

Projet Moteur de recherche - Fouille de Textes

Lansana CISSE ¹
l.cisse@univ-lyon2.fr

Romain DUC ¹
romain.duc@univ-lyon2.fr

10 Janvier 2024

Enseignant : Julien Velcin

¹Université Lumière Lyon 2, Bron, France



Résumé

Rapport synthétisant l'ensemble des travaux dirigés réalisés tout au long du semestre, et exposant les analyses et méthodes de résolution de problèmes qui ont guidé notre approche pour la conception d'un moteur de recherche en Python.

Table des matières

1	Introduction	3
2	Spécifications du Moteur de Recherche	4
2.1	Application du Paradigme Orienté Objet (Classes et Héritage)	4
2.2	Héritage et Polymorphisme	4
2.3	Récupération et Stockage des Données	4
2.4	Mise en Œuvre de Patrons de Conception	4
2.5	Utilisation des Expressions Régulières	5
2.6	Implémentation de Techniques Statistiques	5
2.7	Génération Automatique de Documentation	5
2.8	Interface Graphique	5
3	Analyse	6
3.1	Environnement de travail	6
3.2	Données : ArXiv et Reddit	6
3.3	Description des Classes	6
3.4	Gestion de code source : GitHub	7
4	Conception du Projet	8
4.1	Partage des tâches	8
4.2	Analyse du programme	8
4.2.1	Scraping	8
4.2.2	Nettoyage textuel	8
4.2.3	Analyse avec TFXIDF	9
4.3	Similarité cosinus	10
4.4	Utilisation	11
5	Validation	12
5.1	Tests unitaires	12
5.2	Tests globaux	12
6	Bilan, Maintenance	13

1 Introduction

Dans le cadre du développement d'un moteur de recherche ciblant les articles publiés sur Reddit et ArXiv, ce projet, réalisé en Python, se décompose en plusieurs Travaux Dirigés (TD), chacun contribuant à une étape clé de son développement. L'objectif principal est de construire ce moteur de recherche sans recourir à des bibliothèques pré-existantes, en mettant l'accent sur une compréhension approfondie et l'application pratique des concepts de programmation en Python.

Notre projet a débuté avec l'acquisition et le traitement des données (TD 3), une étape cruciale où nous avons collecté et manipulé des informations depuis des sources comme Reddit et Arxiv, en utilisant la bibliothèque Pandas pour établir la base de notre moteur de recherche. Cette première phase a naturellement évolué vers la structuration et l'organisation du code (TD 4), où nous avons développé des classes et des modules pour une gestion optimisée des documents et des auteurs, ainsi que pour la manipulation d'un corpus de documents. Puis, nous avons intégré des concepts plus complexes tels que l'héritage et les patrons de conception Singleton et Factory (TD 5), ce qui a permis d'enrichir notre structure de projet avec des classes spécifiques pour les documents Reddit et Arxiv, et d'implémenter une interface utilisateur intuitive via des notebooks Jupyter. Enfin, nous avons approfondi notre analyse du contenu textuel (TD 6) en utilisant des expressions régulières pour la recherche de mots-clés et la construction de concordanciers, et en appliquant des méthodes statistiques pour étudier le corpus, incluant le comptage des mots et l'analyse de leur fréquence.

Ce rapport synthétise l'ensemble des TD réalisés tout au long du semestre, tout en exposant les analyses et méthodes de résolution de problèmes qui ont guidé notre approche.

2 Spécifications du Moteur de Recherche

Pour ce projet, l'objectif principal était de concevoir un outil d'analyse de textes capables de collecter des données textuelles sur Internet (grattage), d'analyser ces données et d'être accessible et compréhensible pour des utilisateurs qui n'ont pas nécessairement de compétences informatiques. Une approche intéressante serait de développer une interface graphique permettant aux utilisateurs de classer facilement les documents selon leur origine, de filtrer par auteurs, date de publication, et d'évaluer la pertinence des mots dans les ensembles de données.

