RAPPORT DU PROJET EN PROGRAMMATION PYTHON

Lien vers le dépôt GitHub:
https://github.com/laouabdiasellami/mix2.git

RAPPORT DU PROJET EN PROGRAMMATION PYTHON

Table des matières

RAPPORT DU PROJET EN PROGRAMMATION PYTHON

1. Introduction

Dans le cadre de notre Cours de Programmation de spécialité : Python en Master 1 Informatique à l'Université Lumière (Lyon 2), nous avons réalisé un projet visant à développer une application Python complète. Ce projet nous a permis de traverser toutes les étapes de conception d'un logiciel, de la définition des spécifications à la mise en place des tests et de la maintenance.

L'objectif principal était de développer une application bien structurée, ambitieuse et fonctionnelle, intégrant à la fois les enseignements dispensés dans les TDs 3 à 10 et les connaissances acquises en programmation orientée objet. Nous avons travaillé en binôme pour assurer une répartition équilibrée des tâches et pour maximiser l'efficacité de notre travail collaboratif.

2. Spécifications

Les spécifications de notre projet les suivantes. Nous avons identifié les fonctionnalités principales suivantes :

La réalisation d'un socle de base, inspiré des concepts des TDs 3 à 5, constituant les fondations de notre application, ensuite l'intégration d'un moteur de recherche, développé à partir des TDs 6 et 7, permettant d'effectuer des requêtes efficaces et de gérer des résultats pertinents, et pour finir l'ajout d'une interface utilisateur et de fonctionnalités d'extension issues des TDs 8 à 10, rendant notre application plus interactive et accessible.

Notre application est développée en Python 3.10, ceci garantissant une compatibilité multiplateforme. Par ailleurs, nous avons veillé à utiliser des bibliothèques adaptées et à fournir un fichier requirements.txt pour simplifier l'installation des dépendances.

3. Analyse

Pour mener à bien ce projet, nous avons choisi un environnement de travail adapté et avons défini une architecture claire pour notre application. Le langage Python a été sélectionné pour sa flexibilité

et aussi car c'est le langage utilisé en classe, tandis que l'IDE Visual Studio Code nous a permis de bénéficier d'un environnement de développement stable et ergonomique. GitHub a été utilisé pour la gestion des versions et la collaboration.

Dans un premier temps, nous avons identifié les données nécessaires au bon fonctionnement de notre application. Il s'agissait principalement de données structurées, permettant de gérer les requêtes utilisateur, les résultats de recherche et les interactions avec l'interface utilisateur.

Nous avons également conçu un diagramme des classes simplifié pour clarifier la structure du programme. Notre application se compose de trois classes principales :

Classe Document

- **Description** : Classe de base représentant un document générique.
- **Attributs** :
  - title: Titre du document.
  - author: Auteur du document.
  - date: Date de création du document.
  - text: Contenu du document (par défaut vide).
  - type: Type du document (défini par la méthode getType).
- **Méthodes** :
  - getType(): Retourne "Document".
  - __str__(): Représentation textuelle du document.

Classe ArxivDocument (hérite de Document)

- **Description** : Spécialisation pour les documents provenant d'Arxiv.
- **Attributs** :
  - Hérite tous les attributs de Document.
  - authors: Liste des auteurs du document.
- **Méthodes** :
  - getType(): Retourne "Arxiv".
  - __str__(): Représentation spécifique pour inclure plusieurs auteurs.
  -

Classe Corpus

- **Description** : Gère un ensemble de documents comme un corpus unique .
- **Attributs** :
  - _instance: Instance unique de la classe Corpus.
  - documents: Liste des documents contenus dans le corpus.
- **Méthodes** :
  - __new__(): Implémente le design pattern Singleton.

- add_document(document): Ajoute un document au corpus.
- display_documents(): Affiche tous les documents dans le corpus.

Classe WikipediaDocument (Hérite de Document)

- **Description** : Représente un document provenant de Wikipedia.
- **Attributs** :
  - Hérite tous les attributs de Document.
- **Méthodes** :
  - getType() : Retourne "Wikipedia".
  - __str__() : Fournit une représentation textuelle incluant le titre, l'auteur, la date et le contenu.

