UNIVERSITE FARHAT ABBAS - SETIF-1

FACULTE DES SCIENCES

DEPARTEMENT D'INFORMATIQUE

MODULE M2: RECHERCHE D'INFORMATION

ENSEIGNANT DU COURS : DR. FOUZI HARRAG

TRAVAUX PRATIQUES EN RECHERCHE D'INFORMATION

Mise en œuvre du cours de recherche d'information : implémentations de recherches séquentielles, prétraitements des corpus textuels, exercices sur les modèles booléen et vectoriel.

COMPREHENSION DES MOTEURS DE RECHERCHE

EXERCICE 1 - ÉTUDE DE DIFFERENTS MOTEURS

Nous considérons différents systèmes gérant des documents et proposant des moteurs de recherche. La liste (non exhaustive) ci-dessous est rangée par famille d'applications :

- web global : Google, Qwant, Bing, Yahoo,
- sites d'actualités : Le Monde, L'Équipe,
- systèmes documentaires : catalogues des bibliothèques de l'université de Lille SHS
- encyclopédies : wikipédia, Larousse, Universalis
- sites marchands : amazon, Furet du Nord
- réseaux sociaux : facebook, twitter
- banques d'images : google images, gettyImages
- vidéos : youtube, dailymotion
- musique :
  - Zeffyr, midomi, Deezer
  - Paroles.net, musixmatch,
  - AudioTag.info, midomi, musipedia,
  - Wat Zat Song, Trouve Ta Musique

Il s'agit de comprendre la nature des documents stockés, la syntaxe et la sémantique des requêtes permises, et enfin de deviner les traitements qui mettent en lien requête et documents. Pour chaque question et pour les sites choisis, il faut imaginer des requêtes qui permettent de répondre et examiner :

- le nombre de réponses,
- la pertinence des propositions,
- le critère de classement des résultats,
- la nature des documents fournis (pages web ? documents textuels ? pdf ? etc.).

Il faut également comprendre pourquoi ces documents sont proposés par le système : les éléments de la requête sont-ils présents dans les documents proposés ? Recherche-t-on dans le contenu des documents ? Enfin, quand c'est possible, on comparera la première page de résultats sur différents sites.

1. Recherche avec un unique mot-clé : essayer avec des mots généraux, puis des mots très spécifiques. Construire un mot ne fournissant aucune réponse.
2. Recherche à plusieurs mots clés : tester et établir la sémantique d'une telle requête. Un document résultat contient-il toujours tous les mots-clés de la requête ?
3. Recherche par phrase : tester avec des phrases choisies, puis avec ces mêmes phrases privées des mots sans importance. Construire une phrase sans aucune réponse.
4. Imaginer des requêtes qui permettent de comprendre comment sont traités les accents dans les requêtes. Idem pour la casse : les différences minuscules/majuscules sont-elles prises en compte ? Tester avec des acronymes avec ou sans points, des mots composés avec ou sans tirets, etc.
5. Tester des noms au singulier puis pluriel. Idem avec des verbes et différentes conjugaisons.
6. Des mots (ceux vides de sens) sont-ils systématiquement ignorés dans les requêtes ?
7. Les requêtes sont-elles robustes aux erreurs de saisie ? Des suggestions sont-elles faites pour corriger ces erreurs ?
8. Explorer la recherche avancée si elle est disponible et établir la liste des raffinements possibles des requêtes.
9. Les résultats incluent-ils des recommandations en fonction des autres utilisateurs ?
10. Tester si les recherches sont personnalisées par l'historique de recherche, la localisation, etc.

IMPLEMENTATIONS DE RECHERCHES SEQUENTIELLES

Dans cette partie, nous développons des fonctionnalités de recherche pour le corpus constitué des fichiers du dossier `textes` fourni (ou pour un autre corpus de votre choix).

Exercice 2 - Avec Python

1. Obtenir la liste des fichiers du dossiers `textes`.
2. Écrire une boucle sur cette liste pour chaque fichier, ajouter une boucle sur chaque ligne.
3. Au choix :
    - Tester la présence d'un mot dans la ligne.
    - Découper la ligne en mots puis chercher la cible dans la liste obtenue.
    - Concaténer toutes les lignes du fichier, découper en phrases, puis faire la recherche phrase par phrase.
4. Reprendre les questions précédentes pour, cette fois, tester l'absence d'un mot dans un fichier.
5. Utiliser des expressions régulières pour tenter de faire abstraction des fautes de frappes, des accents, des accords et conjugaisons, etc.
6. Tester l'insensibilité à la casse : soit en passant en minuscules requête et lignes des documents, soit en intervenant sur les expressions régulières et leurs méthodes.
7. Compter les occurrences du mot dans chacun des fichiers, afficher chaque nom de fichier et son compteur.
8. Présenter les résultats : extraits montrant le mot en contexte, noms et localisations des fichiers, numéros des lignes, couleurs.
9. Généraliser pour plusieurs mots-clefs : ils peuvent être liés par ET OU NON.
10. Ajouter un critère de proximité : tester si les mots sont sur la même ligne, dans la même phrase, à une certaine distance comptée en mots.

Pretraitements des textes et constitution d'un vocabulaire

Exercice 4 - Extraction du corpus

À partir des fichiers (pdf, doc, png) du dossier `misc` fourni, récupérer le texte de ces documents.

Exercice 5 - Caracteristiques du français et outils linguistiques

1. Parcourir le site Lexique, en particulier sa partie statistiques et son système d'interrogation.
2. Tester en ligne l'extraction d'entités nommées : ici, là ou ailleurs.
3. Utiliser dans le terminal TreeTagger sur des textes de votre choix.
4. Lire sur wikipédia la loi de Heaps et la loi de Zipf.

