Université Lumière Lyon 2

M1 Informatique

https://github.com/Naghan1132/TextMiningPython

Rapport

Text Mining

Élèves :
Nathan GRIMAULT
Léo GONDOUIN

Enseignant :
Julien Velcin

10 janvier 2023

Table des matières

1 Introduction

Dans le cadre de l'unité d'enseignement Programmation de spécialité, nous nous sommes intéressés à la fouille de texte, à la création de corpus et à l'analyse de données textuelles (ADT) de ceux-ci.

Le Text Mining (ou fouille de texte), est une technique de traitement des données qui vise à extraire de l'information et des connaissances à partir de textes non structurés. Cette technique est utilisée dans de nombreux domaines, tels que la recherche, le marketing, la finance et les médias, pour découvrir des patterns et des tendances cachées dans les données textuelles et pour prendre des décisions éclairées. Elle utilise des méthodes de traitement du langage naturel, de l'analyse sémantique et de l'apprentissage automatique pour analyser et traiter les données textuelles. Elle permet de découvrir des informations cachées dans les textes, telles que les opinions, les sentiments et les préférences des auteurs, ainsi que de déterminer la structure et le sens des textes.

Cette technique est souvent exploitée en conjonction avec d'autres techniques d'analyse de données, telles que l'analyse de réseaux et l'analyse de séries temporelles, pour obtenir une vue complète et approfondie des données. Elle peut aussi servir pour de nombreuses applications, telles que l'analyse de sentiments, la classification de documents, l'extraction de résumés, la génération de texte, etc... C'est une discipline en constante évolution et de nouvelles techniques et outils y sont développés régulièrement.

Figure 1

2 Spécifications

Pour ce projet l'objectif était de créer un outils d'analyse de données textuelles, capable d'aller chercher des données textuelles sur le Web, de permettre une analyse ces données et d'être facilement utilisable et compris par des utilisateurs qui ne font pas partie du domaine informatique. Une solution intuitive et facile d'utilisation serait de réaliser une interface visuelle qui va permettre aux utilisateurs de directement séparer les documents selon leur source, trier les auteurs, la date de publication, voir l'importance des mots dans les corpus. Le coeur du projet devait inclure des méthodes d'analyse de texte en passant par toutes les étapes, c'est à dire dans un premier temps la recherche et la préparation de données, la création d'objets (Document, Auteurs, Corpus), la création d'un vocabulaire. Dans un second temps l'objectif était de développer des méthodes d'analyses statistiques comme des fonctions de concordance de mots dans des textes, des méthodes de pondération (TF, TF-IDF), des méthodes de recherche avec similarité et plus encore.

En résumé notre programme devait donc :
- Rechercher et récupérer des données sur Internet
- Créer des Documents/Auteurs avec les données
- Les ajouter à des Corpus
- Nettoyer les données
- Utiliser des méthodes d'analyses complexes tels que le calcul des TF-IDF, la similarité cosinus etc...
- Permettre à l'utilisateur d'analyser et de manipuler facilement toutes ces données, avec une interface graphique

3 Analyse

3.1 Environnement

Étant habitué à l'univers de l'entreprise JetBrains, Le développement de l'outil s'est fait sur PyCharm, un IDE simple d'utilisation, intuitive, possédant de l'auto-complétion intelligente, une grande variété de plugins et bien plus encore.

Nous avons choisis de travailler pour avec la librairie de ntlk (couplée avec la librairie re) pour nettoyer les textes des documents, cette librairie satisfaisant tous nos besoins et ayant déjà était utilisé pour une autre unité d'enseignement il était naturel pour nous de l'utiliser pour notre projet. L'interface quand a elle à été réalisée avec la librairie Dash que l'on a utilisé durant les séances de TPs. A noter que pour utiliser l'interface **l'utilisateur doit largement privilégier Google Chrome à Mozilla Firefox**, car nous avons décelé des bugs pour une utilisation sur Firefox, notamment des bugs d'affichage. Nous aurions pût utiliser des fonctions existantes pour le calcul de TFxIDF etc... issue de librairies type 'sklearn', mais nous avons préféré le faire nous même d'une part pour apprendre à le faire de nous même et d'autre part pour être plus libre dans notre code. On peut aussi citer quelques autres librairies utilisées comme pandas, numpy etc...

