TD 1 - Indexation

Exercice 1 – Pondération td-idf

Soient :
- un document qui contient le texte "maison belle maison"
- une collection de 100 documents
- le terme "maison" apparaît dans 20 documents pour un nombre d'occurrences de 35 au total
- le terme "belle" apparaît dans 35 documents pour un nombre d'occurrences de 40 au total.

Q 1.1 Quelle est la pondération "tf-idf" des termes "maison" et "belle" pour le document ? Commentez les valeurs obtenues.

Exercice 2 – Structure des fichiers d'indexation et pondération

On considère la collection de documents suivantes :
- Doc 1 : the new home has been saled on top forecasts
- Doc 2 : the home sales rise in july
- Doc 3 : there is an increase in home sales in july
- Doc 4 : july encounter a new home sales rise

Ainsi qu'une liste de mots vides : the, a, an, have, be, on, behind, under, there, in, on, ...

Q 2.1 Identifier l'ensemble des termes devant être indexés pour chaque document.

Q 2.2 Calculer les poids des termes pour chaque document selon la pondération tf.

Q 2.3 Calculer les poids des termes pour chaque document selon la pondération idf.

Q 2.4 Modéliser l'index et l'index inversé pour cette collection de documents en considérant la pondération $tf - idf$.

Q 2.5 Normaliser la pondération Calculer les normes de chaque vecteur document.

Q 2.6 Quels sont les documents retournés pour les requêtes suivantes :
- Q1 : sales home
- Q2 : july new

Dérouler le processus d'interrogation des index (index et index inversés) vous permettant de trouver les documents pertinents. Cela vous aidera pour l'implémentation.

Exercice 3 – Construction des fichiers

Maintenant que vous avez compris la structure des fichiers index, l'objectif ici est de réfléchir comment l'index inversé est construit lors de l'étape d'indexation.
Remarque : La construction du fichier index semble naturelle. Prenez quand même le temps d'être sûr de savoir l'implémenter...

Q 3.1 Soit un ensemble **statique** de documents $\{d_1, d_2, ..., d_n\}$, quelles sont les étapes intermédiaires permettant de construire l'index inversé de cette collection.

Q 3.2 Pour effectuer l'étape de tri, on utilise généralement une indexation par bloc à base tri. Cette étape consiste à stocker les paires dans un tampon (*buffer*) tout au long du parcours de la collection. Si la collection est parcouru ou si le *buffer* est plein, on arrête le parcours et on trie la liste de paires contenues dans le *buffer*, d'abord par les identifiants des termes, ensuite celui des documents. On regroupe ensuite les paires par termes. On répète ce processus jusqu'à lecture complète de la collection et on fusionne ensuite toutes

les listes de paires triées. Faire un schéma et écrire l'algorithme permettant d'effectuer une indexation par bloc à base de tri.

Q 3.3 Pour fusionner les listes triées, on ouvre toutes les listes intermédiaires, chaque liste est associée à un pointeur. On choisit l'identifiant de terme le plus petit qui n'a pas encore été traité. On fusionne toutes les listes correspondant à cet identifiant et on descend le pointeur dans les listes intermédiaires. Faire un schéma et écrire l'algorithme permettant la fusion des listes intermédiaires.

Q 3.4 Ecrire l'algorithme MapReduce permettant de réaliser l'index inversé.

Q 3.5 Lorsque la collection de document est dynamique, il est nécessaire de mettre à jour l'index. Il est alors possible de reconstruire l'index dans sa globalité, mais c'est très coûteux en temps et peut dégrader les performances du moteur de recherche. Que proposez-vous pour régler ce problème ?