Master 1 Informatique

Université de Lyon 2

Projet en Programmation Python

Réalisé par :

SACKO Ismaila

MAHAMOUD ISMAN Abdoulrazack

Enseignant : Julien VELCIN

Année académique 2023-2024

Table des matières

Lien du projet github :

https://github.com/IsmalSacko/programmation-specialite-python

I. Introduction

Dans le cadre du cours de « programmation de spécialité python », nous sommes amenés à réaliser un projet qui consiste à faire une analyse comparative entre deux corpus : Reddit et Arxiv.

Lancé en 2005, Reddit est un site web communautaire américain qui couvre une grande variété de sujet allant de l'actualité à la science en passant par la technologie, la culture et les loisirs. Les utilisateurs de cette plateforme appelé « Redditeur » décortiquent ces diverses thématiques de façon humoristiques, plus sérieuses ou informatives. Ils ont aussi la possibilité de soumettre des liens, des images, des vidéos ou du texte pour animer la discussion.

Arxiv, créé en 1991, est une bibliothèque scientifique contenant plus d'un million de publications. En accès libre, Arxiv traite de diverses thématiques dans différents domaines comme l'informatique, la mathématique, la biologie, la finance, etc.

Cette plateforme permet aux chercheurs de partager librement leurs travaux de recherches avec les autres chercheurs du monde entier avant qu'ils soient publiés dans des revues académiques, ce qui favorise la collaboration et l'échange entre la communauté scientifique.

Ce projet a pour objectif de créer un programme capable de récupérer des données textuelles dans les deux plateformes (Reddit et Arxiv) et d'analyser ces données.

Pour que le programme soit à la portée des utilisateurs qui ne sont pas forcément spécialistes du domaine informatique, il est nécessaire de créer une interface visuelle qui permettrait aux utilisateurs d'interagir avec l'application, autrement dit, séparer les documents en fonction de leur source, faire un tri sur les auteurs, la date de publication, observer l'importance des termes dans les corpus.

Le déploiement de cette application passe par différentes étapes :

D'abord, la récupération de données, la création d'objets (Corpus, Auteurs, Document), la création d'un Vocabulaire. Ensuite, l'étape suivante est de développer des méthodes d'analyses Statistiques telle que des fonctions de concordance de mots dans des textes, des

méthodes de pondération (TF, TF-IDF), ainsi que des méthodes de recherche avec similarité, et d'autres méthodes qui seront détaillées dans le programme.

En bref, notre programme s'organise comme suite :

- Récupération des données sur Internet (Reddit, Arxiv)
- Création des Documents et Auteurs avec les données
- Ajout des corpus
- Nettoyage des données
- Implémentation des méthodes d'analyses (TF-IDF, calcul de similarité, etc.)
- Déploiement de l'interface visuelle

II. Description de l'environnement de travail et données

1. L'environnement de travail

En ce qui concerne l'environnement de travail, nous avons gardé le même environnement de travail que les TPs vu en cours, nous avons donc travaillé sur Anaconda pour ce qui est environnement et Spyder pour ce qui est éditeur.

2.Données identifiées et Data cleaning

Les données utilisées proviennent des sites comme Reddit, site communautaire de discussion, et Arvix, site de publication des articles scientifiques. Par le biais de leurs API, nous avons réussi à récupérer des données de discussions et d'articles faisant l'objet de notre étude.

Pour pouvoir procéder à l'extraction des occurrences de mots permettant de servir à la détection de communautés, nous avons sélectionné pour chaque article le texte de son résumé.

Vu que ces textes sont considérés comme de données brutes, il a fallu commencer par effectuer des nettoyages afin de pouvoir l'utiliser ultérieurement dans nos algorithmes.

Nous avons commencé le traitement des textes à l'aide de la librairie des expressions régulières « re » pour :

