ***Boolean request*** va appliquer le même preprocessing que l'on a appliqué au texte, afin de pouvoir retrouver les termes de la requête.
Si on trouve un mot en majuscule dans la requête, on ne traite pas ce mot (ce sera le terme déterminant de la requête (e.g. AND, OR, NOT), sinon, on applique notre preprocessing. On peut ensuite renvoyer la requête preprocessé.
Pour les **AND**, on retire les documents qui n'ont aucun mot correspondant, De même, on supprime les fichiers de notre résultat pour lesquelles nous avons des correspondance avec les mots exclus de la requête à l'aide du préfixe **NOT**. Ensuite on va appliquer tfidf afin de récupérer les coefficients de pertinence des documents suivant les mots de la requête.
On va ensuite classer les documents suivant le « score » obtenu de tfidf
Dans notre version du code, nos requêtes retournent toutes des résultats hormis la dernière (qui renvoie un log avec une balise DEV). Le mot clé « OR » n'est pas géré dans cette version compte tenu de la complexité qu'elle ajoute aux requêtes.

_ ***Complex request*** va appliquer le même preprocessing que l'on a appliqué au texte, afin de pouvoir retrouver les termes de la requête.
Les espaces sont considérés comme des **AND**, il faut donc retrouver tous les mots des requêtes complexes. On retire ainsi les documents qui n'ont aucun mot correspondant.
Ensuite on va appliquer TF-IDF afin de récupérer les coefficients de pertinence des documents suivant les mots de la requête et on classe les documents suivant le « score » obtenu.

On va maintenant pouvoir passer à nos fonctions principales, qui mettent tout en lien.
La première va nous créer un matrice d'incidence.
Pour ce faire, on itère sur les documents *.txt* afin de les traiter et de les rajoutés dans un dataframe, préalablement créé (ici on utilise la fonction preproc de notre package tools.)
On sérialise le résultat à l'aide de la librairie ***pickle*** et on sauvegarde le fichier.

La seconde fonction va nous permettre de créer l'index inversé, on part de notre matrice d'incidence. On itère dessus afin de lister les mots et la liste des documents associés dans lesquelles apparaissent ce mot. On le fait sous forme de liste de dictionnaire, qu'on va, encore une fois sérialiser avec la librairie ***pickle*** et l'enregistrer dans un fichier dédié.