Git

Pour ce projet j'ai surtout utilisé les scripts que j'avais écrits pour mon projet de mémoire, que j'ai ensuite adaptés pour en créer un projet qui pourrait être utilisé avec d'autres jeux de données. Il s'agissait surtout de redéfinir les paramètres de certaines fonctions et de créer le fichier main.py qui réunit toutes les fonctions et fournit un workflow cohérent. Je faisais des commits à chaque fois que j'ai fait un changement important (« make column names clearer », « normalise variable names across the scripts », « add tests », « define workflow in main.py »), corrigé un bug (« add missing orthography and pronunciation distance scores »), ou ajouté une vérification (« verify that the Alsatian word is not null before calculating pronunciation »). Certains changements ont été implémentés dans une branche (« clean_Alsatian »), comme l'ajout de la fonction *clean()* et la vérification que le mot alsacien est présent avant de calculer le score de similarité (afin d'éviter d'avoir les scores 0.0 là où le mot alsacien n'a pas été assigné). Il n'y a pas eu de conflits lors du merge des deux branches (fast-forward merge).

Tests

L'écriture de tests m'a permis d'améliorer mon code initial car j'ai pensé à certains cas que je n'avais pas pris en compte auparavant. Il s'est agi notamment d'ajouter des vérifications du type des arguments donnés en entrée, ainsi que de la longueur de l'entrée pour assurer que la chaine de caractères d'entrée n'est pas vide (j'ai ajouté des ValueError qui s'affichent au cas d'un type incorrect passé en entrée). De plus, en pensant aux cas particuliers, j'ai ajouté des vérifications additionnelles dans la fonction *tr_retrieve()* afin d'éviter que la fonction considère des chaines de caractères contenant des signes de ponctuation (donc les chaines qui ne sont pas des mots) comme des mots valides. Une exception à cette règle est la présence du tiret, qui peut bien se trouver dans des mots ; pour cette raison j'ai ajouté un test supplémentaire qui vérifie que les mots avec un tiret ne sont pas exclus. Finalement, j'ai automatisé les tests en utilisant GitHub Actions.

Documentation et type hints

J'ai utilisé l'outil IA intégré à PyCharm (JetBrains) pour générer les docstrings et les type hints. J'ai trouvé que la qualité des résultats est en général très bonne : les descriptions étaient pertinentes et spécifiques à mon projet, et il y avait très peu d'erreurs. L'exemple d'une erreur que l'outil a générée est d'assigner le type « dictionnaire » au paramètre *tr* dans la fonction *tr_retrieve()*, alors qu'il s'agit d'une liste (c'était avant que j'ai ajouté des vérifications du type dans la fonction), ou d'assigner les type hints de la fonction *orth_score()* comme « def orth_score(x: str, y: str) -> float », alors que le type de sortie peut également être *str*.

J'ai légèrement modifié certaines descriptions afin de les rendre un peu plus claires et liées les unes aux autres. Par exemple, dans le docstring de la fonction *clean()* j'ai ajouté des précisions entre parenthèses : « param w : Input string (series of Alsatian words) containing irrelevant substrings (codes indicating sources of the words)… ». La génération des docstrings et type hints est la seule partie pour laquelle j'ai utilisé l'IA. J'ai déployé la documentation via GitHub Actions.

Licence

Comme l’un des objectifs principaux de mon projet de mémoire est de contribuer à la préservation de l’alsacien et au développement de ses ressources digitales, j’ai choisi la licence la plus permissive (MIT) afin de faciliter tout futur travail potentiel.