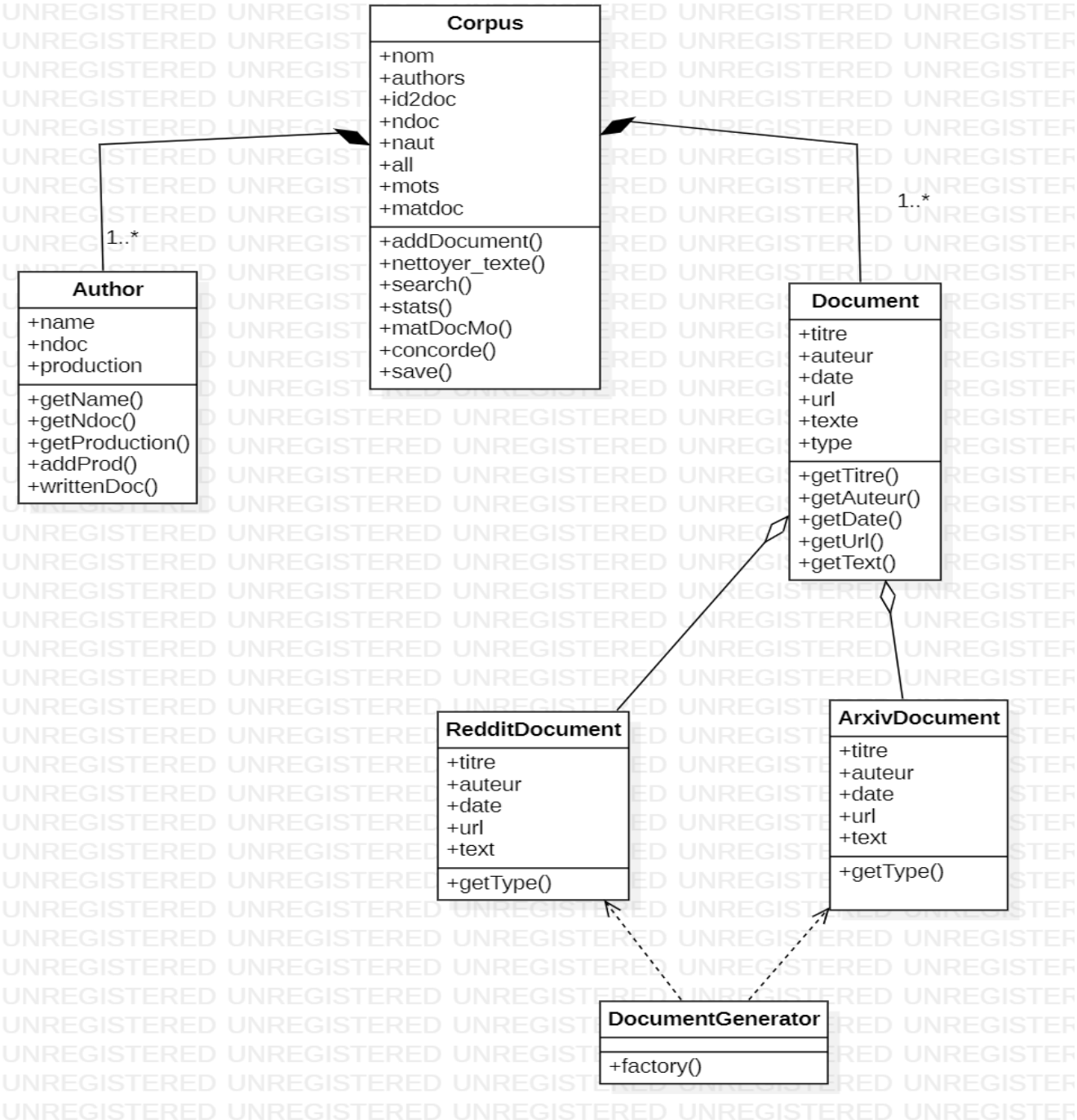
- Mettre en minuscule le texte de chaque document,
- Supprimer toutes les expressions entre crochets, accolades ou parenthèses,
- Enlever la ponctuation et de certains caractères spéciaux comme « %, *, &, ², ~ »
- Supprimer des « s »
- Enlever tous les chiffres
- Supprimer des expressions relatives aux dates ou des nième « 1st, 3rd, etc. »

Après avoir effectué ce traitement, nous avons continué le processus en s'appuyant sur des méthodes de NLP notamment de la libraire NLTK pour rendre le texte exploitable et donc prêt à être exploité. Parmi ces méthodes nous pouvons citer :

La tokenisation, qui est un processus qui a permis de transformer le texte des documents en un mot. Autrement dit, le texte de chaque document est transformé en une liste de mots qu'on appelle tokens.

