Master Modélisation Informatique des Connaissances et du Raisonnement (MICR)

Cours Recherche et Extraction d'Information

TP Recherche d'Information

L'objectif global du TP de Recherche d'Information est de faire tourner un moteur de recherche sur des données d'évaluation réelles (à la façon d'une campagne d'évaluation), et d'évaluer dans ce cadre plusieurs stratégies de recherche. Les trois séances du TP correspondent aux trois tâches suivantes :

- indexation des documents : choix et extraction des termes d'indexation
- utilisation d'un moteur de recherche : le moteur proposé est un petit moteur jouet implémentant le modèle vectoriel et réalisé en Java
- évaluation des résultats pour différentes stratégies

Le travail se fait en binôme. Il sera évalué sur la base

- d'un rapport présentant les différentes stratégies évaluées et leurs résultats
- des programmes et scripts utilisés.

Le rapport et les programmes devront être envoyés par e-mail, après la dernière séance, à l'adresse antoine.rozenknop@lipn.univ-paris13.fr dans un seul message, dont le sujet doit commencer par la balise [TP-RI].

Utiliser la même adresse e-mail pour toute question concernant le TP.

TP Recherche d'Information - Séance 1 (06/03/2009) – Pré-traitement

L'objectif de la première séance sera d'extraire les termes d'indexation des documents en vue de leur indexation. Différents pré-traitements sont à disposition pour cette opération.

Avant toute chose, récupérez l'archive `TP_RI.tgz` depuis votre compte SERCAL. Cette archive est située sur le répertoire ~rozenknop/MICR_REI/TP_RI. Par exemple, en ligne de commande :

```
cp ~rozenknop/MICR_REI/TP_RI/TP_RI.tgz .
```

(Vous pouvez aussi utiliser un gestionnaire de fichiers ; dans ce cas, il faut taper le chemin d'accès complet – *les répertoires intermédiaires ne sont pas ouverts en lecture !*).

Décompressez l'archive quelque part et lancez la prise en compte de l'environnement local de travail (pour l'utilisation du programme d'analyse). Par exemple, en ligne de commande :

```
mkdir TP
cd TP
tar xvfz TP_RI.tgz
cd TP_RI
source TP_RI.env
```

Le répertoire de travail est structuré de la façon suivante :

+--data | | | +--corpus | | | | | | | +--test | | | | | | | | | | | +--assessments | |	***Les données*** Le corpus d'évaluation : 3000 documents et 60 requêtes extraits de la campagne d'évaluation CLEF 2003 Un corpus de test au même format : il peut être utilisé pour la mise au point des programmes (pour diminuer les temps de traitements) Les jugements de pertinence (pour l'évaluation)
+--sources | | | +--preprocessing | | | | | | | | +--stemmers | | | | | +--stoplist | | | | | +--tagger | | | | | +--searchengine | | | +--evaluation | +--docs	***Les programmes*** Les programmes de prétraitement pour extraire les termes d'indexation Des stemmers Une stoplist Un outil pour la désambiguïsation morpho-syntaxique Le moteur de recherche en java Les programmes pour l'évaluation ***Les documents*** (contient le présent énoncé)

La première étape de l'indexation est l'analyse des documents. Plusieurs outils (disponibles gratuitement sur le Web) vous sont fournis dans le répertoire `sources/preprocessing` pour analyser les documents : ces outils sont présentés plus précisément plus bas.

Vous devez réaliser pour ce TP des programmes ou scripts permettant d'appliquer les différents outils aux documents, disponibles au format XML pour extraire les termes à indexer, et les mettre dans le format d'entrée de l'indexeur (défini ci-dessous). Vous êtes libres d'utiliser le(s) language(s) que vous voulez pour cela (je conseille l'utilisation de scripts Perl ou Python, mais un ensemble de scripts shell ou des programmes JAVA, C++ ou autres sont tout-à-fait envisageables). Les termes d'indexation à garder doivent pouvoir être : les formes de surface (mots tels qu'il apparaissent dans le texte original), les stems (après application d'un stemmer), les lemmes ou les couples lemmes+catégorie grammaticale (après l'application d'un tagger).