Le projet devait intégrer les techniques d'analyse textuelle, incluant la collecte et la préparation des données, la création d'entités telles que Document, Auteur, Corpus, et l'élaboration d'un vocabulaire. Ensuite, il s'agissait de mettre en œuvre des méthodes d'analyse statistique, comme les fonctions de concordance de mots, les méthodes de formation (TF, TF-IDF), et des techniques de recherche basées sur la similitude, entre autres.

En somme, notre programme était conçu pour :

- Rechercher et extraire des données sur Internet (posts Reddit, articles ArXiv)
- Générateur des droits comme Document / Auteur à partir de ces données
- Les intégrer dans un Corpus
- Nettoyer les données
- Appliquer des techniques d'analyse avancées telles que le calcul de TF-IDF (fréquence de terme, fréquence de document inverse), la similitude cosinus, etc.
- Fournir une interface graphique permettant à l'utilisateur de manipuler et d'analyser aisément ces données.

Les spécifications détaillées du projet sont présentées ci-dessous :

2.1 Application du Paradigme Orienté Objet (Classes et Héritage)

Le code est structuré rigoureux avec des classes, facilitant une gestion efficace des types de données et des fonctions diverses.

L'utilisation de l'héritage enrichit et personnalise les classes existantes, renforçant la réutilisation et l'organisation du code.

2.2 Héritage et Polymorphisme

Ces principes de la programmation orientée objet sont essentiels pour développer des classes spécialisées, gérant différents formats de documents textuels. Le polymorphisme permet de manipuler des objets de classes variées via une interface unique, augmentant la flexibilité et la scalabilité du moteur.

2.3 Récupération et Stockage des Données

Utilisation de Pandas pour une manipulation efficace et intuitive des données textuelles sous forme de DataFrames.

Sauvegarde des données transformées en format Pickle pour conserver la structure des objets

2.4 Mise en Œuvre de Patrons de Conception

Adoption du patron Singleton pour les classes de gestion de base de données, assurant une instance unique et une gestion centralisée du corpus.

Factory pour la création d'objets (Document, RedditDocument, ArxivDocument), permettant une flexibilité dans la gestion des types d'objets.

2.5 Utilisation des Expressions R guli res

Permettent le nettoyage et la normalisation des donn es textuelles avant leur analyse.

2.6 Impl mentation de Techniques Statistiques

Utilisation des statistiques descriptives pour r sumer les tendances et les caract ristiques dans les ensembles de donn es.

Mise en  uvre d'analyses de corr lation pour identifier les relations entre diff rents termes.

2.7 G n ration Automatique de Documentation

Int gration d'outils comme Sphinx ou Doxygen pour g n rer automatiquement une documentation d taill e du code source.

2.8 Interface Graphique

Int gration de visualisations de donn es, et des statistiques, pour une analyse textuelle plus interactive et visuelle via le framework python Dash, sp cialis  dans la cr ation d'applications web de tableaux de bord. Dash permet de cr er des graphiques interactifs et d'int grer ces graphiques dans des interfaces.

3 Analyse

Dans notre projet de développement, nous avons sélectionné un environnement de travail spécifique pour maximiser l'efficacité et assurer une compatibilité optimale. Voici les éléments clés de cet environnement :

3.1 Environnement de travail

IDE : Visual Studio Code.

Visual Studio Code est un IDE polyvalent et léger, offrant une excellente intégration avec Python, Jupyter Notebook et Python. Sa flexibilité et son grand évènement d'extensions nous permettent d'adapter l'environnement aux besoins spécifiques de notre projet.

3.2 Données : ArXiv et Reddit

En utilisant les techniques du web grattage avec les API nous allons récupérer les publications Reddit et Arxiv avec les attributs suivants :

- Le titre de la publication
- Le texte contenant l'expression
- L'auteur de la publication
- La date et l'heure de publication
- L'URL de la publication

3.3 Description des Classes

- Classe Document :

Cette classe sert de base pour stocker les informations communes à tous les types de documents, telles que le titre, le(s) auteur(s), la date de publication, l'URL et le contenu textuel.