StopWords, ce processus intervient après la tokenisation. Il s'agit d'une méthode qui nous a permis de supprimer tous les mots qui ne sont pas utile à la compréhension du texte (par exemple les articles, les déterminants, etc.).
La lemmatisation, cette méthode nous permet de préserver la racine des mots de tel sorte que deux mots ayant une même souche seront considérés comme un même mot (par ex : « voisine » et « voisinage » peuvent être ramenés à leurs racines « voisin »).

3. Le diagramme des classes



On peut observer à travers ce diagramme de classe le primordiale que joue la classe Corpus. Chaque document peut être considéré comme un corpus et dans un corpus il peut y avoir un ou plusieurs documents et un ou plusieurs auteurs.

4. Fonctionnement des classes

De la même manière que les TDs, La réalisation de ce projet demande l'exploitation de plusieurs ayant chacune leurs caractéristiques et utilités dans le fonctionnement du programme :

- Corpus : cette classe Rreprésente un corpus de document, contient les tous les documents et tous les auteurs d'un même sujet, et possède des méthodes d'analyse.
- Document : cette classe contient les informations sur un document (titre, auteur(s), date, url, texte).
- RedditDocument : cette classe représente un document Reddit avec ses particularités.
- ArxivDocument : cette classe représente un document arXiv avec ses particularités.
- Author : cette classe présente des informations sur un auteur (nom, nombre de documents, productions).
- DocumentGenerator : cette classe génère des documents simplement pendant le téléchargement des données via les APIs.

Le programme contient aussi deux autres fichiers python, qui ne sont pas des classes, et donc ne représentent pas une entité mais qui sont tout autant important au bon fonctionnement du programme comme : index qui permet d'initialiser le Corpus de documents avec les données sur le répertoire et lance le fonctionnement de l'interface graphique

III. Conception

1. La répartition des tâches

Le partage des tâches n'était problématique pour nous sachant qu'on a déjà collaboré pour des projets en dehors des projets académiques. Après la réflexion de chacun de son coté sur les fonctionnalités que nous souhaitons implémentés et le type d'interface que nous voulons utiliser, nous nous sommes partagé les taches ainsi : l'un s'est occupé de la partie chargement, le nettoyage des données, l'implémentation des méthodes d'analyse, la programmation des méthodes de pondération, de recherche, et l'autre le déploiement de l'interface Tkinter et donc moduler les méthodes à cette librairie. Nous avons participé tous les deux à la rédaction du rapport.

2. Programme

- Récupération de données

Dans cette première étape, nous avons procédés par la récupération des données en se connectant avec l'API à l'aide de la bibliothèque PRAW.

Pour ce faire, nous utilisons la bibliothèque Praw pour interagir avec l'API Reddit et récupérer des données à partir de des données Reddit spécifiés.
Voici comment cela fonctionne :

- Initialisation de l'API Reddit

Le programme utilise les identifiants d'application (« client id » et « client_secret ») ainsi qu'un « user_agent » pour se connecter à l'API Reddit. Ces informations sont utilisées pour instancier un objet Reddit via une variable nommée comme suit : **reddit = praw.Reddit**
Cela nous permettra d'aller récupérer les données pour les traiter et stocker ensuite.

```
reddit = praw.Reddit(client_id='u3rloCuzncRYoFhePXGaig',
                         client_secret='gbUMJKAbdAHMjAXz4M88hHAYeuNz1Q',
                           user_agent='projet_ismael_abdoul',
                           check_for_async=False)
    subr = reddit.subreddit(redditSubject)
```

❖ Récupération de données depuis un sous-reddit

Après avoir créé l'objet Reddit, le programme interagit avec un sous-reddit spécifié (dans ce cas, « Philosophie ») en utilisant la méthode « **subreddit ()** » pour obtenir des publications (« post ») depuis ce sous-reddit.

❖ Traitement des publications

Pour chaque publication obtenue (limitée à N nombre grâce à « limit=N »), des informations comme le titre, l'auteur, la date de création, l'URL et le texte associé sont extraites. Ces données sont utilisées pour créer des instances de la classe `Document` avec le type « Reddit » via « DocumentGenerator.factory ».

```
class DocumentGenerator:
    # Factory method

    @staticmethod
    def factory(type, titre, auteur, date, url, texte):
        if type == "Reddit": return RedditDocument(titre, auteur, date, url,
texte)
        if type == "Arxiv": return ArxivDocument(titre, auteur, date, url,
texte)

        assert 0, "erreur "+type
```