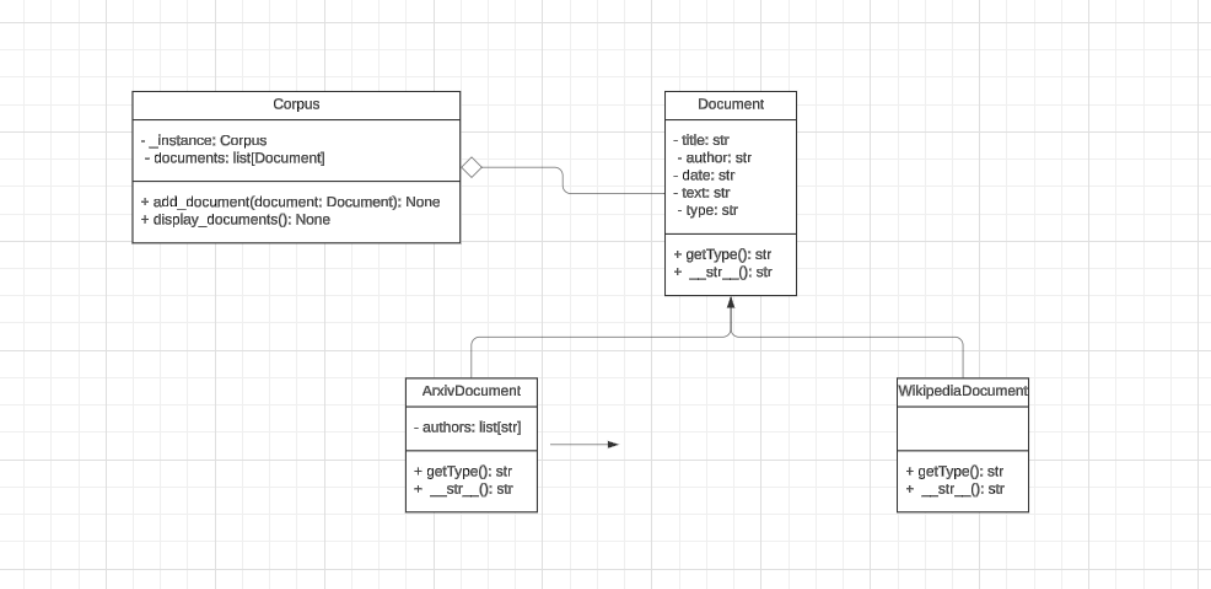

Figure 1: Diagramme de classe simplifié

Ces classes sont interconnectées de manière à assurer une communication fluide entre les différentes parties du programme.

4. Conception

Afin de répartir équitablement le travail, nous avons convenu que l'un de nous se concentrerait sur le développement du socle de base et du moteur de recherche, tandis que l'autre se chargerait de la conception de l'interface utilisateur et de l'intégration des extensions. Cette approche nous a permis de travailler efficacement et de respecter les délais impartis.

Le développement du moteur de recherche a nécessité la mise en œuvre d'un algorithme de recherche performant, capable de traiter des mots-clés et de renvoyer des résultats pertinents en un

temps raisonnable. Nous avons également conçu une interface utilisateur intuitive à l'aide de la bibliothèque tkinter. Cette interface permet à l'utilisateur de saisir des requêtes, de visualiser les résultats et d'interagir avec les différentes options disponibles.

Un exemple concret d'utilisation de notre programme est le suivant : l'utilisateur entre une requête dans l'interface, le moteur de recherche analyse cette requête en interrogeant les différentes sources de données, puis affiche les résultats sous une forme lisible et interactive.

5. Validation

Pour garantir la robustesse de notre application, nous avons effectué des tests rigoureux. Les tests unitaires ont été réalisés sur les principales méthodes de chaque classe, afin de vérifier leur bon fonctionnement de manière isolée. Par exemple, la méthode de recherche a été testée avec divers scénarios. Nous Avons aussi effectué des textes sur l'interface pour vérifier s'il est fonctionnel

