TP Recherche d'informations
Manipulation d'une collection de test

1 Préambule

Ce TP est l'occasion d'utiliser le langage de programmation permettant la manipulation des chaines de caractères et des données textuelles. L'objectif est d'utiliser et de réaliser les scripts pour le traitement de la collection de test, l'indexation et l'accès aux documents.
Vous devez rédiger un compte-rendu de TP (conseil : faire dès maintenant) et fournir les codes de vos scripts dans une archive avec un fichier expliquant comment les utiliser. Ce TP donnera lieu à une présentation orale (22/01/26) à faire avec un support. Travail à remettre (compte rendu de TP, code et présentation) à remettre via filesender pour le 20 décembre 2026.

Le Tp se déroulera en 2 parties
— la mise en place dirigée de la collection et l'utilisation de script pour le « nettoyage des données »,
— la recherche des algorithmes pour répondre aux étapes de préparation de la collection de test et la réalisation de vos propres scripts en python.

2 Utilisation d'une collection de test

2.1 Récapitulatif des étapes à suivre

Ce TP a pour objectif de vous faire utiliser et écrire des programmes concis pour effectuer des traitements basiques sur le texte des documents d'une collection de test.
Pour rappel, ci-dessous le rappel de la description de l'enchaînement des différentes étapes concernant le traitement du contenu textuels, et ce que vous vous allez mettre en place :

1. Nettoyage de la collection
    - — ponctuation & accents
    - — lemmatisation ou radicalisation
    - — sac de termes
    - — stop list / mots vides
2. Analyse de la collection
    - — calcul de la fréquence des termes
    - — fréquence documentaire
    - — observation lois de Zipf, Heaps
    - — terme le plus fréquent dans la collection, dans tous les documents, par document
3. Système de recherche d'informations
    - — indexation
    - — création des fichiers inverses
    - — implémentation des modèles de calcul
    - — interrogation de la collection

3 Traitement des données

3.1 Documentation du code

Préparer comment documenter votre code :

1. Modifier l'entête de chaque script en indiquant : les auteurs, le nom du fichier et les objectifs du programme.
2. Commenter le code. Vous avez besoin d'un éditeur de code qui reconnait le langage perl, par exemple Geany ou notepad++ sur windows ou VSCode.

3.2 Compte github

Vous pouvez utiliser un projet et l'exécution du code en ligne en utilisant une extension pour le langage perl et le langage python.

3.3 Quelques références sur Python et la RI

https://www.python.org/
Mooc python 3 "Python : des fondamentaux aux concepts avancés du langage" sur FUN: http://www.fun-mooc.fr/fr/cours/python-3-des-fondamentaux-aux-concepts-avances-du-langage/

https://www.irit.fr/~Mohand.Boughanem/Enseignements_RI.php
http://www.iro.umontreal.ca/~nie/IFT6255/
Massih-Reza Amini, Éric Gaussier Recherche d'Information - applications, modèles et algorithmes. Eyrolles 2013, ISBN 978-2-212-13532-9
Massih-Reza Amini & al. Data science Eyrolles 2018, ISBN 978-2-212-67410-1

4 Collection de test

De nombreuses approches en recherche d'information utilisent des benchmarks pour comparer les méthodes d'appariement documents/requêtes.
On considère la collection de documents CACM disponible à l'adresse suivante:

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

Télécharger l'archive contenant la collection et récupérer le dossier complet.
— Observer les différents fichiers du répertoire cacm, combien y en a-t-il ?
— Quel est leur rôle pour une expérimentation en recherche d'information ?
— Le fichier cacm est structuré, à quoi correspondent les différentes balises ? et principalement .I ? .T ?.A ? .W?

4.1 Préparer votre compte-rendu

Rédiger un paragraphe qui rappelle ce qu'est une collection de test et expliquer le concrètement en prenant exemple sur les fichiers de la collection CACM. Si le temps le permet, vous travaillerez sur un 2è collection pour éprouver vos scripts. Trouver une autre collection de test.

4.2 Impact de la langue

Tout d'abord, observer la langue utilisée. Quelle est la langue utilisée ? La collection est-elle multilingue, dans ce cas quel problème cela poserait il ?
Rédiger la réponse dans votre compte-rendu.

