Faire interagir bash et python

Mot d'introduction

Le premier objectif est de pratique le travail sur plusieurs *branches* avec `git`. Chaque membre d'un groupe s'occupera d'**une des trois** options de l'exercice **sur sa propre branche**, et il vous faudra ensuite fusionner vos différentes contributions afin que la branche *main* contienne un script **finalisé** qui combine l'ensemble des contributions.

Le second objectif de cette feuille de TP est de mettre en place des ponts entre bash et python. Il s'agit d'appliquer les méthodes vues en cours pour qu'un programme python se comporte comme une nouvelle commande bash et s'intègre correctement dans une *pipeline*.

Pour rappel

- chacun doit avoir un compte gitlab avec une clef publique bien configurée
- un groupe vous a été attribué aléatoirement. Vous y trouverez un projet correspondant au TP2.
  - la branche **main** sert à l'avancée du projet et des exercices, mais elle ne doit contenir que du code finalisé.
  - une branche **page** sert au rendu du journal de bord (un fichier markdown par semaine et du projet final (html),
  - chaque semaine, vous devrez créer des branches individuelles réservées au travail de chaque membre du groupe.
  - des tags doivent être utilisés pour indiquer qu'un exercice et terminé et qu'une branche est prêtre à être fusionnée (**merge**) ou que le travail sur la branche **main** est terminé.
- Attention, pensez à pousser (**push**) toutes les **branches** et tous les **tags** que vous créez sur le **remote**. Pour poussez les branches que vous allez créer, utilisez la commande `git push --set-upstream origin <name>`, qui définit la branche distante **origin** comme branche amont, et où <name> est le nom de votre branche courante. Pour push les tags, utilisez la commande `git push origin -tags`.

Exercice 1 Mise en place des données initiales

1. Dans une **branche individuelle**, chaque membre doit créer un dossier `./Corpus` dans lequel nous travaillerons aujourd'hui
2. Dans le sous-dossier `./Corpus`, ajoutez 10 fichiers textes issus de vos fichiers textes (segmentés si nécessaires) du projet de S1. Faites un **commit** de ces fichiers et un **push** de ce commit
3. Fusionnez (**git merge**) vos contributions de manière à ce que la branche **main** contienne les 30 fichiers textes issus des différentes branches
4. Une fois le corpus constitué, ajoutez un tag au dernier commit de la branche **main** nommé **s2ex1fin**

Exercice 2 Extraction du lexique en python

Pour cet exercice, chaque membre doit écrire une partie différente du programme. Répartissez-vous les rôles entre r1, r2 et r3.

1. depuis la branche **main** contenant toutes les données, créez une nouvelle branche pour chacun et l'activer. Créez un fichier `extraire_lexique.py` dans lequel
   - r1 sera chargé d'écrire une fonction qui construit une liste de chaînes (List[str]), où chaque chaîne correspondra au contenu texte d'un fichier du dossier `./Corpus`.
   - r2 sera chargé d'écrire une fonction prenant comme argument une liste de chaînes et retournant un **dictionnaire** associant chaque mot à son nombre d'occurrences dans le corpus.
   - r3 sera chargé d'écrire une fonction prenant comme argument une liste de chaînes et retournant un **dictionnaire** associant chaque mot au nombre de documents dans lequel il apparaît.
2. Après avoir testé votre fonction, ajoutez un tag **s2ex2r1 s2ex2r2** ou **s2ex2r3** à votre branche.
3. Fusionnez (**merge**) les branches des autres membres vers la vôtre et écrivez une fonction qui combine les trois précédentes pour afficher le lexique du corpus sur trois colonnes : 1) le mot 2) le nombre d'occurrences total 3) le nombre de documents où il apparaît.
4. Après avoir testé sur votre branche, comparez vos trois solutions et choisissez-en une à fusionner dans la branche **main**. Ajoutez un tag **s2ex2fin**

Exercice 3 Lecture des fichiers en bash

Pour cet exercice, chaque membre doit écrire une version différente de la fonction permettant la lecture des fichiers par le programme réalisé précédemment.

Contrairement à l'exercice 2, ici le programme doit laisser à l'utilisateur la possibilité de définir le corpus.

Répartissez-vous les rôles entre r1, r2 et r3.

1. Chaque rôle doit permettre de lancer votre programme dans des *pipelines* similaires à celles présentées ci-dessous.
   - r1 : permettre la lecture d'un corpus comme une liste de fichiers en arguments, en tapant par exemple :
     `python extraire_lexique.py Corpus/*.txt`
   - r2 : permettre la lecture du corpus depuis l'entrée standard, en donnant le contenu d'un document sur chaque ligne (Attention : Cette façon de faire suppose que vos fichiers textes ne contiennent pas de retour à la ligne)
     `cat Corpus/*.txt | python extraire_lexique.py`
   - r3 : permettre de lister les chemins vers les fichiers du corpus sur l'entrée standard du programme en python :
     `ls Corpus/*.txt | python extraire_lexique.py`
2. Après avoir testé votre fonction, ajoutez un tag **s2e3rN** à votre branche, où rN est votre rôle
3. Fusionnez (**merge**) les branches des autres membres vers la vôtre et écrivez une fonction qui combine les trois précédentes options développées indépendamment.

   Le programme finalisé devra permettre à l'utilisateur de choisir une des trois façons de fonctionner au moyen d'options lors du lancement.
4. Comparez vos solutions, fusionnez l'une d'elles dans la branche **main** et ajoutez un tag **s3ex3fin**

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* vers *main*