Programmation de spécialité (python)
TD 5 : héritage et première interface

Julien Velcin

2023-2024

Les deux types de documents que vous avez intégrés à votre corpus (*post* de Reddit et articles d'Arxiv) possèdent de nombreux points communs : un titre, un auteur, une date, etc. Ils possèdent également un certain nombre de différences. Par exemple, les articles Reddit déclenchent des commentaires et l'API permet de récupérer (entre autres) de récupérer leur nombre. Les articles Arxiv sont le plus souvent écrits par *plusieurs auteurs* et il est possible d'en récupérer la liste. Dans cette partie, on vous demande de mettre en place un mécanisme d'héritage en implémentant les deux classes filles `RedditDocument` et `ArxivDocument`. On vous conseille d'implémenter ces deux nouvelles classes dans le fichier `Document.py`, à la suite de la classe mère.

Partie 1 : classe RedditDocument

1.1 Créez une première classe fille `RedditDocument` qui hérite de `Document`. Cette classe doit intégrer au moins une nouvelle variable typique aux documents Reddit et qu'on ne retrouve pas dans les autres sources que vous manipulez. Il s'agit par exemple du nombre de commentaires postés par les internautes en réaction au message. Il faudra notamment implémenter :

- un constructeur, en pensant à appeler le constructeur de la classe mère (en utilisant si possible le mot-clef `super()`),
- des accesseurs/mutateurs pour les champs spécifiques à la classe,
- une méthode spécifique pour s'occuper de l'affichage de l'objet, via la méthode `__str__`.

Partie 2 : classe ArxivDocument

2.1 De la même manière, créez une seconde classe fille `ArxivDocument` qui intègre la gestion des co-auteurs des articles comme élément spécifique à la classe.

Partie 3 : mise à jour de la classe Corpus

3.1 Testez votre objet corpus en y ajoutant des objets des deux classes que vous venez de créer. Le polymorphisme doit vous permettre de réaliser cette opération de manière simple, sans avoir à modifier les méthodes de la classe `Corpus`.

3.2 Vous devez pouvoir afficher la liste des articles contenus dans votre Corpus avec la *source* (Reddit ou Arxiv) d'où ils proviennent. Pour cela, il existe plusieurs solutions, mais vous implémenterez celle qui consiste à ajouter un champ `type` à la classe mère `Document` et à implémenter la méthode `getType()` directement dans les classes filles, comme vu en cours.

Partie 4 : Patrons de conception

4.1 Pour le moment, on considère qu'on ne traite qu'un unique corpus. Profitez-en pour tester l'utiliser d'un patron de type `Singleton`.

4.2 Créez un générateur de documents grâce à un patron de conception d'usine (*factory pattern*).

Partie 5 : Interface notebook

5.1 Jusqu'à présent, votre code était constitué d'une collection de fichiers `.py` pilotés à partir d'un fichier principal (par ex. depuis la classe `Corpus`). Créez un carnet de note (*notebook*) qui servira de point d'entrée à votre programme.

5.2 Ajoutez un formulaire pour que l'utilisateur puisse configurer l'application : mot-clef utilisé pour récupérer les données, nombre d'articles analysés, etc. Pour cela, vous pouvez utiliser la librairie iwidgets (après l'avoir installée et activée pour jupyter), cf. : https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html