3.2 Données Utilisées

Pour un projet d'analyse des données il faut tout d'abord bien sûr disposer de données textuelles utilisable mais aussi libre de droits. Heureusement Internet regorge de ce type de donnée, beaucoup de sites nous permettent d'en avoir et nous en avons choisi deux d'entre-eux, Reddit (plus grand site communautaire de discussion) et arXiv (site de publication d'articles scientifiques). Grâce à leurs API disponible nous avons pu facilement récupérer des données de discussions et d'articles. Parmi ces données nous pouvons citer le titre de chaque document, le texte, le ou les auteurs, leurs dates de publication et l'URL. Les données ne venant pas de la même source d'information, nous avons dû préparer les données en fonction de chaque sources. Avant de pouvoir commencer à analyser nous avons dû gérer les spécifications entre les deux sources, par exemple la gestion des co-auteurs. Après les spécificités, il était impératif de nettoyer les textes des documents par exemple de supprimer les chiffres, remplacer les sauts de lignes, la ponctuation et créer un vocabulaire sur ce textes nettoyés.

3.3 Fonctionnement des classes

Afin de mener à bien ce projet nous avions besoin de plusieurs classes, qui ont chacune leur propre utilité et qui représentent une entité :

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

Le projet possède aussi deux autres fichiers python, qui ne sont pas des classes, qui ne représentent pas une entité mais qui sont tout autant indispensable au bon fonctionnement du programme :

- loadDataFromAPI : Télécharge et enregistre sur le répertoire (dossier /test_data) des données (documents => id2doc et auteurs => id2aut) via les APIs Reddit et arXiv.
- main : Initialise le Corpus de documents avec les données sur le répertoire et gère le fonctionnement de l'interface graphique.

