

Rapport Projet DAAR

Sujet A

Réalisé par :

- Salah Eddine EL MAMOUNI
- Hamza HAJJOU
- Hugo MENDES

1. Définition du problème	3
a. Contexte	
b. Objectifs	
c. Contraintes	
d. Enjeux	
2. Collecte et parsing des données	4
3. Structures des données utilisées (DAO	4
4. Analyse et présentation des Algorithmes utilisées.....	6
a. KMP	
b. Aho Ullman	
c. Jaccard	
5. Réalisation	9
a. Vue d'ensemble de l'application	
b. Architecture	
c. Pré-requis et démarrage de l'application	
6. Conclusion et perspectives	11

1. Définition du problème

Le projet vise à concevoir et développer un moteur de recherche pour une bibliothèque virtuelle, capable de gérer une base de données conséquente de livres en format texte. La bibliothèque doit comporter un minimum de 1664 livres, chacun d'entre eux ayant une taille minimale de 10 000 mots. L'objectif principal est de permettre aux utilisateurs d'effectuer des recherches efficaces dans cette vaste collection de documents.

a. Contexte :

Les bibliothèques numériques, telles que la base de données de Gutenberg, offrent un accès immense à des documents textes. Cependant, avec autant de données, il devient impératif de mettre en place un moteur de recherche performant pour faciliter la recherche et l'accès aux contenus pertinents.

b. Objectifs :

- Construire une bibliothèque virtuelle contenant au moins 1664 livres, chacun avec un contenu textuel d'au moins 10 000 mots.
- Développer un moteur de recherche permettant aux utilisateurs de rechercher des livres par mot-clé.
- Implémenter une fonctionnalité de recherche avancée basée sur des expressions régulières (RegEx).
- Intégrer des fonctionnalités implicites telles que le classement des résultats en fonction de critères de pertinence déterminés par l'indice de centralité (closeness, betweenness, ou pagerank).
- Proposer des suggestions de résultats similaires à la dernière recherche effectuée, basées sur le graphe de Jaccard et d'autres stratégies spécifiques.

c. Contraintes :

- La taille minimale des livres et de la bibliothèque doit être respectée.
- La pertinence des résultats de recherche sera évaluée subjectivement par des tests utilisateurs.
- La performance du moteur de recherche sera mesurée objectivement par des tests de charge sur différents volumes de données.
- Utiliser au moins un critère parmi closeness, betweenness, ou pagerank pour le classement des résultats.

d. Enjeux :

Un moteur de recherche efficace dans une bibliothèque numérique facilite l'accès à l'information, permettant aux utilisateurs de trouver rapidement des documents pertinents.

Cela améliore l'expérience utilisateur et la productivité en évitant une recherche manuelle fastidieuse. De plus, la mise en place de fonctionnalités implicites comme les suggestions basées sur le comportement d'autres utilisateurs contribue à une expérience utilisateur plus personnalisée.

Nous avons donc pour objectif de proposer une exploration efficace d'une vaste bibliothèque de livres numériques, en offrant des fonctionnalités de recherche avancées, de classement pertinent et de suggestions personnalisées.

2. Collecte et parsing des données

Pour constituer la bibliothèque de livres nécessaires pour notre projet (~ 1700 livres), nous avons fait du scraping à partir du site web " The Gutenberg Project " en développant un simple script python qui fait des requêtes vers le site et récupère les livres en format texte et les stocke dans notre dossier data.

Les fichiers texte récupérés à partir du Gutenberg Project ne sont pas adaptés à notre besoin , surtout qu'on cherche à optimiser nos recherches à travers ce projet . C'est pour cette raison que nous avons procédé à un parsing qui va nous aider à extraire les métadonnées et le contenu des livres et éliminer toutes les données qui ne sont pas utiles . Le script de parsing est contenu dans le dossier de notre code source.

