Recherche d'Information

M2 DAC

TME 2. Modèles de Recherche: le Modèle Vectoriel

Au cours de ce TME, le travail à réaliser est d'implémenter un modèle simple de recherche d'information à partir d'un index de documents : le modèle vectoriel. L'ensemble des classes à définir au cours de ce TME seront placées dans un package **modeles** dans lequel les rejoindront d'autre modèles dans les TMEs suivants (modèles de langue, modèle Okapi, etc...).

1 Modèle vectoriel : Représentation de textes

Le modèle vectoriel consiste à considérer les textes, documents et requêtes, comme des vecteurs définis dans l'espace des termes et effectuer des opérations de produits scalaires sur ces vecteurs afin de déterminer des mesures de similarité entre ces textes. Pour ce faire, une fois que l'on dispose d'un index contenant les fréquences d'occurences des termes du vocabulaire dans les documents (ce dont vous devez maintenant disposer), il s'agit de décider d'un schéma de pondérations des termes appartenant aux différents textes. Cette pondération peut suivre différents schémas tf.idf. Par exemple avec $w_{t,d}$ le poids du terme t dans le document d et $w_{t,q}$ le poids du terme t dans la requête q:

- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = 1$ si $t \in q$, 0 sinon;
- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = tf_{t,q}$
- $w_{t,d} = tf_{t,d}$ et $w_{t,q} = idf_t$ si $t \in q$, 0 sinon;
- $w_{t,d} = log(1 + tf_{t,d})$ si $t \in d$, 0 sinon et $w_{t,q} = idf_t$ si $t \in q$, 0 sinon;
- $w_{t,d} = log(1 + tf_{t,d}) \times idf_t$ si $t \in d$, 0 sinon et $w_{t,q} = log(1 + tf_{t,q}) \times idf_t$ si $t \in q$, 0 sinon;

Écrire un classe générique Weighter qui a accès à l'objet Index et contenant au moins les méthodes suivantes:

- La méthode abstraite $getDocWeightsForDoc(String\ idDoc)$ qui retourne les poids des termes pour un document dont l'identifiant est passé en paramètre;

- La méthode abstraite $getDocWeightsForStem(String\ stem)$ qui retourne les poids du terme stem dans tous les documents qui le contiennent;
- La méthode abstraite $getWeightsForQuery(HashMap < String, Integer > \ query)$ qui retourne les poids des termes pour la requête dont les tf sont passés en paramètres.

Déinir ensuite diverses classes Weighter spécifiques correspondant à différents scémas de pondération tels que ceux proposés ci-dessus.

2 Modèle vectoriel : Mesure de Similarité

Définir une classe IRmodel a un accès à l'index utilisé et contient à minima les méthodes suivantes:

- La méthode abstraite $getScores(HashMap < String, Integer > query)$ qui retourne les scores des documents pour une requête passée en paramètre;
- La méthode $getRanking(HashMap < String, Integer > query)$ qui retourne une liste des couples (document-score) ordonnée par score décroissant. La liste contient tous les documents de la collection, ceux qui ne contiennent pas de termes de la requête doivent figurer dans la liste associés à un score minimal. Les documents ayant des scores égaux doivent être ordonnés de manière aléatoire.

Définir une classe Vectoriel dérivant IRmodel. Cette classe possède un Weighter, implémente la méthode *getScores* selon une variable booléenne *normalized*:

- Si *normalized* est faux: *getScores* calcule les scores des documents en réalisant un simple produit scalaire de leurs poids avec ceux de la requête;
- Si *normalized* est vrai: *getScores* considère des scores cosinus (produit scalaires entre vecteurs de norme 1) entre la représentation des documents et celle de la requête. Attention: les normes des vecteurs des documents ne doivent pas être recalculées à chaque nouvelle requête.