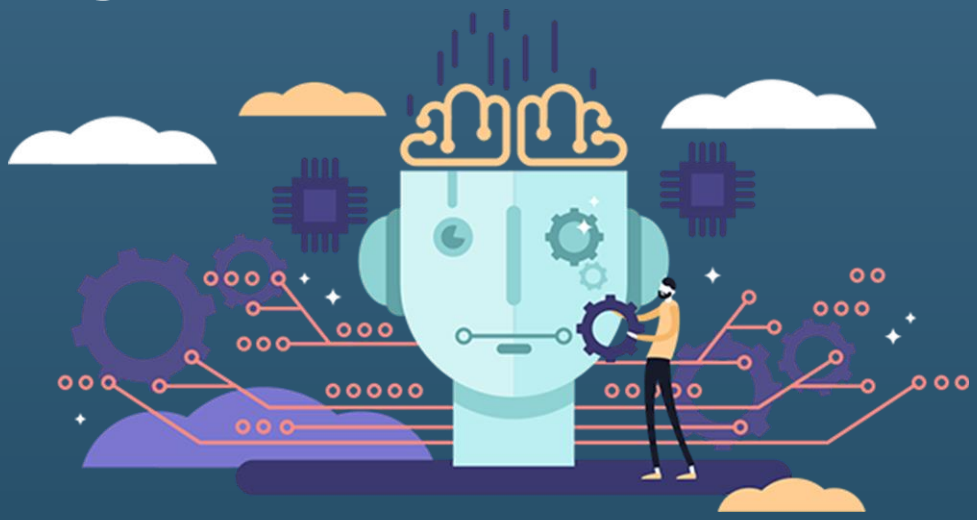
2022-2023

Université Sorbonne Paris Nord

SAE 3.01 – Base de données de films

Algorithme de recherche



TEAM DRACORPORATION

Tamijanebane SARAVANAN

L’algorithme de recherche, n’était pas très simple a codé, en effet l’implémentation d'un algorithme de recherche nécessite une réflexion approfondie sur la manière de gérer les différents cas d'utilisation, les fautes de frappe et les erreurs d'orthographe. Il est important de considérer les différents cas d'utilisation pour lesquels l'algorithme sera utilisé, comme les recherches par nom, par titre, par date, etc. Il peut également être nécessaire de prendre en compte les différentes langues et les caractères accentués pour garantir une recherche fiable et précise. Il est également important de considérer les interactions utilisateur pour rendre l'expérience de recherche la plus fluide et intuitive possible et aussi de considérer tous les aspects de l'algorithme de recherche pour garantir une implémentation efficace et fiable pour l'interaction homme-machine.

J'ai implémenté cet algorithme de recherche en utilisant les langages PHP et SQL. Les requêtes SQL ont été utilisées pour interroger la base de données et récupérer les résultats de recherche correspondants. Le code PHP a ensuite été utilisé pour traiter les données, gérer les cas d'utilisation, les erreurs d'orthographe, et pour rendre l'expérience utilisateur la plus fluide possible. En utilisant ces deux langages en conjonction, j'ai pu créer un algorithme de recherche.

J'ai créé deux codes pour l'algorithme de recherche : un pour la recherche simple basée sur la similarité et les accents, et un autre pour la recherche avancée qui permet de sélectionner des résultats en fonction de différents critères.

Le premier code (recherche simple) utilise la fonctionnalité de similarité de SQL pour trouver des résultats similaires à la requête de l'utilisateur, on a une méthode "search" qui prend en entrée un argument de recherche, et utilise cet argument pour exécuter deux requêtes SQL distinctes. La première requête (variable $queryFilm) utilise la fonction de similarité de PostgreSQL pour rechercher des films dans la table "titlebasics" où le titre ressemble à l'argument de recherche avec un score de similarité supérieur à 0,4. La seconde requête (variable $queryPerso) fait la même chose pour les personnes dans la table "namebasics".
Les résultats de ces requêtes sont ensuite stockés dans des variables de classe appelées $films et $personnes. La méthode getResult() est ensuite utilisée pour retourner un tableau associatif contenant les résultats des films et des personnes. Il y a également un calcul du temps d'exécution entre le moment où les requêtes sont exécutées et le moment où les résultats sont retournés. Cela permet de mesurer combien de temps prend l'exécution de l'algorithme de recherche.

