



---

## MASTÈRE SPÉCIALISÉ BIG DATA GESTION ET ANALYSE DES DONNÉES MASSIVES

---

### *INF 344 : TP Moteur de recherche*

---

Phase 1 : *Crawling du site (Hugo Michel)*

Partie 2 : *Pré Calcul de TF-IDF (Hugo Michel)*

Partie 3 : *Calcul du PageRank (Sara Boutigny)*

Partie 4 : *Requêtage (Julien Lair)*

Télécom Paris

Année 2021 / 2022







Julien LAIR, Sara Boutigny, Hugo  
Michel

## Table des matières

<b>INF 344 :TP Moteur de recherche</b>	<b>1</b>
<b>Phase 1 : Description de la database</b>	<b>3</b>
<b>Phase 1 : Crawling du site</b>	<b>4</b>
<b>Phase 2 : Pré Calcul de TF-IDF</b>	<b>6</b>
Sous-phase 1 : Calcul de l'index inversé	6
Sous-phase 2 : Pré calcul de l'inverse document frequency	9
<b>Phase 3 : Calcul du PageRank</b>	<b>11</b>
<b>Phase 4 : Requêtage</b>	<b>15</b>

# 1. Phase 1 : Description de la database

Nous avons à notre disposition 2 tables SQL dans la base de données original qui nous est fourni :

▼  <b>responses</b>	CREATE TABLE responses (queryURL TEXT, respURL TEXT)	
 <b>queryURL</b>	TEXT	"queryURL" TEXT
 <b>respURL</b>	TEXT	"respURL" TEXT
▼  <b>webpages</b>	CREATE TABLE webpages (content TEXT, URL TEXT)	
 <b>content</b>	TEXT	"content" TEXT
 <b>URL</b>	TEXT	"URL" TEXT

## Table responses

- Contient : 9048 données

queryURL	respURL
URL que le crawler a demandé	URL que le crawler a reçu

## Exemple de données contenu dans la table responses

Table : responses	
Filtrer dans n'importe quelle colonne	
queryURL	respURL
Filtre	Filtre
1 https://wiki.jachiet.com/	https://wiki.jachiet.com/
2 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/index
3 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Pourcentage	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Pourcentage
4 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Valeur_absolue	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Valeur_absolue
5 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Jean-Pierre_Serre	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Jean-Pierre_Serre
6 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Notices_of_the_American_Mathematical_S
7 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Prix_Emil_Artin	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Prix_Emil_Artin
8 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Emil_Artin	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Emil_Artin
9 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Corps_réel_clos	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Corps_r%C3%A9el_clos
10 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Chen_Chung_Chang	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Chen_Chung_Chang
11 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Théorie_des_modèles	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Th%C3%A9orie_des_mod%C3%A8les
12 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Type_(th%C3%A9orie_des_mod%C3%A8
13 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ouvert_(topologie)	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ouvert_(topologie)
14 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Espace_séparé	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Espace_s%C3%A9par%C3%A9
15 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Droite_réelle_achevée	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Droite_r%C3%A9elle_achev%C3%A9e
16 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Compactifié_d'Alexandrov	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Compactifi%C3%A9_d'Alexandrov
17 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ordinal_successeur	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ordinal_successeur
18 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ordinal_limite	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Ordinal_limite
19 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Fonction_de_Veblen	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Fonction_de_Veblen
20 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/HI%C3%A9archie_de_croissance_rapide
21 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_de_Graham	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_de_Graham
22 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Richard_Guy	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Richard_Guy
23 https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_d'Erdf...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_d'Erdf%C5%91e

## Table webpages

- **Contient : 6424 données**

URL	content
<i>URL de la page en question</i>	<i>contenu HTML de la page associé à l'URL au format txt</i>

