CentraleSupélec 2017-2018 **Projet FRI-WEB**

Fondements de la Recherche d'Information-WEB

Avant-propos :

Ce premier projet a pour objectif de mettre en œuvre les notions fondamentales d'indexation et de modèles de recherche vues en cours par la réalisation d'un petit moteur de recherche ad-hoc sur deux bases statiques de documents textuels. En particulier il s'agira de :

- Construire un index non compressé des deux corpus.
- Implémenter un modèle de recherche booléen et un modèle de recherche vectoriel.
- Construire un index compressé du plus gros corpus avec l'approche du code à longueur variable et mettre en oeuvre une recherche booléenne et vectorielle.
- Mettre en place les mécanismes d'évaluation de la recherche sur la collection CACM.

Pour ce projet, seuls les outils de traitement linguistique sont autorisés comme par exemple la bibliothèque python NLTK. Il n'est pas autorisé d'utiliser des plate-formes logicielles de moteur de recherche telles que Apache solr, Apache Lucene ou ElasticSearch.

Aucun langage n'est imposé pour la réalisation de ce projet. Le travail peut être réalisé en binôme.

1 Corpus

Les corpus sur lesquels vous allez travailler sont la collection CACM et le corpus issu du cours CS 276 de l'Université de Stanford.

- La collection CACM (Communications of the ACM) [1] est une base assez classique en recherche d'information qui contient les titres, auteurs et résumés d'un ensemble d'articles scientifiques issus des Communications d'ACM entre 1958 et 1979. Un des avantages principaux de cette base est qu'un ensemble de requêtes et de jugements de pertinence est disponible. Cette base contient 4 fichiers principaux :
  - `cacm.all` : contient le texte de l'ensemble des fichiers
  - `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
- La deuxième collection contient un ensemble de pages web du domaine `stanford.edu`. Les données sont disponibles sur Claroline. C'est un corpus de 170 MBs. Il est organisé en 10 sous-répertoires (numérotés de 0 à 9). Chaque fichier correspond au contenu textuel d'une page web individuelle. Chaque nom de fichier est unique dans chaque sous-répertoire mais ce n'est pas le cas globalement.

2 Tâche 1 : Création d'un index inversé et moteur de recherche booléen et vectoriel

2.1 Traitements linguistiques

Il s'agit ici de mettre en place l'ensemble des traitements utiles vus en cours pour transformer un document donné en une liste de termes d'index.

1. `http://ir.dcs.gla.ac.uk/resources/test_collections/cacm/`

1. Collection CACM
   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 (en vérifiant aussi les marqueurs) et c'est uniquement le contenu de ces champs (.T ou .W ou .K) que l'on considére pour l'indexation.
2. Collection CS276
   Les documents de cette collection ont déjà été pré-traités et contiennent des séquences de mots. Pour cette collection, l'étape de tokenization ne sera donc pas nécessaire.

Les différentes étapes de traitements linguistiques sont :
— **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 alpha-numériques) 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.

Pour chaque collection, vous répondrez aux questions suivantes :

1. **Question 1** : Combien y-a-t-il de tokens dans la collection ?
2. **Question 2** : Quelle est la taille du vocabulaire ?
3. **Question 3** : Calculer le nombre total de tokens et la taille du vocabulaire pour la moitié de la collection et utiliser les résultats avec les deux précédents pour déterminer les paramètres k et b de la loi de Heap.
4. **Question 4** : Estimer la taille du vocabulaire pour une collection de 1 million de tokens (pour chaque collection).
5. **Question 5** : Tracer le graphe fréquence (f) vs rang (r) pour tous les tokens de la collection. Tracer aussi le graphe $\log(f)$ vs $\log(r)$.

2.2 Indexation

Pour la collection de l'index lui-même, une des premières questions à vous poser est le choix des structures de données à utiliser. En plus de l'index inversé lui-même (i.e. liste de postings pour chaque termeID), il faudra pouvoir représenter un dictionnaire de (terme, termeID) et un dictionnaire de (document, documentID). Il faudra aussi pouvoir sauvegarder vos différents index pour chaque collection dans des fichiers. La collection CACM est petite et l'index peut être construit directement en mémoire, ce qui n'est pas forcement le cas de collections plus volumineuses. Il est donc demandé de mettre en place pour la construction de l'index :

— L'algorithme BSBI. On considérera la collection CACM comme un seul bloc et pour la collection CS276, on considérera chaque sous-répertoire comme un bloc.
— Une approche distribuée à l'aide de Map-reduce.

2.2.1 Modèle de recherche 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 peut aussi aboutir à une liste ordonnée des documents mais il n'est pas obligatoire de le mettre en place dans le cadre de ce projet.

2.2.2 Modèle de recherche 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.

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 Evaluation pour la collection CACM

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 implementera les principales mesures vues en cours dont :

— Précision et rappel : il s'agira de tracer la courbe rappel/précision comme vue en cours.
— 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.

3 Tâche 2 : Création d'un index inversé compressé et moteur de recherche booléen et vectoriel

Il s'agit dans cette partie de mettre en œuvre la méthode de compression Variable Byte encoding pour la collection CS276.