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. »

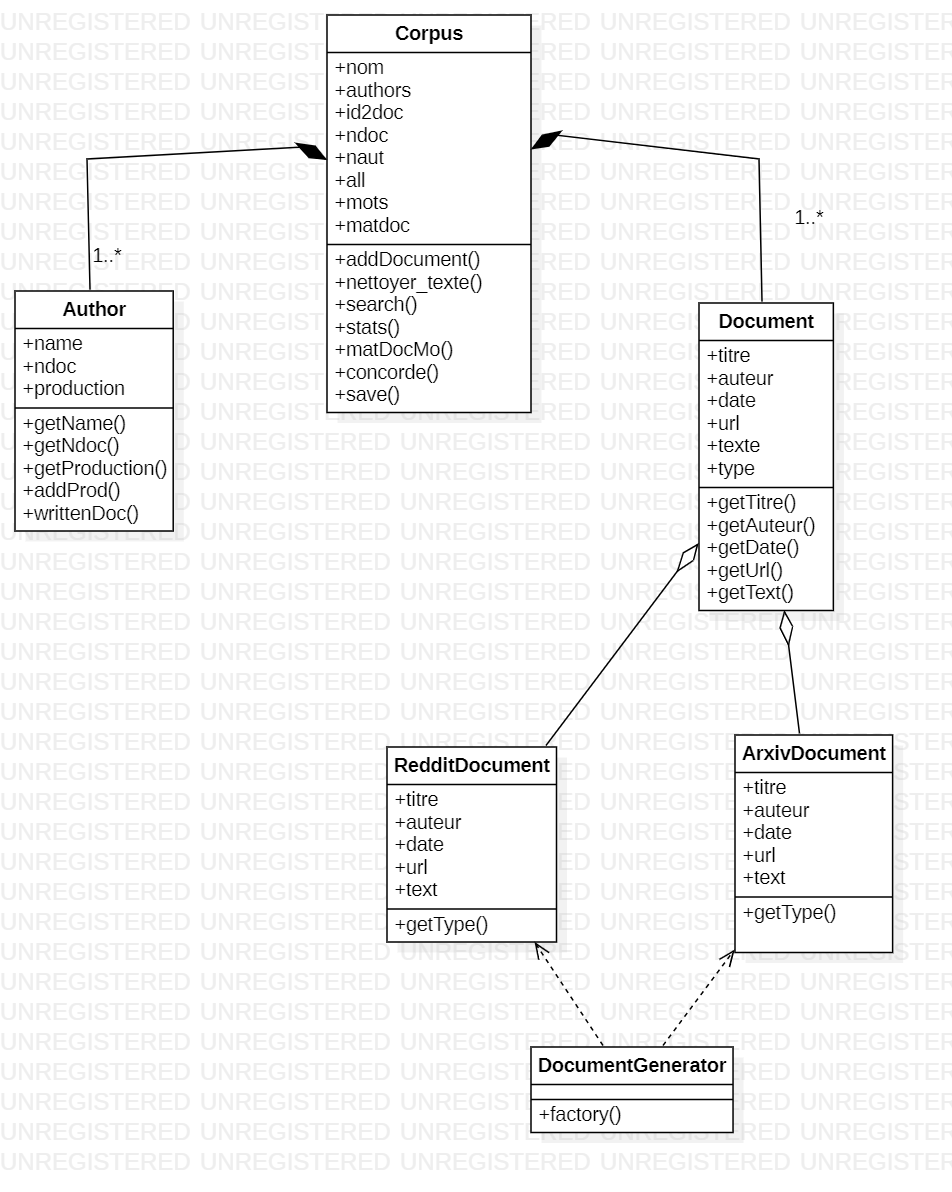
Après avoir effectué ce traitement, nous avons continué le processus en s’appuyant sur des méthodes de NLP notamment de la libraire NLTK pour rendre le texte exploitable et donc prêt à être exploité. Parmi ces méthodes nous pouvons citer :

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

La première étape à faire est la récupération des données en se connectant avec l’API.

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

* Prétraitement des textes