❖ Nettoyage des données

En ce qui concerne le nettoyage de données, nous déployons une fonction nommée « **_nettoyer() »** , qi prend un texte en entrée et effectue plusieurs opérations de nettoyage. Tout d'abord, cette fonction élimine la ponctuation, remplace les retours à la ligne et les tabulations par des espaces, puis met l'ensemble du texte en minuscules. Ensuite, elle divise le texte en mots, enlève les mots qui ne sont pas alphabétiques ou qui ont une longueur inférieure à deux caractères. Enfin, elle applique une racinisation (**stemming**) aux mots restants, ce qui réduit les mots à leur forme racine en anglais avant de les renvoyer.
Le code ci-dessus illustre cette fonction :

```
 # Nettoie le texte donné en retirant la ponctuation, les espaces, en mettant
en minuscules, et en appliquant une racinisation (stemming) aux mots.
    def _nettoyer(self, text):
        ponctuation= string.punctuation
        compiler=re.compile("[%s]" % re.escape(ponctuation))
        text= compiler.sub(" ", text)
        text= text.replace("\n", " ").replace("\t", " ")
        #st= stopwords.words("english")
        texte= text.lower()
        token= texte.split()
        token= list(set([tk for tk in token if tk.isalpha() and len(tk)>1]))
        #token= [tk for tk in token if tk not in st]
        #Pour raciniser les mots
        stemmer=SnowballStemmer("english")
        token=[stemmer.stem(mot) for mot in token]
  return token
```

❖ Stockage des données

Les documents ainsi créés sont stockés dans des structures de données (« id2doc » et « id2aut », où « id2doc ») contient les documents eux-mêmes et « id2aut » conserve les auteurs et leurs productions.
Ces données sont sauvegardées dans des fichiers binaires via « **pickle.dump** ».

```
with open("./file/doc.pkl", "wb") as t:
        pickle.dump(id2doc, t)
    t.close()
    with open("./file/aut.pkl", "wb") as f:
        pickle.dump(id2aut, f)
    f.close()
    return id2doc, id2aut
```

Cette partie sert à préparer l'application à l'affichage de données dans une interface implémentée par TKINTER en Python.

- Interface visuelle avec TKINTER

Cette partie consiste à implémenter une interface utilisateur via la bibliothèque TKINTER pour pouvoir interagir avec le système de recherche et d'analyse de données créé précédemment depuis l'API Reddit.

- Chargement des questions, des réponses et des données

Les questions, les réponses, ainsi que les données précédemment enregistrées (id2doc et id2aut) sont chargées à partir de fichiers qui seront utilisés par le programme par suite.

Cette technique permet à l'application de s'exécuter plus rapidement, en ce sens qu'elle n'aura plus besoin d'aller récupérer les informations depuis l'API, mais plutôt dans les fichiers stockés dans le système pour cette fin.

