Rapport Projet Python QueryPy

Olivier ANDRIKO
N° étudiant : 5240662

Lien vers le repo GitHub du projet : https://github.com/valdoin/QueryPy/tree/main

1. Spécifications

Description de l'application

L'application est un moteur de recherche documentaire conçu pour analyser et explorer des contenus provenant de **l'API Reddit** et d'**Arxiv**. L'objectif principal est de permettre aux utilisateurs de rechercher et filtrer efficacement les contenus en fonction de mots-clés, d'auteurs ou d'années spécifiques. L'application inclut une interface utilisateur ainsi qu'un moteur d'analyse textuelle avancé.

Fonctionnalités principales :

1. **Extraction des données** :
    - Récupération des données depuis Reddit et Arxiv à partir d'un mot-clé spécifié.
    - Création de documents spécifiques à chaque source (Reddit ou Arxiv).
2. **Analyse du corpus** :
    - Statistiques sur les documents (nombre de mots uniques, mots les plus fréquents, tableau de fréquences).
    - Recherche textuelle dans le corpus (par mots-clés et avec concordancier).
3. **Interface utilisateur interactive** :
    - Interface basée sur des widgets notebooks pour rechercher, filtrer, et explorer les résultats.
4. **Tests et validation** :
    - Implémentation de tests unitaires pour garantir la fiabilité du code.
5. **Persistance des données** :
    - Sérialisation/désérialisation en JSON.

Olivier ANDRIKO
N° étudiant : 5240662

2. Analyse

Environnement de travail

L'application est développée en **Python**, avec des bibliothèques principales comme :

- **pandas** pour la manipulation des données
- **numpy** pour les calculs mathématiques
- **scipy** pour la gestion des matrices creuses
- **scikit-learn** pour les calculs de similarité (TF-IDF, cosine similarity)
- **pywidgets** pour l'interface interactive
- **praw** et **xmltodict** pour l'extraction des données de Reddit et Arxiv

Le choix de ces bibliothèques a été motivé par leur robustesse et leur compatibilité avec les besoins du projet.

Données identifiées

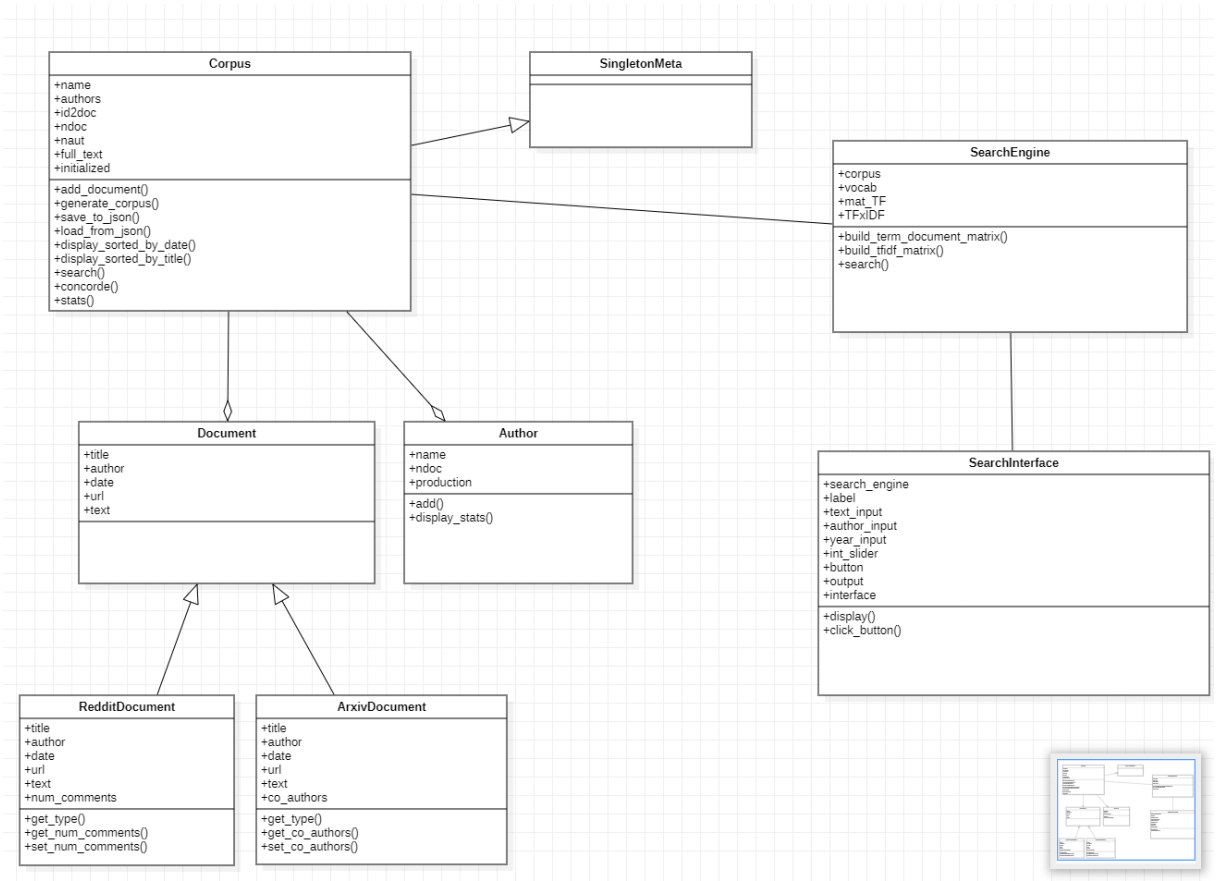
- **Reddit** : titre, auteur, date, URL, texte, nombre de commentaires.
- **Arxiv** : titre, auteur principal, co-auteurs, date, URL, résumé.
- **Corpus** : ensemble des documents collectés.
- **Vocabulaire** : mots uniques extraits des documents, fréquences d'apparition.

