Rapport final
Indexation sémantique et recherche d'information

Yuyan QIAN
21108372

Table des matières

INTRODUCTION

Solr Apache, étant des plateformes de recherche puissantes, offrent une flexibilité et une efficacité impressionnante pour indexer et rechercher des données. Notre moteur de recherche utilise ces technologies pour traiter un corpus de films biographiques, allant des œuvres du début du XXe siècle à nos jours. Ce corpus, riche et diversifié, inclut non seulement des informations basiques des films mais aussi des notes du public, offrant ainsi un panorama complet du genre biographique.

L'objectif de ce projet est double : d'une part, fournir une plateforme intuitive pour explorer facilement ce corpus cinématographique, et d'autre part, exploiter les capacités avancées de Solr/Lucene pour offrir des fonctionnalités de recherche et de filtrage puissantes. En intégrant des facettes, des filtres, et des options de recherche avancées, notre moteur de recherche permet aux utilisateurs d'affiner leurs requêtes, facilitant ainsi la découverte de films spécifiques selon divers critères tels que la période, le réalisateur, ou les acteurs principaux.

Cette introduction présente le contexte et les objectifs de notre moteur de recherche, décrivant brièvement les technologies utilisées et la nature du corpus traité. Les sections suivantes détailleront la structure du corpus, les étapes de pré-traitement, la création du schéma d'indexation, et les fonctionnalités uniques de notre système de recherche. Enfin, nous conclurons en présentant les résultats obtenus et les perspectives d'amélioration de notre moteur de recherche.

1 DESCRIPTION DU CORPUS

Un corpus adéquat est la base de la recherche d'information. Il est primordial de bien choisir un corpus avant d'amener un projet afin d'explore pleinement les fonctionnalités du plateforme *Solr*.

1.1 Origine du corpus

La source principale d'un ensemble de données IMDb de tous les films basé sur le genre serait IMDb, la source la plus populaire et la plus fiable au monde pour le contenu des films, de la télévision et des célébrités. IMDb dispose d'une vaste base de données de films qui est constamment mise à jour avec de nouveaux titres et de nouvelles informations. Le jeu de données que j'ai choisi fait partie d'un grand corpus « *Ensemble de données de films IMDb : tous les films par genre* » sur Kaggle [1] dont la dernière version se compose de 16 fichiers de données, publié tout en licence CC BY-NC-SA 4.0. Chaque fichier contient les informations détaillées d'un genre de film, allant de l'action à la guerre [2].

La raison majeure de mon choix est que ce corpus contient des informations de types variés, par exemple, la chaîne de caractères, le nombre entier, le nombre décimal, etc. En

1. https ://www.kaggle.com/datasets/rajugc/imdb-movies-dataset-based-on-genre/data
2. Les 16 genres sont listées ici : *Action, Aventure, Animation, Biographique, Crime, Famille, Fantastique, Film noir, Historique, Horreur, Mystère, Romantique, Science fiction, Sport, Thriller, Guerre.*

plus, les contenues sont en anglais, ce qui est plus facile pour moi de traiter. Néanmoins, ce corpus est trop grand pour mon projet et il n'est pas nécessaire de créer un moteur de recherche pour tous les genres de films.

Vu la grande taille du corpus, soit 130,45 Mb, j'ai décidé d'utiliser seulement les données du film biographique pour approfondir le projet. De taille 2,73 MB, ces sous-données sont suffiantes pour créer un moteur de recherche puissant. Les attributs sont suavegardés avec séparateur de point-virgule en 14 colonnes. Le fichier *biographie*, sous format de CSV, contient 8289 instances et 14 attributs. Pour illuster les caractéristiques des données, on prend un exemple de film *Elvis* (2019) dans le tableau 1.