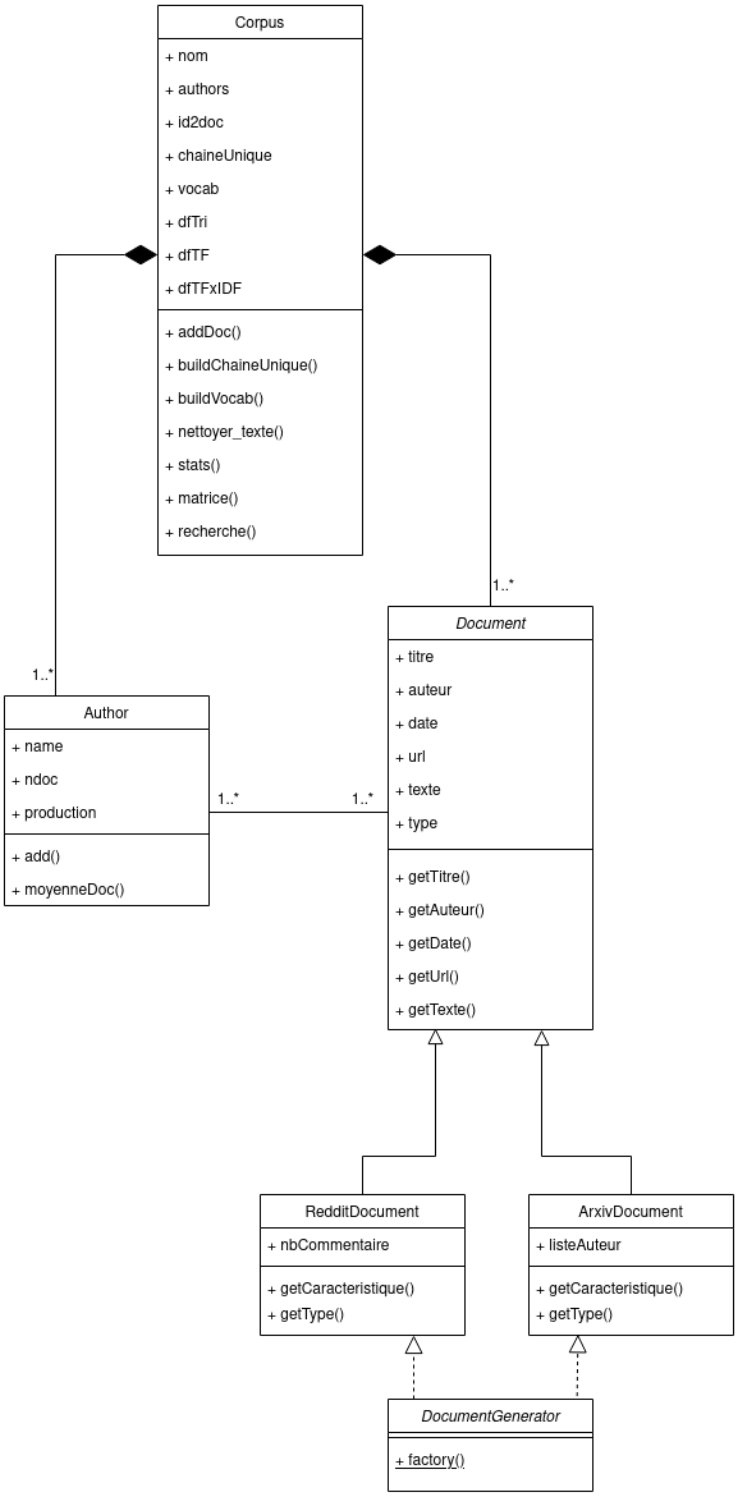

FIGURE 2 – Diagramme de classes

4 Conception

4.1 Tâches Partagées

Le partage des tâches était plutôt simple, l'un s'est occupé de la partie chargement/nettoyage des données, implémentation des méthodes d'analyse, programmation des méthodes de pondération, de recherche etc... et l'autre de l'interface Dash et donc moduler les méthodes à cette librairie. Nous avons tout les deux fait les travaux pratiques de l'UE jusqu'au même point, puis naturellement quand il a fallût commencer le projet nous nous sommes chacun dirigés vers une tâche spécifique.

4.2 Programme

4.2.1 Récupération de données

La première chose à mentionner est la récupération des données, et la création d'entité avec ces données.

```
reddit = praw.Reddit(client_id='KL8AdjqIAdRyS3uaswVLCA',client_secret='5vOe4iyO_1XBrq0LISlxe06MlK1f-Q',user_agent='nath
hot_posts = reddit.subreddit('space').hot(limit=1000) # get n first hot posts from the Space subreddit
```

FIGURE 3 – Récupération données Reddit avec praw

C'est en appelant l'API de chacune des sources que nous pouvons récupérer les données, dans la figure ci-dessus nous récupérons les 5 premiers "hot post" c'est à dire les 5 premiers post populaire du subreddit r/space. Parmi ces posts nous avons accès à plusieurs type de données, le titre, l'auteur, le texte, la date etc... il faut donc créer des entités avec ces données pour les utiliser facilement dans notre corpus.

```
for post in hot_posts:
    dateTime = datetime.datetime.utcfromtimestamp(post.created)
    post.selftext.replace("\r\n", " ")
    if post.selftext != "":
        nb_doc_non_vide += 1
        document = src.DocumentGenerator.DocumentGenerator.factory("Reddit",post.title,post.author.name,post.url,post.selftext,
```

FIGURE 4 – Génération d'un Document

En parcourant les posts que nous avons récupéré, nous pouvons alors créer des Documents avec la classe 'DocumentGenerator', en précisant la source des données.

4.2.2 Nettoyage de texte

Une fois toutes les données sont récupérées, les entités Document, Auteur créées, la prochaine étape est le nettoyage de texte du Corpus, pour, d'une part créer un vocabulaire de mots et d'autre part pour faciliter les calculs complexes sur ces textes.

Pour le nettoyage de textes nous avons choisis d'utiliser la librairie 'ntlk' pour sa facilité d'utilisation, cette librairie fait quelque chose qui parait évident pour l'analyse de textes mais qui aurait été long à faire sans elle, la suppression des 'stops words' (ou mots vide). C'est à dire des mots tellement inutile à utiliser pour de l'analyse de texte, comme 'the', 'that', 'a' etc... en anglais, pour qu'on les supprime complètement du texte.

```python
def nettoyer_texte(self,chaine):
    # remove links
    chaine = re.sub(r'https?://\S+|www\S+', '', chaine) # enlève les mots qui contiennent des liens (https
    # tokenize
    tokens = word_tokenize(chaine)
    #convert in lower case
    tokens = [w.lower() for w in tokens]
    #prepare regex for char filtering
    re_punc = re.compile('[%s]' % re.escape(string.punctuation))
    # remove punctuation from each word
    stripped = [re_punc.sub('',w) for w in tokens]
    # remove remainning token that are not alphabetic
    words = [word for word in stripped if word.isalpha()]
    #filter out stop words
    stop_words = set(stopwords.words('english'))
    words = [w for w in words if not w in stop_words]
    # filter token with 1 char
    words = [w for w in words if len(w)>1]
    cleaned_doc = ' '.join(word for word in words) # pour faire un paragraphe entier (concatene les mots)
    return cleaned_doc
```

FIGURE 5 – Nettoyage de texte avec nltk et re

Ce qui est très pertinent quand on veut analyser l'importance d'un mot dans un Corpus à thème, on ne veut pas connaître l'importance de ces stops words qui n'ont aucun intérêt mais des mots qui sont en rapport avec le thème. Nous avons aussi choisi de supprimer les liens des textes, pour beaucoup de raisons, car les liens contiennent des mots qui peuvent fausser les résultats, un lien n'est pas un mot, on ne veut pas faire d'analyse de liens mais de mots, chaque lien est différent alors cela aurait été beaucoup trop compliqué pour les calculs suivants, nous avons donc trouvé cela pertinent de juste omettre les liens dans l'analyse.

