Projet de TP en Recherche d'information

Ce projet de TP vise à vous familiariser avec les procédures (modules) de base en recherche d'information (RI) à savoir : (1) l'indexation des documents, (2) la pondération des termes et (3) l'appariement requête-document. Dans ce TP, on traite la collection CACM. Cette collection est stockée dans le fichier ~nie/Smart/colls/cacm/cacm.all. Nous allons réaliser, avec le langage Python, le modèle Booléen, le modèle Vectoriel et l'évaluation des formules du modèle Vectoriel.

1. La collection à utiliser

On utilise la collection CACM. Dans cette collection, les documents sont séparés par des marqueurs. Chaque document peut contenir un ou plusieurs champs. Chaque champ à l'intérieur d'un document est aussi identifié par un marqueur.

Voici le premier document de la collection CACM :

```
.I 1
.T
Preliminary Report-International Algebraic
Language
.B
CACM December, 1958
.A
Perlis, A. J.
Samelson,K.
.N
CA581203 JB March 22, 1978  8:28 PM
.X
100     5       1
123     5       1
164     5       1
1       5       1
1       5       1
1       5       1
205     5       1
210     5       1
214     5       1
1982    5       1
398     5       1
642     5       1
669     5       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
1       6       1
165     6       1
196     6       1
196     6       1
1273    6       1
1883    6       1
324     6       1
43      6       1
53      6       1
91      6       1
410     6       1
3184    6       1
```

Voici le document N ° 20 de la collection CACM :

```
.I 20
.T
Accelerating Convergence of Iterative
Processes
.W
A technique is discussed which, when applied
to an iterative procedure for the solution
of an equation, accelerates the rate of
convergence if the iteration converges and
induces convergence if
the iteration diverges. An illustrative
example is given.
.B
CACM June, 1958
.A
Wegstein, J. H.
.N
CA580602 JB March 22, 1978  9:09 PM
.X
20      5       20
20      5       20
20      5       20
```

où les éléments en gras sont des marqueurs. En particulier dans cette collection CACM, on a les marqueurs suivants :

.I indique qu'on commence un nouveau document
.T introduit le champ « titre »
.W introduit le champ « résumé »
.B introduit le numéro où l'article est publié
.A introduit les auteurs
.N identifie quand ce document a été ajouté dans la collection
.X identifie les références

Ces marqueurs doivent apparaître au début d'une ligne pour être reconnus comme marqueur. Pour ce TP, nous ne nous intéressons qu'aux champs .I, .T et .W. Les autres champs sont ignorés. Dans la procédure de l'indexation, on doit d'abord identifier un document (dans la collection CACM, en vérifiant qu'une ligne commence par .I). Une identification unique (le nombre après .I) est alors associée à ce document.

Á l'intérieur de chaque document, on doit d'abord identifier chaque champ (aussi en vérifiant les marqueurs). Pour un champ qu'on va traiter (.T ou .W), on effectue les traitements suivants :

- Tokenisation : ce traitement consiste à séparer chaque ligne en une séquence de mots. Autrement dit, il reconnaît les mots dans une séquence de lettres ou des symboles. Pour ce TP, on procède d'une façon simplifiée : on considère que les espaces et toutes les ponctuations constituent un séparateur de mots.
- Comparaison avec Stoplist (par NLTK) : Pour chaque mot reconnu, il faut le comparer avec une Stoplist qui contient tous les mots non-significatifs (en utilisant NLTK dans Python). Si un mot fait partie de cette Stoplist, on l'ignore.

Á la fin de ces étapes, on doit obtenir le résultat suivant : Pour chaque document, on doit connaître quels mots y apparaissent, et avec quelle fréquence. C'est donc une structure comme suit :

numéro du document -> liste de <mots et fréquences>

Par exemple, pour le document exemple, on devrait avoir le résultat comme suit :

```
1 -> {<preliminary, 1>, <report, 1>, <international,1>, <algebraic, 1>, <language, 1>}
```

ou si vous voulez obtenir plus d'efficacité, de créer une liste triée comme suit :

```
1 -> {<algebraic, 1>, <international,1>, <language,1>, <preliminary, 1>, <report, 1>}
```

1.2. Création d'un fichier inversé par fréquences

L'utilisation d'un fichier inversé peut grandement améliorer l'efficacité de la RI. On vous demande donc de convertir votre résultat de l'indexation en un fichier inversé. Ce fichier inversé correspond à la structure suivante pour chaque mot et chaque document :

(Mot, Num de document) -> fréquence

1.3. Fonctions d'accès

Pour connaître quels sont les mots qui apparaissent dans un document et avec quelles fréquences, et quels documents répondant à une requête simple d'un seul mot et avec quelles fréquences, on doit programmer deux fonctions d'accès de base. La première accepte un numéro de document, et retourne la liste de mots avec leurs fréquences. La deuxième accepte un mot, et retourne la liste de documents et les fréquences d'occurrences du mot.

2. Réalisation du modèle Booléen

La deuxième partie concerne la recherche dans la base de documents indexés en utilisant le modèle booléen. Vous devez établir une interface, qui peut recevoir une requête au format booléen, et retourne une liste de documents pertinents, en utilisant la formule d'appariement du modèle booléen vue en cours.

3. Création d'un fichier inversé par poids

Une fois le fichier inversé des fréquences est créé, créer le fichier inversé des poids en utilisant la formule de pondération TF*IDF: poids(ti, dj)=(freq(ti,dj)/Max(freq(dj))*Log((N/ni) +1)

4. Réalisation du modèle Vectoriel

Cette partie concerne la recherche dans la base de documents indexés en utilisant le modèle vectoriel. Vous devez établir une interface, qui peut recevoir une requête au format vectoriel, et retourne une liste de documents pertinents, en utilisant les formules d'appariements du modèle vectoriel vues en cours. Vous devez implémenter quatre fonctions d'appariements : 1- Produit Interne ; 2-Coef de Dice ; 3-Cosinus ; 4-Jaccard.

5. Evaluation du modèle vectoriel

En utilisant les deux fichiers de la collection CACM : (query.text et qrels.text), développer une interface qui permet de calculer le rappel et la précision pour chacune des formules du modèle vectoriel, pour une requête de test du fichier query.text.

6- Rapport à remettre

Vous devez remettre un rapport écrit, qui contiendra (au minimum) les points suivants :

- Explication de vos algorithmes
- Format (structure) de vos fichiers d'indexation, avec des exemples.
- Montrez quelques résultats de requêtes (simples et composés) pour chacun des modèles : choisissez des requêtes qui illustrent bien les possibilités de votre implémentation.
- Comparaison des résultats du modèle booléen et du modèle vectoriel, et entre les quatre formules du modèle vectoriel.
- Analyse et discussion de vos résultats.