Nb	**Attribut**	**Description**	**Exemple**	**Type**
1	movie_id	ID du film IMDB	*tt3704428*	string
2	movie_name	Titre du film	*Elvis*	TG
3	year	Année de sortie	*2022*	pint
4	certificate	Classification du film	*PG-13*	string
5	run_time	Durée d'exécution	*159 min*	pint
6	genre	Genre du film	*Biography, Drama, Music*	TG
7	rating	Note moyenne du film	*7.3*	pfloat
8	gross (in $)	Revenus bruts en dollar	*151040048.0*	pdouble
9	votes	NB de votes sur IMDB	*189732.0*	pdouble
10	director	Réalisateur/réalisatrice	*Baz Luhrmann*	TG
11	director_id	ID de réalisateur(trice)	*/name/nm0525303/*	–
12	star	Casting principal	*Tom Hanks, Austin Butler, Olivia DeJonge*	TG
13	star_id	ID des acteurs(trices)	*/name/nm0000158/, /name/nm2581521/, /name/nm4609822/, /name/nm0861013/*	–
14	description	Description du film	*The life of American music icon Elvis Presley, from his childhood to becoming a rock and movie star in the 1950s while maintaining a complex relationship with his manager, Colonel Tom Parker.*	TG

Table 1 – Caractéristiques des données du fichier *biographie*. Pour raccourcir, TG signifie *Text_general*. Le symble « – » marqué dans la colonne *Type* indique que cette catégorie d'information est enlevée dans le pré-traitement du corpus en raison de l'adéquation thématique.

1.2 Faiblesses du corpus

Suite à un examen détaillé de l'ensemble de données, j'ai repéré cinq faiblesses majeures. Les trois premières sont liées directement aux films biographiques. D'abord, le corpus contenait également des informations redondantes pour le projet, par exemple les informations sur l'identifiant du réalisateur, l'identifiant d'acteur, qui ne sont pas nécessaires pour la recherche du film. Si l'on crée un moteur de recherche pour les réalisateurs et les acteurs, ces informations seront adéquates, comme le site source IMBd. Ensuite, j'ai également rencontré un problème concernant les années de sortie des films. Normalement, les dates sont indiquées par des années comme 1999, mais il y a des entrées anormales, comme « XVI », qui ne correspondent pas aux années de sortie réelles des films après vérification. En outre, la catégorie « votes » compte les nombres de votes sur le site IMBd qui sont récupéré comme un nombre décimal, (par exemple, *189732.0* montré dans le tableau 1), tout avec un zéro après la virgule.

Les deux faiblesses mentionnées par la suite reflètent également les déficiences inhérentes au corpus source. Une lacune de ce corpus cinématographique : l'auteur n'a pas mentionné la date de récupération des données. Cet ensemble de données en absence de la marque temporelle empêche les utilisateurs de confirmer la validité temporelle des données et de vérifier leur contenu, car le site source des données est constamment mis à jour. Le problème engendré sera discuté dans la section suivante. Enfin, plusieurs attributs sont vides dans le corpus.

2 PRÉ-TRAITEMENT DU CORPUS

Pour répondre aux cinq questions mentionnées dans la section précédente, j'ai effectué un prétraitement du corpus. En raison du format CSV, l'opération a été accomplie à l'aide de l'outil *Excel* au lieu de *GATE*. Les deux premières concernent les données de sous-corpus que je vais présenter dans la recherche de moteur. J'ai choisi ainsi de supprimer les informations redondantes et les années de sortie anormales. Les trois dernières sont liées à l'extraction depuis le site officiel d'IMDb et l'imparfait des données source, et elles ont été conservées telles quelles. Je ne souhaite pas altérer les données. Il est crucial de ne pas altérer les données pour maintenir leur intégrité et leur fiabilité. La modification des données peut entraîner une représentation inexacte de l'information, compromettre les résultats de l'analyse et potentiellement induire en erreur les utilisateurs finaux. Par conséquent, il est essentiel de conserver les données dans leur état original pour assurer la validité et la crédibilité de toute analyse ou conclusion dérivée de ces données. En un mot, le traitement des données pourrait être composé par trois méthodes totalement différentes, soit la modification, soit l'insertion, soit la suppression. Dans ce projet, suelement *supprimer* est appliqué. La dernière étape de prétraitement est l'uniformisation des données de la colonne *Durée* qui permet de fonctionner le tri en ordre dans le plateforme *Solr*.

2.1 Enlèvement de deux catégories d'informations redondantes

Ce projet vise à créer un moteur de recherche spécifique pour les films biographiques de sorte que les identifiants des réalisateurs et des acteurs ne sont pas nécessaires. Garder ces informations peut entraîner l'incompréhension des utilisateurs, puisque les informations des réalisateurs et des acteurs ne sont pas dans ma base de données. L'identifiant du réalisateur et l'identifiant de l'acteur ne sont pas indispensable dans la recherche du film biogragique de sorte que ces deux colonnes de données sont supprimés dans le prétraitement du corpus en raison de l'adéquation thématique.