### Exemple de données contenu dans table webpages

Table : webpages		
	content	URL
	Filtre	Filtre
1	<!DOCTYPE html>...	https://wiki.jachiet.com/
2	<html><head><link type="text/css" href="/skin/jquery-ui/jquery-ui.min.css" ...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/index
3	<!DOCTYPE html> <html class="client-js"><head><link type="text/css" href="/skin/jquery-ui/jquery-ui.min.css" rel="stylesheet" /> <link type="text/css" href="/skin/jquery-ui/jquery-ui.theme.min.css" ...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Pourcentage
4	<!DOCTYPE html> <html class="client-js"><head><link type="text/css" href="/skin/jquery-ui/jquery-ui.min.css" rel="stylesheet" />...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Valeur_absolue
5	<!DOCTYPE html> <html class="client-js"><head><link type="text/css" href="/skin/jquery-ui/jquery-ui.min.css" rel="stylesheet" /> <link type="text/css" href="/skin/jquery-ui/jquery-ui.theme.min.css" ...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Jean-Pierre_Serre
6	<!DOCTYPE html> <html class="client-js"><head><link type="text/css" href="/skin/jquery-ui/jquery-ui.min.css" rel="stylesheet" /> <link type="text/css" href="/skin/jquery-ui/jquery-ui.theme.min.css" ...	https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Notices_of_the_American_Mathematical_Society

## 2. Phase 1 : Crawling du site

Auteur : Hugo Michel

Pour établir la connexion avec la base de données “data.db” depuis un fichier python (.py) on utilisera les lignes codes suivantes :

```
import sqlite3

conn = sqlite3.connect('data.db')
cursor = conn.cursor()
```

**Question 1 : Combien il y a de pages indexées ?**

Pour répondre à cette question, j'ai effectué la requête suivante :

```
# QUESTION 1 : Combien il y a de pages indexées ?
cursor.execute("SELECT COUNT(*) FROM webpages")
query_1 = cursor.fetchone()
print("QUESTION 1")
print("Il y a " + str(query_1[0]) + " indexées\n")
```

Retour console :

```
QUESTION 1
Il y a 6424 indexées
```

**Conclusion :** 6424 pages sont indexées

**Question 2 : Combien de pages ont la même URL requêtée et répondue ?**

Pour répondre à cette question, j'ai effectué la requête suivante :

```
# QUESTION 2 : Combien de pages ont la même URL requêtée et répondue ?
cursor.execute("SELECT COUNT(*) FROM responses WHERE responses.queryURL == responses.respURL")
query_2 = cursor.fetchone()
print("QUESTION 2")
print(str(query_2[0]) + " ont la même URL requêtée et répondue\n")
```

Retour console :

```
QUESTION 2
3197 pages ont la même URL requêtée et répondue
```

**Conclusion :** 3197 pages ont la même URL requêtée et répondue

**Question 3 : Certaines pages comme [https://wiki.jachiet.com/wikipedia\\_fr\\_mathematics\\_nopic\\_2020-04/A/Surface\\_de\\_Delaunay](https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Surface_de_Delaunay) ne sont pas indexées, pourquoi ? Pouvez-vous en trouver d'autres ?**

Pour répondre à cette question, j'ai effectué la requête suivante :

```
# QUESTION 3 : Certaines pages comme https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Surface_
print("QUESTION 3")
page_requete = 'https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Surface_de_Delaunay'
cursor.execute("SELECT COUNT(respURL) FROM responses WHERE queryURL = ?", (page_requete,))
if cursor.fetchone()[0] == 0:
    print("La page 'https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Surface_de_Delaunay' n'est
```

Retour console :

```
QUESTION 3
La page 'https://wiki.jachiet.com/
wikipedia_fr_mathematics_nopic_2020-04/A/Surface_de_Delaunay' n'est pas
indexée
```

On peut confirmer que la page indiquée dans l'énoncé n'est pas indexée.

Certaines URL ne sont pas indexées car elles ne sont pas accessibles depuis la homepage.

