Projet Web sémantique

Chenrui Zhu

1 mars 2024

1 Introduction

Pour ce mini-projet axé sur le Web sémantique, je propose de développer une application qui recueille des données sur des films, les modélise en RDF(S), permet aux utilisateurs de rechercher et de filtrer les films selon différents critères, et fournit des informations détaillées sur chaque film.

Voici les fonctionnalités principales de cette application :

1. **Collecte de données sur les films :** Nous pouvons collecter des données sur les films à partir de sources telles que IMDb, TMDb, ou des bases de données de films open source.

2. **Modélisation en RDF(S) :** Les données sur les films seront modélisées en RDF(S) en utilisant des ontologies appropriées, telles que Schema.org, FOAF (Friend of a Friend), ou une ontologie personnalisée si nécessaire. Chaque film serait représenté comme une instance de la classe rdfs:Movie, avec des propriétés telles que le titre, le réalisateur, les acteurs, la durée, le genre, etc.

3. **Interrogation avec SPARQL :** Les utilisateurs pourront interroger la base de données RDF à l'aide de requêtes SPARQL pour rechercher des films selon différents critères, tels que le genre, la date de sortie, le réalisateur, etc.

4. **Fonctionnalités de l'application Web :** L'application Web permettra aux utilisateurs de rechercher des films, de filtrer les résultats, et d'afficher des détails sur chaque film sélectionné. Les fonctionnalités supplémentaires peuvent inclure la possibilité de noter et de commenter les films, de créer des listes de films favoris, et de recommander des films en fonction des préférences de l'utilisateur.

En résumé, cette application permettra aux utilisateurs d'explorer et de découvrir des films en utilisant les technologies du Web sémantique, tout en offrant des fonctionnalités pour interagir avec les données de manière significative.

2 Présentation des données

Pour ce projet, le jeu de données original est un fichier CSV "rotten_tomatoes_movies.csv", qui contient des informations sur les films disponibles sur Rotten Tomatoes. Voici un aperçu des champs disponibles dans ce fichier :

- **id**: Identifiant unique du film

- **title**: Titre du film
- **audienceScore**: Note donnée par le public
- **tomatoMeter**: Note donnée par Rotten Tomatoes
- **rating**: Classification du film (PG, PG-13, R, etc.)
- **releaseDateTheaters**: Date de sortie en salles
- **releaseDateStreaming**: Date de sortie en streaming
- **runtimeMinutes**: Durée du film en minutes
- **genre**: Genre du film
- **director**: Réalisateur du film
- **boxOffice**: Recettes au box-office

J'ai utilisé ces données pour créer ma base de connaissances RDF. Voici quelques descriptions sur la modélisation des données :

1. J'ai définit une classe 'Movie' dans mon ontologie RDF, avec des propriétés telles que 'title', 'audienceScore', 'tomatoMeter', 'rating', 'releaseDateTheaters', 'releaseDateStreaming', 'runtimeMinutes', 'genre', 'director', et 'boxOffice'.
2. Chaque film dans le fichier CSV est représenté comme une instance de la classe 'Movie', avec des valeurs correspondant à chaque propriété.
3. J'ai utilisé des ontologies RDF standard telles que Schema.org pour modéliser les propriétés comme 'writer', 'director', et 'genre'. Pour les propriétés spécifiques à mon application comme 'audienceScore' et 'tomatoMeter', je les ai définir dans mon ontologie personnalisée.

Je détail la modélisation par ce tableau 1 ci-dessous:

La structure de données décrite dans le tableau suit le modèle RDF (Resource Description Framework) en utilisant le schéma RDF (RDF Schema). Voici une description générale de cette structure de données :

Entité : représente une entité dans notre système RDF, par exemple, un film, une personne, une organisation ou un genre.

Propriétés : sont les caractéristiques ou les attributs de chaque entité, définies par des prédicats RDF.