2.2 Filtrage des valeurs anormales

J'ai utilisé Excel pour filtrer et éliminer les valeurs anormales. Ce processus impliquait la sélection de critères spécifiques pour isoler les données pertinentes, suivie de la suppression des entrées incorrectes. Il est possible que ces entrées soient des erreurs apportés par l'extraction automatique. Dans ce cas, j'ai filtré et supprimé les lignes qui contiennent des années de sortie anormales, comme « XVI » ou « XVII ». En outre, dans la description du films, plusieurs films sont marqué « Add the plot« . J'ai également les supprimé.

2.3 Uniformisation des données de la colonne *Durée*

Cette étape de purification servit à assurer la bonne fonction du tri en ordre dans le plateforme *Solr*, le marqueur de « min » dans la colonne de durée est enlevé afin de rendre les années en format *Int*. L'éxecution de cette opération est réalisée par la fonction *Formules* dans Excel en appliquant la condiiton *SI*.

2.4 Ajout des valeurs booléennes pour l'existence de description

Pour avoir une facette pivot, j'ai ajouté une catégorie de valeur booléenne afin de distinguer les films qui ont une description et ceux qui n'en ont pas. Cette opération est réalisée par la fonction dans Excel en appliquant les expressions régulières.

3 CRÉATION DU SCHÉMA

Le schéma est un fichier de configuration qui définit les champs et les types de données qui seront indexés dans Solr. Une fois que le corpus est prêt, il est temps de midifier le schéma d'indexation afin que *Solr* puisse bien identifier les noms du champ (*fieId*) et les types de données. Les noms du champs sont définis par les attributs du corpus sauf que les les espaces sont rempalcées par un tiret bas et que le symbole de « $ » est écrit en dollar. Tous les chagements se sont fait en correspoandant des labels de colonne dans le fichier CSV. Les types de données sont définis par les types de données de *Solr* qui sont listés dans la liste 1. L'identifiant et la classification du film contient des informations normales qui n'ont pas besion d'être tokenisées ou analysées, donc le type de données est *string*.

Les autres attributs qui se composent des informations textuelles, sont utilisés pour la recherche de texte avec mot-clé. Ce type de données est *Text_general*.

Listing 1 – managed-schema.xml

```
<field name="id" required="true" type="string" indexed="true" stored="true" multiValued="false"/>
<field name="movie_name"  type="text_general" indexed="true" stored="true" required="true" />
<field name="certificate" type="string"       indexed="true" stored="true"/>
<field name="description" type="text_general" indexed="true" stored="true"/>
<field name="director"    type="text_general" indexed="true" stored="true"/>
<field name="genre"       type="text_general" indexed="true" stored="true"/>
<field name="star"        type="text_general" indexed="true" stored="true"/>
<field name="runtime_min" type="pint"         indexed="true" stored="true"/>
<field name="year"        type="pint"         indexed="true" stored="true"/>
<field name="rating"      type="pfloat"       indexed="true" stored="true"/>
<field name="gross_in_dollar" type="pdouble"  indexed="true" stored="true"/>
<field name="votes"       type="pdouble"      indexed="true" stored="true"/>
<field name="with_description" type="booleans" indexed="true" stored="true"/>
```

En Solr, le type de champ numérique doit être choisi avec soin, en fonction des valeurs maximales et minimales que le champ est susceptible de contenir. La méthode de trouver les valeurs extrêmes est de trier les données dans l'ordre croissant ou décroissant.

L'année de sortie du film est un nombre entier, allant de 1903 à 2024. La durée d'exécution a la plus grande valeur de 442 minutes. Ells sont eses nombres entiers, et donc leurs types de données sont *pint*. La note moyenne du film ne passe pas 10, mais elle a toujours un chiffre après la virgule, donc le type de données est *pfloat*. Le revenu brut est un nombre décimal, allant de 0.0 à 936662225.0. Le nombre de votes est un nombre décimal, allant de 5.0 à 189732.0. Ils sont des nombres assez grands qui ont plus de 7 chiffres, de sorte que j'ai choisi *pdouble* pour bien inclure toutes les possibilités.