Il s'agit d'une page orpheline car une page orpheline c'est une page web qui est encore présente au sein d'un site web mais à laquelle on ne peut pas accéder en utilisant les liens proposés sur le site

Les autres URL orphelines peuvent être trouvées manuellement en naviguant sur la page du référençant les pages sur la thématique des mathématiques

## 3. Phase 2 : Pré Calcul de TF-IDF

### 3.1 Sous-phase 1 : Calcul de l'index inversé

L'objectif de cette première sous-phase de la phase 2 consiste à calculer un index inversé.

Pour ce faire, il convient d'associer à chaque mot clef  $w$  une liste  $(u_1; f_1) \dots (u_k; f_k)$  où les  $u_i$  sont des URL de documents qui contiennent le mot  $w$  et  $f_i$  est la fréquence d'occurrence de  $w$  dans  $u_i$ .

#### 1ère étape : Construction d'une fonction "countFreq()"

La fonction "countFreq()" prend une liste de mots en entrée et calcule l'occurrence d'apparition du mot dans la liste.

La fonction retourne une liste de tuple qui associe à chaque mot de la liste son occurrence d'apparition  $[(w_1; f_1) \dots (w_k; f_k)]$

#### countFreq

```
def countFreq(word_list):
    word_freq = [word_list.count(word) for word in word_list]
    dico_word_freq = dict(list(zip(word_list, word_freq)))

    # On ordonne le dictionnaire dans l'ordre décroissant (descending order)
    sorted_freq_dico = dict(sorted(dico_word_freq.items(), key=operator.itemgetter(1), reverse=True))

    # On convertit le nouveau dictionnaire en ordonnée en liste de tuple
    # format du tuple (word, occurrence)
    list_tuples_word_freq = [(k, v) for k, v in sorted_freq_dico.items()]
    return list_tuples_word_freq
```

**2ème étape Calcul de l'index inversé**

Avant de réaliser le calcul de l'index inversé, il convient de créer la table "inverted\_index" qui va contenir l'index. Cette table se présente comme suit :

keyword	URL	frequency
correspond à un mot dans le document à l'adresse "URL"	adresse URL de la page en question	correspond à la fréquence d'apparition du mot dans le document

Pour créer la table "inverted\_index" j'ai exécuté les lignes de codes suivantes :

```
conn.execute("DROP TABLE IF EXISTS inverted_index")
conn.commit()
conn.execute("CREATE TABLE inverted_index (keyword TEXT, URL TEXT, frequency REAL)")
conn.commit()
```

▼	inverted_index	CREATE TABLE inverted_index (keyword TEXT, URL TEXT, frequency REAL)
	keyword	TEXT "keyword" TEXT
	URL	TEXT "URL" TEXT
	frequency	REAL "frequency" REAL

Une fois la table créée, il convient d'insérer les données en question. Pour ce faire, j'ai réalisé les étapes suivantes :

1. Récupération de la liste des URL de la table webpages (url des pages indexées) en itérant simplement sur les lignes de données contenus dans cette table et en y récupérant l'URL de la page en question
2. Extraction des mots contenu dans le code HTML de la page en utilisant la méthode extractListOfWords()
3. Stemmatisation de chaque mot extrait
4. Calcul de la fréquence d'apparition de chaque mot extrait et "stemmatisé"
5. Insertion de chaque tuple (word,frequency) dans la base de données "inverted\_index"

**Table inverted\_index**

	keyword	URL	frequency
	Filtre	Filtre	Filtre
1	kiwix	https://wiki.jachiet.com/	0.1111111111111111
2	mathématic	https://wiki.jachiet.com/	0.1111111111111111
3	wikipédia	https://wiki.jachiet.com/	0.1111111111111111
4	welcom	https://wiki.jachiet.com/	0.0555555555555556
5	to	https://wiki.jachiet.com/	0.0555555555555556
6	server	https://wiki.jachiet.com/	0.0555555555555556
7	par	https://wiki.jachiet.com/	0.0555555555555556
8	un	https://wiki.jachiet.com/	0.0555555555555556
9	sélection	https://wiki.jachiet.com/	0.0555555555555556
10	d	https://wiki.jachiet.com/	0.0555555555555556
11	articl	https://wiki.jachiet.com/	0.0555555555555556
12	sur	https://wiki.jachiet.com/	0.0555555555555556

