Fiche Documentaire du Code : *Segmented*

Titre du projet :

Analyse personnalisée des articles scientifiques (à l'aide des LLM et d'un paradigme RAG)

Auteurs :

- **Benlahcen Souad** : Responsable de la division des données en paragraphes

Description du code :

Ce code a pour objectif de segmenter automatiquement les articles scientifiques collectés en paragraphes en divisant les articles en sections et sous-sections pertinentes. Ce processus est essentiel pour améliorer l'extraction d'informations et la génération de réponses personnalisées en utilisant un paradigme de *Retrieval-Augmented Generation* (RAG).

Structure du code :

- **Segmented_files.ipynb** : Le notebook principal qui effectue la segmentation des fichiers texte en paragraphes et sections, en utilisant des techniques de traitement automatique du langage naturel (NLP) et des embeddings.

Structure des fichiers générés :

- **new_segmented_files/** : Dossier contenant les fichiers texte des articles segmentés après le traitement. Chaque fichier porte le nom de l'article et est suivi de l'extension _segmented.txt.

Instructions d'installation :

1. **Installer les bibliothèques nécessaires :** Les bibliothèques requises incluent *Spacy*, *tiktoken* et *Google Drive API*.

```
pip install spacy tiktoken gspread oauth2client
python -m spacy download en_core_web_sm
```

2. **Accéder à Google Drive** : Monter Google Drive pour accéder aux fichiers stockés, puis utiliser l'API Google Drive pour automatiser le traitement des fichiers.

Prérequis :

- **Python 3.10**
- **Spacy**, **tiktoken**, et **Google Colab**.
- **Clé API Google Drive** pour accéder aux fichiers.

Fonctionnement du code :

1. **Chargement des données** : Le code commence par charger les articles scientifiques enregistrés dans Google Drive via un fichier Excel qui contient les liens vers les fichiers texte des articles collectés.
2. **Segmentation automatique** : Chaque article est segmenté en sections et paragraphes, en se basant sur la structure du texte

- Les fonctions comme is_section_digit() et is_section_letter() permettent de détecter automatiquement les sections et sous-sections dans le texte, en identifiant les titres et en divisant le texte en segments cohérents.

3. **Sauvegarde des fichiers segmentés** : Les paragraphes segmentés sont ensuite sauvegardés dans un répertoire spécifique sous forme de fichiers texte.

Entrées et sorties :

Entrées :

- **Fichiers texte non segmentés** des articles scientifiques, récupérés via Google Drive.
- **Fichier Excel** contenant les liens vers les fichiers texte dans Google Drive.

Sorties :

- **Fichiers texte segmentés** : Chaque fichier contient les sections et paragraphes des articles segmentés.

Exemples d'utilisation :

Pour exécuter le code de segmentation, voici un extrait pour lancer le traitement de segmentation d'un fichier :

```
# Processus de traitement du texte avec structure par section et division de paragraphes longs
def process_text_file(file_path, title, token_threshold=250, max_tokens=300, min_tokens=200, model="gpt-4"):
    content = load_file(file_path)  # Charger le fichier
    paragraphs = structure_text(content)  # Structurer le texte en sections et paragraphes
    save_file(paragraphs, title)  # Sauvegarder le contenu segmenté
```

Tests et validation :

- Des tests ont été réalisés pour vérifier la qualité de la segmentation.
- La validation manuelle des fichiers segmentés a été effectuée pour garantir la précision de la segmentation.

Problèmes connus :

- Les articles dont la structure est très complexe peuvent présenter des erreurs mineures lors de la segmentation, comme des divisions inappropriées des paragraphes.
- Certains fichiers peuvent contenir des caractères ou des formats spéciaux qui compliquent l'analyse automatique du texte.