Pour les films, le plus essentiel est de posséder un identifiant et un titre dans une base de données où l'on pourrait effectuer la recherche. Dans ce cas, les deux paramètres sont indispensables au cours de l'indexation. De plus, l'identifiant doit être unique par film de manière que les duplications de l'ID ne sont pas acceptables par le schéma. Le schéma d'indexation est présenté dans la liste 1.

4 INDEXATION DU CORPUS

Listing 2 – Ligne de commande pour le corpus

```
./post -c biographie -type text/csv -url "http://localhost:8983/solr/biographie/update/csv
    ?commit=true&separator=%3B" -params "f.champs.trim=true" biography1.csv
```

L'indexation du corpus n'a pas été executée avec résussite au début. J'ai rencontré un problème de *java.io.IOException : Server returned HTTP response code : 400 for URL* qui est causé par le délimiteur du fichier CSV. En effet, le corpus est délimité par un point-virgule. Pour résoudre ce problème, il me faudrait bien identifier le délimiteurs dans les lignes de commandes. La solution est d'ajouter *separator=%3B*. Ici, le paramètre *separator=%3B* est utilisé dans le contexte des requêtes ou de la configuration pour spécifier

le point-virgule avec l'encodage URL. La ligne de commande est présentée dans la liste 2.

5 CRÉATION DES FACETTES

Les deux facettes de champs sont le certificat de classification du film et le genre du film. Bien que j'ai utilisé les films biographiques, certains d'entre eux sont encore classés comme « Action », « Historique », etc. dans le corpus. Les deux facettes d'intervalles sont la durée d'exécution, l'année de sortie et la note. Afin de créer des intervalles adéquates, j'ai exploité les requêtes sur l'interface admin de *Solr*. En appliquant la fonction *Sort*, j'ai trouvé les valeurs maximales et minimales. Cette étape a vérifié également la validité des types de données que j'ai défini dans le schéma. De ce fait, l'intervalle de l'anée de sortie est de 1903 à 2024, en pas de 20 ans. L'intervalle de la durée d'exécution est de 0 à 500 minutes, en pas de 100 minutes. L'intervalle de la note moyenne du film est de 0 à 10, en pas de 1.0.

La facette pivot est l'existence de description. Ici, j'ai fait appel à une catégorie d'information booléenne qui est ajoutés dans le corpus afin de distinguer les films qui ont une description et ceux qui n'en ont pas. En fait, tout est pour réaliser une fonction de recherche avancée. La facette de requête est laissé vide puisque la requête vise majoritairement à rechercher les titres de films, et les 7978 instances ne sont pas susceptibles de tout apparaître sur une seule page.

6 MODIFICATION SUR LE BROWSE

6.1 Trois parties de Browse

Pour illustre, j'ai essayé de distinguer les trois grandes parties fonctionnelles dans l'interface d'utilisateur de *Solr* : Filtrage et facettes, Barre de recherche et paramètre avancé, Résultat de recherche. La figure 1 montre l'exemple d'une page de recherche de *Solr* avec les trois parties encadrées et annotée en rouge. La sous-partie de résultat est encadrée en vert. En réalité, les modifications de *Filtrage et facettes* sont analysées dans la section précédente. Dans cette section, je vais présenter les modifications de *Barre de recherche et paramètre avancé* et *Résultat de recherche*.

6.2 Paramètres avancés

La fonction d'origine est de *Boost by price*, qui permet de trier les résultats par prix. Notre corpus, les films sont évalués essentiellement par les notes du pulique de sorte que j'ai modifié cette fonction en *Tri par note en ordre décroissant* , qui permet de trier les résultats par note moyenne du film. La modification a été faite dans deux fichiers *query.vm, query_form.vm*. Voici le code de *query.vm* dans la liste 2.