Le traduction en code de ses différentes et décrit comme suit :

### Calcul de TF

```
counter_insert = 0
# On récupère la liste des URL de la table webpages (url des pages indexées)
for row in cursor.execute("SELECT * FROM webpages"):

    word_list_extracted = list(extractListOfWords(row[0]))
    #print(word_list_extracted)

    word_list_stemmed = list()

    # On stemmatize chaque mot extrait de l'url de la page
    for word in word_list_extracted:
        word_stemmed = stem(word)
        word_list_stemmed.append(word_stemmed)
    #print(word_list_stemmed)

    # On crée la liste de tuple (word, frequence)
    tuple_word_freq = countFreq(word_list_stemmed)
    #print(tuple_word_freq)

    # On nourrit la table inverted_index avec la liste de tuple que l'on vient de créer
    for elm in tuple_word_freq:

        cursor1.execute('INSERT INTO inverted_index VALUES (?, ?, ?)', (elm[0], row[1], elm[1]/len(word_list_stemmed)))

    # Log pour le suivi de l'insertion des données en DB
    counter_insert = counter_insert + 1
    print(counter_insert)
```

Note : J'ai utilisé un compteur "counter\_" pour avoir un suivi de l'évolution de l'insertion des données au sein de la base de données.

Maintenant que la table "inverted\_index" est alimentée en donnée, il convient de créer l'index "inv\_ind" sur la colonne "keyword"

```
# On crée un index inverse pour la table inverted_index en se basant sur la colonne 'keyword'
conn.execute("DROP INDEX IF EXISTS inv_ind")
conn.commit()
conn.execute("CREATE INDEX inv_ind ON inverted_index(keyword)")
conn.commit()
```

### Index inverted\_index

<div> <div>inv_ind</div> <div>keyword</div> </div>	CREATE INDEX inv_ind ON inverted_index(keyword) "keyword"
--	--



**Question 4 : Dans quelle page apparaît le plus souvent (en fréquence) le terme “matrice” ?**

Pour répondre à cette question, j’ai effectué la requête suivante :

```
# Question 4 : Dans quelle page apparaît le plus souvent (en fréquence) le terme “matrice” ?
cursor.execute("SELECT keyword, URL, MAX(frequency) FROM inverted_index WHERE keyword LIKE('matric')")
page_matrice = cursor.fetchone()
print("\n\nQUESTION 4")
print("Le terme " + page_matrice[0] + " apparaît le plus souvent dans " + page_matrice[1] + " avec une fréquence de " + str(page_matrice[2]))
```

Retour console :

```
QUESTION 4
Le terme matric apparaît le plus souvent dans https://wiki.jachiet.com/
wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_involutive avec une
fréquence de 0.07792207792207792
```

**Conclusion :**

Ainsi le terme matrice apparaît le plus dans la page

[https://wiki.jachiet.com/wikipedia\\_fr\\_mathematics\\_nopic\\_2020-04/A/Matrice\\_involutive](https://wiki.jachiet.com/wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_involutive)

avec un fréquence de 0.0779.

Note : Le terme matrice devient “matric” après stemmatisation.

### 3.2 Sous-phase 2 : Pré calcul de l’inverse document frequency

Rappel formule de IDF :  $\log(N/n_w)$

avec  $N$  le nombre de documents et  $n_w$  le nombre d'occurrence où le mot  $w$  apparaît.

Comme pour la sous-phase 1, avant de calculer l’IDF de chaque mot, il convient de créer la table qui contiendra l’IDF correspondant à chaque mot extrait et stemmetisé.