L'indexeur utilisé dans le moteur de recherche prend en entrée des fichiers au format suivant :

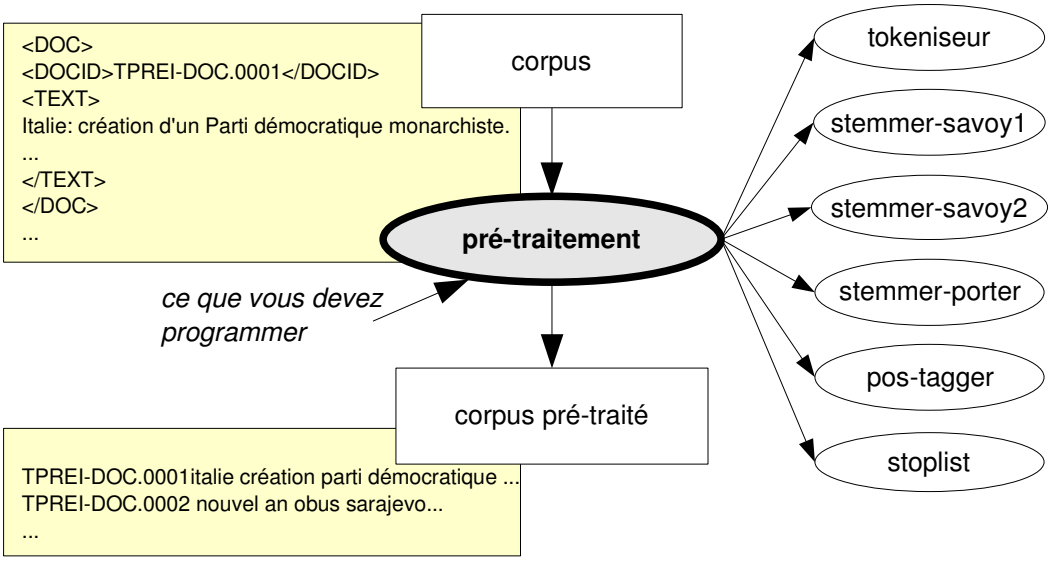
- chaque ligne représente un document

- l'espace est le caractère séparateur au sein de chaque ligne
- le premier élément de la ligne est le nom du document
- les éléments suivants de la ligne sont les termes d'indexation retenus pour ce document (il sont dans l'ordre des mots du texte du document original et ils peuvent être répétés).

Un exemple de fichier XML et de *fichiers à produire* sont disponibles dans le répertoire `data/test`

- le fichier `test.xml` est un exemple de fichier XML contenant le corpus
- le fichier `test.porter.txt` est un exemple de fichier prétraité par le stemmer de Porter et reformaté pour être fourni en entrée de l'indexeur.
- le fichier `test.tagger.txt` est un exemple de fichier prétraité par une désambiguïsation morpho-syntaxique et reformaté pour être fourni en entrée de l'indexeur.

Concrètement:



Présentation des outils :

Pour pouvoir utiliser les outils de pré-traitement, il faut compiler les programmes nécessaires et les installer, en faisant :

```
cd TP_RI/sources/preprocessing
make install
```

Les outils de pré-traitement disponibles dans ce répertoire sont les suivants :

- un tokeniseur simple en perl, qui prend du texte en entrée et sort le texte tokenisé (sur la sortie standard).

pour le lancer, utiliser la commande

```
tokenizer.pl nom_de_fichier
```

- trois stemmers différents : le stemmer de Martin Porter
  http://snowball.tartarus.org/algorithms/french/stemmer.html
  et deux stemmers proposés par Jacques Savoy :
  http://www.unine.ch/info/CLEF/frenchStemmer.txt
  http://www.unine.ch/info/CLEF/frenchStemmerPlus.txt

  Ces stemmers prennent en entrée un fichier contenant un texte tokenisé et donnent en sortie le même texte avec les termes tronqués. Pour lancer les stemmers, utiliser les commandes respectives

```
stemmer-porter-fre nom_de_fichier
stemmer-savoy1-fre nom_de_fichier
stemmer-savoy2-fre nom_de_fichier
```

- une stoplist du français, de Jacques Savoy également:
  http://www.unine.ch/info/CLEF/frenchST.txt
  pour utiliser la stoplist, un programme perl est proposé, qui prend un texte en entrée et qui donne en sortie le même texte sans les mots de la stoplist. Ce programme est lancé avec la commande

```
apply-stoplist.pl fichier_stoplist nom_de_fichier
```

- un analyseur permettant de faire la désambiguïsation morpho-syntaxique : le *tree-tagger*, de Helmut Schmid :
  http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

  Ce programme prend en entrée un fichier contenant le texte tokenisé et produit en sortie le texte étiqueté au format suivant :

mot	*catégorie*	*lemme*
La	DET:ART	le
Grèce	NAM	Grèce
entend	VER:pres	entendre
promouvoir	VER:infi	promouvoir
le	DET:ART	le
livre	NOM	livre
blanc	ADJ	blanc
de	PRP	de
la	DET:ART	le
Commission	NOM	commission

  Le programme est lancé avec la commande

```
tree-tagger-french nom_de_fichier
```

Le traitement étant plus complexe qu'un simple stemming, le traitement est relativement long : l'analyse du corpus complet (les 3000 documents) peut prendre 30 minutes ou plus.