Un problème rencontré est que certains post Reddit ne comporte pas de texte mais juste un lien (vers une image le plus souvent). Pour contourner ce problème, nous ne prenons pas les posts qui contiennent seulement un lien, durant les tests nous avons observé que sur 100 posts Reddit, environ, 12 étaient vraiment utilisables (ne possédait pas qu'un seul lien) et donc 88 inutilisables. Afin d'avoir le même nombre de documents venant de chaque source, donc nous devions connaître le nombre de documents utilisable de Reddit puis demander ce même nombre de documents à l'API d'arXiv.Une manière de faire ceci est de demander à l'API Reddit les 1000 premiers 'hot post' (pour prendre large) et dès que nous avons le nombre de posts utilisables voulu nous arrêtons la recherche Reddit puis nous récupérons ce même nombre de documents grâce à l'API d'arXiv.

Pour la création du vocabulaire commun qui sera utilisé comme base tout le long du programme, c'est plutôt simple, on prend tous les textes, sans exceptions, on les rassemble en un seul gros texte, on le nettoie, puis chaque mots restant après le processus devient un mot du vocabulaire, ce mot sera unique (pas de doublons).

4.2.3 Pondération TFxIDF

Pour calculer l'importance d'un mot dans un texte nous avons utilisé la méthode **TFxIDF** (term frequency-inverse document frequency), plus précise que la méthode **TF**

seule. Dans notre cas elle est utilisée pour comparer l'importance des les mots recherchés par l'utilisateur entre les différents documents et corpus.

La méthode TF est définie par :

$$TFi,j = \frac{Nombre\ d'apparitions\ du\ terme\ i\ dans\ le\ document}{Nombre\ total\ de\ termes\ dans\ le\ document\ j}$$

Et la méthode IDF par :

$$IDFi = \log \frac{Nombre\ total\ de\ documents\ dans\ le\ corpus}{Nombre\ de\ documents\ ou\ le\ terme\ ti\ apparait}$$

Une multiplication des ces deux résultats nous donne le TFxIDF, donc l'importance d'un mot, le plus il est élevé, le plus il est important/présent.

$$TF.IDF\ i,j = TF\ i,j\ .\ IDF\ i$$

Pour avoir le TFxIDF global d'un terme nous additionnons chacun de ses TFxIDF de chaque document où il est présent, ce qui nous donne un indicateur de vision générale d'un terme dans tout le corpus. On peut voir cette information dans la section "**Informations Générales**" de l'interface.

4.2.4 Similarité cosinus

Nous avons implémenté une recherche par mots-clés, pour que l'utilisateur puisse entrer les mots-clés qu'il désire et directement observer les documents qui sont le plus similaire à ceux-ci.

La méthode que nous avons utilisée pour la recherche par mots-clés, est la similarité cosinus, définie par :

$$\cos\ \theta = \frac{A\ .\ B}{||A||\ ||B||}$$

Cette méthode calcule la similarité entre deux vecteurs, le vecteur des mots-clés entrés par l'utilisateur (A) et le vecteur d'un document (B). Plus la similarité cosinus est élevée, plus le document est pertinent par rapport aux mots-clés entrés.

```
cosine_similarity = dot(vectorMotCles, vector) / (norm(vectorMotCles) * norm(vector))
```

FIGURE 6 – Calcul d e la similarité

Afin de créer le vecteur qui correspond aux mots-clés le programme parcourt le vocabulaire, si le mot clés est dans le vocabulaire alors on ajoute 1 au vecteur sinon on ajoute 0.

```
# vecteur 'motsCles' :
vectorMotCles = []
for word in self.vocab.keys():
    if word in motsCles:
        vectorMotCles.append(1)
    else:
        vectorMotCles.append(0)
```

FIGURE 7 – Vecteur mot clés

Même principe pour calculer les vecteurs des documents, pour chaque texte du corpus on parcourt le vocabulaire global, si un mot dans le texte contient un mot du vocabulaire on ajoute 1, sinon 0. Cela nous donne des deux vecteurs de même taille que le vocabulaire, et nous pouvons évaluer donc la similarité entre les mots-clés et les documents.

