Refactorisation du code et sérialisation du corpus

Mot d'introduction

Après un changement de groupe, il vous faudra maintenant vous familiariser avec le code laissé par le groupe précédent et continuer le travail. La première étape sera donc de relire et et vérifier que les attentes des semaines précédentes sont bien remplies (à commenter dans le journal de bord).

Pour cette semaine, le nouvel objectif sera premièrement de séparer le code existant en plusieurs fichiers pour rendre le projet plus clair. Vous devrez améliorer le code du dépôt avec les suggestions présentées pendant la correction.

Il faudra ensuite ajouter des fonctions permettant de sauvegarder les données lues et filtrées dans différents formats afin de pouvoir recharger un corpus de travail plus rapidement.

Pour rappel

— un nouveau groupe gitlab vous a été attribué aléatoirement. Vous y trouverez un dépôt à cloner.
— la branche **main** sert à l'avancée du projet et des exercices, mais elle ne doit contenir que du code finalisé. il faut y pousser le moins souvent possible.
— une branche **doc** sert au rendu du journal de bord (un fichier markdown *différent* par semaine),
— chaque semaine, vous devrez créer des branches individuelles réservées au travail de chaque membre du groupe.
— un tag xxx-fin doit être utilisé pour indiquer qu'un exercice et terminé et un tag xxx-relu indiquera qu'il a été relu par un tiers et est prêt à être fusionné (**merge**). Un dernier tag indiquera que le travail sur la branche **main** est terminé.
— pour rappel, les **xxx** d'un tag seront à remplacer par **xy-sTrN**, où xy sont vos initiales, T le numéro de la séance et N votre rôle.

Exercice 1 Lecture et division du code précédent

Pour commencer, vous devrez relire et tester le code de l'équipe précédente. La première tâche consiste à apporter les modifications présentées lors de la correction qui précède ce TP. Veillez à ce que :

— le code fonctionne correctement (au moins avec **etree** et **feedparser**, et au moins une fonction de parcours d'arborescence),
— les parties non fonctionnelles ne doivent pas être proposées lors de l'appel à une fonction principale (à supprimer de l'**ArgmentParser**),
— soit séparé en 3 fichiers :

datastructures.py qui contiendra les **dataclass** utilisées dans les autres fichiers (**Corpus** et **Article**). **On écrira les noms des champs en anglais**.

rss_reader.py qui contiendra le code pour lire un unique fichier RSS. Il contiendra une fonction main pour lire un fichier depuis bash.

rss_parcours.py qui contiendra le code pour parcourir tout une arborescence de fichiers RSS et la filtrer. Il contiendra une fonction main pour lire un fichier/dossier et (dé)sérialiser les données depuis bash.

— le code utilise bien les **dataclass**

Pour ce premier exercice, vous pourrez vous répartir le travail librement. Veillez tout de même à travailler dans des branches qui en seront fusionnées vers la branche **main** qu'une fois le travail stabilisé.

Vous écrirez vos commentaires dans le journal.

Exercice 2 Nouvelles fonctionnalités

L'ajout de fonctionnalité se limitera à la possibilité de sauvegarder et recharger le corpus dans ou depuis un fichier, en proposant différent formats.

Ici vous vous répartirez la tâche en trois rôles sur trois branches comme pour les semaines précédentes, avant de fusionner vers **main** pour combiner les fonctionnalités.

r1 proposera les fonctions de sauvegarde et chargement en **XML** (on peut utiliser **etree** pour le créer)

r2 utilisera le format **json** (et le module du même nom)

r3 utilisera le format **pickle** (et le module du même nom)

Il s'agira pour chaque rôle d'ajouter une fonction d'écriture et une fonction de lecture dans le fichier **datastructures.py**. Ajoutez également une fonction `main` pour passer d'un format de sérialisation à un autre depuis bash (avec un parser d'arguments comme avant).

Les fonctions auront les prototypes suivants :

```
save_xxx(corpus: Corpus, output_file: Path) -> None
```

```
load_xxx(input_file: Path) -> Corpus
```

où **xxx** est le format de fichier (xml, json ou pickle)

Une fois votre travail terminé, ajoutez un tag xxx-fin à votre branche.

Exercice 3 Mise en production

Comme pour la semaine précédente, validez le code d'un(e) de vos camarade et ajoutez un tag **xxx-relu** quand le résultat est satisfaisant.

Finalement, fusionnez vos travaux et proposez une version finale combinant les différentes contributions sur la branche **main**.

La fonction principale et les arguments proposés avec **argparse** doivent être adaptés en conséquence pour permettre à l'utilisateur de préciser un format et un fichier de sortie où sauvegarder.

Indiquez que le travail du groupe est terminé au moyen d'un tag.

Exercice 4 Mise à jour du journal de bord

Pour ce travail, chaque membre renseigne sa partie du journal de bord, qui sera hébergé sur la branche **doc**. Commentez :

1. vos difficultés
2. vos solutions
3. les choix lors des *merges*

N'hésitez pas à ajouter quelques indications et conseils pour le groupe qui reprendra votre code la semaine suivante !