Recherche d'information

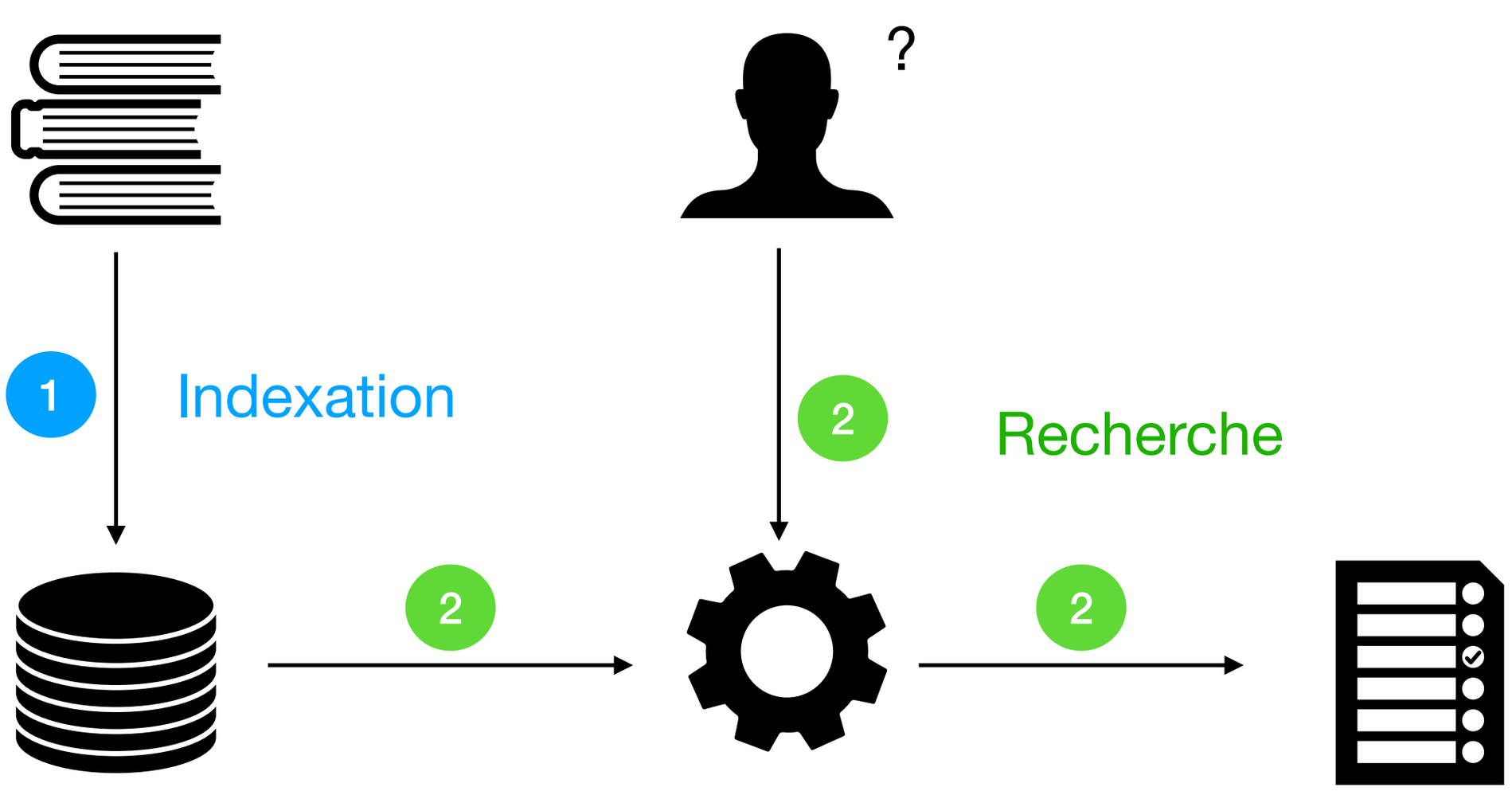
Travaux pratiques

Lorraine Goeuriot - http://mrim.imag.fr/User/lorraine.goeuriot/isn2/

Objectif

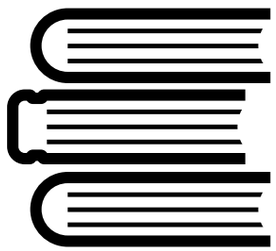
Deux étapes :

