

Rapport :

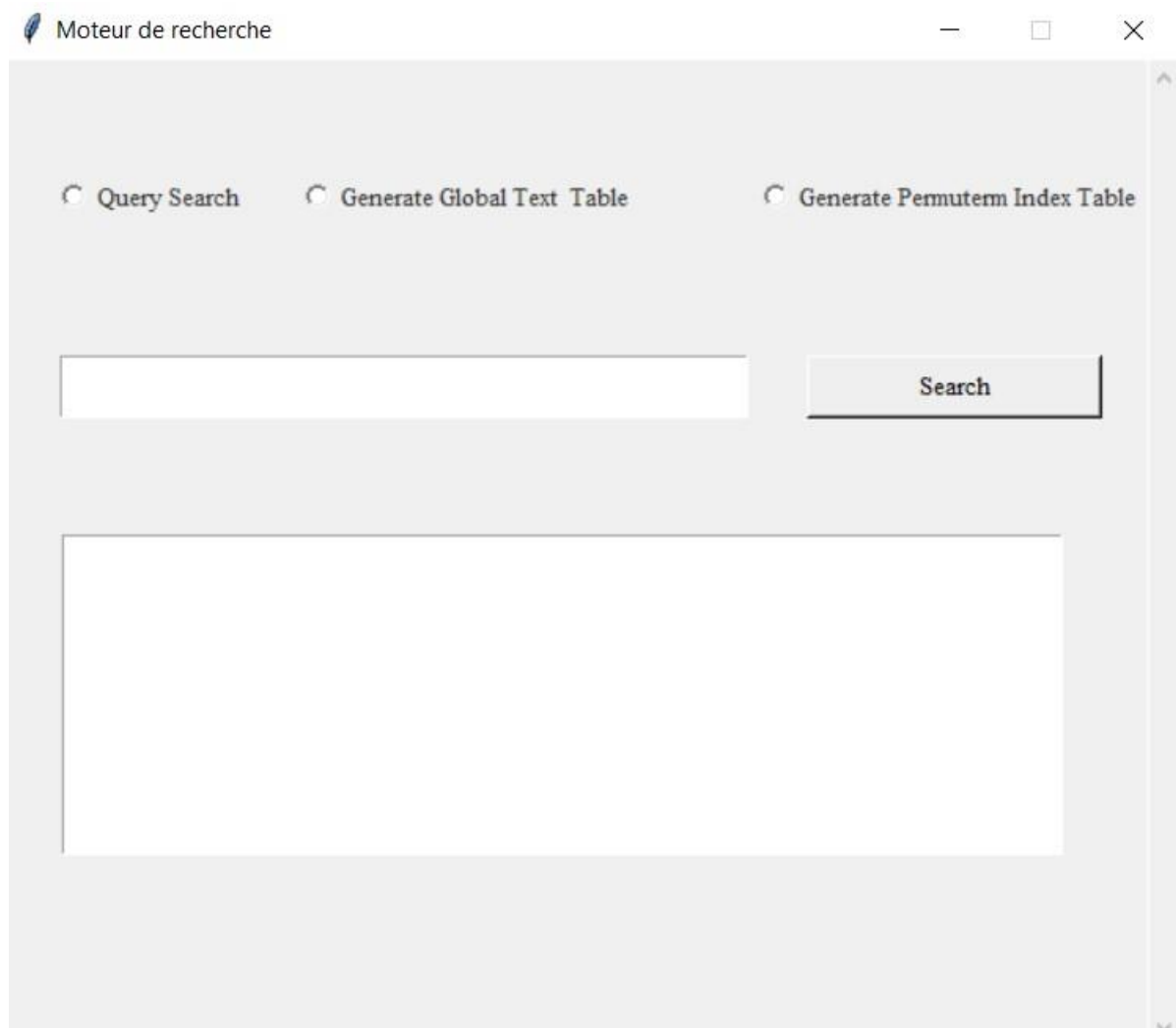
Technologie de moteurs de recherche

Table de matières

Partie 1 : Présentation générale du moteur de recherche	3
Partie 2 : Etude de conception	4
1.Diagramme de cas d'utilisation	4
2.Diagramme de classe	4
3.Les opérations principales	5
 Partie 3 : Exécution du moteur de recherche	 6
1. Exemple du mot « Health »	6