Diagramme de classes

(page suivante)

Rapport Projet Python QueryPy

Olivier ANDRIKO
N° étudiant : 5240662



Relations :

- **Héritage : RedditDocument** et **ArxivDocument** héritent de **Document**.
- **Composition : Corpus** contient une collection de documents et d'auteurs.
- **Utilisation : SearchEngine** utilise les données du **Corpus**, et **SearchInterface** interagit avec **SearchEngine.**

Olivier ANDRIKO
N° étudiant : 5240662

3. Conception

Partage des tâches

Le projet a été réalisé en totale autonomie, dans l'ordre des instructions des différents TD :

1. **Extraction des données** : (TD3 – TD6)
    - Implémentation des modules d'extraction pour Reddit et Arxiv.
    - Gestion des exceptions et validation des données extraites.
    - Outils d'analyses textuelles pour le corpus.
2. **Moteur de recherche** : (TD7)
    - Création du modèle TF-IDF et implémentation des filtres de recherche.
3. **Interface utilisateur** : (TD 8 – TD 10)
    - Développement des widgets interactifs et intégration avec le moteur de recherche.
4. **Tests** : (TD 9 – TD 10)
    - Écriture des tests unitaires pour chaque classe.

Algorithmes spécifiques

1. **TF-IDF et cosine similarity** :
    - Les documents sont transformés en une matrice TF-IDF.
    - Le score de similarité entre la requête utilisateur et les documents est calculé pour classer les résultats.
2. **Filtrage** :
    - Les résultats peuvent être filtrés par auteur et par année grâce à des conditions appliquées sur les métadonnées des documents.

Olivier ANDRIKO
N° étudiant : 5240662

Problèmes rencontrés et solutions

1. **Singleton et tests unitaires** :
    - Problème : La classe Corpus implémentée en tant que singleton causait des problèmes de persistance d'instance lors des tests.
    - Solution : Ajout d'une méthode *reset_instance* pour réinitialiser le singleton lors des tests.
2. **Filtrage des résultats** :
    - Problème : Les filtres d'auteur et d'année nécessitaient des modifications dans plusieurs parties du code.
    - Solution : Centralisation des conditions dans le moteur de recherche pour une meilleure maintenance.

Exemple d'utilisation

1. **Extraction des données** :
    - L'utilisateur lance main.ipynb et spécifie un mot-clé pour générer un corpus ou se sert du corpus sample fourni (*coronavirus_data.json*).
2. **Recherche** :
    - Dans l'interface il entre un mot-clé, un filtre d'auteur et une année pour obtenir des résultats.
3. **Résultats** :
    - L'application affiche les résultats classés par pertinence avec les métadonnées correspondantes.

4. Validation

Tests unitaires

Les tests unitaires sont implémentés avec **pytest**. Les commandes pour exécuter les tests :

python -m pytest Tests/searchengine_tests.py Tests/corpus_tests.py Tests/author_tests.py

Olivier ANDRIKO
N° étudiant : 5240662

Tests globaux

Les tests globaux ont été réalisés directement via l'interface :

1. **Cas particulier** : Recherche avec un filtre d'année non valide.
    - Résultat : Aucune erreur, mais aucun résultat affiché.
2. **Cas particulier** : Recherche avec une requête vide.
    - Résultat : Message demandant de saisir une requête.

5. Maintenance

Évolutions possibles

1. **Support multi-langues** :
    - Ajouter un support pour analyser des corpus dans des langues autres que l'anglais.
2. **Visualisation avancée** :
    - Ajouter des graphiques pour visualiser les statistiques du corpus (histogrammes des mots les plus fréquents, courbes de fréquence cumulée).
3. **Amélioration des filtres** :
    - Ajouter des filtres avancés, comme le type de document (Reddit ou Arxiv) ou une plage de dates.

Facilité de mise à jour

- L'application est bien structurée avec des classes indépendantes.
- L'utilisation de bibliothèques standard permet une intégration rapide des nouvelles fonctionnalités.
- Les tests unitaires garantissent une validation rapide des nouvelles modifications.