3. Structure de données utilisée (DAO)

Dans notre projet, nous avons opté pour stocker les métadonnées de nos livres en format JSON. Ces fichiers sont particulièrement adaptés pour stocker les métadonnées associées à chaque document, telles que le nom de l'auteur, la langue et le titre du livre. L'utilisation de JSON facilite la manipulation et l'accès aux données grâce à sa structure claire, tout en offrant une compatibilité étendue avec divers systèmes et langages de programmation. Le contenu des livres est directement accessible dans le répertoire data pour faciliter la recherche . Nous avons renoncé à l'utilisation d'une base de données distante MongoDB Atlas à cause des dégradations de performance considérables et la lenteur des entrées/sorties de la base de données qu'on a pu constaté après implémentation de cette solution.

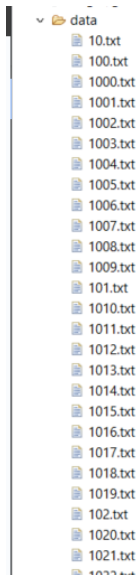
```
[
  {
    "id": "10",
    "title": "The King James Version of the Bible",
    "author": "Unknown",
    "releaseDate": "August 1, 1989 [eBook #10]",
    "language": "English"
  },
  {
    "id": "100",
    "title": "The Complete Works of William Shakespeare",
    "author": "William Shakespeare",
    "releaseDate": "January 1, 1994 [eBook #100]",
    "language": "English"
  },
]
```

Pour la partie back-end de notre application, nous avons choisi Spring Boot, un choix robuste et efficace pour la création d'API. Spring Boot nous permet de développer rapidement des services RESTful efficaces, facilitant ainsi la communication avec le front-end. Cette architecture nous permet de traiter les requêtes de manière efficace, en assurant une interaction fluide avec le front-end, réalisé avec jQuery. L'emploi de jQuery pour le front-end nous permet de gérer les requêtes GET de manière simple et efficace, en assurant une expérience utilisateur fluide et réactive.



Sur le plan métier, notre structure repose sur la classe modèle Document. Cette classe est au cœur de notre application; elle encapsule les propriétés essentielles de chaque document, telles que id, title, author, releaseDate, language, occur. les objets de cette classe sont souvent renvoyés à notre frontend pour s'afficher aux utilisateurs quand ils lancent une recherche.

```
public class Document {  
  
    private int id;  
    private String title;  
    private String author;  
    private String releaseDate;  
    private String language;  
    private int occur;  
}
```

Pour résumer, notre structure de données, articulée autour de fichiers JSON pour le stockage, de Spring Boot pour le back-end, et d'une classe modèle Document bien définie, forme une base solide pour notre application. Cette structure nous permet de gérer efficacement les données, de traiter les requêtes utilisateur et de fournir une expérience fluide et intégrée du côté du client.



l'arborescence de nos classes java utilisés pour les différents algorithmes de recherche est comme suit:

- >  BinaryTree.java
- >  BookConfiguration.java
- >  ClosenessCentrality.java
- >  DfaTraversal.java
- >  JaccardGraph.java
- >  KMP.java
- >  LeafNode.java
- >  Node.java
- >  ReaderFile.java
- >  RegexToDfa.java
- >  State.java
- >  SyntaxTree.java

4. Analyse et présentation des Algorithmes utilisées

Il existe de nombreux algorithmes de recherche de pattern, nous allons nous intéresser à ceux utilisés dans le cadre de notre projet qui sont KMP, Aho Ullman et Jaccard

a. Knuth-Morris-Pratt (KMP)

L'algorithme de KMP est un algorithme de recherche de sous-chaîne efficace. Cet algorithme cherche une sous-chaîne dans une chaîne principale et est particulièrement efficace car il évite de repasser sur des caractères déjà comparés, contrairement à des approches plus naïves.

Le principal avantage de l'algorithme KMP réside dans sa capacité à maintenir les informations des correspondances partielles grâce à un tableau de préfixes. Cela lui permet de sauter des positions dans le texte sans manquer de possibles correspondances, offrant une complexité temporelle optimale de $O(n + m)$, où n est la longueur du texte et m celle du motif.

Dans le contexte de notre bibliothèque virtuelle, l'implémentation de KMP apporte une réponse efficace et rapide à la recherche de chaînes spécifiques dans le contenu volumineux des livres. Cela est particulièrement utile pour des recherches répétitives où la performance et la rapidité sont cruciales.