- Sous-Classes spécialisées :

RedditDocument : Adaptée pour les spécificités des documents issus de Reddit, cette classe-fille hérite de Document et ajoute ou modifie les attributs nécessaires pour refléter les particularités des publications Reddit.

ArxivDocument : De manière similaire, cette classe-fille est dédiée aux documents provenant d'ArXiv, avec des ajustements spécifiques pour ce type de source.

- Classe Auteur :

Contient des informations détaillées sur les auteurs, comme leur nom, le nombre de documents publiés et leurs contributions diverses.

- Classe DocumentGenerator :

Conçue pour faciliter la génération et le téléchargement de documents à partir des APIs. Cette classe automatise la création d'instances de RedditDocument ou ArxivDocument pendant l'acquisition des données.

— Classe Corpus :

Représente un ensemble de documents et d’auteurs liés à un sujet spécifique. Cette classe englobe l’ensemble des documents (Reddit et ArXiv) et offre des méthodes pour leur analyse.

— Fichier LoadData :

Enregistre sur le répertoire /data les données collectées via l’API Reddit et ArXiv via pickle

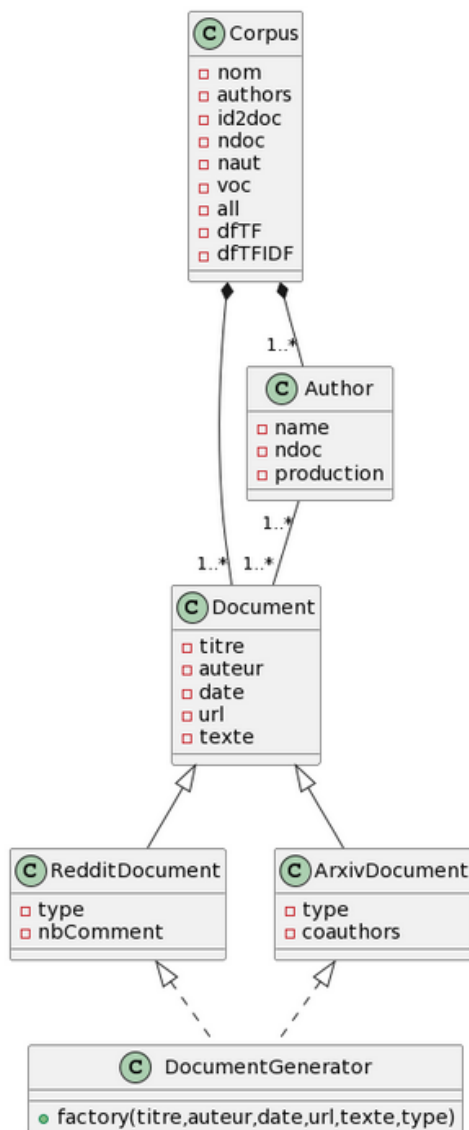


FIGURE 1 – Diagramme de classes et attributs

3.4 Gestion de code source : GitHub

Nous avons opté pour GitHub pour le dépôt de nos codes sources et leur gestion de versions.
URL vers le dépôt GitHub du projet <https://github.com/lansanacisse/ProjetProgrammationPython.git>

4 Conception du Projet

4.1 Partage des tâches

Concernant les travaux pratiques liés à l'unité d'enseignement, nous avons progressé de manière équivalente jusqu'à un certain point. Par la suite, pour le projet, nous nous sommes naturellement orientés vers des tâches qui correspondaient à nos compétences et appétence spécifiques

4.2 Analyse du programme

4.2.1 Scraping

On doit au préalable récupérer les données à utiliser. Le point de départ de notre projet est la collecte et l'organisation des données. Nous utilisons les API des sources concernées pour extraire les informations nécessaires. Par exemple, comme illustré dans la figure mentionnée, nous extrayons les 50 posts les plus populaires (« hot posts ») du subreddit **r/MachineLearning**. Ces posts offrent une diversité de données telles que le titre, l'auteur, le contenu, la date, les commentaires, etc. Pour gérer efficacement ces informations, nous créons des entités correspondantes, facilitant leur intégration dans notre corpus. À travers l'exploration des posts récupérés, nous générons des instances de la classe 'DocumentGenerator', en spécifiant clairement la source de chaque donnée.



```
reddit = praw.Reddit(client_id='tIn-EBuYy_LRYNlp6g_HVQ', client_secret='Y_77xj7yIgUY6ucEdIfzqjmYz_PxGw', user_agent='romain')
ml_posts = reddit.subreddit('MachineLearning').hot(limit=1000) # Retrieve hot posts from the ML subreddit
```