En calculant cette similarité avec tous les documents du corpus, nous pouvons déterminer avec précision quels sont les documents les plus similaires, et générer un classement du plus au moins similaire. Ainsi quand l'utilisateur recherche des mots-clés, l'interface classe les documents de chaque source selon leur similarité, l'utilisateur peut alors facilement trouver ce qu'il cherche.

4.3 Interface

Nous avons fait face à de nombreux bugs durant le développement de l'interface, nous n'avons pas réussi à correctement implémenter la multi-sélection d'auteur, à cause d'un problème de CSS.

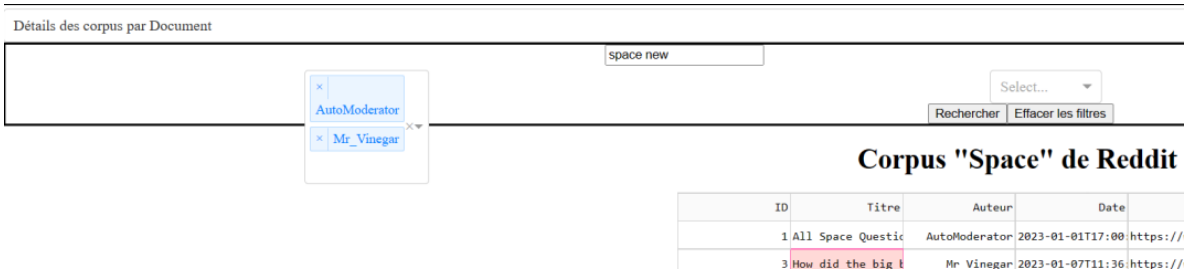

FIGURE 8 – Bug multi-sélection d'auteurs

Nous nous sommes alors tournés vers une mono-sélection mais avec la possibilité de choisir un auteur sur chacune des sources.

FIGURE 9 – Mono-sélection d'auteurs

4.4 Utilisation

Pour modéliser l'utilisation de l'interface voici un diagramme d'utilisation simplifié ci-dessous.