L'algorithme fonctionne de la manière suivante :

- Prétraitement du Motif

L'algorithme commence par construire un tableau de préfixes en $O(m)$, m étant la longueur du motif. Ce tableau permet de déterminer, pour chaque position dans le motif, la plus longue sous-chaîne propre qui est à la fois préfixe et suffixe du motif jusqu'à cette position.

Cela signifie qu'en cas de non-correspondance, l'algorithme sait combien de caractères il peut sauter sans manquer une possible correspondance.

- Recherche dans le Texte

Une fois le tableau de préfixes construit, l'algorithme commence la recherche dans le texte principal. Il parcourt le texte et le motif simultanément, et en cas de non-correspondance, il utilise le tableau de préfixes pour déterminer la prochaine position du motif à comparer. Cela évite de revenir en arrière dans le texte, permettant ainsi à l'algorithme de fonctionner en temps linéaire par rapport à la longueur du texte.

- Exemple

Considérons le texte "ABABDABACDABABCABAB" et le motif "ABABCABAB" que nous voulons rechercher dans le texte.

Prétraitement :

On construit le tableau de préfixes pour "ABABCABAB".

Motif: A B A B C A B A B

Tableau: 0 0 1 2 0 1 2 3 4

Ce tableau indique, par exemple, que si une non-correspondance survient à la position 4 du motif, nous devons reprendre la comparaison directement à la position 0.

Recherche :

On parcourt le texte "ABABDABACDABABCABAB" avec le motif.

À chaque étape, si les caractères correspondent, on continue.

Si une non-correspondance est trouvée, on utilise le tableau pour déterminer combien de caractères on peut sauter dans le motif.

En utilisant KMP, on trouve que le motif ABABCABAB apparaît dans le texte à partir de la position 10.

b. Aho Ullman

Théorème : Tout langage régulier est accepté par un automate fini déterministe.

La méthode d'Aho-Ullman est basée sur deux concepts clés : la construction de l'automate fini du motif et l'utilisation de cet automate pour rechercher le motif dans le texte

Pour parvenir à notre objectif, nous utilisons une série d'algorithmes qui transforment l'expression régulière en un automate fini déterministe (AFD).

- Transformation d'une expression régulière en arbre de syntaxe : Cette étape consiste à convertir l'expression régulière en un arbre de syntaxe pour faciliter le traitement ultérieur. Cette étape a une complexité linéaire par rapport à la taille de l'expression, c'est-à-dire $O(n)$, où n est le nombre de caractères dans l'expression
- Construction d'un automate fini non déterministe avec ϵ -transitions : En utilisant la méthode d'Aho-Ullman, nous transformons l'arbre de syntaxe en un automate fini non déterministe qui peut gérer les ϵ -transitions. La complexité est linéaire par rapport à la taille de l'expression, soit $O(n)$.

- Conversion en automate fini déterministe : Nous convertissons ensuite l'automate fini non déterministe en un automate fini déterministe (AFD) en appliquant la méthode des sous-ensembles. La complexité est exponentielle dans le pire des cas. Cependant, dans la plupart des cas pratiques, n est relativement petit, de sorte que la conversion reste efficace.
- Minimisation de l'AFD : Si possible, nous minimisons l'AFD obtenu pour réduire le nombre d'états et optimiser sa représentation.
- Reconnaissance des sous-chaînes : Enfin, nous utilisons l'AFD pour vérifier si un suffixe de chaque ligne du fichier texte est reconnaissable par cet automate. La recherche du motif dans le texte a une complexité en $O(n)$ avec n la longueur du texte

c. Jaccard

Le graphe de Jaccard est souvent utilisé pour représenter les relations entre des ensembles d'éléments. Dans le contexte de la recherche de documents, les nœuds du graphe pourraient représenter des mots-clés ou des documents, et les arêtes indiqueraient une relation d'intersection entre documents

La closeness centrality (centralité de proximité) est un indice de centralité qui mesure la proximité d'un nœud avec tous les autres nœuds du graphe. Plus précisément, elle évalue la distance moyenne entre un nœud donné et tous les autres nœuds. Un nœud avec une closeness centrality plus élevée est considéré comme plus central dans le réseau .