- Gestion des événements et des interactions

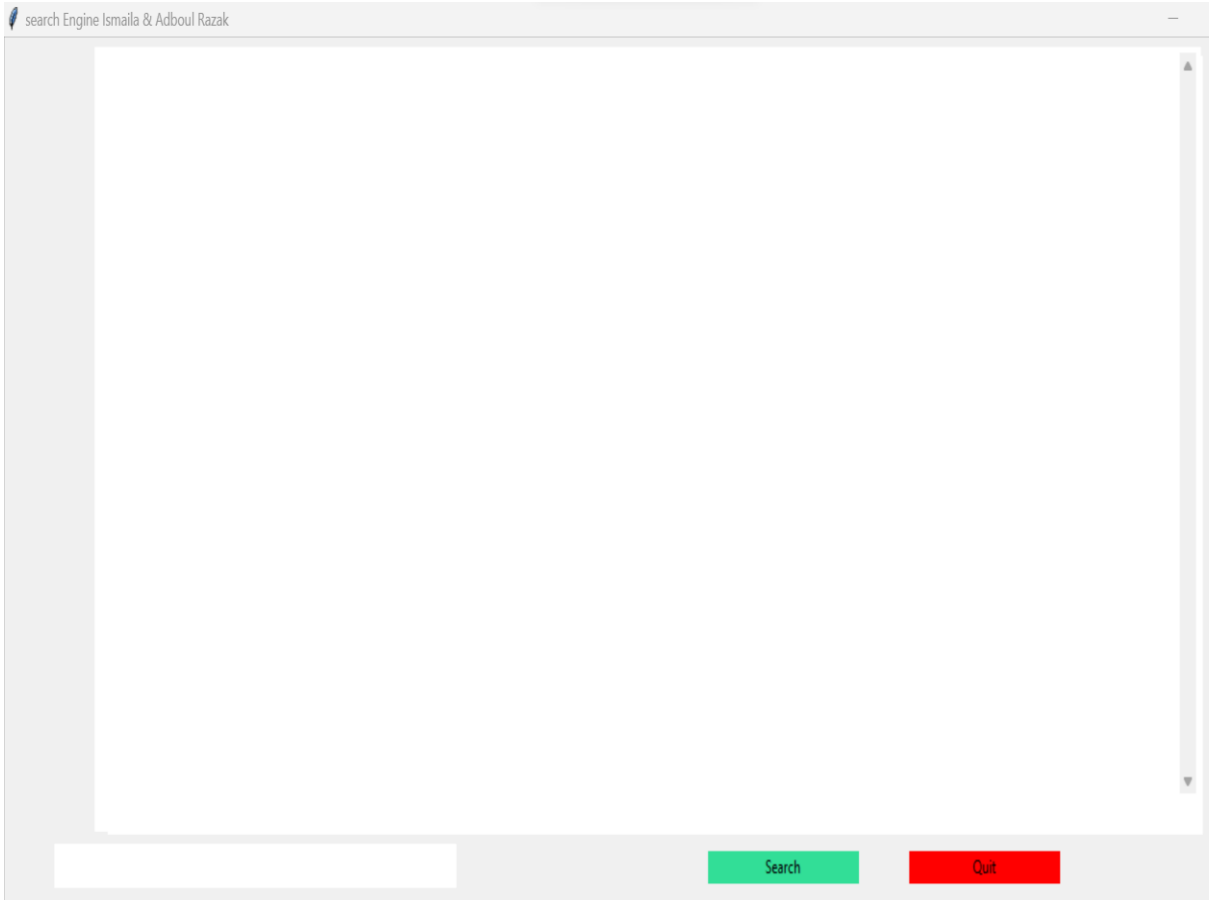
Des fonctions sont définies pour gérer les événements utilisateur, comme la saisie d'une recherche (**saisie ()**), la demande de fermeture de l'application (**quitter ()**), et la liaison de ces fonctions aux boutons correspondants.

