**Développement d'un Moteur de Recherche d'Information Maison**

Table des matières

[Présentation du projet 1](#_Toc155778512)

[Spécifications 1](#_Toc155778513)

[Analyse 2](#_Toc155778514)

[Conception 2](#_Toc155778515)

[Gestion du projet 2](#_Toc155778516)

[Algorithmes 3](#_Toc155778517)

[Problèmes rencontrés 5](#_Toc155778518)

[Exemple d’utilisation 6](#_Toc155778519)

[Validation 6](#_Toc155778520)

[Maintenance 6](#_Toc155778521)

# Présentation du projet

Ce projet vise à créer une solution personnalisée de recherche d'information, en mettant l'accent sur la mise en œuvre de nos propres algorithmes développer en ***Python*** plutôt que de recourir à des bibliothèques existantes telles que ***scikit-learn*** ou ***nltk***. L'objectif principal est d'approfondir notre compréhension des mécanismes fondamentaux qui sous-tendent ces outils "prêts à l'emploi" tout en travaillant la programmation orienté objet et notamment son utilisation avec ***Python***.

Vous pouvez retrouver ce projet à l’adresse suivante : <https://github.com/AlysonTT/Python>

# Spécifications

L'interface graphique offre la possibilité d'explorer diverses actions, Il est possible de suivre l'évolution temporelle d'un mot, analyser la fréquence d'un terme, effectuer une recherche par mots-clés, auteurs ou date spécifique, ainsi que visualiser l'intégralité du corpus et des matrices TFxIDF pour chaque mot du vocabulaire du corpus. De plus, la comparaison de deux documents est également incluse dans les fonctionnalités.

L'évolution temporelle d'un mot est mise en évidence à travers une frise temporelle, offrant une représentation visuelle claire des variations de la fréquence de ce mot au fil du temps.

Les utilisateurs peuvent accéder à la fréquence de tous les mots présents dans le corpus, ainsi qu'à la mesure de l'importance d'un terme, calculée selon la méthode TFxIDF, grâce à un bouton dédié sur notre interface graphique.

La fonctionnalité de recherche avancée permet aux utilisateurs de spécifier des critères tels que des mots-clés, des auteurs, et/ou une date spécifique, simplifiant ainsi l'exploration ciblée du corpus.

Une option pratique est disponible pour afficher l'ensemble du corpus, offrant une vue d'ensemble facilitant l'analyse globale des documents présents.

Une fonctionnalité intégrée permet de comparer deux documents et d'indiquer la présence éventuelle de mots communs entre eux.

Ces caractéristiques combinées font de notre interface graphique un outil complet et convivial, permettant aux utilisateurs d'explorer, d'analyser et de comparer efficacement le contenu de leur corpus de textes.