La table “invert\_document\_frequency” contient les champs suivants :

keyword	idf
mot contenu dans la page en question	valeur idf correspondant au mot

Pour créer la table “invert\_document\_frequency” j’ai executé le lignes de code suivantes :

```
# Création de la table invert_document_frequency (keyword, idf)
conn.execute("DROP TABLE IF EXISTS invert_document_frequency")
conn.commit()
conn.execute("CREATE TABLE invert_document_frequency (keyword TEXT, idf REAL)")
conn.commit()
```

invert_document_frequency		CREATE TABLE invert_document_frequency (keyword TEXT, idf REAL)
keyword	TEXT	"keyword" TEXT
idf	REAL	"idf" REAL

Ensuite pour calculer la valeur IDF pour chaque mot j'ai procédé de la manière suivante :

1. Récupération le nombre d'url total contenu dans la table webpages pour le calcul de IDF. Cette valeur correspond tout simplement au nombre de documents  $N$  de la formule IDF que nous avons énoncée précédemment nous sera utile pour le calcul de l'IDF.
2. Insertion en db le mot et sa valeur idf

```
# On récupère le nombre d'url contenu dans la table webpages pour le calcul de IDF
cursor.execute("SELECT COUNT(*) FROM webpages")
nb_url = cursor.fetchone()[0]
print(nb_url)

# On insère en db le mot et sa valeur idf (idf = log(N/n_w))
# N est le nombre de documents
# n_w le nombre de document où w apparaît
counter_insert_bis = 0

for row in cursor.execute("SELECT keyword, COUNT(URL) FROM inverted_index GROUP BY keyword"):
    cursor1.execute("INSERT INTO invert_document_frequency VALUES (?,?)", (row[0], log(nb_url / row[1])))
    counter_insert_bis = counter_insert_bis + 1
    print(counter_insert_bis)

# On sauvegarde les insertions en db
conn.commit()
```

Table invert\_document\_frequency

	keyword	idf
	Filtre	Filtre
1		0.00577630206500836
2	_	1.50516765465236
3	_____	8.07464907506665
4	_____...	8.7677962556266
5	__swaqaamaaj	8.7677962556266
6	_daniel	8.7677962556266
7	_generalized	8.7677962556266
8	_hom	8.7677962556266
9	_icmat	8.7677962556266
10	_p	8.07464907506665
11	a	0.186689738466707
12	a_	2.86516292222523

De manière analogue à la sous-phase 1, une fois la table alimentée en donnée, il convient de créer un index “inv\_doc\_freq\_ind” pour le champ “keyword”. La création de cet index s’effectue avec les lignes de code suivantes :

```
# Création de l'index pour la table "invert_document_frequency"
conn.execute("DROP INDEX IF EXISTS inv_doc_freq_ind")
conn.commit()
conn.execute("CREATE INDEX inv_doc_freq_ind ON invert_document_frequency(keyword)")
conn.commit()
```

### Index inv\_doc\_freq\_ind

inv_doc_freq_ind	CREATE INDEX inv_doc_freq_ind ON invert_document_frequency(keyword)
keyword	"keyword"

### Question 5 : Quel est l’IDF de “matrice” ?

Pour répondre à cette question, j’ai effectué la requête suivante :

```
# Question 5 : Quel est l’IDF de “matrice” ?
cursor.execute("SELECT * FROM invert_document_frequency WHERE keyword LIKE('matric')")
idf_matrice = cursor.fetchone()
print("\n\nQUESTION 5")
print("IDF de " + idf_matrice[0] + " = " + str(idf_matrice[1]))
```

Retour console :

```
QUESTION 5
IDF de matric = 2.0695282015111847
```

**Conclusion :** L’IDF du terme “matrice” est de 2.069

## 4. Phase 3 : Calcul du PageRank

Dans un premier temps, on réalise le graphe sur lequel on va ensuite faire le PageRank.