1. L’indexation d’une collection de documents
2. La recherche dans cette collection par mots-clés



Indexation

L’objectif de cette étape est de construire un **index inverse :**



	doc 1	doc 2	doc 3
racine 1	w1,1	w1,2	w1,3
racine 2	w2,1	w2,2	w2,3
racine 3	w3,1	w3,2	w3,3
racine 4	w4,1	w4,2	w4,3

Pour cela, nous allons réaliser plusieurs traitements :

1) Pré-traitement des documents : doc —> liste de racines

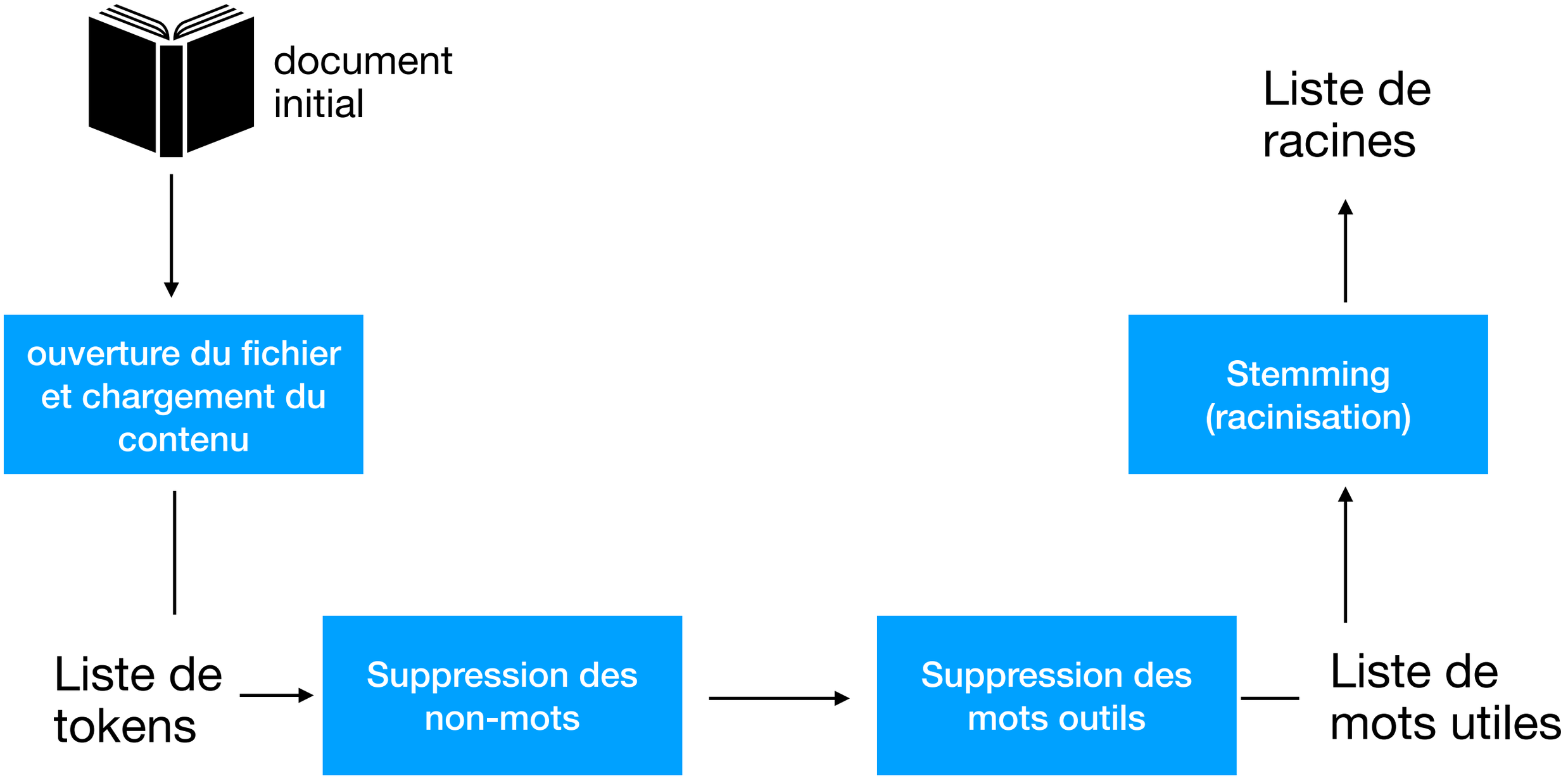
2) Calcul des tf et des df

3) Création de l’index et inversion

Indexation

1) Pré-traitement des documents : doc —> liste de racines

Pour chaque document, effectuer le traitement suivant :



Indexation

1) Pré-traitement des documents : doc ==> liste de racines

```
fonction doc2listeRacines

    – Entrée : chemin du document d

    – Sortie : liste des racines contenues dans d

    Ouverture du fichier et chargement de la chaine

    Transformation de la chaine en liste de mots

    Suppression des non-mots

    Suppression des mots-outils

    Stemming

    Renvoyer la liste de racines
```

fonction *loadFile* dans outils.py

fonction *string2list* dans outils.py

la fonction *contientLettre* de outils.py permet de déterminer si un mot ne contient pas de lettre (=non-mot)

la constante MOTSOUTILS de outils.py contient une liste de mots outils (l'opérateur *in* teste l'appartenance d'un élément à une liste)

fonction *mot2racine* dans outils.py

Indexation

2) Calcul des tf : dico (doc —> liste racines) ==> dico (doc —> racine —> tf)

```
fonction calculeTF

   – Entrée : liste des racines pour tous les documents (dico (doc –> liste racines))

   – Sortie : dictionnaire doc –> (racine –> tf)

   Initialisation du dictionnaire de sortie

   Pour chaque document et sa liste de racines Ld :

      Pour chaque racine r ∈ Ld :