FIGURE 2 – Récupération des données



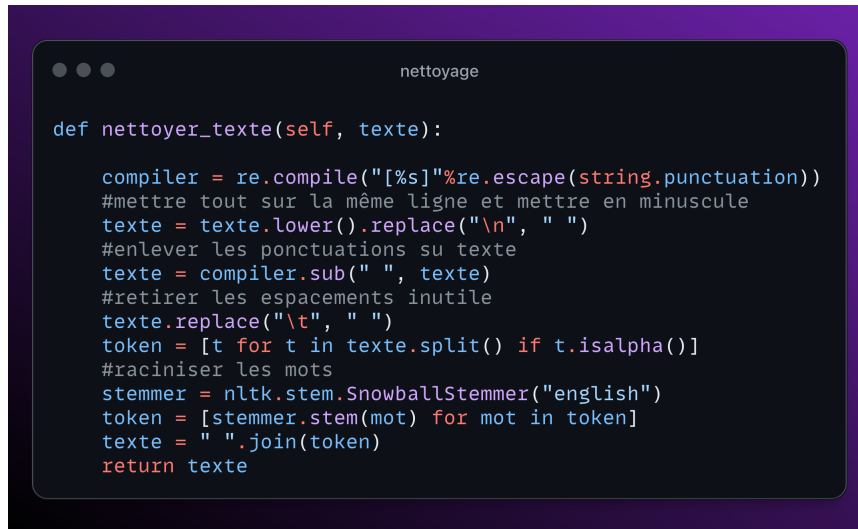
```
for post in ml_posts:
    if valid_docs_count < 50:
        post_timestamp = datetime.datetime.utcfromtimestamp(post.created)
        post.selftext.replace("\r\n", " ")
        if len(post.selftext.strip()) > 60 and post.author:
            valid_docs_count += 1
            reddit_doc = DocumentGenerator.DocumentGenerator.factory(post.title, post.author.name, post_timestamp, post.url, post.selftext, "Reddit")
            doc_repository[doc_index] = reddit_doc
            doc_index += 1
```

FIGURE 3 – Création d'un document

4.2.2 Nettoyage textuel

Après avoir rassemblé toutes les données nécessaires et constitué les entités Document et Auteur, nous passons à l'étape cruciale du nettoyage des textes au sein du Corpus. Cette étape a pour double objectif de construire un vocabulaire pertinent et de simplifier les traitements analytiques complexes sur les textes. Pour le nettoyage des textes, notre choix s'est porté sur la bibliothèque 'nltk', reconnue pour sa simplicité d'utilisation. Elle offre une fonctionnalité clé pour l'analyse de texte : l'élimination des 'stop words'. Ces mots, tels que 'the', 'that', 'a' en anglais, sont écartés car ils n'apportent pas de valeur significative pour l'analyse thématique. En les supprimant, nous pouvons nous concentrer sur les mots réellement pertinents en lien avec le sujet du Corpus. Nous avons également décidé d'exclure les liens des textes. Cette suppression est justifiée car les liens peuvent introduire des biais dans les résultats d'analyse, d'autant plus que chaque lien est unique et complexifierait inutilement les calculs.