TP Recherche d'Information - Séance 2 (13/03/2009) – Moteur de recherche

La deuxième séance a pour but d'utiliser un moteur de recherche pour indexer les documents et de rechercher les documents pertinents par rapport aux requêtes proposées.

La compilation et l'installation du moteur de recherche se font par la commande :

```
cd TP_RI/sources/searchengine
make install
```

Les prétraitements des documents étant faits, l'indexation des documents s'effectue par la commande :

```
indexer.sh nom_index nom_de_fichier
```

En sortie, le fichier `nom_index` contiendra alors l'index inversé des documents : cet index est lisible (format texte), et contient la structure inversée des documents ainsi que le lexique des documents (liste des mots).

Les requêtes sont contenues dans le fichier XML `TP_RI/data/corpus/topics.xml` (pour la mise au point, un exemple de requête au même format se trouve dans le fichier dans le fichier `TP_RI/data/test/topic-test.xml`). Ce fichier XML est au format des requêtes TREC.

Le moteur de recherche prend en entrée un index et un fichier de requêtes analysées, au même format que les documents analysés :

- chaque ligne représente une requête
- le premier élément de la ligne est l'identifiant de la requête
- les éléments suivants de la ligne sont les termes d'indexation retenus pour la requête

Un script Perl est fourni pour faire la conversion entre le fichier XML des requêtes et un fichier au même format que ceux des documents, permettant l'utilisation des programmes/scripts développés dans la première partie. Ce script s'utilise de la façon suivante :

```
reformat.pl -[t][d][n] topics_file.xml > topics_file_2.xml
```

les options `t`,`d`,`n` indiquent les parties des requêtes à utiliser (`t` pour *title*, `d` pour *desc*, `n` pour *narr*). Ces options sont fournies pour permettre de tester les différences de performances en utilisant plus ou moins d'éléments dans les requêtes.

Le programme de recherche prend en entrée le fichier contenant l'index et le fichier contenant les requêtes analysées et lance la recherche pour chacune des requêtes. Il retourne un maximum de 1000 documents par requête, au format TREC. Ce format

est décrit ici à titre informatif : chaque ligne représente un document retourné pour une requête, et contient les éléments suivants, séparés par des espaces :

- le premier élément de la ligne est l'identifiant de la requête
- le deuxième élément de la ligne est constant et vaut 1
- le troisième élément de la ligne est l'identifiant du document retourné
- le quatrième élément de la ligne est le rang du document dans la liste des documents retournés (ce rang commence à 0 et finit à 999). Les documents doivent être rangés dans l'ordre de leur rang
- le cinquième élément de la ligne est le score associé au document (ce score doit être décroissant avec le rang: plus le document est pertinent, plus le score est élevé)
- le dernier élément de la ligne est un identifiant du résultat (il est le même sur toutes les lignes du fichier): cet identifiant est utile pour comparer des résultats.

La commande pour lancer le moteur de recherche est la suivante :

```
searchengine.sh index topics weighting_scheme runId
```

Le premier argument est le nom du fichier contenant l'index, le second argument est le nom du fichier contenant les requêtes analysées, le troisième argument est le schéma de pondération (au format SMART), le dernier argument est l'identifiant du test :un fichier contenant le résultat au format TREC est produit, qui a le nom de l'identifiant du test.

TP Recherche d'Information - Séance 3 (20/03/2009) - Evaluation

La troisième séance de TP a pour objectif l'évaluation des résultats de différentes stratégies de recherche sur le corpus d'évaluation présent dans le répertoire `TP_RI/data`. Ce répertoire contient :

- `corpus/corpus.xml`: une collection de documents composée de 3010 documents ;
- `corpus/topics.xml`: une collection de requêtes composée de 60 requêtes au format TREC ;
- `assessments/qrels`: des résultats de référence : la liste des documents pertinents pour chacune des 60 requêtes.