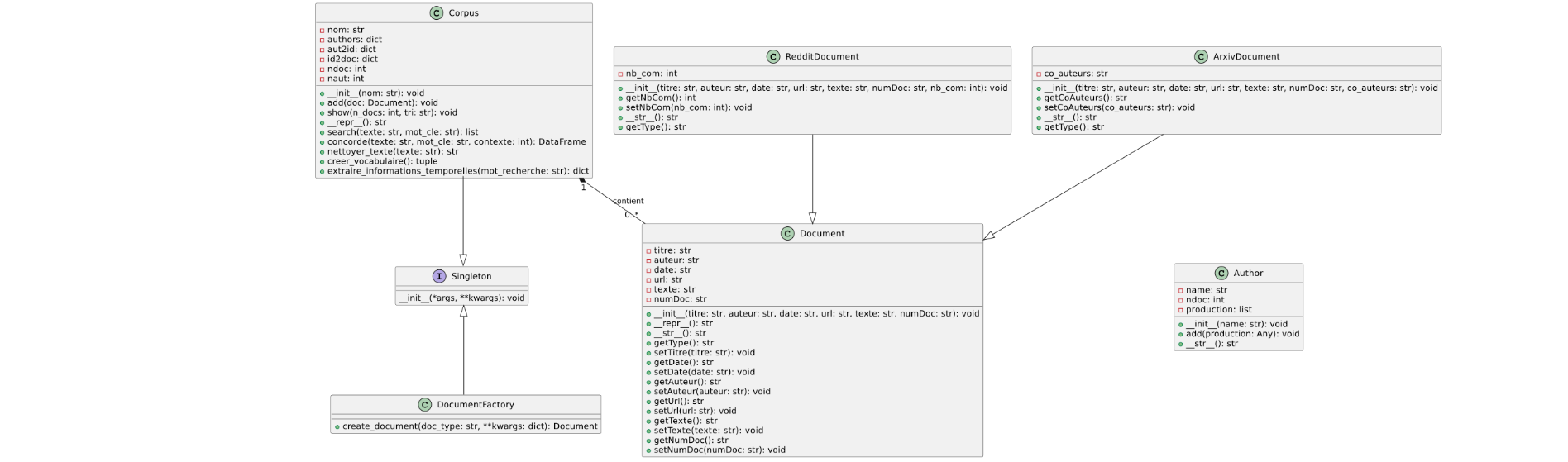
# Analyse

Notre environnement de travail implémente sur Anaconda Navigator s'est articulé autour de Visual Studio Code, GitHub, et la bibliothèque ***Tkinter*** pour l'interface graphique. Visual Studio Code a été notre IDE principal, offrant une expérience de développement fluide. GitHub a facilité la collaboration au sein de notre binôme en permettant le suivi des versions et la gestion du code source. Pour l'interface utilisateur, nous avons choisi ***Tkinter***, une bibliothèque Python, pour sa simplicité d'utilisation et son intégration harmonieuse avec notre projet.

En ce qui concerne les données identifiées dans les spécifications, nous avons utilisé les API de Reddit et arXiv. Pour interagir avec l'API de Reddit, nous avons employé la bibliothèque ***PRAW***, tandis que pour l'API arXiv, nous avons utilisé la bibliothèque ***urllib***.

Pour Reddit, nous avons récupéré tous les articles disponibles, cependant, une limite maximale d'articles a été définie alors que pour arXiv, une requête a été formulée pour extraire des articles contenant les termes "clustering" et "Dirichlet". Ces données extraites serviront de base pour notre analyse ultérieure.

Voici le diagramme de classes final de ce projet :



# Conception

## Gestion du projet

Au début du projet, Alyson débutait les travaux dirigés, car elle avait des cours avant Maude. Ensuite, Maude prenait le relais pendant son créneau horaire de cours. Si le travail dirigé n'était pas terminé, nous le poursuivions ensemble lorsque nous n'avions pas de cours en commun, profitant parfois de ces moments pour travailler en pair programming. Tout au long, l'objectif était d'être à jour dans la réalisation des travaux dirigés.

D'un commun accord, il a été décidé d'utiliser en partie la correction de G. Poux-Médard, 2021-2022 pour les TD3/4. Cela nous a été très utile pour corriger notre première version du code et poursuivre plus sereinement notre projet.

Pour le TD9/10, il y a eu une répartition plus précise des tâches. Au vu du sujet, qui est plus libre, il était préférable de faire une partie chacune et de réaliser une mise en commun plus tard, en utilisant la méthode de « diviser pour mieux régner ». Ainsi, notre attention a été focalisée sur des problèmes plus petits, offrant une meilleure gestion. Nous sommes tout de même partis d'un code identique et fonctionnel, soit l'interface et le moteur de recherche, puis sur des fichiers réalisant des tâches différentes.

Concernant la partie graphique, Alyson a pris en charge la création de l'ensemble de l'interface, incluant l'affichage des résultats ainsi que la section dédiée à la comparaison de deux documents. Maude, de son côté, s'est focalisée sur la mise en place d'options permettant à l'utilisateur de sélectionner des critères tels que les sources, les auteurs, et une date spécifique. Elle a également travaillé sur la frise temporelle en intégrant les éléments graphiques nécessaires pour faciliter ces différentes actions.

Alyson a ensuite mis les fichiers en commun en les adaptant. Pendant ce temps, Maude a pu réaliser le diagramme de classe et commencer la rédaction de ce rapport qui a été réalisé à quatre mains.

Tout au long, chacune a testé et a eu un regard sur le travail de l'autre pour avancer ensemble.

Pour finir, comme vous avez pu le voir, les parties du projet ont été réalisées intégralement, conjointement.

## Algorithmes

**Fonction selection\_unique(index, variables) :**

Description : permet d'avoir seulement un type de source sélectionné en décochant les autres.

Paramètres : index - l'index du type de source à conserver sélectionné.

Variables - la liste des variables associées aux types de source.

Algorithme : parcours les variables associées aux types de source et les décoche, sauf celle à l'index spécifié.

**Fonction checkbutton\_selection(source, varialbles) :**

Description : retourne les types de source sélectionnés sous forme de chaîne, séparés par des virgules.

Paramètres : source - la liste des éléments à afficher.

variables - la liste des variables associées aux cases à cocher.