         Calcul de la fréquence de r dans Ld

         Mise à jour du dictionnaire de sortie

   Renvoyer le dictionnaire
```

Indexation

2) Calcul des df : dico (doc —> liste racines) ==> dico (racine —> df)

```
fonction calculedf

   — Entrée : liste des racines pour tous les documents (dico (doc —>
liste racines))

— Sortie : dictionnaire (racine —> df)

Initialisation du dictionnaire de sortie

Pour chaque document et sa liste de racines Ld :

    Pour chaque racine r ∈ Ld :

        Mise à jour du dictionnaire de sortie

Renvoyer le dictionnaire
```

Indexation

3) Création de l'index : dicoTF et dicoIDF —> dico (doc —> racine —> tf.idf)

```
fonction créationIndex

   – Entrée : dicoTF et dicoDF (cf diapos précédentes)

   – Sortie : dictionnaire (doc –> (racine –> df))

Initialisation du dictionnaire de sortie dicoTFIDF

Pour chaque document d :

    Pour chaque racine r :

       dicoTFIDF[d][r] = TF.IDF normalisé

Renvoyer le dictionnaire
```

Indexation

3) Inversion de l’index :

dico (doc —> (racine —> tf.idf)) ==> dico (racine —> (doc —> tf.idf))

`Je vous laisse réfléchir à l’algo !!`

Indexation

Comment allons nous stocker l'index et les structures intermédiaires ?

Données	Structure Python
Contenu d'un document	str
Liste de mots dans un document	list<str>
Liste de racines dans un document	list<str>
Racines et leurs fréquences dans un document racine —> tf	dict<str, int>
Racines et leurs fréquences dans tous les documents doc -> racine -> tf	dict<str, dict<str, int>>
Fréquence documentaire des racines dans le corpus racine —> df	dict<str, int>
Index : Racines et leurs tf.idf par document doc —> racine —> tf.idf	dict<str, dict<str, float>>
Index inverse : Documents et tf.idf pour chaque racine racine —> doc —> tf.idf	dict<str, dict<str, float>>