La compilation et l'installation des programmes pour l'évaluation se font par la commande :

```
cd TP_RI/sources/evaluation
make install
```

Les programmes d'évaluation disponibles sont :

- le programme d'évaluation standard `trec_eval` :
  http://trec.nist.gov/trec_eval

  il prend en entrée un fichier de résultat retourné par le moteur de recherche et un fichier de jugements de pertinence (ce fichier est disponible dans le répertoire `TP_RI/data/assessments`). La syntaxe de l'appel à ce programme est la suivante (avec redirection de la sortie standard sur un fichier) :

  ```
  trec_eval qrels results > results.eval
  ```

  où `qrels` est le nom du fichier contenant les résultats de référence
  `results` est le nom du fichier contenant les résultats du moteur de recherche
  `results.eval` contient le résultat de l'évaluation :

  Vous pouvez regarder directement le fichier d'évaluation : le format de cette évaluation est de la forme

  ```
  mesure1          requête    valeur
  mesure2          requête    valeur
  ...
  ```

  Par défaut, les valeurs données sont une moyenne sur toutes les requêtes (le champ de la deuxième colonne vaut « `all` »).
  Les valeurs calculées contiennent en particulier :
  ➔ `num_ret` = nombre (absolu) de documents retournés
  ➔ `num_rel` = nombre (absolu) de documents pertinent
  ➔ `num_rel_ret` = nombre (absolu) de documents pertinents retournés

➔ `map` = précision moyenne
➔ `ircl_prn.0.xx` = valeur interpolée de précision pour un rappel de 0.xx

Le détail complet de toutes les mesures peut être obtenu en faisant
`trec_eval -h`

- un programme Perl, nommé `ireval.pl`, qui permet d'interpréter les résultats du programme trec_eval, et sortir les courbes précision/rappel avec gnuplot (programme interactif pour l'affichage de courbes sous Linux/Unix). Il prend en entrée la sortie du programme `trec_eval`. La syntaxe d'appel de ce programme est la suivante :
  `ireval.pl results.eval`

  Pour lancer les courbes comparatives de plusieurs résultats, il suffit de faire
  `ireval.pl results1.eval results2.eval`

  Pour comparer les résultats sur certaines valeurs (par exemple, précision moyenne) on peut faire
  `ireval.pl -text -value=map results1.eval results2.eval`
  ou
  `ireval.pl -best -value=map results1.eval results2.eval`

  Plus d'informations sont disponibles avec l'option `-h`

- un programme Perl nommé `Wilcoxon.pl` qui applique le test de Wilcoxon sur deux résultat pour établir si la différence est statistiquement significative, basé sur l'implémentation du test de Wilocoxon par Rob van Son, disponible à l'adresse : http://www.fon.hum.uva.nl/rob/SignedRank/SRTest.pl

  Ce programme s'utilise avec la syntaxe :

  `Wilcoxon.pl -value=map qrels results1 results2`

  la valeur indiquée est la mesure sur laquelle on veut faire le test (sans option, fait le test sur toutes les mesures)

L'évaluation complète d'une stratégie de recherche se fait avec les étapes suivantes (les premières étapes ont pu être réalisées dans les séances précédentes) :

- lancer l'analyse sur la collection (avec les options souhaitées : stemming, stoplist...)
- créer un fichier d'index à partir de cette analyse
- lancer l'analyse sur les requêtes (avec les mêmes options)
- lancer le moteur de recherche avec l'index et les requêtes analysées
- lancer le programme `trec_eval` pour évaluer les résultats obtenus

La comparaison des résultats obtenus avec différentes stratégies de recherche se fait

par la comparaison des courbes précision/rappel, par la comparaison de valeurs de performance globales, comme la précision moyenne (*map*), et par des tests statistiques pour vérifier si les différences sont significatives.

Parmi les choses à évaluer, vous pourrez regarder en particulier les suivantes :

- influence du choix des termes d'indexation sur les résultats :
  - application ou non d'un stemmer ou d'un tagger (avec les catégories ou non) ;
  - utilisation ou non d'une stoplist (comparer aussi les tailles d'index) ;
- influence des champs utilisés dans la requête (titre, description, narratif)
- comparer les résultats de différents schémas de pondération du modèle vectoriel
- comparer les résultats des différentes requêtes : quelles requêtes sont plus difficiles quelle que soit la stratégie utilisée ? Qu'est-ce qui les rend difficiles ? par exemple, quels sont les mots de la requête que le moteur n'a pas trouvés, pourquoi certains documents pertinents ont été ratés ?

Pour des comparaisons avancées, vous pouvez également, si vous le souhaitez, tester sur ce corpus d'évaluation d'autres moteurs de recherche disponibles sur Internet (des vrais moteurs), implémentant d'autres modèles de recherche :

- Lemur (Indri) : http://www.lemurproject.org/
- Apache Lucene : http://lucene.apache.org/java/docs/
- Terrier : http://ir.dcs.gla.ac.uk/terrier/
- MG : http://www.math.utah.edu/pub/mg/
- zettair : http://www.seg.rmit.edu.au/zettair/