Le projet présente une application Streamlit simple. Il s'agit d'un moteur de recherche pour les articles scientifiques en traitement automatique des langues. La partie déployée ne concerne que l'application et les fichiers nécessaires à son fonctionnement en local (fichiers de scores des différents modèles). Le fichier dans lequel sont calculés les différents scores selon les modèles et les requêtes, et à partir duquel les fichiers de scores sont générés, n'est pas l'objet central. Néanmoins, il est présent sur le repo à titre d'information.

Organisation des commits

Les commits ont été réalisés de manière régulière, suivant les différentes étapes d'avancement du projet. Au fur et à mesure que des modifications étaient effectuées, de nouveaux commits les suivaient. Nous avons créé une branche (theme) contenant des modifications sur l'application (appy.py), notamment relatives aux thèmes (couleurs) et à l'ajout d'une sidebar. Le merge avec la branche principale s'est effectué sans problème. C'est toutefois lors du commit concernant les triples guillemets, nécessaires à la génération des docstrings, que nous avons rencontré un conflit, car des modifications différentes avaient été faites sur la version en ligne et en local. Nous avons résolu le conflit en conservant les modifications que nous avions effectuées sur la version en ligne.

Tests

Nous avons créé des tests afin de vérifier la conformité entre ce que l'utilisateur saisit en entrée et le format attendu. Pour cela, nous avons créé une fonction dans un fichier à part (validation.py) sur laquelle nous avons effectué des tests. Nous avons procédé ainsi car les tests portent sur un champ d'entrée Streamlit (st.text_input), sur lequel il est difficile d'effectuer des tests directement, puisqu'il ne s'agit pas d'une fonction. L'entrée de l'utilisateur (via Streamlit) est passée par cette fonction et, si elle est valide, le résultat de la requête est affiché. La fonction a ensuite été importée dans le fichier de tests.

Le test porte notamment sur le nom du modèle de recherche d'information que l'utilisateur souhaite utiliser (soit BM25, Modèle dense ou Modèle hybride). Dans la fonction créée dans validation.py, le type de données à entrer doit être une chaîne de caractères. Ainsi, dans les tests, nous vérifions si l'entrée est un nombre. Si c'est le cas, le test retourne False. Nous testons également avec des réponses correctes mais dont la casse n'est pas respectée (bm25, modèle dense), qui sont toutes classées comme False. Les tests sont effectués avec Pytest.

Licence

En nous inspirant du site https://choosealicense.com/, nous avons choisi la licence MIT. Nous l'avons choisie car il s'agit d'un petit projet en TAL qui, d'après nous, peut parfaitement être distribué gratuitement. En outre, cette licence permet un accès libre à toute personne désireuse de modifier ou d'apporter une contribution au projet. Cela laisse la voie libre à d'éventuelles améliorations de cette idée, par nous ou par d'autres à l'avenir.

Utilisation de l'IA

Nous avons utilisé ChatGPT à différentes étapes du projet. Nous l'avons notamment employé pour interpréter des bugs lors de l'utilisation de GitHub Actions pour générer automatiquement

la documentation avec pdoc. Par exemple, nous l'avons utilisé pour corriger le fichier pdoc.yml sur la base duquel la doc a été générée par GitHub Actions. En effet, après plusieurs difficultés pour générer automatiquement la doc, c'est grâce à l'aide de l'IA concernant les instructions d'installation de Python et de pdoc, notamment en ce qui concerne les indentations et la commande pour générer la doc (run: pdoc ./appy.py --output-dir docs/), que nous avons pu générer la page HTML.

Nous avons également utilisé l'IA pour obtenir des explications sur comment effectuer des merges et sur l'arborescence des fichiers (dans quels dossiers créer différents fichiers pour faire fonctionner le code). Enfin, nous avons utilisé l'IA pour la relecture et la correction de ce fichier de justification, en lui précisant de ne corriger que les fautes d'orthographe et les phrases ambiguës.