**Rapport – LI Bo et PARTHIPAN Agentan**

**Projet IN104 : Search engines**

L'objectif du projet était d'implémenter en langage Python un moteur de recherche qui opérerait sur une base de données locale fournie à l'avance. Cette base de données est un ensemble de fichiers textes (articles du journal *The Guardian*) qui ne contiennent pas d'images ni de vidéos.

Les deux premiers TPs sur les marchés de fruits et les nombres romains, qui n'étaient pas directement en lien avec le projet, nous ont permis de nous familiariser avec les avantages d'une programmation orientée objet, notamment avec l'usage de classes et de fonctions sur les objets de ces classes. Nous n'avons pas eu de difficultés majeures à comprendre ces concepts car le langage Python nous était déjà connu.

À partir de l'algorithme proposé par le professeur pour rechercher des articles à l'aide de mots clés, **voici la méthode que nous avons réussi à implémenter** :

- création de deux index (étape dite d'indexation) : dans le premier, on associe à chaque mot de la base de données un unique nombre, et dans le deuxième, on fait de même pour chaque article. Puis, dans le premier index, pour chaque mot, on garde en mémoire des statistiques : les index des articles où apparaît ce mot, et le nombre d'occurrence de ce mot dans ces articles.
- Calcul des TF-IDFs de la requête et de chaque document où apparaît au moins un mot de la requête (même si c'est un petit mot insignifiant comme « et », « le », « la », « de », … . Mais la TF-IDF sera alors proche de zéro.)
- calcul des similarités entre la requête et chaque article à partir des TF-IDFs
- tri des articles en fonction de leur similarités avec la requête, pour afficher les chemins qui mènent à chacun d'eux, du plus similaire au moins similaire

Nous pensions que la détermination des différentes quantités numériques (nombre d'occurrence d'un mot dans un article, nombre d'articles contenant un mot, …) intervenant dans le calcul des TF-IDFs se ferait au moyen d'une comparaison des mots de la requête et des mots des articles **lettre par lettre.** Cela nous avait conduit à écrire un algorithme assez lent.

Nous avons alors saisi l'importance de l'étape d'indexation. Associer un unique nombre à chaque mot de la base de données permettait en fait de déterminer si un mot apparaît dans un article simplement en regardant **si le nombre correspondant à ce mot apparaît dans la liste des index des mots de l'article considéré**. On a alors obtenu un algorithme bien plus rapide.

La structure d'un **dictionnaire de dictionnaires** pour indexer les mots de la base de données, et pour garder en mémoire les articles où ils apparaissaient et les occurrences avec lesquelles ils apparaissaient, s'est révélée plus intéressante que celle d'un tableau de tableaux. La première structure nous permet d'accéder aux statistiques d'un mot en se servant directement du mot que l'on rencontre lorsqu'on parcourt un article. Alors qu'avec un tableau, on devrait identifier l'index correspondant à ce mot rencontré avant de pouvoir utiliser l'entier qu'est cet index pour accéder aux statistiques du mot stockées dans le tableau.

Dans ce projet, le temps que mettrait l'algorithme à créer l'index des articles et l'index de tous les mots de la base de données demeurait quelques peu incompressible car la quantité de données était très importante. Mais le professeur nous a fait prendre conscience que **si l'on gardait les index dans la mémoire** de l'ordinateur, on pourrait aller les consulter à chaque fois que l'on a besoin d'utiliser le moteur de recherche. Cela est plus efficace que de passer du temps à recréer les

mêmes index à chaque fois que l'on utilise le moteur de recherche.