Présentation générale du moteur de recherche :

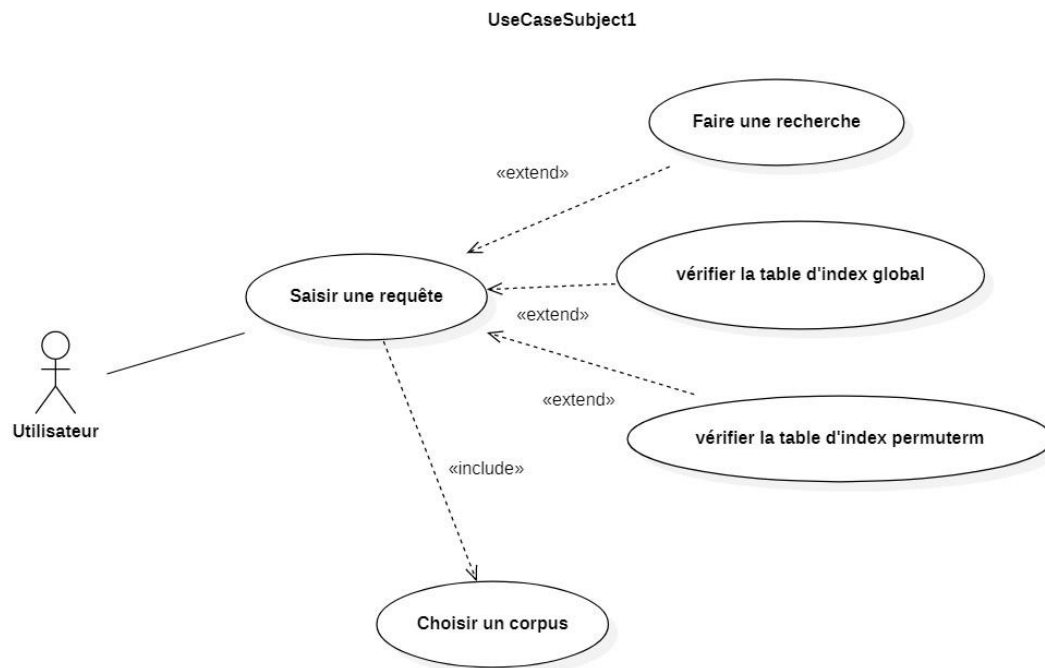
- On a réalisé un moteur de recherche alimenté avec un corpus (documents, articles, etc.) dont l'utilisateur peut saisir sa requête et obtenir le résultat correspondant.
- Ce moteur de recherche prend en charge les requêtes de termes/booléens, les requêtes de phrases ainsi que les requêtes de caractères génériques.
- Il respecte le processus du traitement de l'indexation dans les opérations des recherches.
- Ce moteur a été réalisé par Python, en se divisant par 2 tâches principales :
 - La création du moteur de recherche : processus de traitement.
 - La réalisation de l'interface graphique qu'on va implémenter.



- Voici notre interface du moteur de recherche.