Dans cette structure de données :

Les films sont représentés par l'entité :name_movie et ont les propriétés suivantes :

- `rdf:type rdfs:Movie` : indique que l'entité est de type "Movie".
- `rdfs:aggregateRating` : représente la note agrégée du film, contenant les sous-propriétés suivantes :

Entité	Propriétés
:name_movie	**rdf:type** rdfs:Movie **rdfs:aggregateRating** [rdfs:type rdfs:AggregateRating :audienceScore double :tomatoMeter double] **rdfs:author** :name_person **rdfs:datePublished** [rdf:type rdfs:DatePublished releasedStreamingDate double releasedTheaterDate double] **rdfs:director** :name_person **rdfs:duration** double **rdfs:genre** :name_genre **rdfs:inLanguage** string **rdfs:maintainer** :name_organization **rdfs:title** string
:name_person	**rdf:type** rdfs:Person **foaf:name** string
:name_organization	**rdf:type** rdfs:Organization **rdfs:name** string
:name_genre	**rdf:type** rdfs:Genre **rdfs:label** string

Table 1: Explaining RDF data structure

  - `rdfs:type rdfs:AggregateRating` : indique que l'entité est de type "AggregateRating".
  - `:audienceScore double` : représente la note donnée par le public.
  - `:tomatoMeter double` : représente la note donnée par Rotten Tomatoes.
- `rdfs:author` : représente l'auteur du film, lié à l'entité :name_person.
- `rdfs:datePublished` : représente la date de publication du film, contenant les sous-propriétés suivantes :
  - `rdf:type rdfs:DatePublished` : indique que l'entité est de type "DatePublished".
  - `releasedStreamingDate double` : représente la date de sortie en streaming du film.
  - `releasedTheaterDate double` : représente la date de sortie en salle du film.
- `rdfs:director` : représente le réalisateur du film, lié à l'entité :name_person.
- `rdfs:duration double` : représente la durée du film en minutes.
- `rdfs:genre` : représente le genre du film, lié à l'entité :name_genre.
- `rdfs:inLanguage string` : représente la langue du film.
- `rdfs:maintainer` : représente l'organisme maintenant la distribution du film, lié à l'entité :name_organization.
- `rdfs:title string` : représente le titre du film.

Les personnes sont représentées par l'entité :name_person et ont la propriété suivante :

- `rdf:type rdfs:Person` : indique que l'entité est de type "Person".
- `foaf:name string` : représente le nom de la personne.

Les organisations sont représentées par l'entité :name_organization et ont la propriété suivante :

- `rdf:type rdfs:Organization` : indique que l'entité est de type "Organization".
- `rdfs:name string` : représente le nom de l'organisation.

Les genres sont représentés par l'entité :name_genre et ont la propriété suivante :

- `rdf:type rdfs:Genre` : indique que l'entité est de type "Genre".
- `rdfs:label string` : représente le libellé du genre.

Un extrait du fichier rdf (turtle) obtenu est comme suit:

```
:gett_the_trial_of_viviane_amsalem
    rdf:type                rdfs:Movie ;
    rdfs:aggregateRating    [ rdf:type          rdfs:AggregateRating ;
                              :audienceScore    "81.0"^^xsd:double ;
                              :tomatoMeter      "100.0"^^xsd:double
                            ] ;
    rdfs:author             :RonitElkabetz , :ShlomiElkabetz ;
    rdfs:datePublished      [ rdf:type                  rdfs:DatePublished ;
                              :releasedStreamingDate    "2015-06-09"^^xsd:date ;
                              :releasedTheaterDate      "2015-02-15"^^xsd:date
                            ] ;
    rdfs:director           :ShlomiElkabetz , :RonitElkabetz ;
    rdfs:duration           "115.0"^^xsd:double ;
    rdfs:genre              :Drama ;
    rdfs:inLanguage         "Hebrew" ;
    rdfs:maintainer         :MusicBoxFilms ;
    rdfs:title              "Gett: The Trial of Viviane Amsalem" ;
    :boxOffice              "$987.8K" .

:RichardCurtis  rdf:type  rdfs:Person ;
    foaf:name  "Richard Curtis" .

:War    rdf:type    rdfs:Genre ;
    rdfs:label  "War" .

:UniversalPictures  rdf:type  rdfs:Organization ;
    rdfs:name  "Universal Pictures" .
```