Pour mettre en place l'algorithme de recherche, j'ai utilisé plusieurs extensions et fonction pour PHP et SQL :

Extension / Fonction	Utilités
PDO	La classe utilise l'extension PDO (PHP Data Objects) pour se connecter à la base de données PostgreSQL et exécuter des requêtes SQL. PDO est une extension de PHP qui permet de se connecter à différents types de bases de données à l'aide d’une seule et même syntaxe. Cela facilite la maintenance et la portabilité du code.
SIMILARITY	La fonction de similarité de PostgreSQL (SIMILARITY) permet de comparer deux chaînes de caractères et de renvoyer un score de similarité compris entre 0 et 1. Il est utilisé ici pour trouver les titres et les noms qui ressemblent le plus à l'argument de recherche donné.

Microtime(true)	"microtime()" qui est une fonction de PHP qui renvoie le nombre de secondes écoulées, avec une précision de microsecondes. Il est utilisé ici pour calculer le temps d'exécution de l'algorithme de recherche.
UNCCENT	La fonction "unaccent" est une extension de PostgreSQL qui permet de supprimer les accents et les diacritiques d'une chaîne de caractères. Elle est utilisée ici pour garantir que les recherches de similarité fonctionnent correctement, même si l'argument de recherche ou les données de la base de données contiennent des accents ou des diacritiques

Le deuxième code (recherche avancée) permet de créer une page web qui permet à l'utilisateur de rechercher des films et des personnes dans une base de données en utilisant une requête SQL. Le code est divisé en deux parties principales : une partie HTML pour l'interface utilisateur, et une partie PHP pour la logique de recherche.

La première partie du code est un formulaire HTML qui permet à l'utilisateur de saisir le nom de la personne ou du film qu'il recherche, ainsi que des options pour trier les résultats par nom ou date de naissance, et pour limiter ou décaler les résultats retournés. Lorsque l'utilisateur soumet le formulaire, les données sont envoyées à un script PHP appelé "search.php" pour traitement. La seconde partie du code est écrite en PHP. Il inclut deux fichiers importants : "config.php" et "FilmModel.php". Le fichier "config.php" contient les informations de connexion à la base de données, comme le nom d'utilisateur, le mot de passe et l'URL. Le fichier "FilmModel.php" contient la classe "FilmModel" qui est utilisée pour effectuer la recherche.

Lorsque le formulaire est soumis, le script PHP valide les données envoyées par l'utilisateur, puis crée une instance de la classe "FilmModel" en utilisant les informations de connexion à la base de données. Il appelle ensuite la méthode "search" de cette classe en lui passant les données du formulaire. La méthode "search" construit une requête SQL en utilisant les informations fournies par l'utilisateur, puis l'exécute en utilisant les fonctionnalités de la classe PDO (PHP Data Object) pour se connecter à la base de données. La requête SQL est composée de deux requêtes distinctes une pour les films et une pour les personnes.

La requête pour les films utilise la fonction "similarity" pour trouver les films dont le titre est similaire au nom saisi par l'utilisateur, la requête pour les personnes utilise la fonction "similarity" pour trouver les personnes dont le nom est similaire au nom saisi par l'utilisateur. Et pour les personnes la requête utilise aussi une clause WHERE pour filtrer les personnes selon la date de naissance.

Enfin, les résultats de la requête sont stockés dans des variables de classe, et la méthode "getResult" est utilisée pour retourner les résultats sous la forme d'un tableau associatif contenant les films et les personnes trouvés, le code HTML affiche les résultats de la recherche en utilisant des boucles pour parcourir les tableaux de résultats retournés par la méthode "getResult" et afficher chaque film ou personne trouvé sous forme de liste.

Le seul problème que j'ai rencontré lors de la mise en place de cet algorithme de recherche était le temps d'exécution, en particulier lorsque la base de données est volumineuse et que le nombre de résultats de recherche est élevé. J'ai réfléchi a une solution pour améliorer les performances de l'algorithme de recherche et réduire le temps d'exécution pourrait consister à utiliser AJAX.