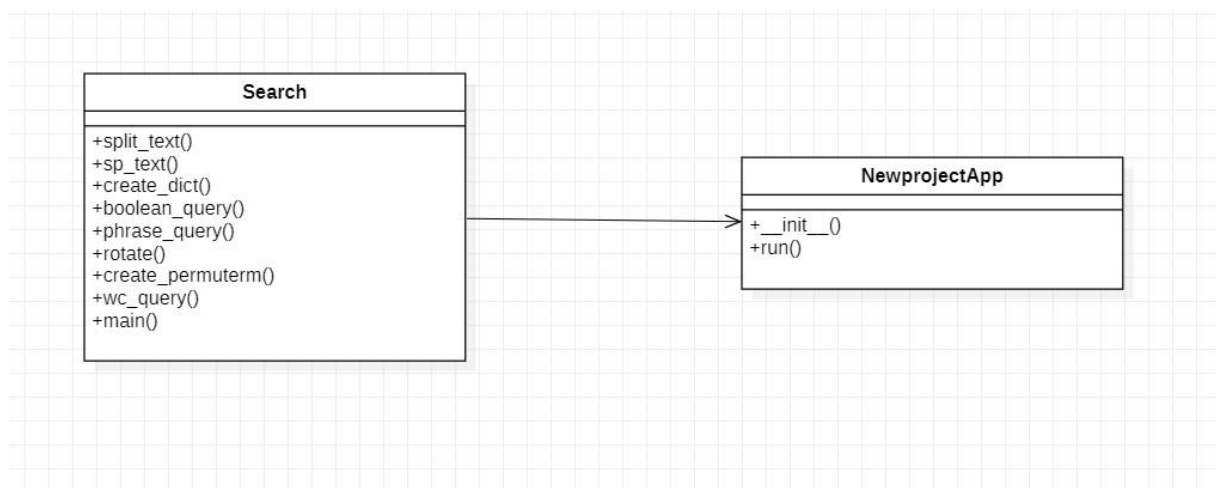
Etude de conception :

1) Diagramme de cas d'utilisation



- Ce diagramme nous présente les différentes fonctions principales de l'utilisateur du moteur de recherche.

2) Diagramme de classe



- Ce diagramme nous présente les deux classes utilisées dans la création de notre moteur de recherche et leurs attributs : la classe search et la classe App.
- La classe App concerne la création de l'interface graphique GUI.
- La classe Search concerne l'algorithme utilisée dans la création du moteur de recherche, qu'on va implémenter dans l'interface graphique pour avoir un moteur avec interface afin de faciliter les tâches.

