Documentation Technique : Moteur de Recherche en Genesis en Python avec Flask

Introduction :

Ce document technique décrit la conception et l'implémentation d'un moteur de recherche web en utilisant Python et Flask. Le moteur de recherche sera capable de crawler des pages web, d'extraire leur contenu, d'indexer les informations pertinentes et de fournir des résultats de recherche à travers une interface web.

Fonctionnement :

Crawler

Indexation + CSV

Recherche

Interface Web

Aperçu :

Ce projet a été réalisé en Python avec Flask. Voici les dépendances utilisées et les raisons :

- **Flask : Interface web et création d'API pour communiquer avec les fonctions Python**
- **Queue : (Sans installation) Nous permet de faire un système de first in first out pour la distribution des URLs et du contenu aux autres fonctions**
- **Requests : Essentiel pour pouvoir accéder au contenu d'une page internet**
- **BeautifulSoup (BS4) : Nous permet de scrapper des balises et des éléments du HTML, essentiel pour indexer et récupérer les nouveaux liens**
- **urlJoin : Essentiel pour vérifier si les liens sont valides avant de les ajouter à la file d'attente**
- **csv : Communiquer avec notre CSV qui nous sert de base de données**
- **Json : Interagir avec le JSON**

Comment ça marche ?

Rendez-vous dans le readme.md pour connaître les étapes du projet.

Initialisations :

Créations des files d'attentes :

urls_to_crawl = queue.Queue()
content_to_add = queue.Queue()
links_to_add = queue.Queue()

keywords_to_add = queue.Queue()

Chacune est utilisée pour distribuer les informations à d'autres fonctions. Lorsque l'on lance le fichier, on ajoute une URL de départ à commencer à crawler de notre choix.

Crawler :
Le crawler s'exécute indéfiniment tant que la file d'attente n'est pas vide.

Il récupère une URL à crawler de la file d'attente urls_to_crawl. Puis, il accède à cette URL, récupère les h1, h2, h3 et le titre, puis les ajoute dans la file d'attente keywords_to_add. Ensuite, il récupère du HTML toutes les balises à vérifier si elles sont valides, puis les rajoute à la file d'attente links_to_add qui va vérifier si elles sont déjà dans la file d'attente ou non.

Indexation + CSV :

La fonction add_to_csv sert à ajouter les informations extraites des pages web dans un fichier CSV. Elle fonctionne indéfiniment en attendant que de nouvelles données soient disponibles dans la file d'attente content_to_add.

- **Récupération des données : La fonction récupère continuellement les informations sur une page web, notamment son URL, son titre et sa description depuis la file d'attente content_to_add.**
- **Ajout au fichier CSV : Les informations extraites sont ajoutées au fichier CSV webpages.csv avec l'URL, le titre et la description de la page.**

Boucle d'exécution continue : La fonction s'exécute en boucle, permettant ainsi de traiter les données en continu tant qu'il y en a de disponibles.

- **La fonction update_keywords_csv est responsable de la mise à jour du fichier CSV contenant les mots-clés et leurs fréquences associées.**
- **Récupération des données : La fonction récupère les données sur les mots-clés et leurs fréquences associées pour une URL donnée depuis la file d'attente keywords_to_add.**
- **Mise à jour du fichier CSV : Elle met à jour le fichier CSV keywords.csv en ajoutant ou en mettant à jour les données des mots-clés avec les nouvelles informations extraites.**

- **Boucle d'exécution continue : Comme la fonction add_to_csv, cette fonction s'exécute en boucle pour traiter les données de manière continue.**
- **Gestion des exceptions : La fonction gère les exceptions liées à la lecture ou à l'écriture du fichier CSV, assurant ainsi la robustesse du processus même en cas d'erreur.**

Recherche :

La fonction search_keywords permet de rechercher les sites internet les plus pertinents en se basant sur la fréquence d'apparition des mots-clés dans les pages web indexées.

- **Normalisation de la Requête : La requête de recherche est normalisée en enlevant les accents, en convertissant en minuscule et en excluant les mots-clés inutiles ou "stop words". Cela permet d'obtenir une liste de mots-clés pertinents pour la recherche.**
- **Recherche dans les Mots-Clés : La fonction parcourt le fichier CSV keywords.csv, qui contient les mots-clés et leurs fréquences associées pour chaque URL. Elle cherche les mots-clés correspondant à ceux de la requête normalisée et calcule un score de pertinence pour chaque URL en additionnant les fréquences des mots-clés correspondants.**
- **Tri des Résultats : Les URLs sont triées en fonction de leur score de pertinence, avec les URLs ayant les scores les plus élevés en tête de liste. Seuls les 5 meilleurs résultats sont conservés.**
- **Extraction des Informations : Les informations des 5 meilleurs résultats sont extraites du fichier CSV webpages.csv, qui contient les URLs, titres et descriptions des pages web indexées. Les informations sont récupérées pour chaque URL trouvée dans les résultats de la recherche.**
- **Retour des Résultats : Les résultats sont retournés sous forme d'une liste d'objets, chaque objet contenant l'URL, le titre et la description d'une page web pertinente.**

1. Interface Web

Pour notre interface web, le front-end est constitué simplement de HTML, CSS et JS.

Pour pouvoir accéder aux résultats de notre fonction de recherche et connecter notre HTML à Python, nous avons opté pour Flask. Nous avons créé 3 API.

La première est /keywording qui reçoit le mot-clé et renvoie les résultats de recherche en fonction du mot-clé.

Pour la deuxième, c’est /crawler qui permet de simuler un crawler. On lance un crawler avec le choix de l'URL de départ.

Enfin, la route /clientkeywording permet de faire une recherche en fonction des données que l'utilisateur a collectées grâce à son crawler.