Travaux Dirigés d'informatique linguistique n°2

Cours d'Introduction à l'informatique linguistique

—Licence Informatique 3ème année—

Représentation vectorielle des textes et recherche documentaire

Le but de ce TP est de manipuler la notion de représentation vectorielle des textes dans le cadre de la recherche documentaire.

---

1 Ressources

Pour cette séance, nous vous fournissons des ressources que vous pourrez trouver dans le répertoire :
`http://igm.univ-mlv.fr/~mconstan/enseignement/l3/infolingu/data/`

Vous y trouverez :

- `textSpaceVector_td2.py`, un module python (avec des trous) permettant de charger et manipuler des collections de documents textuels sous la forme de vecteurs
- `td2.py`, un exemple d'utilisation du précédent module, pour lequel vous aurez besoin de créer trois fichiers texte `data/corpus.txt`, `data/corpus2.txt` et `data/corpus3.txt`.
- `reuter-collection.zip`, une archive contenant un ensemble de textes anglais (dépêches Reuter)

2 Manipulation d'un document

1. Créer un fichier texte en anglais (copier-coller une dépêche Yahoo par exemple)
2. A l'aide de la classe `textVector` et de son constructeur, charger ce texte sous la forme d'un vecteur. Vous utiliserez l'expression rationnelle que vous avez utilisée dans le TD 1 pour tokeniser le texte (paramètre `pattern` du constructeur).
3. Afficher le vecteur associé au document à l'aide de la méthode `printVector`, de telle manière que les composantes soient les fréquences des tokens-mots.

3 Manipulation d'une collection de documents

1. Créer cinq fichiers textes (dépêches Yahoo).
2. A l'aide de la classe `textVectorCollection`, charger ces trois fichiers sous la forme de vecteurs (utiliser le paramètre `list` de son constructeur).

3. Afficher pour chaque token-mot de la collection, le nombre de documents dans lequel il apparaît. Vous utiliserez la méthode `printSortedTokens` de la classe `textVectorCollection`.
4. En faisant varier les paramètres `with_freq` et `with_idf` de la méthode `printVector` de la classe `textVector`, afficher les vecteurs des textes de la collection de trois manières :
   - chaque composante est la fréquence du token-mot associé
   - chaque composante est le nombre de documents de la collection dans lesquels apparaît le token-mot associé
   - chaque composante est la pertinence tf.idf du token-mot associé dans le contexte de la collection de textes

   Ne pas oublier d'affecter la collection au paramètre `collection` dans `printVector`.

4 Application de requêtes sur une collection de documents

1. Charger maintenant 1000 textes de la collection Reuter qui vous est fournie (cf. ci-dessus). Pour cela, vous créerez un fichier qui listera les chemins de ces textes (un chemin de fichier par ligne) en utilisant les programmes `find` et `head` dans un shell. Vous utiliserez le paramètre `file` du constructeur de `textVectorCollection`.
2. Afficher pour chaque token-mot de la collection, le nombre de documents dans lequel il apparaît. Vous utiliserez la méthode `printSortedTokens` de la classe `textVectorCollection`. Que constatez-vous ?
3. Compléter la méthode `scalarProduct` de `textVector`, qui permet de calculer le produit scalaire entre le textVector `self` et le textVector `vector`. Les paramètres `with_freq` et `with_idf` permettent de spécifier le type de composante des vecteurs (ex. fréquence, pertinence tf.idf, ...). Le paramètre `with_norm` indique si l'on doit normaliser le résultat (i.e. un résultat compris entre 0 et 1).
4. Appliquer différentes requêtes à la collection de documents à l'aide de la méthode `applyQuery` de `textVectorCollection`, en faisant varier la nature de composantes des vecteurs des textes (poids binaire, fréquence, pertinence tf.idf,...). Que constatez-vous ?