Fondements de la Recherche d'Information

Projet: Moteur de Recherche Ad-hoc de documents

L'objectif de ce projet est de construire (dans le langage de votre choix) un moteur de recherche sur une base statique de documents textuels et qui implémente les trois modèles classiques de la RI vus en cours : **le modèle booléen, le modèle vectoriel et le modèle probabiliste**. Ce moteur de recherche sera testé sur deux bases classiques : la base CACM (Communications of the ACM)[1] et la base Wikipedia-INEX 2007 [2]. La base CACM est une base assez classique en recherche d'information qui contient les titres, auteurs et résumés d'un ensemble d'articles scientifiques. Un des avantages principaux de cette base est qu'un ensemble de requêtes et de jugements de pertinence est disponible. La base Wikipedia-INEX 2007 est une base un plus réaliste en terme de taille. Si vous avez le temps, vous pourrez aussi tester votre moteur sur d'autres bases classiques utilisées en RI et qui seront décrites en cours (GOV2...).

L'objectif est de partir de l'information textuelle brute présente dans ces bases de données et de programmer l'ensemble des étapes permettant de construire un moteur de recherche opérationnel. Ces étapes sont les suivantes : pré-traitement linguistique, indexation et implémentation des modèles de recherche. Vous développerez aussi un module d'évaluation de votre moteur de recherche.

Dans la suite, nous décrivons les différentes étapes pour la collection CACM mais vous devrez aussi traiter la collection Wikipedia-INEX 2007 en prenant plus d'initiatives. Il faudra donc faire attention à la généricité et à la modularité de la solution proposée.

1 Collection CACM

Cette base contient 6 fichiers :

- `cacm.all`: contient le texte de l'ensemble des fichiers

[1] (`http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/` ou `http://www.search-engines-book.com/collections/`)

[2] (`http://www-connex.lip6.fr/~denoyer/wikipediaXML/`)

- `common-words`: stop-list que vous pouvez utiliser pour les pré-traitements
- `query.text`: contient 64 requêtes vectorielles
- `qrels.text`: contient les jugements de pertinence correspondant aux requêtes

2 Indexation des documents

2.1 Premières étapes

On suppose un ensemble de documents stockés dans un ou plusieurs fichiers. Dans le cas de cette collection, on peut ne manipuler qu'un seul fichier qui contient un ensemble de documents (`cacm.all`). Dans ce fichier, 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. Les marqueurs de la collection CACM sont les suivants :

- .I indique qu'on commence un nouveau document. Identifiant du document.
- .T introduit le champ titre
- .W introduit le champ résumé
- .B introduit la date de publication
- .A introduit les auteurs
- .N identifie quand ce document a été ajouté dans la collection
- .X identifie les références
- .K identifie des mots clés

Ces marqueurs doivent apparaître au début d'une ligne pour être reconnus comme marqueur. Pour ce projet, nous ne nous intéressons qu'aux champs .I, .T, .W et .K. Les autres champs sont ignorés.

Dans la procédure d'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.

A 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 ou .K), 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 de symboles. Pour ce projet, on procède d'une façon simplifiée : on considère que les espaces et toutes les ponctuations (caractères non alphanumériques) constituent un séparateur de mots.

- Comparaison avec une stop-liste (`common-words`). Pour chaque mot reconnu, il faut le comparer avec une stop-liste qui contient tous les mots non-significatifs. Si un mot fait partie de cette stop-liste, on passe au prochain mot. Note : Un traitement qu'on fait souvent avant cette comparaison est la transformation de majuscule en minuscule. Cette transformation est aussi demandée.

- Pour un mot significatif, on peut procéder à la lemmatisation ou à une troncature. Dans ce TP, on peut négliger cette étape pour la collection CACM mais c'est une étape utile pour toute autre collection plus réaliste. Tous les mots significatifs que vous rencontrez sont considérés comme terme d'index.

- Statistique : Dès qu'un mot significatif est rencontré, vous devez incrémenter de 1 sa fréquence d'occurrences dans le document.

- Une fois que chaque champ est traité, une liste de mots avec leur fréquence d'occurrences est établie. Dans ce projet, on ne vous demande pas de séparer les champs .T et .W et .K. Ces trois champs sont traités ensemble, comme s'ils ne constituaient qu'un seul champ. Donc, vous pouvez établir une seule liste de mots avec leurs fréquences pour un document.

A la fin de ces étapes, on doit avoir pour chaque document quels mots y apparaissent et avec quelle fréquence. La structure est :

- numéro document $->$ liste de <mots,fréquence>

2.2 Pondération des termes dans les documents

Plusieurs méthodes de pondération devront être testées dans le cadre de ce projet parmi lesquelles :

- la pondération tf-idf

- la pondération tf-idf normalisée

- la fréquence normalisée : $w_{t_i,d} = \frac{tf_{t_i,d}}{max_{t_j \in d}(tf_{t_j,d})}$

- ...