❖ Affichage de la fenêtre

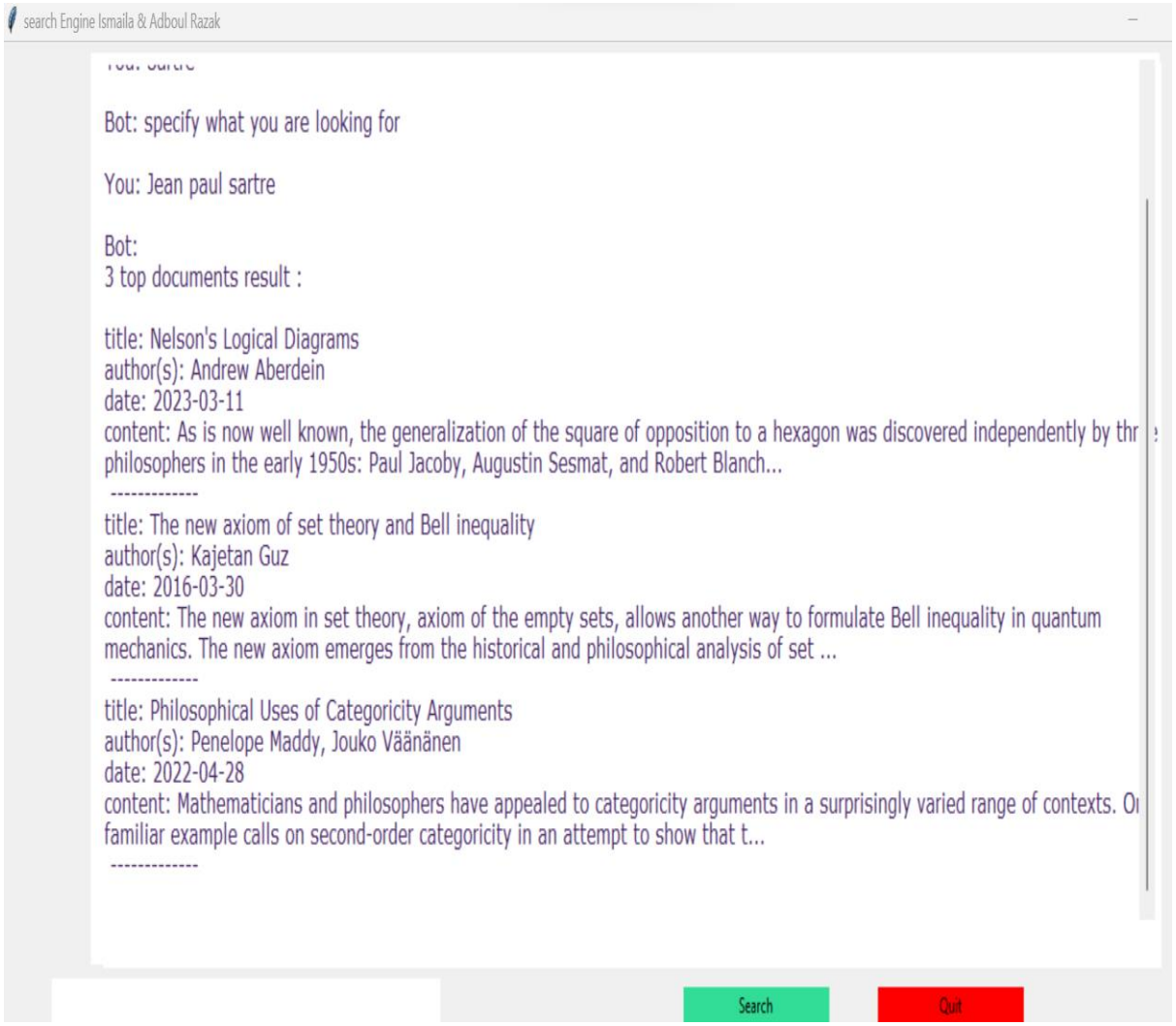
Cette partie d'affichage gère l'interface utilisateur (**fenetre.mainloop()**) et est lancée pour afficher et gérer l'interface pour permettre à l'utilisateur d'effectuer une recherche sur un auteur ou le titre d'un ouvrage de celui-ci. La fonction (**fenetre.mainloop()**), qui est une boucle, rend l'interface interactive en répondant aux événements déclenchés par l'utilisateur.

En fin de fin compte, cette partie en place une interface où l'utilisateur peut saisir des requêtes de recherche. Ces requêtes sont ensuite traitées par le système de recherche en utilisant les documents et les auteurs récupérés précédemment depuis Reddit et Arxiv. Les résultats de la recherche sont affichés dans la zone de chat (**ChatLog**), offrant une interaction utilisateur simple pour l'exploration et l'accès aux informations stockées.

Voici un exemple de l'interface utilisateur à l'ouverture de l'application :



Un exemple de l'interface utilisateur après avoir lancer la recherche :



IV. Validation du programme

1. Les tests

Pour la compréhension des différents tests réalisés, suivons cette approche :

- ❖ Test de fonctionnement de l'interface utilisateur

Pour procéder aux différents tests, nous avons pu :

- Vérifier si l'interface utilisateur se lance sans erreur
- Vérifier que la fenêtre principale de l'application s'affiche correctement avec les champs prévus pour l'entrée utilisateur et l'affichage des résultats.
- Vérifier si les boutons de recherche et de sortie (**quit**) fonctionnent correctement et s'ils arrêtent l'application de manière appropriée.

- ❖ Test de saisie utilisateur

Nous avons également pensé à réaliser un test sur la saisie des utilisateurs.
Pour cela, nous avons pu :

- Vérifier si l'entrée utilisateur est correctement récupérée par le champ de texte.
- Nous avons vérifié que la fonction « **saisie() »** fonctionne correctement et que les données entrées sont correctement transmises pour traitement.

- ❖ Test de traitement des données

En ce qui concerne le test au niveau de traitement des données, nous avons pu :

- Vérifier si les données entrées par l'utilisateur sont correctement analysées et traitées par l'application.
- Vérifier si les réponses du chatbot sont correctement générées en fonction des saisies utilisateur.