Quant à la création du vocabulaire commun, le processus est relativement direct. Nous compilons tous les textes en un seul corpus global, le nettoyons, et ensuite, chaque mot distinct restant après ce processus de nettoyage est intégré au vocabulaire. Cette méthode garantit un vocabulaire unique, sans redondances.



```
def nettoyer_texte(self, texte):

    compiler = re.compile("[%s]"%re.escape(string.punctuation))
    #mettre tout sur la même ligne et mettre en minuscule
    texte = texte.lower().replace("\n", " ")
    #enlever les ponctuations su texte
    texte = compiler.sub(" ", texte)
    #retirer les espacements inutile
    texte.replace("\t", " ")
    token = [t for t in texte.split() if t.isalpha()]
    #raciniser les mots
    stemmer = nltk.stem.SnowballStemmer("english")
    token = [stemmer.stem(mot) for mot in token]
    texte = " ".join(token)
    return texte
```

FIGURE 4 – Nettoyage du texte (re, nltk)

4.2.3 Analyse avec TFxIDF

Afin d'évaluer le poids d'un mot spécifique dans un texte, nous avons opté pour l'approche TFxIDF (fréquence du terme - fréquence inverse du document), qui s'avère plus efficace que la simple méthode de fréquence du terme (TF). Cette technique est particulièrement utile pour mesurer et comparer la pertinence des mots clés recherchés par l'utilisateur à travers divers documents et corpus. La méthode TF est calculée comme suit :

$$TF(i, j) = \frac{\log_2(\text{Freq}(i, j)) + 1}{\log_2(L)}$$

avec i : Terme dont le Term Frequency dans le document doit être calculé, j : document à analyser, L_j : nombre de mots dans j , $\text{Freq}(i, j)$: fréquence d'un mot i dans j .

Quant à l'IDF, elle se définit par :

$$IDF(i) = \log \left(\frac{N_D}{f_i} + 1 \right)$$

avec i : Terme dont l'Inverse Document Frequency doit être calculée, N_D : nombre de tous les documents dans le corpus, f_i : nombre de les docs dans lesquels i apparaît.

En combinant ces deux mesures, nous obtenons le TFxIDF, qui reflète l'importance d'un mot dans un texte. Plus la valeur du TFxIDF est élevée, plus le mot est considéré comme significatif.

$$TF(i, j) = TF_{i,j} \times IDF_i$$

Pour déterminer la valeur globale du TFxIDF d'un terme, nous additionnons les TFxIDF de ce terme pour chaque document où il apparaît. Cela nous fournit une vue d'ensemble de la présence et de la

pertinence du terme dans l'ensemble du corpus.

$$tf_{i,d} = \frac{\text{frequency of term } i \text{ in document } 'd'}{\text{total terms in document } 'd'}$$

$$idf_t = \log_{10} \left(\frac{\text{total number of documents}}{\text{total documents with term } 't'} \right)$$

4.3 Similarité cosinus

La similitude cosinus est une mesure utilisée pour calculer le degré de similitude entre 2 vecteurs. Les étapes sont les suivantes :

- Représentation vectorielle : Chaque document est transformé en un vecteur. Chaque élément du vecteur représente un mot ou une caractéristique du document, souvent réfléchi par des mesures comme TFxIDF .
- Calcul de la similitude : La similitude cosinus entre deux documents est calculée en prenant le cosinus de l'angle entre leurs vecteurs respectueux. Mathématiquement, cela se fait en divisant le produit scalaire des deux vecteurs par le produit de leurs normes. La formule est généralement la suivante :

$$\cos \theta = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

où A et B sont les vecteurs des documents.

- Interprétation : La valeur résultante varie entre -1 et 1. Une valeur de 1 indique une similitude parfaite (les documents sont identiques), 0 signification aucune similitude, et -1 indique une dissemblance totale (les documents sont complètement différents).

On peut également utiliser cette méthode pour la recherche par mots-clés. Plus la similitude cosinus est importante, plus le document correspond aux mots-clés entrés.