On a créé une **fonction de sauvegarde et une fonction de chargement pour garder et utiliser les index en mémoire**. On avait d'abord écrit les algorithmes de sorte à stocker les dictionnaires sous forme de chaîne de caractères. Cependant, lorsqu'on essayait de charger ces dictionnaires ainsi stockés pour faire une deuxième requête, une erreur bloquait l'exécution de la recherche. Sur les conseils du professeur, nous avons alors **stocké puis chargé les index non pas sous forme de chaîne de caractères, mais directement sous forme binaire**, avec les arguments optionnels *« wb »* (write binary) et *« rb »* (read binary) de la fonction *open*. Il nous a aussi fallu comprendre que l'on ne pouvait pas enregistrer un objet d'une classe que nous avions nous-même créé, et qu'il nous fallait donc enregistrer **séparément** les deux dictionnaires correspondant aux deux index de notre classe « Tableau ». Par ailleurs, comme on ne peut sauvegarder que des index déjà créés et par conséquent déjà stockés dans des variables, il nous a semblé plus cohérent d'écrire la fonction « save » comme une fonction de la classe « Tableau ». Alors que, comme on peut charger des index qui ne sont pas forcément déjà manipulés dans le code, il nous a semblé cette fois-ci plus cohérent d'écrire la fonction « load » hors de la classe « Tableau ».

```
11
12      def save(self, save_path):  #sauvgarde les 2 dictionaires du tableau
13          article = open(save_path, 'wb')
14          pickle.dump(self.INDEX, article)
15          pickle.dump(self.ID_docs, article)
16          article.close()
17
93
94  def load(save_path): #charger l'index precedemment enregistre
95      f = open(save_path, 'rb')
96      D = Tableau()
97      D.INDEX = pickle.load(f)
98      D.ID_docs = pickle.load(f)
99      f.close()
100     return D
```

Un des points essentiels de notre travail sur le code est la construction d'une **boucle récursive pour pouvoir explorer tous les sous-dossiers de la base de données afin de ne pas manquer un article.** Nous avions eu du mal à comprendre en quoi il était **nécessaire de procéder récursivement**, mais il nous a finalement semblé clair que **si ce n'était pas le cas, les index seraient modifiés indépendamment dans les différentes branches de l'arborescence de la base de données, ce qui conduirait potentiellement à perdre l'unicité du nombre associé à un mot ou à un article.**

```
            pickle.dump(self.ID_docs, article)
            article.close()

        def initialisation(self, bdd_path, id):
            for element in os.listdir(bdd_path):
                if element.endswith('.txt'):
                    element_path = bdd_path + '/' + element
                    self.ID_docs[id] = element_path
                    article = open(element_path, encoding='utf-8', errors='ignore') #parame
                    article_content = article.read()
                    transf_article_content = word_tokenize(article_content)
                    mots_dejavus_ici = []

                    for word in transf_article_content:
                        if word not in mots_dejavus_ici:
                            nb_occ = transf_article_content.count(word)
                            if word not in self.mots_dejavus:
                                self.INDEX[word] = {}
                                self.mots_dejavus.append(word)
                            self.INDEX[word][id] = nb_occ
                            mots_dejavus_ici.append(word)
                    article.close()
                    id += 1

                elif os.path.isdir(bdd_path + '/' + element):
                    id = self.initialisation(bdd_path + '/' + element, id)
            return id  # pour continuer a numeroter les documents

        def partie_log(self, document, word):
```

À noter que nous avons aussi eu besoin de préciser des paramètres **d'encodage de caractères** pour pouvoir tester nos algorithmes sur la base de données : cela a quelques peu ralenti notre progression car il ne nous a pas été évident de trouver ces astuces.

```
        def initialisation(self, bdd_path, id):
            for element in os.listdir(bdd_path):
                if element.endswith('.txt'):
                    element_path = bdd_path + '/' + element
                    self.ID_docs[id] = element_path
                    article = open(element_path, encoding='utf-8', errors='ignore') #parametres optio
                    article_content = article.read()
                    transf_article_content = word_tokenize(article_content)
```

Enfin, nous avons décidé de **séparer notre code en deux fichiers** : **l'un** contenant les programmes de création d'index, de calculs, et d'affichage, tandis que **l'autre** est le fichier d'exécution dans lequel nous avons travaillé à rendre la recherche la plus simple et la plus claire possible pour l'utilisateur. **Il n'y a pas besoin de préciser soit-même où enregistrer les index**, mais on peut le décider en modifiant directement le fichier main.py . **Seul sera demandé le chemin de la base de données**, car c'est une information que l'on ne peut pas deviner à l'avance. Mais l'**on a pris soin de ne pas exiger de l'utilisateur qu'il indique ce chemin à chaque fois qu'il fait une recherche**, mais seulement lors de la première recherche ou lors d'un renouvellement de la base de données.