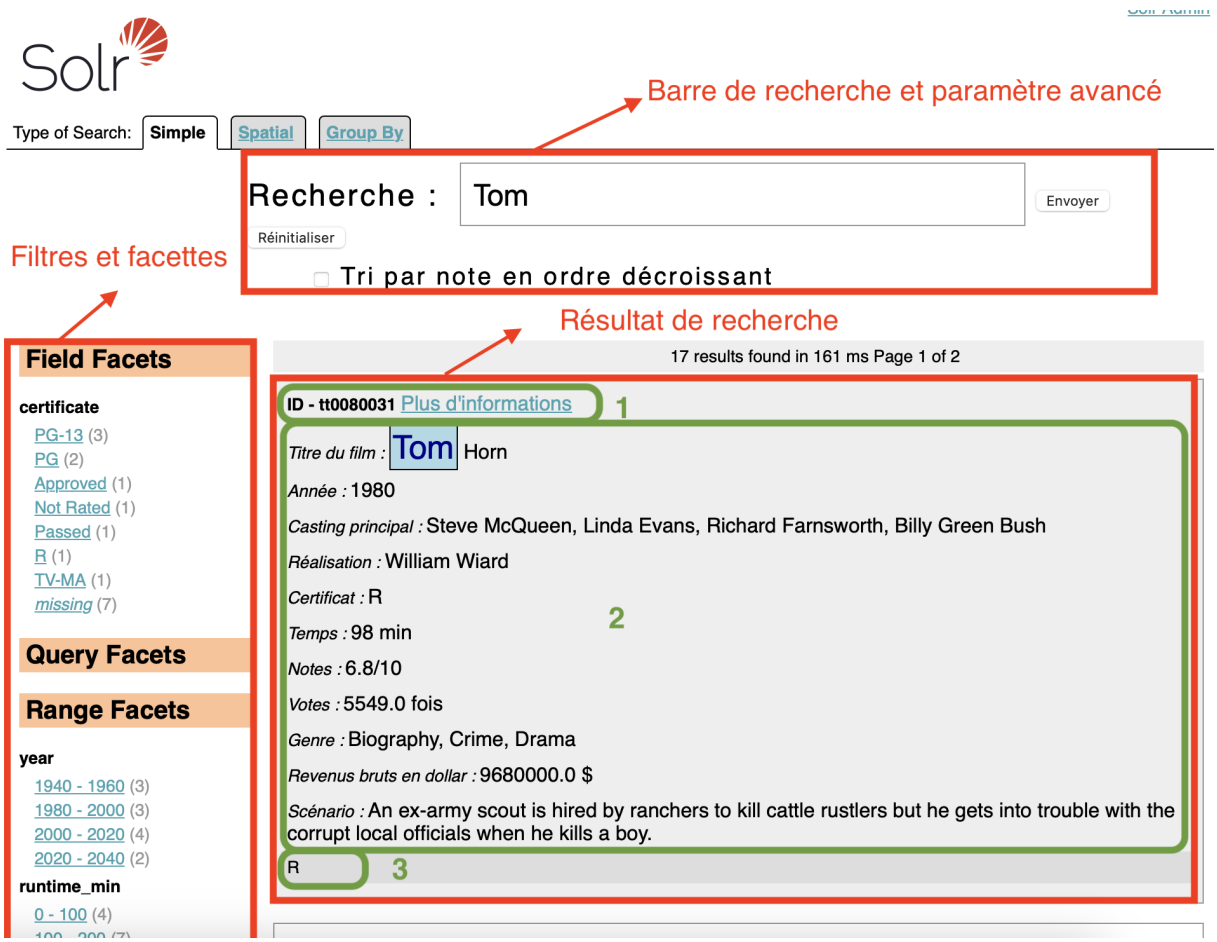

Figure 1 – Modification de Browse

Listing 3 – Query.vm

```
<div class="query-box">
  <form id="query-form" action="#{url_for_home}" method="GET">
    <div class="inputs">
      <span #annTitle("Add the query using the &q= parameter")>Recherche :
          <input type="text" id="q" name="q" value="$!esc.html($params.get('q'))"/>
          <input type="submit" id="querySubmit"/>
          <input type="reset"/>
      </span>
      <div class="query-boost">
        <span #annTitle("Add the sort parameter &sort=rating desc to the query")>
          <input type="checkbox" name="sort" value="rating desc"
              #if($request.params.get('sort') == 'rating desc') checked="true"#end>
              Tri par notre en ordre decroissant
          </input>
        </span>
      #parse("querySpatial.vm")
      #parse("queryGroup.vm")
    </div>
  </div>
```

6.3 Trois composants de *Résultat de recherche*

Les résultats de recherche sont composés de trois parties : icône, titre et lien pour MLT (Plus de résultats similaires) ; contenu du film ; la badge de classification en bas.