2.3 Création du (des) fichier(s) inversé(s)

Convertir le (les) résultats de l'indexation en un fichier inversé.

- mot $->$ liste de <document contenant le mot, sa fréquence...>

Pour la création du fichier inversé, vous prendrez bien soin de réfléchir et de justifier le choix de la structure de données retenue pour l'implémentation. Dans le cadre de la collection CACM, de très petite taille, les choix d'implémentation ne sont pas très *importants* mais ils le sont dès que les collections grandissent et notamment dès qu'il n'est plus possible de construire intégralement l'index en mémoire.

3 Recherche des documents

3.1 Modèle vectoriel

On s'intéresse au modèle vectoriel avec le calcul de similarité par la mesure du cosinus.

Vous devez réaliser le processus de recherche en vous basant sur les résultats d'indexation de l'étape précédente. Votre programme doit accepter une requête, et produire une liste ordonnée des documents comme réponse.

Pour le modèle vectoriel, la requête est entrée sous une forme libre. Votre programme doit indexer cette requête comme pour l'indexation de documents, et évaluer cette requête en utilisant le fichier inversé. La similarité est calculée selon la mesure cosinus. Vous êtes bien sûr invité à tester d'autres mesures de similarité pour comparaison.

3.2 Modèle booléen

Pour le modèle booléen, la forme de la requête est une expression booléenne. Vous devez accepter les opérateurs : ET, OU, et la négation. La façon dont vous entrez une requête booléenne (y compris la façon d'entrer les opérateurs logiques) est laissée libre. Dans le cas d'une requête booléenne, les opérandes sont des mots simples. On suppose que ces opérandes ne sont pas des mots vides (stopwords). L'évaluation d'une requête booléenne doit aussi aboutir à une liste ordonnée des documents (il vous est demandé de vous documenter sur comment faire pour ordonner les résultats d'une requête booléenne).

4 Modèle probabiliste

L'objectif de cette partie est d'implémenter un système de recherche basé sur le modèle probabiliste et en particulier sur le Binary Independance Retrieval.

Dans un premier temps, on effectuera une estimation a priori de p_i et q_i:

- On donne une valeur fixe à p_i

- Pour estimer q_i, on utilise la fraction des documents de notre collection qui contient le terme t_i

Si vous avez le temps, vous pouvez aussi, dans un deuxième temps, proposer d'autres estimations de p_i et q_i et utiliser le modèle probabiliste pour déterminer quels sont les mots significativement significatifs pour qu'on les indexe.

5 Evaluation

Afin d'évaluer vos différents systèmes de recherche, i.e. les systèmes correspondants aux différents modèles implémentés, deux types d'évaluation sont proposés :

- les mesures de **performances** : temps de calcul pour l'indexation, temps de réponse à une requête et occupation de l'espace disque par les différents index.
- les mesures de **pertinence** en utilisant les requêtes et les jugements de pertinence disponibles.

  En particulier, on implémentera les principales mesures vues en cours dont :

  - Précision et rappel
  - F-Measure, E-Measure et R-Measure
  - Mean Average Precision

L'objectif est d'une part d'évaluer de manière quantitative vos systèmes et d'autre part de vous comparer entre vous.

6 Collection Wikipedia-INEX 2007

De manière plus autonome, indexer la collection Wikipedia-INEX 2007 et tester vos moteurs de recherche sur cette collection. On s'intéressera uniquement aux mesures de performances. Pour cette collection, vous pourrez aussi prendre en compte l'aspect multi-linguisme ou encore la catégorisation des documents.

7 Améliorations et extensions

Toute initiative d'amélioration et d'extension de ce projet se basant sur des éléments du cours est la bienvenue (compression des index, mise en place de retour de pertinence, ordonnancement par apprentissage...).

8 Modalités

- Le travail peut être effectué par groupe de 2 personnes.
- Vous devez remettre vos programmes sources ainsi qu'un petit rapport décrivant votre réalisation lors de la dernière séance du cours.
- Une séance de démonstration pourra être organisée afin que vous puissiez montrer le fonctionnement de vos programmes.
- Le projet comptera pour 70 % de la note globale. La notation comportera les points suivants :
  - Fonctionnement correct de vos programmes (conversion des poids et recherche). Le code source rendu devra pouvoir être exécuté et testé (fournir un readme). La validation sur les collections CACM et Wikipedia sera bien sûr aussi un aspect important de l'évaluation.
  - Choix des structures de données et des algorithmes utilisés
  - Clarté et utilité du rapport