N.B :

- Réaliser une des propositions suivantes.
- Les systèmes implémentés doivent utiliser un SGBD NoSQL pour gérer le fichier inverse-Exemple MongoDB.
- Le mini-projet doit être réalisé par des groupes de 4 étudiants.
- L'exposé du Mini-projet aura lieu probablement le 4/12/2024.
- Langage de programmation au choix.
- Fichiers envoyés : stoplist, collection TIME, la liste des collocations appartenant au dictionnaire WordNet.

Proposition 1 :

Réaliser un SRI classique qui utilise le pseudo algorithme suivant pour identifier les termes d'indexation :

Algorithme de détection des termes

Entrée : Doc D_i , Stoplist={the, of, in, on, to, a, has, been, most, around, due, are, that, can, lot}, $Index_i$={}, $Freq_i$={}
Sortie : $Index_i$, $Freq_i$ //les 2 ensembles seront remplis à la sortie

Début

$Tokens$ = Tokeniser (D_i) // Tokeniser par rapport aux blancs et tous les caractères spéciaux
Pour chaque $Tok_{j\ /j=1\ldots|Tokens|} \in Tokens$
Si ($Tok_j \notin$ Stoplist)
Mot=Tok_j ; L = longueur(Mot) ; // La fonction Longueur(M) retourne la longueur de la chaine de caractères M
Si(L>3)
Si(Mot [L]= 's' and Mot[L-1]= 'e' and Mot[L-2]= 'i')
//Mot[L] permet d'accéder à l'élément L de la chaine de caractères
Mot=Extraire(Mot,L-2) ;//La fonction Extraire(M,L) retourne le sous Mot M commençant de 1 à L
L = L-2 ;
FinSi ;
Si(Mot [L]= 's')
Mot=Extraire(Mot,L-1) ; L = L-1 ;
FinSi ;
Si(Mot[L]= 'd' and Mot[L-1]= 'e' and Voyelle(Extraire(Mot, L-2)))
// La fonction Voyelle(M) vérifie si le mot M contient une voyelle
Mot=Extraire(Mot,L-1) ; L = L-1 ;
FinSi ;
Si(Mot[L]= 'y')
Mot=Extraire(Mot,L-1) ; L = L-1 ;
Mot=Mot+'i' ;
FinSi ;
Si(length(Mot)>6)
Si(Mot[L]= 'n' and Mot[L-1]= 'o' and Mot[L-2]= 'i' and Mot[L-3]= 't' and Mot[L-4]= 'a' and m(Mot)>2)
// La fonction m(M) calcul le nombre de voyelles consonnes dans le Mot M
Mot=Extraire(Mot, L-5) + 'ate' ; L = L-2 ;
FinSi ;
FinSi ;
Si(Mot[L]= 't' and Mot[L-1]= 'n' and Mot[L-2]= 'a' and m(Mot)>1)
Mot=Extraire(Mot, L-3) ; L = L-3 ;
FinSi ;
Si(Mot[L]= 'e' and Mot[L-1]= 't' and Mot[L-2]= 'a' and m(Mot)>1)
Mot=Extraire(Mot, L-3) ; L = L-3 ;
FinSi ;
Si(Mot[L]= 's' and Mot[L-1]= 's' and m(Mot)>1)

```
                Mot=Extraire(Mot, L-1) ; L = L-1 ;
            FinSi ;
            Si(Mot[L]= 'l' and Mot[L-1]= 'a' and m(Mot)>1)
                Mot=Extraire(Mot, L-2) ; L = L-2 ;
            FinSi ;
            Si(Mot[L]= 'e' and m(Mot)>1)
                Mot=Extraire(Mot,L-1) ; L = L-1 ;
            FinSi ;
            Si(Mot[L]= 't' and m(Extraire(Mot,L-1)>1)
                Mot=Extraire(Mot,L-1) ; L = L-1 ;
            FinSi ;
        FinSi ;
        Si(Mot ∉ Index_i)
          Inserer(Mot, Index_i) ;  InsererF(1, Freq_i)
        Sinon
          Pos = Position(Mot, Index_i) ;  // La fonction Position retourne la position de Mot dans la liste Index_i
          ModifierF(Pos, Freq_i) ; //La fonction ModifierF permet d'ajouter 1 à l'élément à la position Pos dans  Freq_i
        FinSi ;
Fin.
```

Les termes d'indexation sont ensuite pondérés avec *tf*idf.*

Le processus de recherche s'appuie sur une mesure du modèle vectoriel (Produit scalaire ou cosinus).

Proposition 2 :

Réaliser un SRI classique qui utilise les traitements d'indexation suivantes : Tokenisation, Elimination des mots vides, radicalisation avec la méthode bigram et la mesure de distance lexicale Dice (avec un seuil >75%) . Et enfin pondérer les termes avec *tf *idf.* Le processus de recherche s'appuie sur une mesure du modèle vectoriel (Produit scalaire ou cosinus) dans le classement des documents sélectionnés.

Proposition 3 :

Construire un fichier inverse positionnel constitué de lemmes et leurs positions respectives dans chaque document. Utiliser une librairie qui assure la lemmatisation (selon le langage de programmation utilisé, par exemple spacy en python). Le processus de recherche traite des requêtes de type : texte libre, expression entre " ", ou combinaison des deux.

Proposition 4 :

Réaliser un SRI classique basé sur le modèle booléen (standard ou flou). Les termes d'indexation sont des racines obtenues avec l'algorithme de Porter (programme ou librairie : disponibles sur le Net). Dans le cas du flou, la pondération est *tf=fréquence normalisée par la somme des fréquences.*

Proposition 5 :

Réaliser un SRI qui permet de représenter les contenus textuels par les entités nommées qu'ils contiennent. Les entités nommées sont identifiées avec la librairie spacy en python. La pondération est *tf*Idf.* Le processus d'appariement utilise une mesure du modèle vectoriel (produit scalaire ou cosinus).

Proposition 6 :

Implémenter un SRI conceptuel, où les concepts sont identifiés en utilisant le lexique wordnet. Pour cela, proposer un algorithme d'identification des concepts ou implémenter celui donné dans le dernier exo de la série 2. La pondération des concepts est *tf*idf.* Le processus d'appariement utilise dans le classement des documents la mesure produit scalaire du modèle vectoriel.