Exercice 6 - Constitution du vocabulaire

Répondre aux questions suivantes à l'aide de programmes Python.

1. Calculer la taille du corpus en nombre de mots.
2. Compter le nombre de mots différents dans le corpus.
3. Compter pour chacun son nombre d'occurrences dans le corpus.
4. Les ordonner et afficher le rang, observer les mots les moins fréquents et les plus fréquents.
5. Vérifier la loi de Zipf.
6. Exclure cette fois les mots de la *stop-list* fournie.
7. Quelle est la nouvelle taille du corpus (toujours en nombres de mots) ? Quels sont maintenant les mots les plus fréquents ?
8. Écarter les mots qui contiennent des chiffres, écarter les tirets bas.
9. Remplacer les mots au pluriel et les verbes conjugués, chacun par leur lemme.
10. Stocker les mots du vocabulaire dans un fichier. Quelle est la taille du fichier obtenu ?

Modele booleen

Exercice 7 - Requetes booleennes avec matrices d'incidence

Nous avons la connaissance d'un corpus et de son vocabulaire à travers la matrice d'incidence suivante (extrait) :

	DOC#11	DOC#19	DOC#175	DOC#257	DOC#332	DOC#789	DOC#819	DOC#956
HASARD	0	0	0	0	0	1	0	0
ESPIONNAGE	1	0	1	0	0	1	0	0
IRRIGATION	0	0	0	0	0	0	1	0
LENDEMAIN	0	1	0	1	1	0	0	0
SECRET	1	1	0	0	0	0	0	0

IMPOSSIBLE	1	0	0	0	1	1	0	0
COMPOSITEUR	0	0	0	0	0	1	1	0
VIVANT	1	0	1	1	1	0	1	0

Calculer les résultats des requêtes suivantes :

1. `lendemain`
2. `lendemain et vivant`
3. `lendemain et vivant et hasard`
4. `lendemain ou vivant`
5. `lendemain ou vivant ou hasard`
6. `lendemain et non(vivant)`
7. `non(lendemain) ou non(vivant)`

Pour cette dernière, trouver une requête équivalente et vérifier qui les résultats sont identiques.

Exercice 8 - Construction d'une matrice d'incidence

1. Relire le fichier-vocabulaire, parcourir le corpus et calculer la matrice d'incidence.
2. Stocker cette matrice dans un fichier.
3. Calculer quelques statistiques : taille du fichier, nombre et pourcentage de zéros dans la matrice.
4. Implémenter la recherche d'un mot unique : ce nouveau programme Python devra relire la fichier contenant la matrice et l'utiliser au mieux.

Exercice 9 - Requetes booleennes et index

Nous reprenons la base de documents précédemment évoquée et la matrice d'incidence déjà utilisée.

1. Transformer la matrice d'incidence précédente sous forme d'un index.
2. En utilisant cette nouvelle structure, résoudre les requêtes suivantes en discutant les stratégies de résolution
    1. `lendemain et vivant`
    2. `lendemain et vivant et hasard`
    3. `(lendemain ou vivant) et non (hasard ou impossible)`

Exercice 10 - Index

Nous reprenons l'implémentation d'une recherche dans le modèle booléen, cette fois avec une structure d'index.

1. Relire le fichier-vocabulaire, parcourir le corpus et construire l'index.
2. Stocker cet index dans un fichier.
3. Quelle est cette fois la taille du fichier ?
4. Implémenter la recherche d'un mot unique basée sur l'index.

EXERCICE 11 - PLUS LOIN

1. Dans les deux programmes Python de recherche, mesurer les temps d'exécution liés au chargement des structures de données et ceux associés aux recherches.
2. Implémenter une boucle interactive pour enchaîner les recherches après un seul chargement de la matrice ou de l'index en mémoire.
3. Mettre en place la gestion des combinaisons booléennes de plusieurs mots-clés et optimiser les calculs, dans le cas d'une matrice d'incidence comme dans celui d'un index.

MODELE VECTORIEL

EXERCICE 12 - CALCUL DE TF-IDF ET CALCUL DE SCORE

On considère une base de 800,000 documents.

1. Étant données la table des occurrences et la table des df ci-dessous, calculer les valeurs de tf-idf pour les termes `car`, `auto`, `insurance` et `best` pour chacun des 3 documents.

	DOC#1	DOC#2	DOC#3
CAR	27	4	24
AUTO	3	33	0
INSURANCE	0	33	29
BEST	14	0	17

	DF
CAR	18165
AUTO	6723
INSURANCE	19241
BEST	25235

3. On considère la requête `q = best car insurance`. Calculer le score (cosinus) de chacun des trois documents pour cette requête.

EXERCICE 13 - IMPLEMENTATION PYTHON - REPRESENTATION VECTORIELLE DES DOCUMENTS ET SIMILARITE COSINUS

Nous travaillons sur le même corpus que précédemment et nous repartons du vocabulaire déjà calculé et stocké.

1. Calculer les tf, df, idf, tdf-idf.
2. Observer les mots avec les plus faibles idf. Quel est la plus petite valeur théorique de idf ? Comparer aux mots de la *stop-list*.
3. Quelle est la plus grande valeur de tf-idf observée ?

Nous comparons maintenant les documents entre eux avec le cosinus comme mesure de similarité.

4. Quels sont les deux textes du corpus les plus proches ? Quels sont les deux documents les plus différents ?
5. Implémenter un moteur de recherche qui à partir d'une requête donne les 10 documents les plus pertinents.

EXERCICE 14 - PLUS LOIN

Reprendre les deux exercices précédents pour étudier les variantes du calcul de tf-idf (à partir de ce point de départ par exemple).

En particulier comparer les différences dans les résultats produites par ces différentes possibilités. Notez qu'il est possible de faire des choix différents pour le codage des documents et pour le codage de la requête.