Préparation des fichiers

A cette étape, les traitements ne sont pas à réaliser, il s'agit de prévoir les extensions de fichiers pour distinguer les versions des fichiers contenant le texte des documents.
Les fichiers sources seront traités pour enlever les caractères inutiles, les mots vides, récupérer seulement le texte des parties intéressantes, pas de mots au pluriel, etc. Nous utiliserons plusieurs scripts pour réaliser cela, prévoir le nom de vos extensions de fichier. Vous pouvez faire un tableau du type de fichier en fonction des critères fichiers.

1. Sans caractères spéciaux et avec un seul espace entre les mots
2. Sans les mots vides
3. Application du stemming. Avec un traitement pour les pluriels (trouver une api pour faire cela, par exemple en python ou enlever brutalement les « ies » puis « s ». Renseignez-vous sur l'algorithme de Porter pour la racinisation du texte en anglais.

Rem : vous pouvez aussi gérer ceci en mettant les fichiers dans des sous-répertoires.

Mettre la réponse dans votre compte-rendu.

5 Prise en main du processus en python: utilisation des scripts, manipulation des documents

Pour comprendre le processus de nettoyage des données, vous allez manipuler les scripts permettant de :

- ***Decode*** Créer un fichier par document à partir du fichier ***cacm.all***
- ***Clean*** Enlever les caractères spéciaux
- ***Remove*** Enlever les mots vides

Lire la question en entier avant de commencer.

Script ***Decode.pl*** permet de récupérer les informations des documents. Extraire les parties de texte intéressantes des fichiers de la collection CACM. Quelles sont les parties à extraire pour constituer un fichier dans lequel rechercher l'informations ?
Script ***Clean.pl*** permet de créer un fichier pour chaque document ne contenant pas de caractères spéciaux ni d'accents et de supprimer les espaces inutiles. Ce script utilise la collection de tous les documents (répertoire Collection) et le fichier Collection qui contient la liste des noms de fichiers de la collection CACM. Enlever les caractères spéciaux. La question est d'éliminer les caractères spéciaux, les supprimer ou les remplacer.
Script ***Remove.pl*** permet d'enlever les mots vides qui se trouvent dans le fichier ***commonwords***. Les mots vides sont dans ce fichier, le document de la collection doit être réécrit sans ces mots.

Avant de manipuler les scripts, écrire les algorithmes. Les reporter dans votre compte-rendu.

Le script ***Decode.pl*** ajoute les fichiers sont dans le dossier Collection et crée un fichier avec le nom des documents en utilisant leur identifiant.
Observer les documents et comparer au fichier ***cacm.all***. Quelles sont les portions de texte qui ont été utilisées pour créer les documents ?
Mettre la réponse dans votre compte-rendu. Utilisez votre analyse de ces scripts pour vérifier votre algo. Rédiger dans votre compte-rendu, l'explication du script : donner en gros la démarche ou l'algorithme utilisé pour répondre au besoin s'il y a une différence importante par rapport à ce que vous aviez pensé.

NB : Pour la dernière question, moteur de recherche / fonction de correspondance, « Créez une V2 du script ***clean*** et du script ***remove*** ou bien un autre script qui utilisera tous les documents d'une extension/version de document donné pour regrouper en un seul fichier HTML chaque document dans un article ou un paragraphe.
Utiliser une classe pour marquer le document comme étant de la collection de test cacm
Ces fichiers seront nommés Collection1.html (pour la version sans caractères spéciaux/blanc) et Collection2.html (pour la version sans mots vides).

> Je montrerai rapidement le principe du langage html et les balises pour faire l'exercice. Pour plus de précision voir
> cf. petits mémos html https://course.oc-static.com/courses/1603881/Fiche+HTML+CSS.pdf
> https://jenseign.com/html/wp-content/uploads/2018/01/memo-html-apprendre-v2.50.pdf

6 Calcul des valeurs classiques – dictionnaire des termes

De nombreuses fonctions de correspondance utilisent les valeurs de fréquence des termes dans les documents et de fréquence documentaire (nombre de documents qui contient un terme).

6.1 Vocabulaire

Ecrire l'algorithme qui permet de créer un fichier contenant le ***vocabulaire*** de la collection. Le vocabulaire de la collection est sauvegardé dans un fichier (un mot par ligne).
Ecrire le script ***vocabulaire.py***
Rédiger votre CR : indiquer le nom du fichier de script et celui contenant le vocabulaire. Expliquer votre démarche pour l'écriture du script.