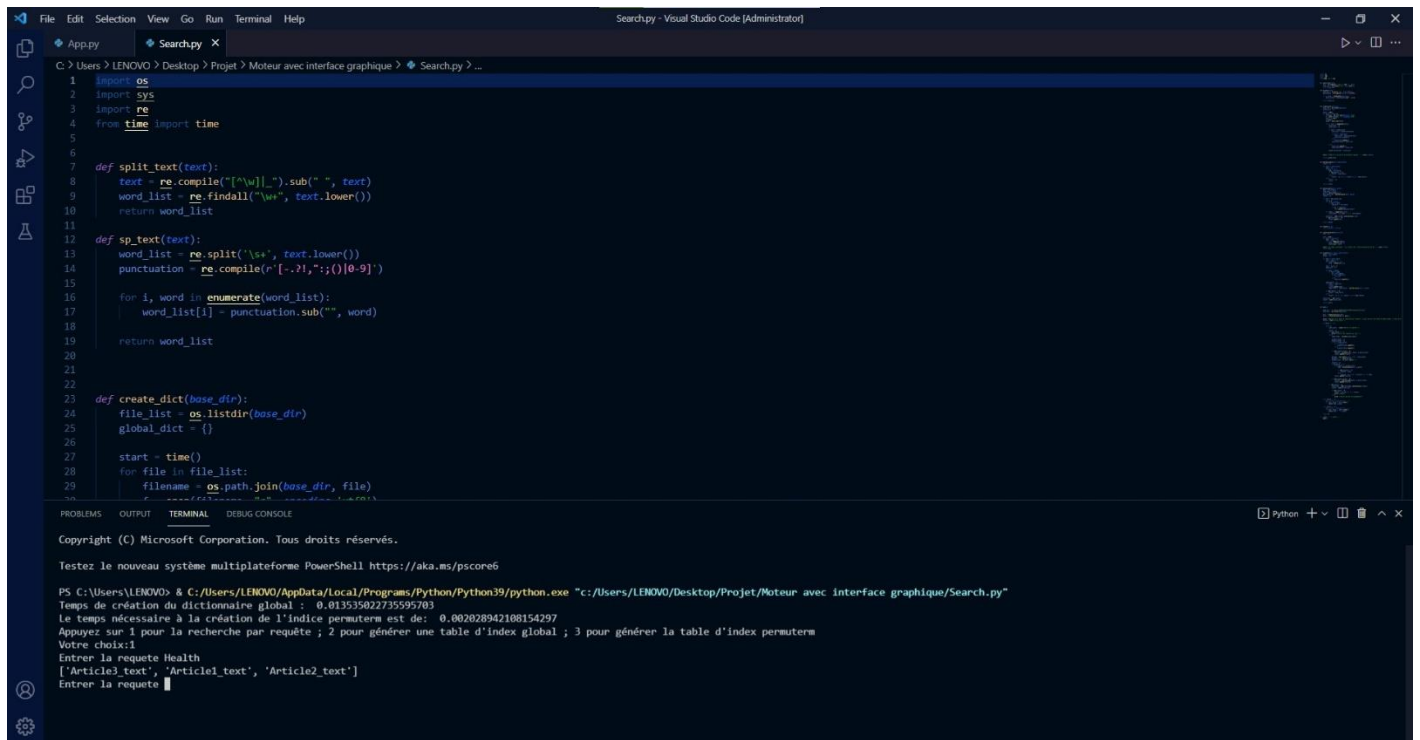
3) **Les opérations principales**

- On présente les différents attributs/opérations principaux pour notre projet :
- create_dict
 - Paramètres : l'emplacement du répertoire de base où sont placés tous les fichiers de test.
 - Renvoie : dictionnaire avec clé comme chaîne et valeur comme autre dictionnaire avec key comme nom de fichier et value comme index de position.
- split_text
 - Paramètres : chaîne.
 - Renvoie : liste de jetons.
- Boolean_query
 - Paramètres : dictionnaire global qui est créé avec create_dict() et liste de termes à interroger.
 - Renvoie : liste contenant les noms de fichiers résultants satisfaisant la requête booléenne sur query_terms.
- rotate
 - Paramètres : chaîne à faire pivoter et nombre d'indices à prendre en compte lors de la rotation.
 - Renvoie : chaîne tournée.
- create_permuterm
 - Paramètres : liste de chaînes.

- Renvoie : un dictionnaire avec clé comme permuterm et valeur comme chaîne de la liste de mots à partir de laquelle le permuterm est créé.
- wc_query
- Paramètres : dictionnaire du create_dict() ,dictionnaire de create_permuterm() liste des chaînes à prendre en compte pour la requête générique.
 - Renvoie : liste contenant les noms de fichiers résultants pour la requête avec caractères génériques.
- Main()
- Paramètres : aucun.
 - Cette fonction sert principalement à obtenir la requête d'entrée de l'utilisateur, puis séparer la requête d'entrée et appeler les fonctions de requête nécessaires (boolean_query(), phrase_query(), wc_query()) puis peigner les résultats obtenu à partir de différentes fonctions de requête, puis génère le résultat final.

Exécution du moteur de recherche :