Algorithme : utilise les variables des types de source pour identifier ceux qui sont sélectionnés.

**Fonction auteurs\_selection(listebox\_auteurs) :**

Description : retourne les auteurs sélectionnés sous forme de chaîne, séparés par des virgules.

Paramètres : listebox\_auteurs - la listebox contenant les auteurs

Algorithme : utilise la sélection actuelle dans la listebox des auteurs pour obtenir les auteurs sélectionnés.

**Fonction deselectionner\_tous\_les\_auteurs(listebox\_auteurs, checkbutton\_deselection) :**

Description : désélectionne tous les éléments de la listebox des auteurs et décoche le checkbutton associé.

Paramètres : listebox\_auteurs - la listebox contenant les auteurs.

checkbutton\_deselection - le checkbutton associé à la désélection.

Algorithme : utilise les méthodes appropriées pour désélectionner tous les éléments et mettre à jour l'état du checkbutton.

**Fonction est\_date\_valide(annee, mois, jour) :**

Description : vérifie si une date est valide.

Paramètres : annee, mois, jour - composants de la date à valider.

Algorithme : Vérifie si l'année est entre 1900 et 2024, le mois entre 1 et 12, et le jour en fonction du mois (prend en compte les années bissextiles).

**Fonction effectuer\_recherche(corpus, zone\_texte, mots\_clefs, date, source, variables, listebox\_auteurs, vars\_afficher, vars\_comparer):**

Description : effectue une recherche dans le corpus en fonction des mots-clés, de la date, du type de source, et des auteurs spécifiés par l'utilisateur. Affiche les résultats avec un score de similarité et la possibilité de sélectionner ou comparer des documents.

Paramètres : corpus : l'objet corpus contenant les documents.

zone\_texte : le widget de zone de texte dans lequel afficher les détails.

mots\_clefs : mots clés rentrez par l’utilisateur

date : date entrez par l’utilisateur

source, variables : paramètres utilisé lors de l’appel de **checkbutton\_selection**

listebox\_auteurs : paramètre utilisé lors de l’appel de **auteurs\_selection**

vars\_afficher : un dicionnaire de variables associées au bouton "Afficher" pour chaque document.

vars\_comparer : un dictionnaire de variables associées au bouton "Comparer" pour chaque document

Algorithme :

Obtient les mots-clés, la date, le type de source, et les auteurs spécifiés par l'utilisateur. Mise en évidence des mots-clés, des auteurs, et de la date dans le texte des documents.

Utilise la méthode creer\_vocabulaire pour obtenir le vocabulaire.

Transforme les mots-clés en un vecteur sur le vocabulaire construit.

Calcule la similarité entre le vecteur de requête et tous les documents du corpus.

Filtrage des documents en fonction de la présence des mots-clés, du type de source, de la date et des auteurs.

Affiche les résultats triés par score de similarité.

Met en évidence les mots-clés, les auteurs et la date dans le texte des documents.

Permet à l'utilisateur d'afficher ou de comparer des documents sélectionnés. À l'aide des fonctions afficher\_details\_selectionnes et comparer\_documents.

Les résultats sont affichés dans la zone de texte avec des boutons "Afficher" et "Comparer" pour chaque document.

**Fonction afficher\_details\_selectionnes(corpus, zone\_texte, numDoc, vars\_afficher) :**

Description : affiche les détails des documents sélectionnés dans la zone de texte. Elle est généralement utilisée en conjonction avec la fonction de recherche pour afficher en détail les résultats sélectionnés par l'utilisateur.

Paramètres :

corpus : l'objet corpus contenant les documents.

zone\_texte : le widget de zone de texte dans lequel afficher les détails.

numDoc : le numéro du document à afficher.

vars\_afficher : un dictionnaire de variables associées aux boutons "Afficher" pour chaque document.

Algorithme :

Récupère le document associé au numDoc dans l'objet corpus.

Active la modification de la zone de texte.

Efface le contenu précédent de la zone de texte.

Récupère les documents sélectionnés en fonction des variables associées aux boutons "Afficher".

Affiche les détails des documents sélectionnés : le titre, l'auteur, la date, l'URL et le contenu du document.

**Fonction afficher\_corpus(corpus, zone\_texte, checkbutton, vars\_afficher, vars\_comparer):**

Description : Affiche tous les documents du corpus dans la zone de texte, avec des options pour afficher ou comparer certains documents.