3 Requêtes Sparql

```
# This SPARQL query aims to retrieve the titles and durations of movies, ordered by duration in descending order, and limited to the top 10 longest movies.

    prefix :      <http://localhost:3333/>
    prefix foaf:  <http://xmlns.com/foaf/0.1/>
    prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
    prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
    prefix vcard: <http://www.w3.org/2006/vcard/ns#>
```

```
prefix xsd:   <http://www.w3.org/2001/XMLSchema#>

SELECT ?title ?duration
WHERE {
  ?movie rdf:type rdfs:Movie ;
         rdfs:title ?title ;
         rdfs:duration ?duration .
}
ORDER BY DESC(?duration)
LIMIT 10
```

```
# This SPARQL query retrieves the titles and box office earnings of movies. It converts the box office earnings to decimal format,
# handling values expressed in thousands (ending with "K") or without, removing the dollar sign if present.
# The results are ordered by box office earnings in descending order and limited to the top 20 highest-earning movies.

prefix :      <http://localhost:3333/>
prefix foaf:  <http://xmlns.com/foaf/0.1/>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix vcard: <http://www.w3.org/2006/vcard/ns#>
prefix xsd:   <http://www.w3.org/2001/XMLSchema#>

SELECT ?title ?boxOffice
WHERE {
  ?movie rdf:type rdfs:Movie ;
        rdfs:title ?title ;
        :boxOffice ?boxOfficeValue .

  BIND(IF(STRLEN(?boxOfficeValue) > 1 && STRENDS(?boxOfficeValue, "K"),
          xsd:decimal(REPLACE(REPLACE(?boxOfficeValue, "\\$", ""), "K", "")),
          xsd:decimal(REPLACE(?boxOfficeValue, "\\$", ""))
        ) AS ?boxOffice)
}
ORDER BY DESC(?boxOffice)
LIMIT 20
```

```
# This SPARQL query aims to find pairs of movies directed by the same
```

```
        director where the first movie was released after the second one.

prefix :      <http://localhost:3333/>
prefix foaf:  <http://xmlns.com/foaf/0.1/>
prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#>
prefix vcard: <http://www.w3.org/2006/vcard/ns#>
prefix xsd:   <http://www.w3.org/2001/XMLSchema#>

SELECT ?releaseDate1 ?releaseDate2 ?title1 ?title2
WHERE {
  {
    SELECT DISTINCT ?director ?releaseDate1 ?releaseDate2 ?title1 ?title2
    WHERE {
      ?director rdf:type rdfs:Person .
      ?movie1 rdf:type rdfs:Movie ;
              rdfs:title ?title1;
              rdfs:director ?director ;
              rdfs:datePublished/:releasedTheaterDate ?releaseDate1 .
      ?movie2 rdf:type rdfs:Movie ;
              rdfs:title ?title2;
              rdfs:director ?director ;
              rdfs:datePublished/:releasedTheaterDate ?releaseDate2 .
      FILTER (?movie1 != ?movie2)
      FILTER (?releaseDate1 > ?releaseDate2)
    }
    LIMIT 20
  }
}
```

4 Présentation de l'application

La capture d'écran ci-dessous est une page de mon application.

Pour l'instant, l'application a trois fonctions principales:

1. Camembert
2. Diagramme à barres
3. Diagramme de force

Chaque graphique contient un point d'arrivée(#endpoint) pour la base de données locale et un point d'arrivée(#endpoint) pour la base de données distante, ainsi qu'un bouton de test du cache. Les détails de l'installation et de l'utilisation sont donnés dans la documentation git myGit, je ne les répéterai donc pas ici.