Dans le contexte de la recherche dans une base de données de livres, construire un graphe de Jaccard pourrait signifier représenter les relations entre les mots-clés associés à chaque livre. Ensuite, en utilisant la closeness centrality, on peut identifier les livres qui sont plus "centraux" dans le réseau, c'est-à-dire ceux qui ont des relations plus étroites avec d'autres livres en termes de mots-clés partagés.

5. Réalisation

a. Vue d'ensemble de l'application

Rechercher Par regex :

Cette fonctionnalité permet aux utilisateurs de faire des recherches avancées en utilisant des expressions régulières, offrant une grande flexibilité et précision pour isoler les informations désirées.



Rechercher Par mots clé:

Cette option permet une recherche rapide et intuitive basée sur des mots-clés, adaptée pour des requêtes plus générales ou lorsque l'utilisateur a besoin de résultats rapidement sans paramétrages complexes.



b. Architecture

Notre application se compose de deux parties principales : le front-end et le back-end.

Le front-end est une page HTML simple, rendue dynamique et interactive grâce à jQuery. Cette structure permet une manipulation aisée des éléments et une interaction fluide avec les requêtes AJAX, offrant ainsi une expérience utilisateur intuitive et réactive.

Pour la partie back-end, nous avons opté pour une application web robuste basée sur Spring Boot. Spring Boot facilite le développement d'applications autonomes, en minimisant la configuration et le déploiement, tout en offrant une riche boîte à outils pour répondre aux besoins métier.

c. Pré-requis et démarrage de l'application

Pour démarrer l'application, une série de prérequis et d'étapes doivent être suivies :

Assurez-vous que **Java 17** est installé sur votre système. Cette version de Java est nécessaire pour garantir la compatibilité avec les fonctionnalités utilisées par notre application Spring Boot.

Maven est essentiel pour la gestion des dépendances et la construction de notre application. Si Maven n'est pas déjà installé sur votre système, vous devez l'installer.

Démarrage de l'Application :

Une fois que Java 17 et Maven sont installés, vous pouvez démarrer l'application en utilisant le Maven Wrapper inclus dans le projet. Ouvrez votre terminal, naviguez vers le répertoire racine du projet, et exécutez la commande suivante :

```
./mvnw spring-boot:run
```

```
C:\Users\Public\ws\search-book>mvnw spring-boot:run
[INFO] Scanning for projects...
[WARNING]
[WARNING] Some problems were encountered while building the effective model for org.ms.java:search-book:jar:0.0.1-SNAPSHOT
OT
```

6. Conclusions et perspectives

Nous avons implémenté deux types de recherche dans notre application : la recherche par mots-clés et la recherche par expressions régulières (regex). Cependant, nous faisons face à des contraintes de performances, notamment lors de la recherche sur une base de données de 1664 livres au format texte. Même en utilisant des threads pour paralléliser le processus, la recherche prend actuellement environ 20 secondes. Nous sommes conscients de cette limitation et cherchons des moyens d'optimiser les performances. Dans le but d'améliorer les performances de notre système de recherche, nous examinons différentes approches telles que l'optimisation des algorithmes de recherche, la gestion efficace des ressources, ou l'exploration de méthodes de stockage de données plus rapides. Nous reconnaissons l'importance de réduire le temps de recherche pour offrir une expérience utilisateur plus fluide et travaillons activement à résoudre ce problème spécifique.

En tant que prochaine étape de notre développement, nous sommes en train de mettre en place la logique métier du graphe de Jaccard avec l'utilisation de la centralité de proximité (closeness centrality). Cette fonctionnalité a été testée avec succès sur un ensemble de 120 livres à des fins de validation. Nous considérons cette implémentation comme une perspective prometteuse pour améliorer la pertinence des résultats de recherche et pour offrir une expérience utilisateur plus riche et ciblée. Nous sommes impatients d'étendre ces tests à l'ensemble complet de livres dans notre base de données pour évaluer pleinement l'efficacité de cette nouvelle fonctionnalité.