Enrichissements du corpus avec des analyseurs morphosyntaxiques

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). Il faut vérifier que le code de départ est capable de lire et filtrer l'arborescence de fichiers RSS et de sauvegarder le corpus filtré sur le disque.

Pour cette semaine, le nouvel objectif sera d'enrichir le corpus avec les sorties de différents analyseurs morphosyntaxiques. Il faudra commencer par prendre en main les différents outils (ne pas hésiter à consulter leurs documentations !)

Le code obtenu en fin de semaine devra pouvoir recharger un corpus, ajouter les analyses d'un des trois analyseurs proposés dans un format unifié et de sauvegarder le résultat de l'analyse.

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. 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 Prise en main d'outils

La première étape sera donc de prendre en main différents outils d'analyses. Vous devez avoir déjà installé **spacy**, **stanza** et **trankit**.

Chaque membre du groupe travaillera de façon autonome à prendre en main un des trois outils.

Vous pouvez consulter leurs documentations respectives :

spacy `https://spacy.io/`

stanza `https://stanfordnlp.github.io/stanza/`

trankit `https://trankit.readthedocs.io/en/latest/`

Vous devez pouvoir lancer l'analyse d'un petit texte et expliquer comment récupérer pour chaque **token** du texte :

— sa forme,
— son lemme,
— sa partie du discours (*part of speech*, POS, ou catégorie grammaticale).

Notez vos observations dans le journal.

Exercice 2 Script de démonstration indépendant

Dans un premier temps, vous pouvez écrire un petit script de démonstration de l'outil, **demo_xxx.py** (où xxx est spacy, stanza ou trankit). Ce script vous servira à appréhender les interfaces des outils (méthodes, structures de données, logique générale de l'utilisation).

Exercice 3 Intégration au projet

Enfin, dans un nouveau fichier **analyzers.py**, écrivez une fonction qui prend en argument un **Item** et le retourne, enrichi avec le résultat de l'analyse. Chaque analyseur pourra avoir sa propre fonction d'analyse.

Vous pourrez avoir besoin d'écrire des fonctions annexes (par exemple pour charger le modèle), et de modifier le fichier **datastructures.py**, en particulier de modifier et d'ajouter des **dataclasses**. On attend notamment

une *dataclass* `Token` qui servira d'interface commune pour stocker les résultats différents analyseurs (chaque *token* ne stocke qu'une analyse, mais on doit pouvoir utiliser le type `Token` avec chaque outil).

Proposez une fonction principale (**main**) dans le fichier **analyzers.py** afin que celui-ci puisse être utilisé comme une **commande bash** qui charge un corpus précédemment sauvegardé, l'analyse avec votre outils et sauvegarde le résultat. Les fonctions de (dé)sérialisation devront être mises à jour pour intégrer les analyses (les lire ou les écrire quand elles sont présentes).

Exercice 4 Mise en production

Comme pour les semaines précédentes, 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 choisir un des trois outils d'analyse.

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

Exercice 5 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 !