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

```

similarité cosinus

def searchCosine(self, keywords):
    # Création du vecteur pour les mots-clés
    # 1 si le mot est dans les mots-clés, 0 sinon
    vecteur_mots_cles = np.array([1 if mot in keywords else 0 for mot in self.voc])

```

FIGURE 5 – Vecteurs de mots-clés

```

similarité cosinus

def searchCosine(self, keywords):
    ...
    similarite_cosinus = (vecteur_mots_cles @ np.array(vecteur)) / (norme_vecteur_mot_cles * norme_vecteur_doc)

```

FIGURE 6 – calcul de la similitude cosinus dans la fonction

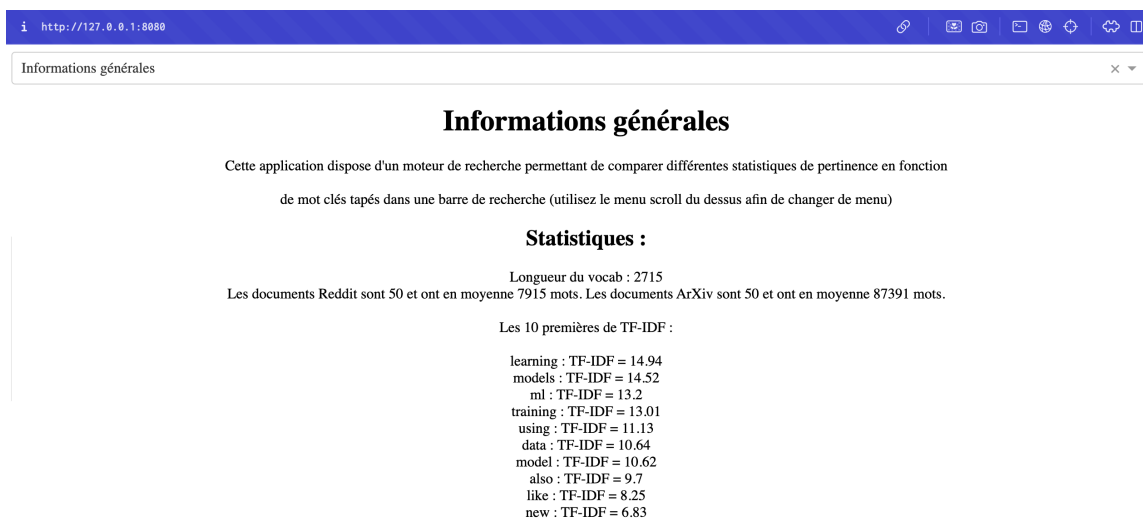
4.4 Utilisation

Afin de lancer le programme, il s'agit d'exécuter le code du fichier *index.py* et copier-coller l'URL suivante dans un navigateur : `http://127.0.0.1:8080/`

```
Dash is running on http://127.0.0.1:8080/

* Serving Flask app 'index'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:8080
Press CTRL+C to quit
```

Sur une alors cette interface de départ :



The screenshot shows a web browser at `http://127.0.0.1:8080`. The page title is 'Informations générales'. The main content area has a heading 'Informations générales' and a description: 'Cette application dispose d'un moteur de recherche permettant de comparer différentes statistiques de pertinence en fonction de mot clés tapés dans une barre de recherche (utilisez le menu scroll du dessus afin de changer de menu)'. Below this, there is a section 'Statistiques :' with the following data: 'Longueur du vocab : 2715', 'Les documents Reddit sont 50 et ont en moyenne 7915 mots. Les documents ArXiv sont 50 et ont en moyenne 87391 mots.', 'Les 10 premières de TF-IDF :', and a list of TF-IDF values for various models: learning (14.94), models (14.52), ml (13.2), training (13.01), using (11.13), data (10.64), model (10.62), also (9.7), like (8.25), and new (6.83).

A l'aide du menu troublant du haut, on peut effectuer une analyse plus détaillée des documents, en choisissant le 2e menuItem :



The screenshot shows the same web browser, but the sidebar menu now has 'Détails des corpus par Document' selected. The main content area is identical to the previous screenshot, showing the 'Informations générales' page with the same statistics and description.

On arrive sur le moteur de recherche à proprement parler :

i http://127.0.0.1:8080
 🔗 📄 📷 📧 ⚙️ 🔄 📱

Détails des corpus par Document

Saisir les mots clés...

Select... 12/08/2015 → 05/01/2024 Select...

