1. **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.

L’objectif principal de ce projet est d’observer l’importance d’un mot ou plusieurs mots dans un corpus à l’autre.

Reddit est un site web communautaire américain d'actualités sociales mais aussi d’un forum où le contenu est classé par sous-catégorie et est accessible aux utilisateurs.

Arxiv, quant à lui, est une bibliothèque scientifique contenant plus d’un million de publications. En accès libre, Arxiv aborde de diverses thématiques dans différents domaines comme l’informatique ou la mathématique.

1. **Description de l’environnement de travail et données**

## 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.

## 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. »

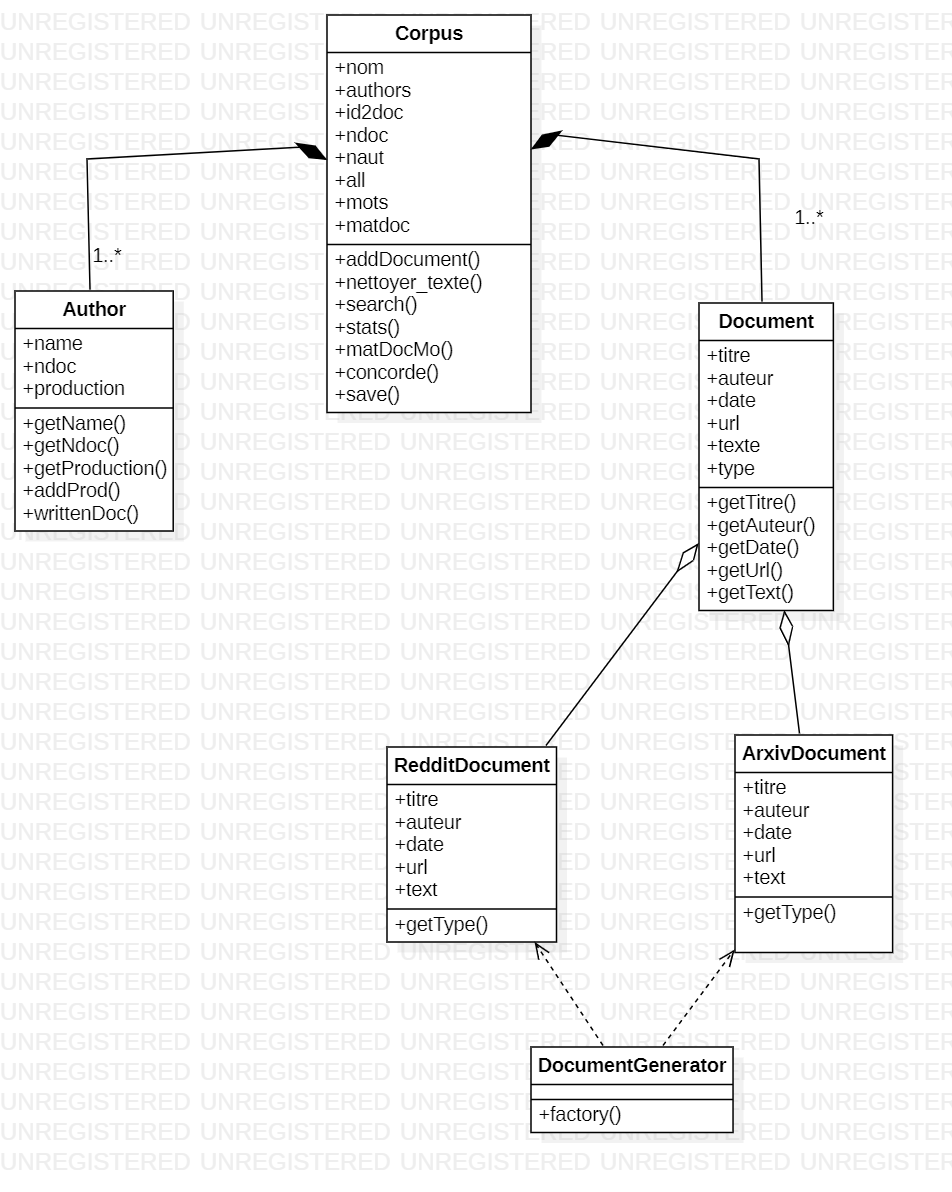
Après avoir effectué ce traitement, nous avons continué le processus en s’appuyant sur des méthodes de NLP notamment de la libraire NLTK pour rendre le texte exploitable et donc prêt à être exploité. Parmi ces méthodes nous pouvons citer :

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 »).

1. 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.

1. 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

1. **Conception**
2. 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.

1. 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 reddits 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,ce qui nous permettra d’aller récupérer les données pour les triter et stocker ensuite.

* **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.

reddit = praw.Reddit(client\_id='u3rloCuzncRYoFhePXGaig',

                         client\_secret='gbUMJKAbdAHMjAXz4M88hHAYeuNz1Q',

                           user\_agent='projet\_ismael\_abdoul',

                           check\_for\_async=False)

    subr = reddit.subreddit(redditSubject)

* **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`.

* **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`.

Cette partie prépare l’application à l’affichage de données dans une interface implémentée par TKINTER de Pyhon.

with open("C:/Users/ismal/projets/moteur-recherche/file/doc.pkl", "wb") as t:

        pickle.dump(id2doc, t)

    t.close()

    with open("C:/Users/ismal/projets/moteur-recherche/file/aut.pkl", "wb") as f:

        pickle.dump(id2aut, f)

    f.close()

    return id2doc, id2aut

* **Interface utilisateur avec TKINTER**

Cette parie consiste à implémenter une interface utilisateur via la bibliothèque TKINTER pour interagir avec le système de recherche ou 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 résumé, 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 :

Une image contenant texte, affichage, capture d’écran, Rectangle

Description générée automatiquement

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

Une image contenant texte, logiciel, Police, Page web

Description générée automatiquement

---> ici on va afficher une capture d’écran de la partie du code qui traite la connexion avec l’api

* Prétraitement des textes