6.2 DF – fréquence documentaire

Ecrire l'algorithme qui permet de sauvegarder dans un fichier le ***df*** (document frequency) d'un terme pour chaque document. Les valeurs de fréquence documentaire sont sauvegardées dans un fichier de la forme #mot #df (un mot par ligne). Est-ce qu'un terme est dans tous les documents ? Si oui, que peut-on faire ?
Ecrire le script ***df.py***
Rédiger votre CR : indiquer le nom du fichier de script et celui contenant les valeurs de df. Expliquer votre démarche pour l'écriture du script.

7 Analyse de la collection

7.1 Script count et termes fréquents

Ecrire les algorithmes dans un premier temps et ensuite les scripts, puis les utiliser. Vous pouvez utiliser le langage python si vous préférez mais il est tout aussi facile de partir de la base des scripts précédents.

- ***Count.py*** Compter le nombre d'occurrences de chaque mot dans la collection. Le parcours du fichier Collection permet de connaitre le nom de chaque document de la collection de test. Le parcours d'un fichier permet de calculer le nombre d'occurrence d'un même mot. Ce programme écrit dans un fichier de sortie les informations sous la forme : Rang du mot, Compte du mot et Mot trié par ordre croissant des rangs.
  Rédiger votre CR : noter la taille du vocabulaire c'est-à-dire le nombre de mots différents de la collection ; expliquer votre démarche pour l'écriture du script.
  Mettre dans votre CR le résultat, quelle est la taille du vocabulaire. Utiliser ce script les 2 types de documents pour donner la taille de la collection.
- ***TermFreq.py*** Calculer le nombre moyen d'apparitions du terme le plus fréquent dans les documents : c'est-à dire trouver la moyenne d'apparition d'un terme quand il apparait dans un document. Vous pouvez ouvrir un fichier de sortie en mode ajout (»$filepath) et tester ce programme sur plusieurs termes très fréquents et moyennement fréquents.
  Rédiger votre CR : noter le résultat ; expliquer votre démarche pour l'écriture du script.

Remarque : essayer de rendre ces scripts génériques par rapport à la version de fichier. Sinon modifier le script à chaque fois que vous l'appliquer sur une version de la collection (sans ou avec mots vides, lemmatisation ou pas).

7.2 Utilisation des résultats

Ploter (gnuplot) ou utiliser un tableur pour créer un graphique visualisant la fréquence des mots (ordonnée) en fonction du rang (abscisse). Vous pouvez aussi utiliser un programme python avec matplotlib pour faire le graphique, les tableaux ordonnée et abscisse doivent avoir la même taille.
Quel constat faites-vous ?
Rechercher de la documentation sur la loi de Zipf.
Améliorer votre graphique en sélectionnant quelques mots et le rang/fréquence associé.

7.3 Vecteurs de termes

7.3.1 Binaire

Ecrire l'algorithme qui permet de créer un fichier pour la ***représentation vectorielle*** des documents sous forme binaire. (Exemple : 10 termes pour le vocabulaire, le document contient les termes 1,3,7, on représente ce document sur une ligne par 1 :1 3 :1 7 :1).
Ecrire le script correspondant ***vecteurBinaire.py***

7.3.2 TF fréquence des termes

Réaliser une version améliorée pour une représentation avec la fréquence des termes tf
Exemple 1 :2 3 :5 7 :1

7.3.3 TF IDF (à faire en plus)

Réaliser une troisième basée version sur tf . idf en utilisant les données du fichier des df.
Rédiger votre CR : indiquer le nom des fichiers de scripts et expliquer votre démarche pour l'écriture de chaque script.
Montrer des captures d'écran des résultats, prendre quelques lignes des fichiers en exemple.

8 Construction de fichier inversé

Chercher l'algorithme pour créer ***l'index inversé*** des termes à partir du vocabulaire. Regarder les indications ci-dessous.
Ecrire un script pour créer ***l'index inversé*** des termes à partir du vocabulaire.
Vous pouvez utiliser la méthode suivante pour la construction du fichier inverse en 3 étapes

1. Extraction des paires d'identifiants (terme, doc), passe complète sur la collection
2. Tri des paires suivant les id. de terme, puis les id. de docs
3. Regroupement des paires pour établir, pour chaque terme, la liste des docs A condition que la collection puisse tenir en mémoire ce qui est le cas ici.