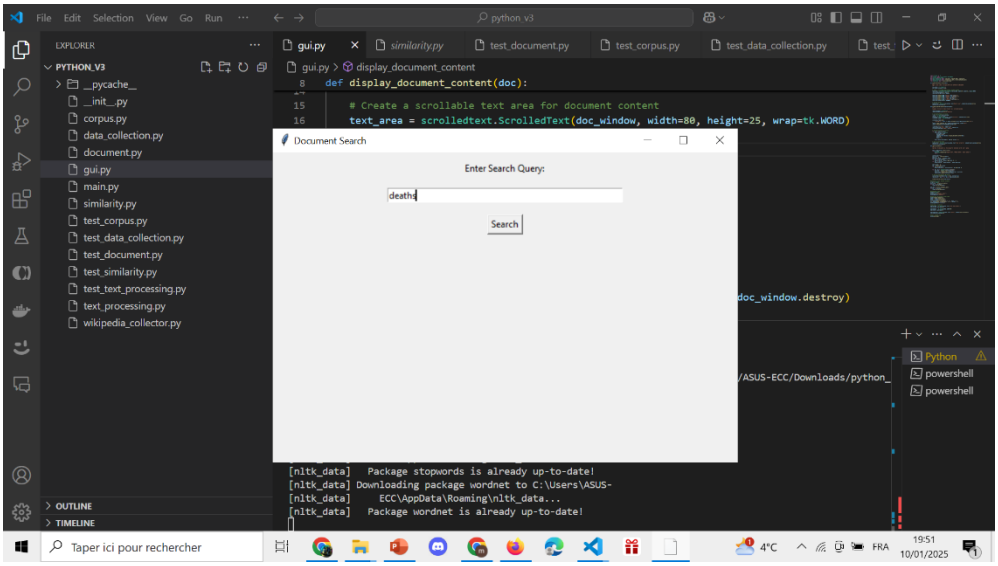

Figure 2 : Interface de recherche

RAPPORT DU PROJET EN PROGRAMMATION PYTHON

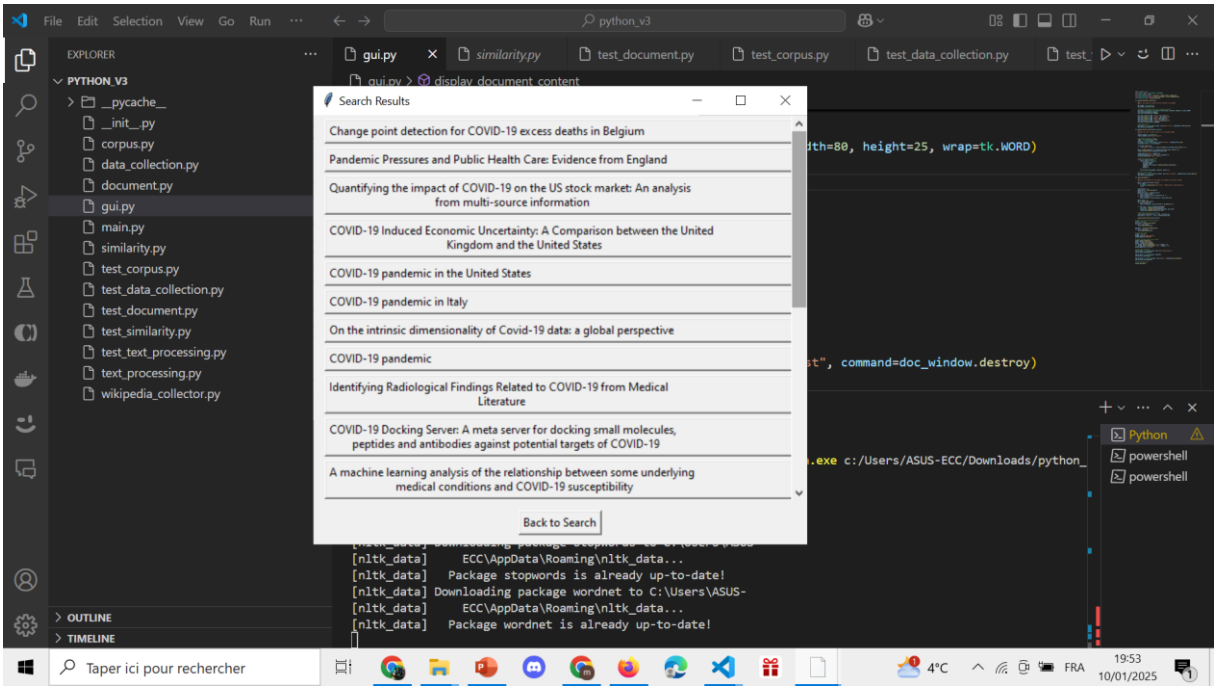

Figure 3 : Interface de résultat après recherche

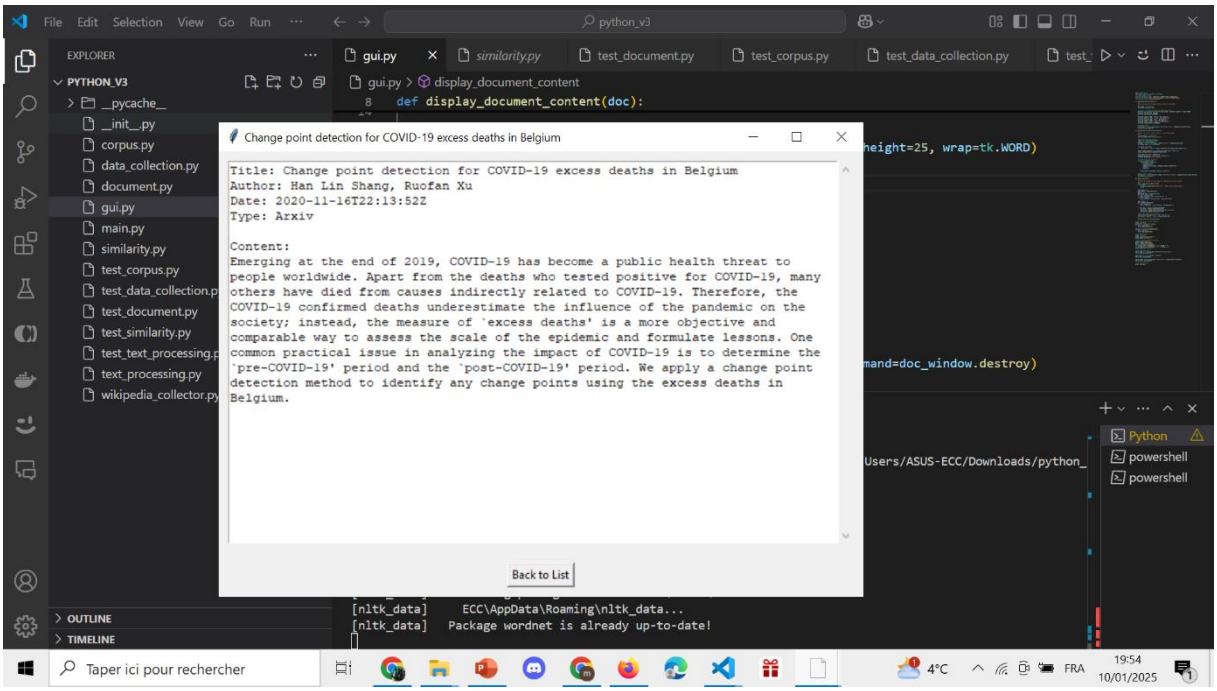

Figure 4 : Affichage du contenu d'une recherche

6. Maintenance et perspectives

Dans une optique de maintenance et d'évolution, nous avons réfléchi à plusieurs fonctionnalités qui pourraient être ajoutées à notre application. Parmi elles, nous envisageons l'intégration d'un système d'analyse de données statistiques et l'ajout de fonctionnalités de personnalisation de l'interface utilisateur. Ces évolutions seraient relativement simples à implémenter grâce à la modularité de notre architecture.

7. Conclusion

Ce projet a été une expérience enrichissante qui nous a permis de développer nos compétences techniques et organisationnelles. Malgré les défis rencontrés, nous avons réussi à produire une application fonctionnelle et bien structurée. Nous sommes fiers du travail accompli et espérons que notre projet pourra être utilisé, amélioré et adapté à d'autres contextes.

8. Annexes

Lien vers le dépôt GitHub : https://github.com/laouabdiasellami/mix2.git

Bibliothèques utilisées : Fichier requirement.txt