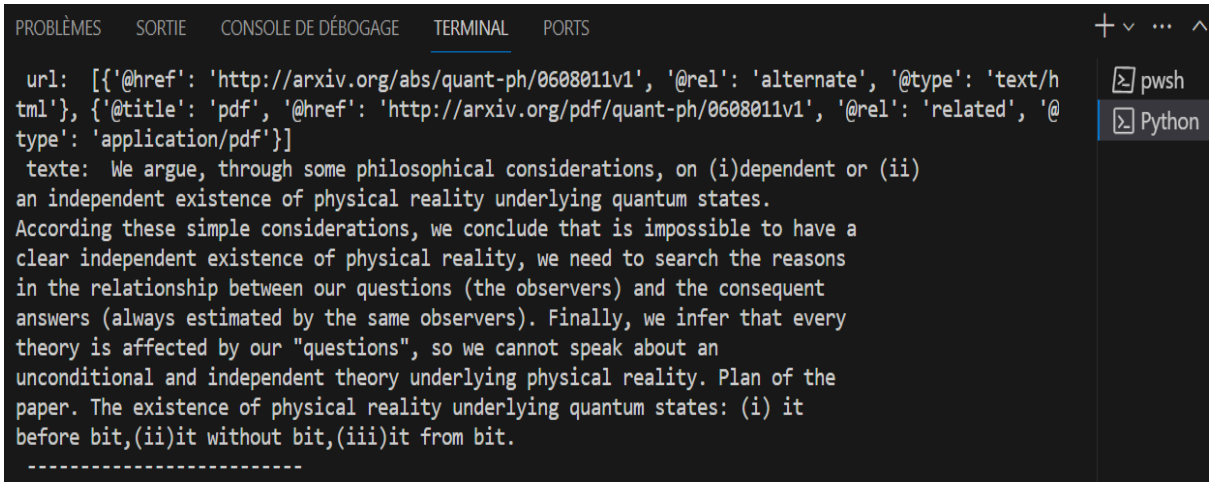
❖ Test de fonctionnalité de recherche

Nous nous sommes également intéressés sur la fonctionnalité de recherche de l'interafce.
Dans ce sens, nous avons pu :

- Tester la fonction de recherche en entrant des termes existants et des termes inexistants pour vérifier si l'application renvoie des résultats pertinents ou des réponses appropriées.

❖ Test d'intégration avec les données enregistrées

Nous avons ici vérifié si les données chargées depuis les fichiers **« questions.json »**, **« answers.json »**, **« doc.pkl »**, et **« aut.pkl »** sont correctement intégrées et utilisées par l'application.
Nous avons aussi vérifié si les données des documents et des auteurs sont correctement récupérées et affichées dans la console avant même de les afficher dans l'interface utilisateur.



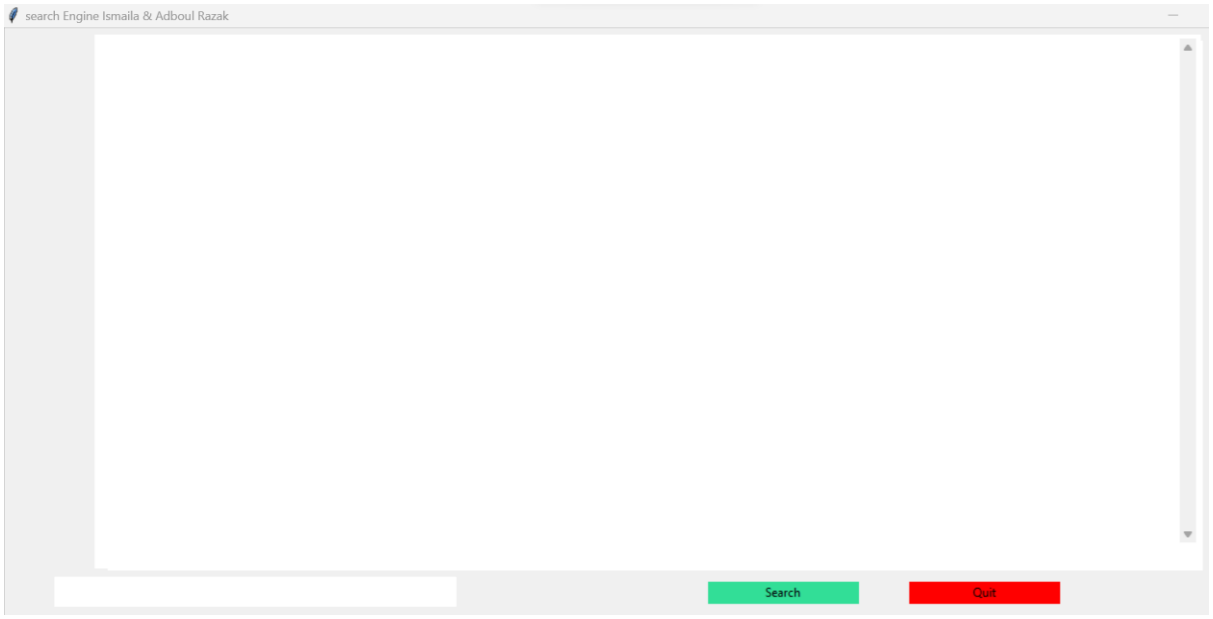
❖ Test d'affichage et fermeture de l'application