Rédiger votre CR : indiquer le nom du/des fichiers de scripts que vous avez écrits et expliquer votre démarche pour l'écriture de chaque script. Montrer des captures d'écran des résultats, prendre quelques lignes des fichiers en exemple.

9 Moteur de recherche

Si vous avez terminé, créer les scripts pour ***interroger la collection*** de test avec une requête, définir le vecteur de la requête, et accéder aux fichiers d'index. Une sortie de la liste de réponse au format html avec le lien vers le fichier permet d'accéder aux résultats.
En premier lieu, vous pouvez utiliser le tf comme fonction de scoring, et ensuite pour améliorer vous pouvez utiliser l'une des méthodes ci-dessous

- prendre en compte la position des termes et implémenter la méthode de proximité basée sur la logique floue thèse A. Mercier (cf. article sur chamilo).
- tester un fonction de scoring du type tf.idf. (cf 2 articles sur chamilo) ou un autre issue d'une de vos recherches sur le web article inforsid 2003.

Rédiger votre CR : indiquer le nom du/des fichiers de scripts que vous avez écrits et expliquer votre démarche pour l'écriture de chaque script. Montrer des captures d'écran des résultats, prendre quelques lignes des fichiers en exemple.

Ce que vous pouvez faire avec les scripts que vous avez écrits :
1. Noter la taille du vocabulaire avec et sans les mots vides, avec la lemmatisation si vous avez fait
2. Vérifier la loi de Zipf sur la collection sans les mots vides
3. Ajouter une autre version de la collection en traitant les données pour faire de la lemmatisation/appliquer l'algorithme de Porter. Utiliser une API pour faire cela. Vous pouvez appliquer le point 1 et 2 ci-dessus sur cette nouvelle version de la collection de documents.
https://www.nltk.org/howto/stem.html https://www.nltk.org/howto/tokenize.html
Rédiger les explications dans votre CR.

10 Informations MAIL

10.1 Scraping web

Pour la question concernant la récupération des documents à partir du fichier html que vous avez créé (cf.extrait CACM exemple, vous pouvez passer à des scripts en python et utiliser la librairie bs4. Ci-dessous quelques liens de pages web qui donnent les explications utiles pour les quelques lignes de code à écrire.

La question est le "scraping" la page HTML générée pour en faire une autre version et faire aussi un autre jeu de fichiers de la collection CACM.

https://code.tutsplus.com/fr/scraping-webpages-in-python-with-beautiful-soup-the-basics--cms-28211t
https://beautiful-soup-4.readthedocs.io/en/latest/
https://www.codespeedy.com/beautifulsoup-lxml-parser-full-tutorial-python/#google_vignette
https://python.doctor/page-beautifulsoup-html-parser-python-library-xml

```
import bs4

file = "Collection/CollectionTestHtml.html"

with open(file,'r',encoding="utf8") as f:

    html =bs4.BeautifulSoup(f,'html.parser')
```

10.2 Lemmatisation

Comme on passe aux scripts en python autant profitez de la librairie nltk pour faire une troisième version des données textuelles en faisant de la lemmatisation avec technique de Porter.

```
from nltk.stem.porter import *

corpus=html.find_all("article", class_="cacm")

stemmer = PorterStemmer()

for myarticle in corpus:

    print(myarticle["id"] + " ")
```

10.3 documentation nltk

(ici sur stem mais à voir sur traitements précédents aussi enlever blancs et stoplist)

https://www.nltk.org/howto/stem.html

10.4 Autres informations

3 articles sont à regarder si vous êtes curieux, deux concernent l'étude des tf idf et l'autre l'utilisation de la proximité des termes. Il existe d'autres collections de test en RI : lien http://ir.dcs.gla.ac.uk/resources/test_collections/ Une autre librairie pyterrier peut être utilisé pour tester les différentes approches pour les fonctions de correspondance : https://github.com/terrier-org/ecir2021tutorial/blob/main/notebooks/notebook1.ipynb
Vous pouvez aussi regarder l'API Lucene.