1. Exemple de recherche du mot « Health » dans notre corpus :



```
1 import os
2 import sys
3 import re
4 from time import time
5
6
7 def split_text(text):
8     text = re.compile("[^w]").sub(" ", text)
9     word_list = re.findall("[w]", text.lower())
10    return word_list
11
12 def sp_text(text):
13     word_list = re.split('[^s]', text.lower())
14     punctuation = re.compile("[^a-zA-Z;0-9]")
15     for i, word in enumerate(word_list):
16         word_list[i] = punctuation.sub("", word)
17     return word_list
18
19
20 def create_dict(base_dir):
21     file_list = os.listdir(base_dir)
22     global_dict = {}
23
24     start = time()
25     for file in file_list:
26         filename = os.path.join(base_dir, file)
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

Copyright (C) Microsoft Corporation. Tous droits réservés.

Testez le nouveau système multiplateforme PowerShell <https://aka.ms/pscore6>

PS C:\Users\LENOVO> & C:\Users\LENOVO\AppData\Local\Programs\Python\Python39\python.exe "c:/Users/LENOVO/Desktop/Projet/Moteur avec interface graphique/Search.py"

Temps de création du dictionnaire global : 0.013535022735595703

Le temps nécessaire à la création de l'indice permuterm est de: 0.002028942108154297

Appuyez sur 1 pour la recherche par requête ; 2 pour générer une table d'index global ; 3 pour générer la table d'index permuterm

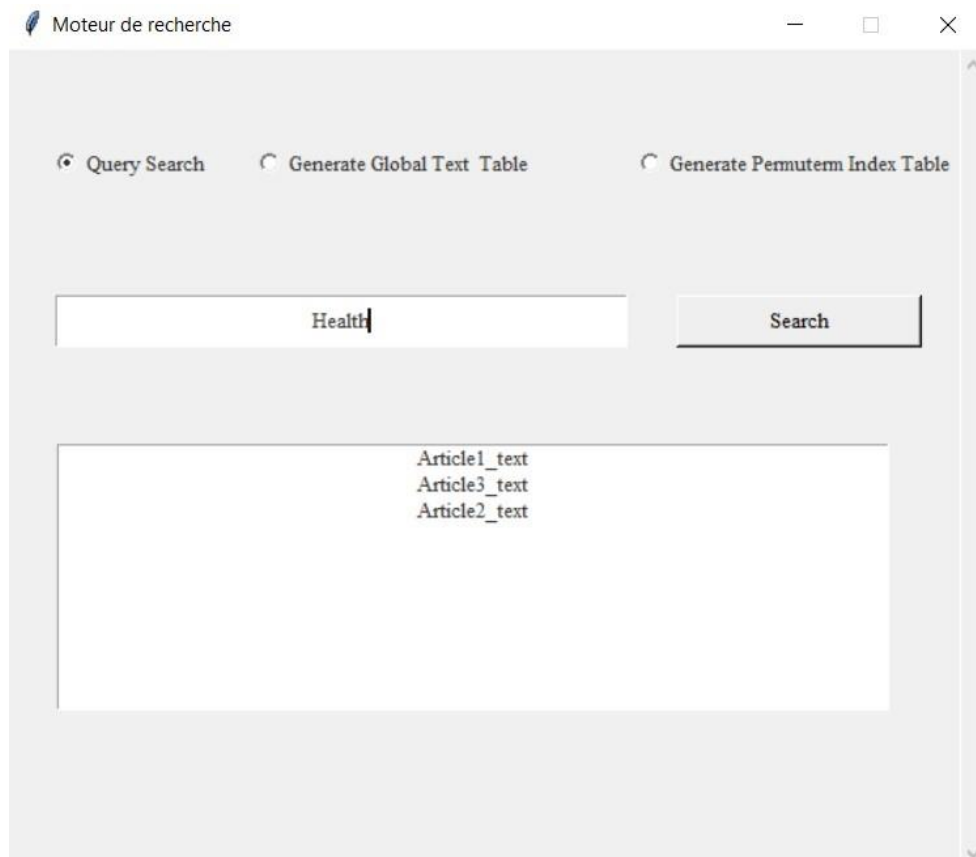
Votre choix:1

Entrer la requete Health

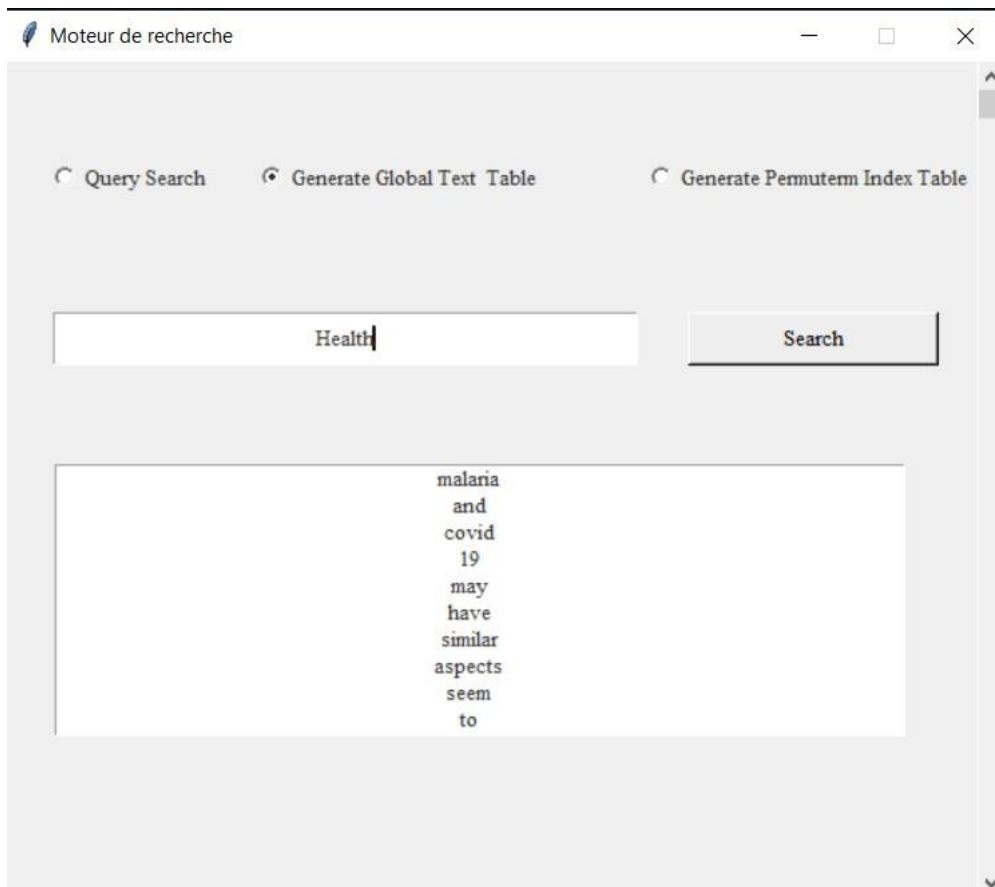
['Article3_text', 'Article1_text', 'Article2_text']

Entrer la requete

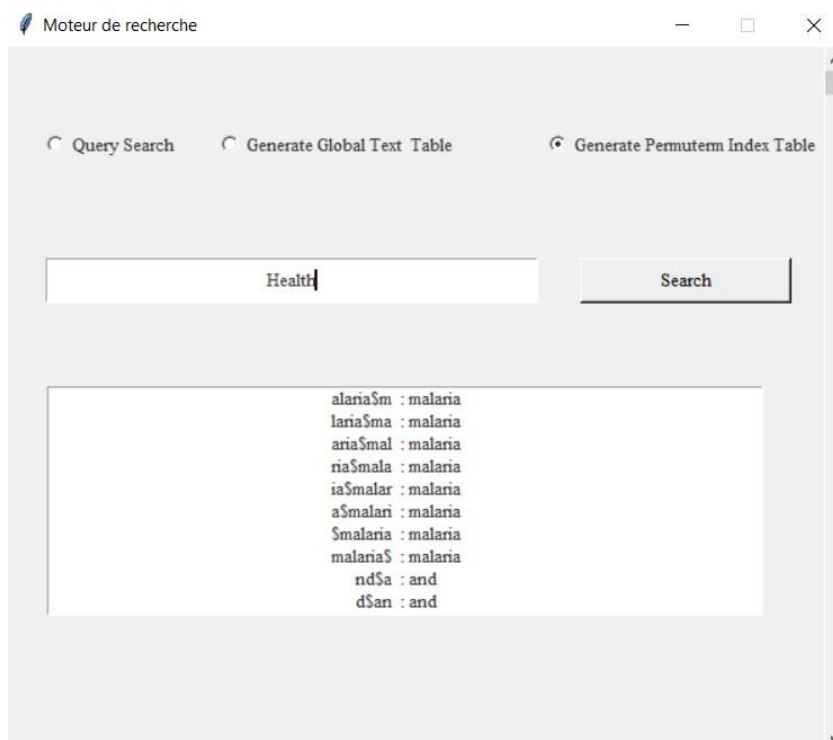
- Voici une exécution de recherche normale du code pour chercher le mot : Health
- On remarque qu'il existe dans les trois articles 1,2 et3.



- On remarque la même exécution mais en utilisant l'interface graphique créée.



- Vérifier le fichier pour voir la table d'index global.
- La table d'index global est un dictionnaire avec une clé comme chaîne et une valeur comme dictionnaire.



- Vérifier le fichier pour voir la table d'index permuterm.
- La table d'index Permuterm est un dictionnaire avec une clé en tant que permuterm et une valeur en tant que clé du dictionnaire global.