Concernant l'affichage et la fermeture de l'application, nous avons pu :

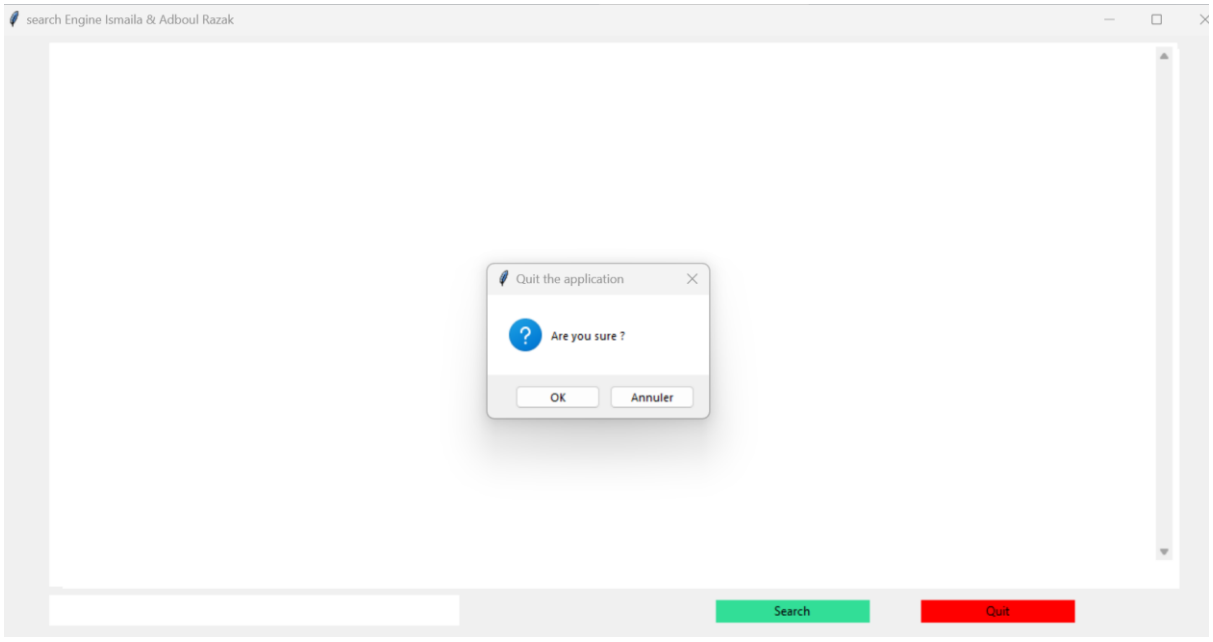
- Vérifier si les résultats générés par le chatbot sont correctement affichés dans l'interface utilisateur.
- Nous nous sommes assurés que les résultats des documents (s'ils existent) sont affichés correctement avec toutes les informations nécessaires.

- Tester la fonctionnalité de fermeture pour nous assurer que l'application se ferme correctement et proprement lorsqu'elle est quittée à l'aide du bouton « **Quit** ».

La fenêtre avant de cliquer sur le bouton « **Quit** » :



La fenêtre après avoir cliqué sur le bouton Quit :



V. Maintenance

Une des améliorations que l'on pourrait proposer concerne la lisibilité et l'optimisation du programme surtout la partie qui traite l'interface visuelle Tkinter. On peut constater que notre interface visuelle pourrait être améliorée davantage en matière de l'ergonomie en apportant plus des couleurs et des écritures plus vives et personnalisées.