**Pour conclure**, nous souhaitons vous présenter les résultats de notre code pour deux recherches quasiment similaires : « Europe development » et « Europe and development ». Le but est de vous montrer, pas seulement que l'indexation a bien fonctionné et permet d'afficher quelque chose, mais aussi que les calculs des TF-IDFs et des similarités sont tels que les mots insignifiants (comme « le », « la », « de », « et », …) n'ont pas une grande influence dans la recherche. **Cela est une des exigences courantes du cahier des charges d'un moteur de recherche usuel, et nos algorithmes y répondent.**

```
C:\Users\李博\Documents\python\venv\Scripts\python.exe C:/Users/李博/Documents/python/in104/projet/main.py
for the first query or to renew the database, enter "r"
for further queries in the same database, enter "c":
c
enter your query :Europe development
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Interview__Paul_Bugeja__CEO_of_Malta_Tourism_Authority.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/UK_troops_sent_to_train_anti_Isis_rebels__have_arrived_in_Syria_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Take_a_peak__fun_new_places_to_stay_in_European_ski_resorts.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Antimicrobial_resistance_a__greater_threat_than_cancer_by_2050_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Theresa_May_s_wardrobe__decoding_her_brand_management.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Argentina_run_away_from_Ireland_to_reach_Rugby_World_Cup_semi_final.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Young_tech_in_a_hurry.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Gibunco__Gibraltar_s_expanding_flagship_firm.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Quality_over_quantity__attracting_industry.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_A_system_of_privilege_and_benefits___is_a_global_tax_body_needed_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Agrichemicals_and_ever_more_intensive_farming_will_not_feed_the_world.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Africa_risks_fresh_debt_crisis_as_levels_of_borrowing_rise_sharply__warns_UN.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Interview__Tim_Bristow__CEO_of_Gibtelecom.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_Europe_pays_a_heavy_price_for_Syria___it_must_act_to_end_the_crisis_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_It_terrifies_me___Britons_in_Europe_on_how_Brexit_is_going_to_affect_them.txt

Process finished with exit code 0
```

```
C:\Users\李博\Documents\python\venv\Scripts\python.exe C:/Users/李博/Documents/python/in104/projet/main.py
for the first query or to renew the database, enter "r"
for further queries in the same database, enter "c":
c
enter your query :Europe and development
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Interview__Paul_Bugeja__CEO_of_Malta_Tourism_Authority.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/UK_troops_sent_to_train_anti_Isis_rebels__have_arrived_in_Syria_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Antimicrobial_resistance_a__greater_threat_than_cancer_by_2050_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Theresa_May_s_wardrobe__decoding_her_brand_management.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Argentina_run_away_from_Ireland_to_reach_Rugby_World_Cup_semi_final.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Take_a_peak__fun_new_places_to_stay_in_European_ski_resorts.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Young_tech_in_a_hurry.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Gibunco__Gibraltar_s_expanding_flagship_firm.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Quality_over_quantity__attracting_industry.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_A_system_of_privilege_and_benefits___is_a_global_tax_body_needed_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Agrichemicals_and_ever_more_intensive_farming_will_not_feed_the_world.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Africa_risks_fresh_debt_crisis_as_levels_of_borrowing_rise_sharply__warns_UN.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/Interview__Tim_Bristow__CEO_of_Gibtelecom.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_Europe_pays_a_heavy_price_for_Syria___it_must_act_to_end_the_crisis_.txt
C:/Users/李博/Documents/python/in104/projet/archive/guardian/09/_It_terrifies_me___Britons_in_Europe_on_how_Brexit_is_going_to_affect_them.txt

Process finished with exit code 0
```

**Lien vers le répertoire où se trouvent nos codes :**

https://github.com/Agentan/IN104_Bo_LI_and_Agentan_PARTHIPAN