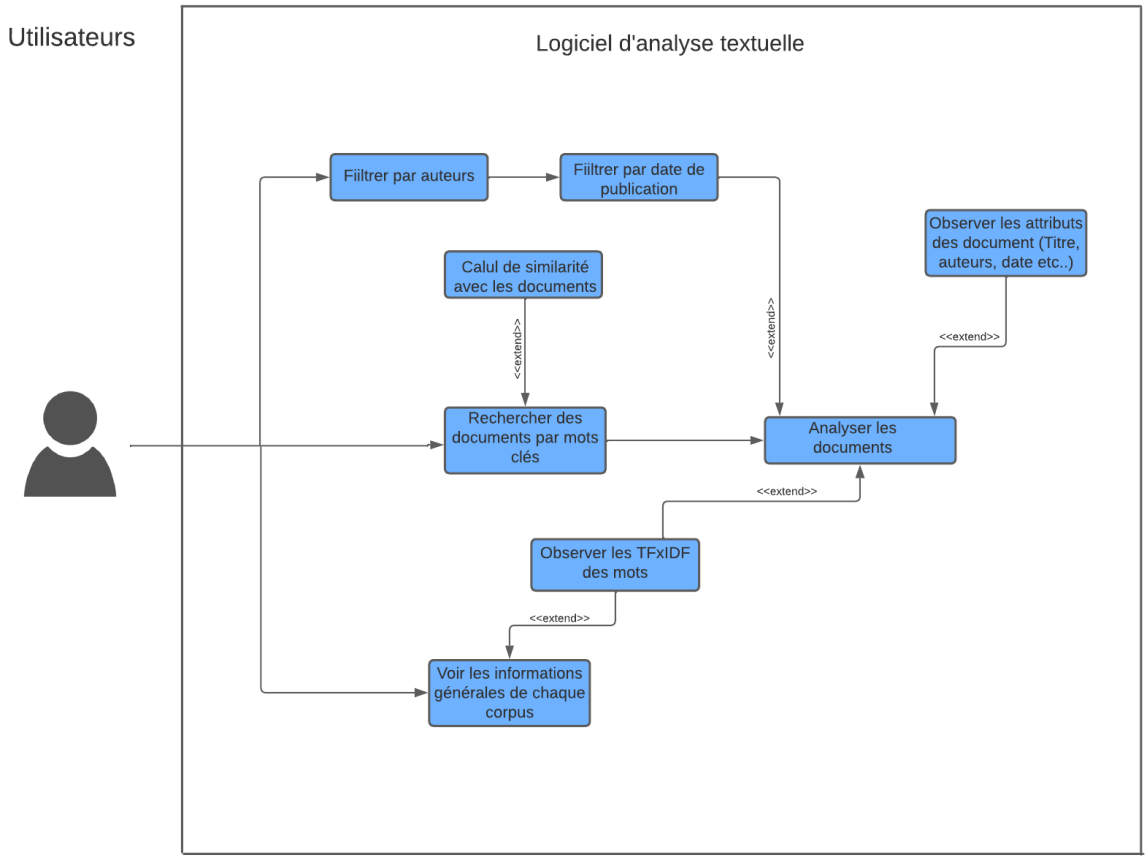

FIGURE 10 – Diagramme d'utilisation

Lorsque l'on lance le programme, la première chose que l'utilisateur voit c'est les **Informations Générales** des deux corpus, Reddit et arXiv, cela permet directement d'avoir une vision globale de l'analyse. On peut y trouver le nombre de documents des corpus, la taille du vocabulaire (créer à partir du texte des documents), le top 20 des mots les plus présents au total, grâce à la somme de leurs TFxIDF.