Etape 1 : calculer un dictionnaire realURL qui retourne la respURL associée à une queryURL

```
conn = sqlite3.connect('data.db')
cursor = conn.cursor()

realURL = {}
for row in cursor.execute("SELECT queryURL, respURL FROM responses"):
    queryURL, respURL = row[0], row[1]
```

```
realURL[queryURL] = respURL
```

Etape 2 : calculer un dictionnaire pointsTo qui stocke la liste neighbors(content) pour chaque respURL

```
pointsTo = {}
for row in cursor.execute("SELECT URL, content FROM webpages"):
    URL, content = row[0], row[1]
    pointsTo[URL] = neighbors(content, URL)
```

Etape 3 : Modifier pointsTo pour que les queryURL soient remplacés par des respURL

```
d = pointsTo.copy()
df = pd.DataFrame(dict([ (k,pd.Series(v)) for k,v in d.items() ]))

list_realURL_key = [i for i in realURL.keys()]

for i in range(df.shape[1]):
    for j in range(df.shape[0]):
        if pd.isnull(df.iloc[j,i]) != True and df.iloc[j,i] in list_realURL_key:
            df.iloc[j,i] = realURL[df.iloc[j,i]]

list_pointsTo_key = [i for i in pointsTo.keys()]

for i in list_pointsTo_key:
    pointsTo[i] = df.loc[pd.isnull(df[i])!=True,i].tolist()
```

Pour l'algorithme du PageRank, nous avons suivi les étapes données dans l'énoncé. Cela nous donne l'algorithme suivant :

```
# PageRank
-----
score, nouveau_score, nombre_de_pages = dict(),dict(), len(pointsTo.keys())

# score = [1/N pour chaque page]
```

```

for r in pointsTo: score[r] = 1/nombre_de_pages

# Pour chaque itération
for i in range(NB_ITERATIONS):

    # nouveau_score = [ 0 pour chaque page ]
    for r in pointsTo: nouveau_score[r] = 0

    # proba_téléportation = 0
    proba_teleportation = 0

    # pour chaque page P:
    for r in pointsTo:

        # s'il y a des liens sur P:
        if len(pointsTo[r]) > 0 :

            # proba_téléportation += ALPHA × score[P]
            proba_teleportation += ALPHA * score[r]

            # pour chaque lien de P vers P' : nouveau_score[P'] += score[P] ×
            (1-ALPHA) / nombre de liens sur P
            for u in pointsTo[r]: nouveau_score[u] += score[r] * (1 - ALPHA) /
            len(pointsTo[r])

            # sinon (s'il n'y a pas de lien sur P):
            else:

                # proba_téléportation += score[P]
                proba_teleportation += score[r]

            # pour chaque page P: nouveau_score[P] += proba_téléportation / nombre de
            pages
        for r in pointsTo: nouveau_score[r] += proba_teleportation / nombre_de_pages

    # score = nouveau_score
    for r in nouveau_score: score[r] = nouveau_score[r]

```

On peut ensuite créer une table dans la base de données afin d'intégrer les scores calculés :

```
conn.execute("DROP TABLE IF EXISTS page_rank")
conn.commit()
conn.execute("CREATE TABLE page_rank (URL TEXT, rank_score REAL)")
conn.commit()

for key, value in score.items():
    conn.execute('INSERT INTO page_rank VALUES (?,?)', (key, value))

conn.commit()
conn.close()
```

La table possède deux colonnes : un URL nommé "URL" et le score associé "rank\_score".