Dans ce contexte, « MLT » fait référence à *More Like This*, une fonctionnalité courante dans les systèmes de recherche qui trouve et affiche des éléments similaires à celui consulté. Tous les adaptations se sont effectuées dans le fichier *richtext_ doc.vm*. Dans notre corpus, il n'existe que des données textuelles et numériques, de manière que l'icône n'est pas nécessaire dans l'affichage.

En outre, j'ai transformé le résumé du film en bas de chaque résultat de recherche en badge de classification qui est plus conci pour moi. Tous les paramètres du contenu du film sont traduits en français.

6.4 *Highlighting*

Le dernier grand morceau du projet est de modifier les paramètres de « Velocity » afin de visualiser le contenu des données plus finement.

Le *Highlighting* est une fonction qui permet de mettre en évidence les mots-clés dans les résultats de recherche. Cette fonction est très utile pour les utilisateurs de localiser les mots-clés dans un contexte des résultats trouvés. Dans ce projet, j'ai modifié la couleur de surlignage en bleu foncé, le fond en bleu clair, l'encadrement en noir et la taille de police en large. La première étape est d'ajouter les différentes classes dans la configuration de *Highlighting*. Ensuite, les commandes *CSS*, par exemple, dans la liste 4, sont ajoutées dans le fichier *richtext_ doc.xml* afin de modifier la couleur et la taille de police.

Listing 4 – richtext_doc.xml

```
1 <str name="hl.simple.pre">&lt;span style="background-color: #add8e6; color: #00008b;
      border: 1px solid black; padding: 2px; font-size: 1.2em;"&gt;</str>
2 <str name="hl.simple.post">&lt;/span&gt;</str>
```

6.5 Troncation de texte à cause de *Highlighting*

Les résultats de recherche à partir de l'interface admin montraient le scénariocomplet du film. Néanmoins, sur l'interface utilisateur, la description du film, qui désigne toujours des textes longs, était tronqué. En fait, c'est un conflit entre la recherche d'attribut et la fonction de Highlighting. Les parties en *Highlighting* sont entournées par les balise de HTML, comme dans l'exemple de la liste 4, la balise de hl. La taille de fragment textuel est délimitée dans le fichier de configuration en 100 caractères. Cette définition par défaut a causé une troncation de texte dans l'affichage. Conséquemment, j'ai augmenté la taille de fragment textuel en 10000 caractères et pour assurer la bonne fonctionnement de *Highlighting*, j'ai utilisé les codes plus complexes dans le fichier *richtext_ doc.vm* pour trouver les mots-clés dans la description du film. La liste 5 montre le code de *troncation*. Ici, l'indexation sur la description du film a apporté une possibilité de « multiValued » de sorte que je devais ajouter *.get(0)* pour enlever les crochets dans l'affichage.

Listing 5 – richtext_doc.xml

```
#set($highlighting = $response.getHighlighting().get($doc.get('id')))
...
$highlighting.get('description').get(0)
```

CONCLUSION

Ce projet m'a permis de comprendre le fonctionnement de *Solr* en profondeur. J'ai appris à créer un moteur de recherche avec *Solr* et à modifier l'interface d'utilisateur de *Solr* avec *Velocity*. En outre, j'ai également appris à choisir le corpsu et à l'indexer pour l'extraction d'information. En conclusion, le plus fondamental est de bien savoir les caractéristiques du corpus et ensuite adapter à l'outil d'indexation, comme ce que j'ai effectué pour le schéma et la configuration du Solr. Les étapes suivantes, telles que la visualisation, permettent d'avoir une amélioration constante pour la recherche. Ce que j'ai créé n'est pas tout à fait parfait et j'ai également vu des parties rudimentaires. L'aboutissement d'un simple moteur de recherche quand même m'aide à acquérir un aperçu du processus de traitement. J'ai compris l'importance de la création d'un corpus adéquat pour la recherche d'information. Tout cela m'aidera à mieux comprendre le plateforme *Solr/lucene* et du cours de *Indexation sémantique et recherche d'information*.