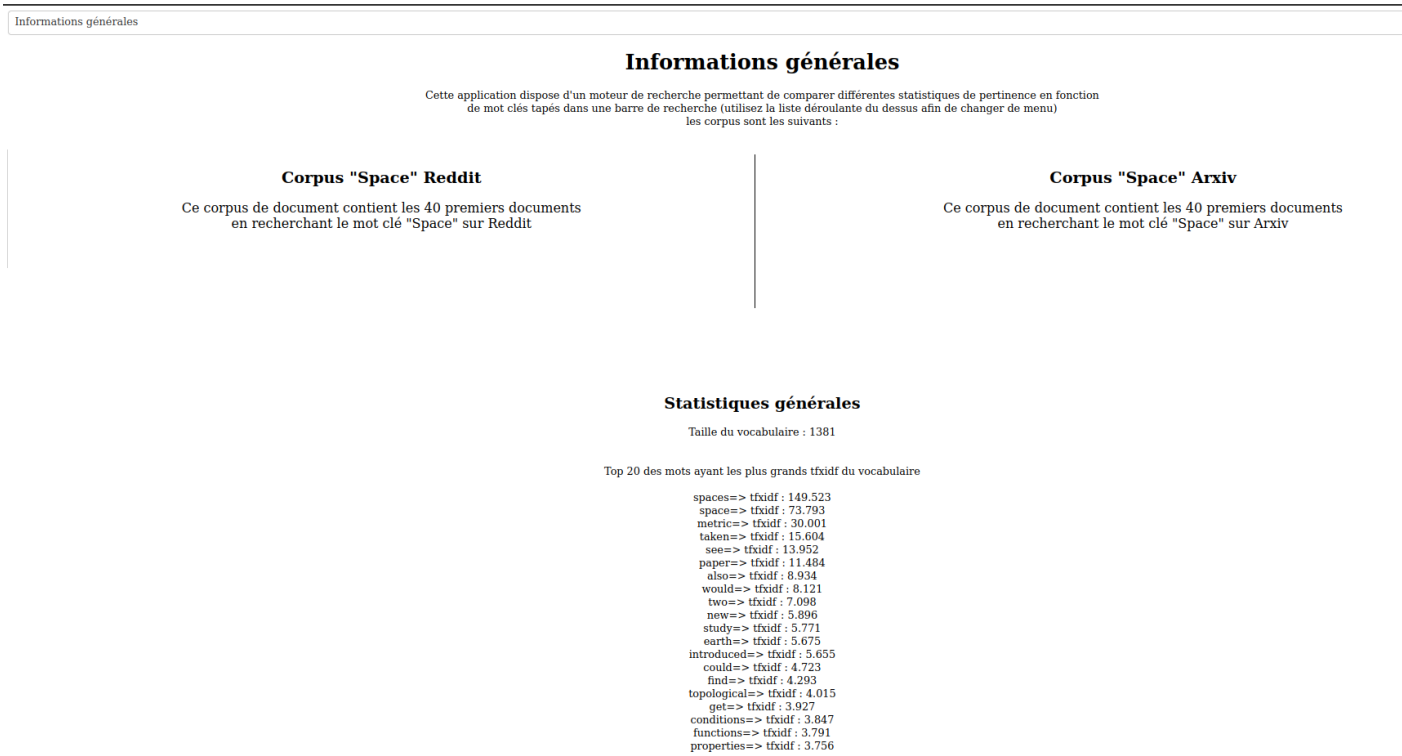

FIGURE 11 – Informations Générales

Ensuite à l'aide de la barre du haut on peut passer à une analyse détaillée des corpus, nous pouvons soit observer simplement les différents documents des corpus ou alors appliquer des filtres, rechercher des mots-clés parmi les documents et donc trier les documents en fonction de leurs similarités, filtrer avec la date de publication de l'article/post, ou encore filtrer avec le nom de l'auteur.

FIGURE 12 – Barre de recherche et filtres

Les données de tous les documents des corpus sont affichés, juste en dessous de cette barre de recherche/filtrage, on peut observer chaque attribut des documents, et comparer les documents voulus ensemble.
NB : Le 'score' correspond à la similarité entre les mots-clés et le document sélectionné.

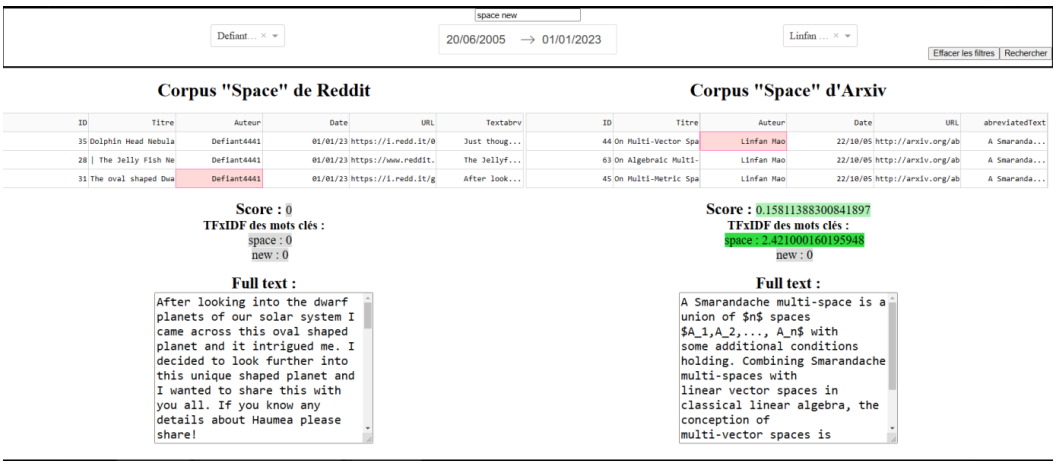

FIGURE 13 – Analyse Détaillée

5 Validation

5.1 Tests Unitaires

Pour simplifier le développement, le programme à tout d'abord était testé sur des petites quantités de données, puis au fur et à mesure la quantité a augmenté.

Avant d'afficher les TFxIDF dans l'interface lors de l'analyse nous avant dû nous assurer que notre méthode de calcul était bonne, c'est pourquoi dans le dossier 'output_data' il y a un fichier csv, TFxIDF.csv, qui permet de rendre compte des calculs effectués pour les données, même chose pour le calcul des TF, TF.csv.

	A	B	D	E	F
1		All Space Questions thread for week	Scientists Worried Humankind Will Descend	China Plans to Build Nuclear-Powered Moon Base V	Why are astronauts' nametags on the left chest now
2	additional	0	0	0	0
3	admissibly	0	0	0	0
4	algebra	0	0	0	0
5	also	0	0	0.0742769384836789	0
6	analysis	0	0	0	0
7	another	0.0590406434101037	0	0	0
8	appears	0	0	0	0
9	approach	0	0	0	0
10	ask	0.118081286820207	0	0	0
11	astronauts	0	0	0	0
12	away	0.0590406434101037	0	0	0
13	baire	0	0	0	0
14	banach	0	0	0	0
15	basic	0	0	0	0
16	betasigma	0	0	0	0
17	brangesrovnyak	0	0	0	0
18	buried	0.0590406434101037	0	0	0
19	cases	0	0	0	0
20	categories	0	0	0	0
21	characteristics	0	0	0	0
22	chest	0	0	0.0742769384836789	0
23	classical	0	0	0	0
24	closely	0	0	0	0
25	color	0	0	0.0742769384836789	0
26	combining	0	0	0	0
27	comments	0.0590406434101037	0	0	0
28	computable	0	0	0	0
29	concept	0	0	0	0
30	conception	0	0	0	0
31	conditions	0	0	0	0
32	construction	0	0	0	0
33	could	0.0590406434101037	0	0	0
34	crew	0	0	0.0742769384836789	0