Effacer les filtres Rechercher

Corpus "ML" de Reddit

ID	Titre	Auteur	Date	URL	Texte
1 [D]	Simple Qu	AutoModerator	31/12/23	https://www.r	Please pos...
2 [D]	Dropping	TheMysticalJa	04/01/24	https://www.r	I am about...
3 [D]	Training	Electronic_Ha	05/01/24	https://www.r	I have to ...
4 [D]	ArXiv alt	osamc	05/01/24	https://www.r	My current...
5 [D]	Setting u	Dr-LucienSanc	05/01/24	https://www.r	I am wanti...

<< < 1 / 10 > >>

Corpus "ML" d'Arxiv

ID	Titre	Auteur	Date	URL	Texte
51	From Cutting	Liva Ralaivol	12/08/15	http://arxiv.	Cutting-pl...
52	A State-of-th	Zainab AL-Oth	02/10/20	http://arxiv.	The Intern...
53	A Survey on R	Syed Muhammad	16/10/19	http://arxiv.	Artificial...
54	Deep Learning	Ismail Irmakc	01/03/20	http://arxiv.	The diagno...
55	Tokamak disru	Joost Croonen	11/05/20	http://arxiv.	Disruption...

<< < 1 / 4 > >>

i http://127.0.0.1:8080
 🔗 📄 📷 📧 ⚙️ 🔄 📱

Détails des corpus par Document

TheMy... x

Neural Deep

12/08/2015 → 05/01/2024

Ismail I... x

Effacer les filtres Rechercher

Corpus "ML" de Reddit

ID	Titre	Auteur	Date	URL	Texte
2 [D]	Dropping ou	TheMysticalJam	04/01/24	https://www.rec	I am about...

Corpus "ML" d'Arxiv

ID	Titre	Auteur	Date	URL	Texte
54	Deep Learning f	Ismail Irmakci	01/03/20	http://arxiv.or	The diagno...

5 Validation

5.1 Tests unitaires

Étant donné que notre projet intègre de nombreuses classes et méthodes, l'exécution de tests unitaires est primordiale. Ces tests étaient particulièrement utiles pour identifier les sources d'erreurs lorsque les résultats obtenus ne correspondaient pas à ce qui était attendu. Le fichier *testUnit.py* comprend les tests unitaires des classes du modèle. Nous avons utilisé le module **unittest** qui est un framework de test s'inspirant à l'origine de JUnit.

5.2 Tests globaux

Nous avons effectué des tests globaux lors de ce projet.

Nous avons déployé notre application dans un nouvel environnement. Pour ce faire, nous avons exploité le fichier *requirements.txt* pour installer toutes les dépendances nécessaires.

Lors de l'utilisation de notre interface, nous avons identifié un souci de compatibilité spécifique avec

Firefox. Nous avons observé que la mise en forme des attributs des documents dans l'interface ne se comportait pas correctement, provoquant un chevauchement sur les colonnes adjacentes. ce qui a eu pour effet de rendre l'analyse des données illisible.

6 Bilan, Maintenance

Durant ce projet, nous avons efficacement élaboré un fichier .pkl en utilisant des données extraites via des requêtes API. Ensuite, nous avons développé une interface permettant de rechercher des mots-clés. Cette fonctionnalité était enrichie par des informations complémentaires telles que les mots les plus pertinents selon les critères de pondération TF et TFxIDF, le nombre total de documents répondant aux critères de recherche. Cette expérience nous a permis d'améliorer considérablement nos compétences en Python orienté objet, en interfaces utilisateur et en gestion de projet à échelle moyenne.

Améliorations, prolongements possibles :

- Ajout de nouvelles sources de données textuelles, amélioration des algorithmes de traitement ;
- Restructuration, réorganisation de l'interface graphique en fichiers distincts ;
- Gestion des synonymes ou en attribuant des poids à certains mots pour améliorer la pertinence des résultats.

Facilité de mise à jour :

- Grâce à l'architecture modulaire, l'ajout de nouvelles fonctionnalités est facilité.