Paramètres : corpus : l'objet corpus contenant les documents.

zone\_texte : le widget de zone de texte dans lequel afficher les détails.

checkbutton : checkbutton associe au type de source

vars\_afficher : un dicionnaire de variables associées au bouton "Afficher" pour chaque document.

vars\_comparer : un dictionnaire de variables associées au bouton "Comparer" pour chaque document

Algorithme :

Efface le contenu précédent du widget de texte et désélectionne le check du type s'il est activé.

Utilise une boucle pour parcourir tous les documents dans le corpus.

Crée des boutons "Afficher" et "Comparer" pour chaque document, associant les fonctions (afficher\_details\_selectionnes et comparer\_documents).

Ajoute les boutons créés à la zone de texte.

Affiche les informations de chaque document, y compris le titre, les auteurs, la date, le type, et le contenu.

Ajoute des lignes de séparation pour une meilleure lisibilité.

Met à jour les dictionnaires vars\_afficher et vars\_comparer avec les nouvelles variables associées aux boutons pour chaque document.

Désactive la modification de la zone de texte.

## Problèmes rencontrés

**????**

Tout d'abord l'un de nous problème à été pour l'implémentation des design pattern, TD ???,

En effet, ca até un sujet nouveau pour nous et assez diffile à implementer au départ, avec certaine logique assser asbrait par momentt pour réussir à réelemnt bien l'implémenter,

Pour ce qui est du design patern singleton, nous avons décider de ne pas l'utilser même si c'était très intéressaint à étudiant. Car si on utilise le singleton, sur les documents nous perdons la focntionnalité permettant de commaper de document, Il aurrait été possible de oder le singleton sur le corpus entier afin de n,en charger qu'un seul à la fois mais ca pourait etre intéressenat de

Ensuite, ca a tété de définir les limites de notres prrojet, jusqu'ou il faut aller pour offrir une expérience utilisateur réussite, A partir, de quel moment peut se dire que notre projet peut dans la mesure du possible etre focntionnel,

Pour le finir, le dernier problème à résider dans la création du moteur de recherche cela à été un peu long et nous à nécessiter pas mal d'esssaie,

Un autre, problème réside dans la mesure TfxIDF, nous voulions commencer proposé aller plus loin que juste avec TF mais il à été très compliqué de l'utilisé,

De base, on voulait l'utiliser pour la mesure TfxIDF pour l'analyse comparative de deux document pour chaque mot en commun, Mais le soucis les documents ne sont pas adapter à ce genre de mesure ainsi pour comparer nous somme rester sur la fréquence des mot qui est assez intéressante au vu de la forme des document, Ensuite, nous avons décider d'afficher la mesuure TfxIDf pour chaque mot du corpus en prenons le corpus comme un seul document, Mais cette, mesure prenant en paramètre??? les valeurs obtenue sembler aberrante, Nous avons aussi essayer de faire la moyen de TdxIdf pour chaque mot du corpus mais la encore les résultat nous sembler assez fosse,

Ainsi nous avons décider d'afficher la matrice Tdxidf afin de ne pas fausser les résultat et perdre de l'information, ainsi l'utilisateur pour l'utiliser comme il lui convient, nous avons aussi décidé d’ajouter un grafique, pour que se soit plus parlant,

il aurait peut etre fallu, traiter le tdfx de manière différentes,

Notre première approche aurait été très pertiente dans le cas d'un corpus pourtant sur un thême (ex ; le jardinnage, le sport,,,) permettant d'avoir plus de chances d'avoir des score Tdxidf élé et de pouvoit en faire ressortir les mots cles les plus important du thême abordé,

## Exemple d’utilisation

**?donnez un exemple comment´e d’utilisation du programme du d´ebut `a la fin. ?**

Cela peut être utile si nous avons un thème

travailler sur tout les document d'un auteur

visulaer l'éolution d'une langue temporerle d'une langue

reconnatre du plagiat

# Validation

Chaque fonctionnalité implémentée, telle que la recherche spécifique ou la frise temporelle, a été testée à l'aide de fichiers dédiés. Cela nous a permis de développer un code fonctionnel sur une branche distincte, que nous avons ensuite fusionnée avec la branche principale. Cette approche nous a assuré d'avoir constamment un projet opérationnel à chaque étape du développement.

# Maintenance

**Evolution possible ??**

Il aurait été possible de coder le singleton sur le corpus entier afin de n,en charger qu'un seul à la fois mais ça pourrait être intéressent de ????

* faire une recherche d'un mots pour le trouver dans mesure corpus
* implementer le chargement de plusieurs corpus
* demander à l'utitisateur d'importer ses propre document et de travailler directement avec eux en désact ==> pas ouf peut etre
* source facile à rajouter
* faciler de charger un document