FIGURE 14 – Fichier TFxIDF.xls généré par le programme

Aussi, pour tester la recherche, par exemple, cela c'est d'abord fait sans l'interface et sur une petite quantité de données, en analysant nous-mêmes si les résultats étaient cohérents.

```
-----------------------------------
Recherche avec :  ['space', 'new']
On the topological equivalence of S-metric and cone S-metric spaces  => score :  0.2886751345948129
Dolphin Head Nebula, 5000 light years away ZWO ASI533MC William Optics Z61 Skywatcher heq5 pro Optolong l-extreme 30 x 300s  => scor
All Space Questions thread for week of January 08, 2023  => score :  0.2581988897471611
All Space Questions thread for week of January 01, 2023  => score :  0.2581988897471611
Tchebyshev's characteristic of rearrangement invariant space  => score :  0.24618298195866545
```

FIGURE 15 – Test recherche pour la similarité cosinus

5.2 Tests Globaux

En utilisant l'interface nous avons remarqué un problème de compatibilité, en effet avec Mozilla Firefox les attributs des documents s'étendent sur les colonnes voisines et rend l'analyse illisible.

Corpus "Space" de Reddit

ID	Titre	Auteur	Date	URL	Textabrv
19	Kerbal Space Program is free on Epic Games!	LMFGamer	2023-01-05T22:44:41	https://www.reddit.com/r/space/comments/10	Fe/comments/10
29	What's your favorite non-fic book about space?	GiftedRealspace	2023-01-02T19:46:21	https://www.reddit.com/r/space/comments/10	T/comments/10
24	On the planet Mercury, a day is longer than a year.	Ah_Nonlogic_Mouse	2023-01-04T06:41:17	https://www.reddit.com/r/space/comments/10	de/comments/10
1	All Space Questions thread for week of January 08, 2023	AutoModerator	2023-01-08T17:00:12	https://www.reddit.com/r/space/comments/10	Pl/comments/10
40	All Space Questions thread for week of January 01, 2023	AutoModerator	2023-01-01T17:00:09	https://www.reddit.com/r/space/comments/10	Pl/comments/10

« ‹ 1 / 8 › »

FIGURE 16 – Diagramme d'utilisation

L'utilisateur est alors "obligé" d'utiliser Google Chrome, ou autres navigateur.

Nous avons aussi remarqué des incompatibilités, beaucoup de conflits, avec l'utilisation de composants bootstrap pour dash issu de la librairie **dash_bootstrap_components**, nous avons donc décidé de ne pas utiliser de librairie CSS pour Dash.

Le fait de sélectionner un document dans le menu "détail des corpus par Document" si aucun mot-clé n'a été saisi, renvoyait une erreur et n'affichait pas le texte, étant donné que le score n'était pas calculé et un None était renvoyé pour cette valeur.

Il y avait aussi une anomalie avec la pagination, les données renvoyées n'étaient pas les bonnes (par exemple si l'on clique sur la première ligne de la deuxième page, les données renvoyées en détail étaient celles de la première ligne de la première page.

6 Maintenance

Une des améliorations possible pour le programme serait de rendre la structure du code plus lisible, notamment pour l'interface, en créant un ou plusieurs fichiers qui permettrait de la gérer.

L'interface en général aurait pu être plus stylisée, avec des couleurs, une écriture particulière, etc...

Il aurait pu aussi être possible de filtrer les filtre entre eux : uand un certain auteur est choisi, l'intervalle se réduit aux dates correspondant aux documents postés par cet auteur et quand on change l'intervalle de dates, la liste déroulante des auteurs se met à jour afin de juste avoir les auteurs des documents dans l'intervalle choisi. De cette manière, l'utilisateur a moins de chances de tomber dans des options de filtre ne lui ramenant aucune donnée.

Pour le clic sur le bouton "Effacer les filtres" il faudrait qu'en plus d'effacer les filtres, les données soient ré-initialisée en incluant aucune valeur de filtre. Cela permettrait que l'utilisateur n'ait pas à recliquer sur le bouton "Rechercher" après avoir cliqué sur "Effacer les filtres" afin de revoir les données de base. Le problème est que l'output data des deux tables est déjà utilisé pour la fonction reliée au bouton recherche et un même Output ne peut normalement pas être exploité par deux fonctions différentes.