Lecture et filtrage du corpus de flux RSS

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 de la semaine précédente sont bien remplies (à commenter dans le journal de bord).

Pour cette semaine, le nouvel objectif sera d'utiliser la fonction existante de lecture d'un (unique) fichier RSS pour charger tout le corpus (en appelant la fonction un grand nombre de fois).

Il faudra aussi ajouter des fonctions permettant de filtrer les données à charger en fonction de la date, de la source ou de la catégorie des articles.

attention, un même article ne doit être affiché qu'une seule fois, même si il est présent dans plusieurs fichiers RSS.

Pour rappel

- un nouveau groupe gitlab vous a été attribué aléatoirement pour la semaine. 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.
- 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 du code précédent

Pour commencer, vous devrez relire et tester le code de l'équipe précédente. Chacun lira et testera la fonction principale (**main**), puis en fonction de votre rôle de la semaine précédente :

- Si vous étiez **r1** vous relirez et commenterez le travail de **r2**
- Si vous étiez **r2** vous relirez et commenterez le travail de **r3**
- Si vous étiez **r3** vous relirez et commenterez le travail de **r1**

Vous écrirez vos commentaires dans le journal.

Exercice 2 Nouvelles fonctionnalités

Pour cette semaine, chacun devra proposer un moyen de lire l'ensemble de l'arborescence du corpus, ainsi qu'une des trois fonctions de filtrage.

En fin de semaine, vous combinerez vos travaux (par des **merge**) afin de conserver une des façons de lire l'arborescence et **les trois** filtres.

r1 utilisera le module **os** [1] (notamment `os.listdir` et `os.path`, et proposera l'option de filtrer en fonction de la date (les articles parus depuis une date et/ou jusqu'à une date).

r2 utilisera le module **pathlib** [2] (notamment son objet `Path`, mais sans utiliser la fonction **glob()**) et proposera une fonction de filtrage en fonction de la ou des sources (noms des journaux/sites comme BFM, Libération, Blast...)

r3 utilisera le module **pathlib** et sa fonction **glob()** et proposera une fonction de filtrage acceptant une ou plusieurs catégories indiquées dans les balises `category` des fichiers XML.

La fonction principale et les arguments proposés avec **argparse** doivent être adaptés en conséquence pour permettre à l'utilisateur de préciser une date de début, une date de fin, une source et des catégories.

1. `https://docs.python.org/fr/3/library/os.html`
2. `https://docs.python.org/fr/3/library/pathlib.html`

Quelques consignes générales :
Pour faciliter le travail et les **merge**s, on modifiera la fonction de lecture d'un document XML pour que celle-ci retourne une liste de dictionnaires (→ `list[dict]`). Chaque dictionnaire représentant un article (`item` du XML) devra inclure les clefs suivantes :

— titre (type **str**, le contenu texte de la balise **title)**
— description (type **str**, le contenu texte de la balise **description)**
— date (type **str** ou **datetime**[3])
— categories (type **list[str]**, une liste de catégories auxquelles appartient l'article)

Pour faciliter le filtrage, vous pouvez commencer par définir une fonction dont la déclaration ressemblerait à :
`filtre_xxx(item: dict, ...) → bool:`
qui retourne **True** si l'article *item* doit être conservé et **False** sinon.
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**.

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 !

3. `https://docs.python.org/fr/3/library/datetime.html`