ENSICAEN - 3A Info, Image
TP 2 - Web Sémantique

Lagarrigue Lucie & Vimont Ludovic
September 29, 2015

1 Structure

Nous avons donc réaliser deux classes afin de réaliser notre implémentation. La classe Document et la classe Word. Voici la structure de la classe Document :

```
public class Document {
  private String documentName;
  private HashMap<Word, Double>  indexes;
  private double sumPonderationSquare;

  ...
}
```

On retrouve donc trois attributs :

- Le nom du document
- Une HashMap contenant tout les mots du document, que l'on va associer à la valeur du TF-IDF de ce mot.
- On stock également, la valeur de la somme des pondérations au carré, comme cette dernière ne dépend pas des éléments de la recherche, on peut la calculer en avance afin d' accélerer la phase de recherche.

Concernant la classe Word, voici sa structure :

```
public class Word {
  private String name;
  private HashMap<String, Integer>  occurences = new HashMap<>();
  ...
}
```

On y voit donc deux attributs :

- Le nom du mot
- Le nom du document dans lequel le mot est présent et le nombre d'occurences dans ce document donné.

Enfin, pour réaliser la phase d'indexation, nous utilisons les trois variables suivantes :

```
LinkedList<Document> documents = new LinkedList<Document>();
HashMap<String, Integer> allWords = new HashMap<String, Integer>();
...
LinkedList<String> stopList = hfStopList.getWords();
```

- La première c'est bien sur tout simplement la liste de tout les documents que l'on désire indexer.
- La seconde, c'est une liste de tout les mots présents dans chaque document. Elle nous permet tout simplement de compter le nombre de fois où un mot est présent dans chaque document.
- Enfin, la dernière c'est une liste de mots contenu dans une stop-liste (celle proposée sur la plateforme). Lors de l'ajout d'un mot dans un document, on vérifie que ce dernier n' est pas présent dans la stop-list.

Enfin, la dernière phase de l'indexation permet d'enregistrer les résultats calculés dans un fichier texte, dont voici un extrait :

```
corpus/Automne_GA.txt 159,58
cache 1,61
hameaux 2,3
...
pauvres 1,61
mourir 2,3
###
```

On commence par fournir le nom du document, ainsi que la valeur du coefficient de Salton correspondant au document. Ensuite, on va simplement lister les mots avec la valeur de leur pondération. Enfin, on termine le fichier par un ensemble de caractères afin de prévenir de l'analyse d'un nouveau document.

2 Défaut(s) de notre structure

Un des principales défaut de notre structure, c'est le stockage des mots par HashMaps dans la la classe Document, lorsque l'on va parcourir le fichier afin de compter les occurences d'un mot la HashMap n'étant pas prévu pour pouvoir récupérer une clé grâce à sa valeur, cela nous oblige à faire un parcours de cette dernière, afin de pouvoir mettre à jour le mot existant au lieu d'en recréer un nouveau.

3 Implémentations

Réalisé : - indexation - recherche - stop list - ligne de commande - ?

4 Conclusion