### Question 6 : Quelles sont les 20 pages qui ont le meilleur PageRank ? Comment l'expliquez-vous ?

Les 20 premiers URLs qui apparaissent sont les suivants :

```
1.976% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Fonction_de_Legendre
1.967% : .../skin/jquery-ui/jquery-ui.theme.min.css
1.967% : .../skin/jquery-ui/jquery-ui.min.css
1.966% : .../wikipedia_fr_mathematics_nopic_2020-04/A/index
1.966% : .../skin/taskbar.css
1.957% : .../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/style.css
1.941% :
.../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/skins.minerva.base.reset%7Cs
kins.minerva.content.styles%7Cext.cite.style%7Csite.styles%7Cmobile.app.pagestyles.andr
oid%7Cmediawiki.page.gallery.styles%7Cmediawiki.skinning.content.parsoid.css
1.941% : .../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/inserted_style.css
1.941% : .../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/content.parsoid.css
1.226% :
.../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/ext.cite.ux-enhancements.css
1.226% : .../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/ext.cite.styles.css
1.133% : .../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/ext.math.styles.css
1.133% :
.../wikipedia_fr_mathematics_nopic_2020-04/-/s/css_modules/ext.math.scripts.css
0.922% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Math%C3%A9matiques
0.521% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Math%C3%A9maticien
0.287% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_complexe
0.269% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Nombre_r%C3%A9el
0.250% : .../wikipedia_fr_mathematics_nopic_2020-04/A/G%C3%A9om%C3%A9trie
0.219% : .../wikipedia_fr_mathematics_nopic_2020-04/A/Polyn%C3%B4me_de_Legendre
0.205% : .../wikipedia_fr_mathematics_nopic_2020-04/A/American_Mathematical_Society
```

Le PageRank consiste à attribuer aux pages un score qui soit proportionnel au nombre de passages qu'un utilisateur ferait dessus s'il se déplaçait aléatoirement d'une page à une autre en cliquant sur les liens qui s'y trouvent. C'est un indicateur de centralité d'une page. Si beaucoup de pages pointent vers une même page, alors cette dernière aura un

score plus élevé. Dans les 20 premiers URLs, on trouve des liens correspondant à des index et des .css de mise en page. En effet, lorsque l'on se déplace sur une page, on fait régulièrement appel aux autres pages nécessaires à son bon fonctionnement et affichage, d'où ce résultat. On trouve aussi des liens supposés revenir souvent dans le texte d'autres pages tels qu'un lien vers la page "Mathématiques" ou "Mathématicien".

## 5. Phase 4 : Requêtage

### 5.1. Avec simplement tf-idf

Dans cette partie, on implémente le tri avec la métrique tf-idf.

**Question 7 : Quelles sont les dix premières pages pour "comment multiplier des matrices" en utilisant seulement la métrique tf-idf ?**

```
Résultat avec tf-idf
1: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_involutive
2: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_%C3%A9l%C3%A9mentaire
3: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_de_permutation
4: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_normale
5: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_de_Hessenberg
6: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_identit%C3%A9
7: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_orthogonale
8: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_des_degr%C3%A9s
9: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_binaire
10: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_unitaire
```

### 5.1. Avec tf-idf et PageRank

**Question 8 : Quelles sont les dix premières pages pour "comment multiplier des matrices" en combinant tf-idf et Page-Rank?**

```
1: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_(math%C3%A9matiques)
2: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_identit%C3%A9
3: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_diagonale
4: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_d'adjacence
5: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_orthogonale
6: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_inversible
7:
.../wikipedia_fr_mathematics_nopic_2020-04/A/Polyn%C3%B4me_caract%C3%A9ristique
8:
.../wikipedia_fr_mathematics_nopic_2020-04/A/Groupe_g%C3%A9n%C3%A9ral_lin%C3%A9aire
9:
.../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_d'une_application_lin%C3%A9aire
10: .../wikipedia_fr_mathematics_nopic_2020-04/A/Matrice_de_permutation
```

Le TF-IDF permet de trouver les pages qui correspondent le plus à la requête tandis que le PageRank renvoie les pages qui reviennent le plus souvent dans les autres pages. En combinant les deux, on obtient un arbitrage entre ces deux méthodes : des pages qui correspondent à la requête et des pages qui sont considérées comme “centrales” (e.g. qui reviennent souvent dans les liens d’autres pages).