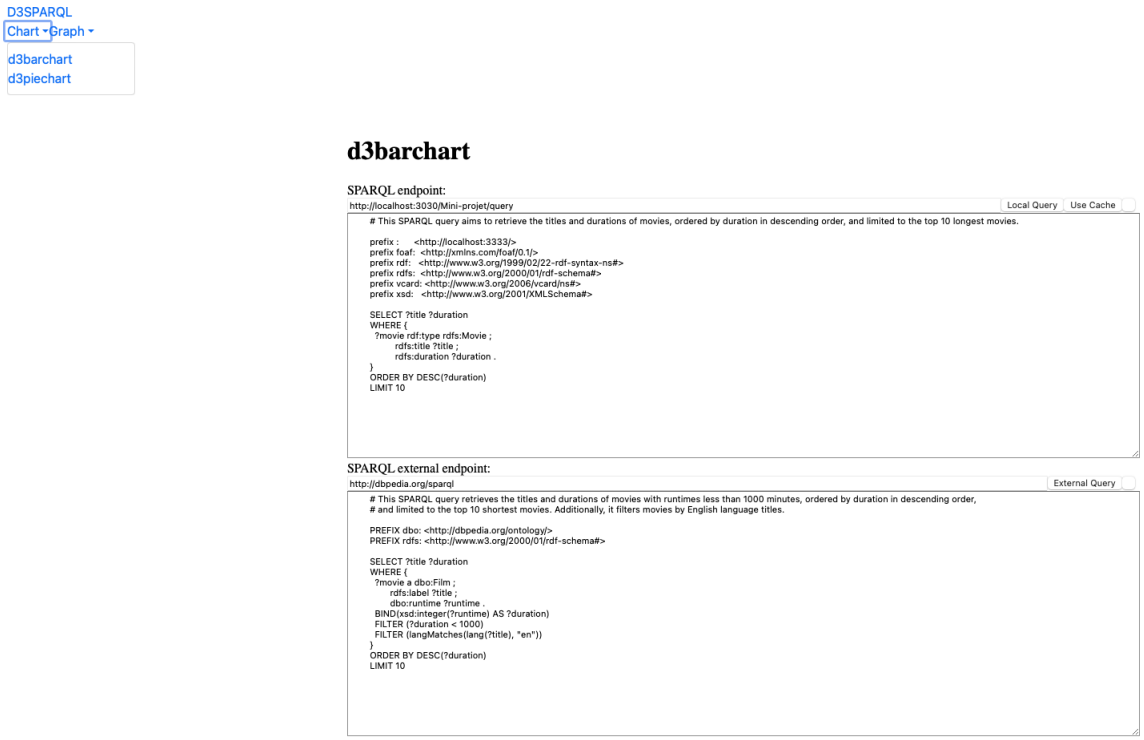

Figure 1: Application fonctionnalité 1

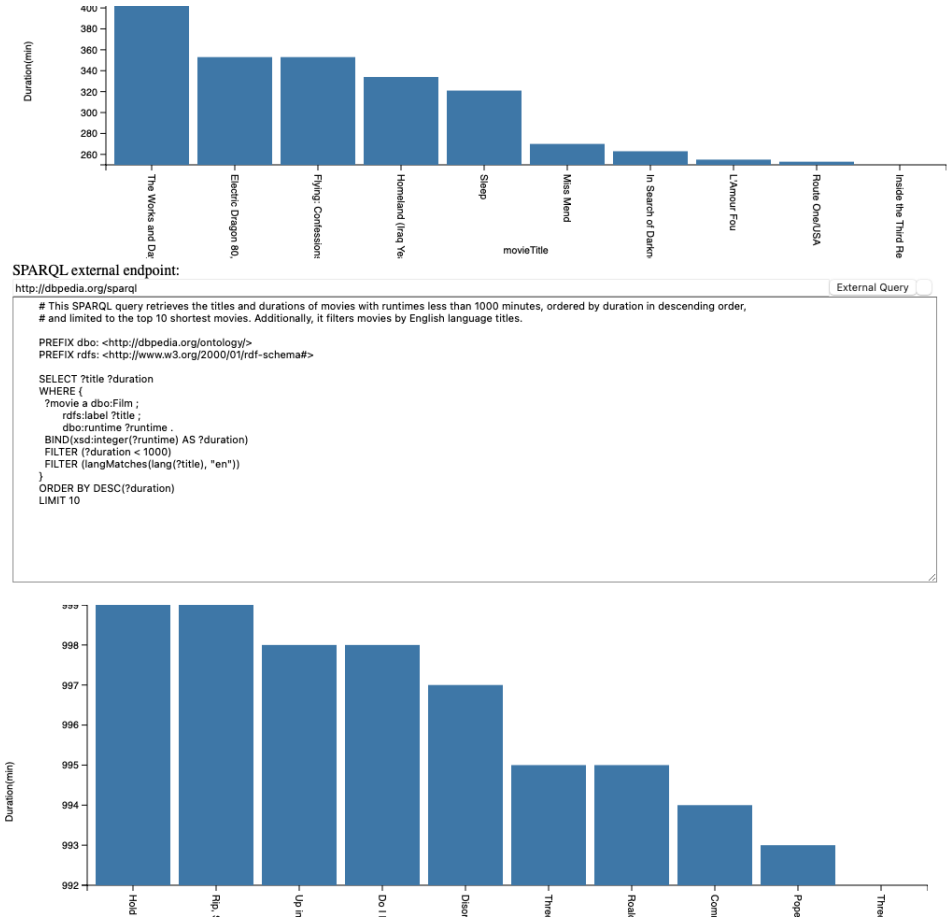

Figure 2: Application fonctionnalité 2