Didier Bastien - Corentin Grandchamps

Projet Web Sémantique

Introduction

Ce rapport est consacré à la présentation du travail réalisé lors du projet de web sémantique en M2ICE à l'université Toulouse 2 Jean Jaurès. Ce projet implique de réaliser une application MashUp sémantique. L'objectif de ce type d'application est de décrire à partir d'une ou plusieurs ontologies (i.e., un vocabulaire commun) des données issues de différentes sources disponibles sur le Web et de permettre, à partir de ces données et des ontologies, la déduction de nouvelles données offrant ainsi une nouvelle vue sur les informations disponibles. Le rôle de l'ontologie dans ce type d'application est double :

• d'une part elle devra permettre de « sémantiser » différentes sources de données ouvertes semi-structurées ;

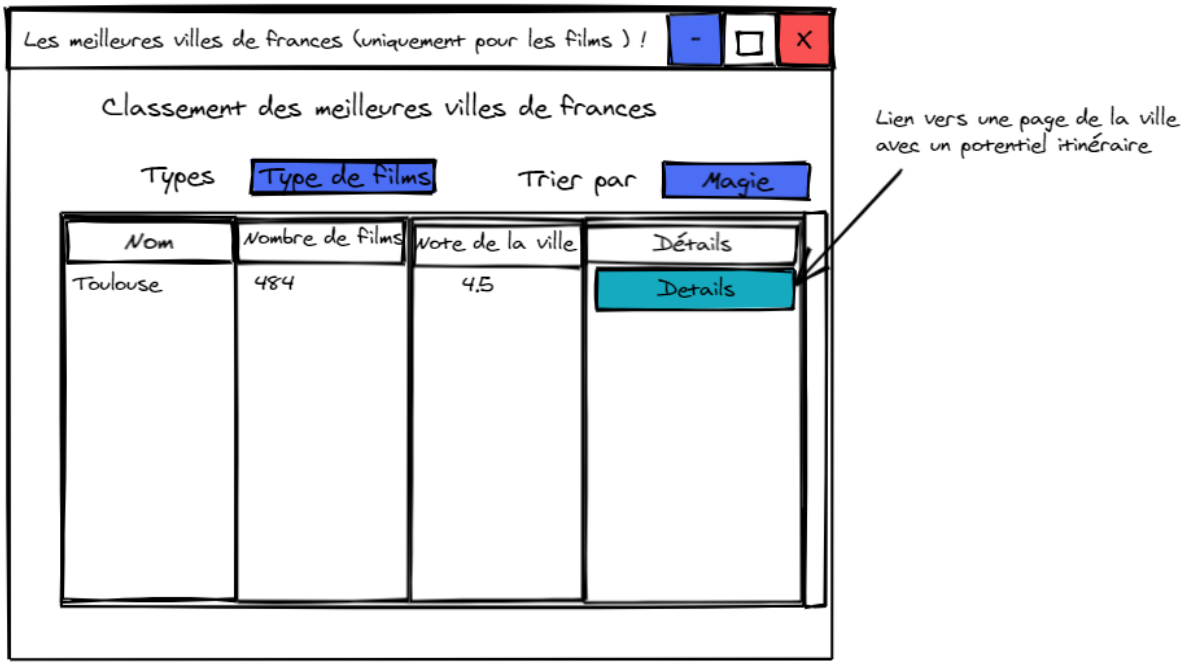
• d'autre part elle devra aider à la mise en place d'une application permettant de valoriser de nouveaux faits générés grâce à l'ontologie une fois les données sémantisées.

Spécification de l’application:

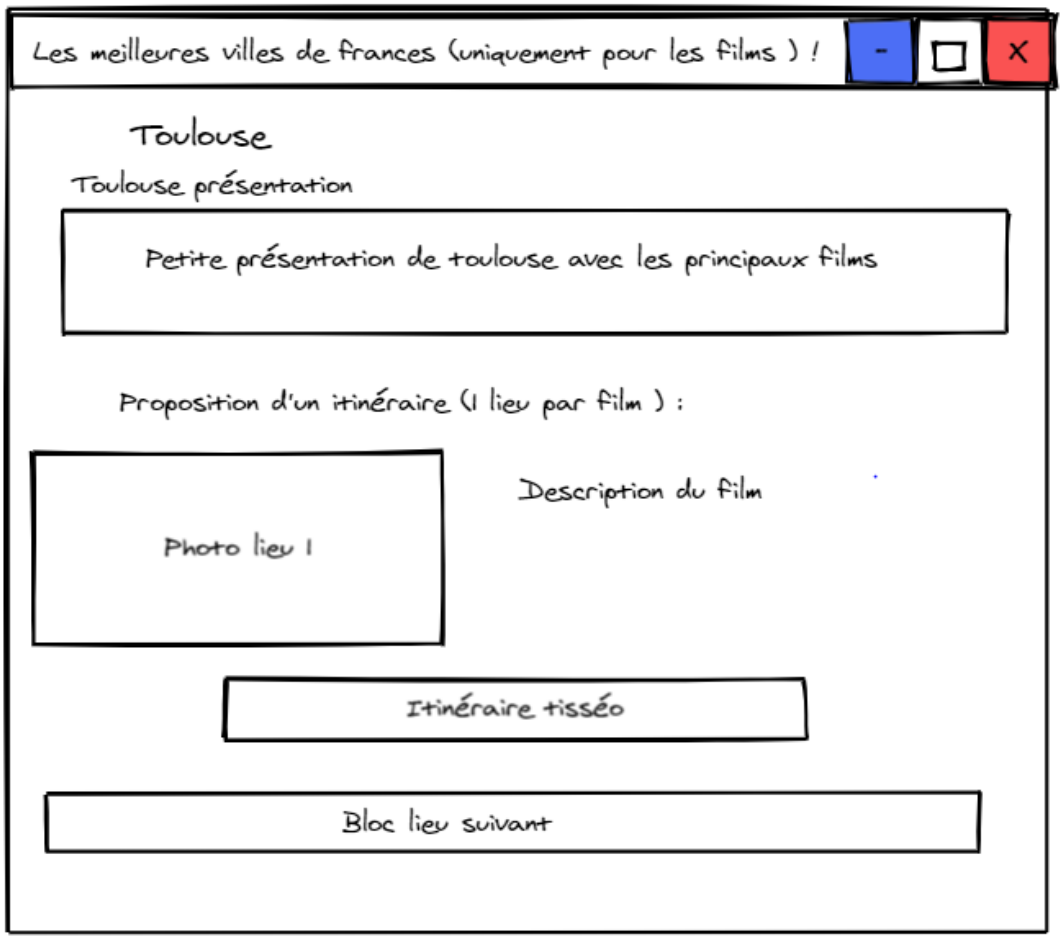
Oskour est une application MashUp sémantique réalisée par Didier Bastien et Grandchamps Corentin qui permet de récupérer la liste des films produits en France puis de classer les diverses villes de France selon la qualité perçue par le public et les critiques ainsi que la popularité des films produits dans la ville. Le classement est fait selon les différents types de films ou médias (série TV , film, ou même différents types de films comme les comédies, documentaires etc..). Cette fonctionnalité s'adresse plutôt aux réalisateurs et leur permettrait de faire un choix éclairé sur le choix de la ville . Afin de choisir la ville la plus adaptée pour leur film selon le genre ou le type de vidéo.

Enfin, l'autre fonctionnalité de notre application est la présentation d’une ville en particulier, et propose un itinéraire pour relier les principaux lieux de tournage de cette ville, permettant une découverte culturelle de la ville à vélo.

Maquettes de l’application

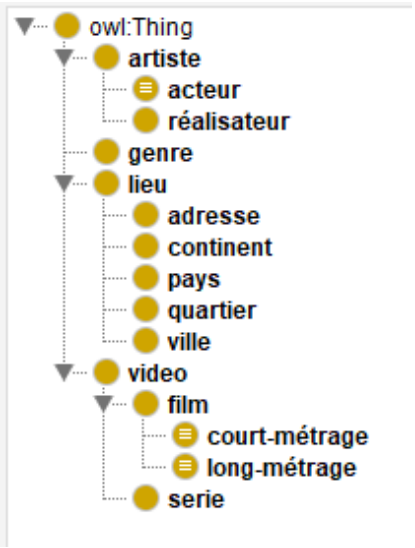


Maquette de la page d’accueil



Maquette de la page des itinéraires

Ontologie:



Descriptif des classes de l'ontologie

Cette ontologie a été créée en utilisant l'ontologie CinémaTP, néanmoins certains changement ont été fait. Les principaux changements sont :

- la suppression des genres pour les acteurs, réalisateurs et comédiens.
- Un lieu peut maintenant être un continent, un pays, une ville , un quartier et une adresse (ajout du type quartier)
- Création d'une classe Vidéo qui peut être un film (court métrage ou long métrage) ou une série
- Création d'une classe Genre
-

Chaque vidéo dispose d'une note, d'une date de sortie, d'un nombre de votes. d'un genre.
Chaque ville dispose d'une description.

Source de données utilisée pour peupler l'ontologie :

- https://www.data.gouv.fr/fr/datasets/lieux-de-tournage-a-paris-2/
- http://data.montpellier3m.fr/dataset/tournages-montpellier
- http://fr.dbpedia.org/
-

Ces données ont été extraites au format csv puis formater à l'aide de la route /transform du serveur node. Une fois que les données ont été transformés correctement, elles sont enrichies grâce aux api suivantes :

IMDB :

https://rapidapi.com/hmerritt/api/imdb-internet-movie-database-unofficial

OmdbApi :

http://www.omdbapi.com/

L'api IMDB nous permet de retrouver pour un nom de film son numéro imdb. Ce numéro sert ensuite pour récupérer les informations comme le nombre de votes, la note du film, la liste des acteurs ... de l'api OMDB API ce qui permet d'enrichir le csv qui sera ensuite inséré dans fuseki.

L'ensemble des données n'a pas été transformé et enrichi avec les apis pour la simple raison que nous sommes limités à 1000 requêtes sur l'api OMDB API. Ceci explique aussi pourquoi on ne retrouve que des films ayant été tournés à Paris dans la base (seulement les 1000 premières lignes du csv sont enrichies, ce qui correspond uniquement à des films produit à Paris les csv de films sont ajoutés les uns à la suite des autres après la transformation), Néanmoins le processus est automatique et pourrait théoriquement fonctionner avec l'ensemble du fichier csv.

Après transformation on se retrouve pour chaque film avec les informations suivantes :

- année du tournage
- Type du tournage
- Nom du tournage
- Nom du réalisateur
- Adresse du lieu de tournage
- Acteurs
- La durée de la vidéo
- Genres
- Metascore
- Date de sortie
- Note
- Nb_votes

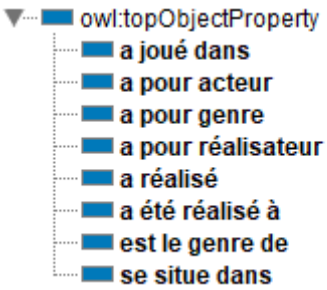
Enfin nous récupérons aussi des données de dbpedia pour chaque ville, en effet nous requérons pour chaque ville avant l'insertion, la description de la ville. On peut imaginer une future version ou l'on pourrait donner la description dans plusieurs langues, et en liant chacune de nos classes a leurs équivalent sur dbpedia.

Insertion des Données

L'insertion des données est un processus automatique qui est appelé en allant sur la route /insert. Cette requête va provoquer l'insertion automatique dans fuseki du csv transformed_datas.csv située dans le dossier server_node. En cas d'erreur la requête

renvoie un code 500, en cas de succès elle renvoie un code 200 avec pour paramètre result: “ok”

Inférences



Descriptif des propriétés de l’ontologie

Un acteur peut être inféré par son apparition dans un film grâce aux deux propriétés **“a joué dans”** (placée du côté de l’acteur) et **“a pour acteur”** (placée du côté du film).
Ces deux inférences sont inverses.

Le genre d’un film peut être inféré par son association à un film grâce aux deux propriétés **“a pour genre”** (placée du côté du film) et **“est le genre de”** (placée du côté du genre).
Ces deux inférences sont inverses.

Le réalisateur d’un film peut être inféré par son association à un film grâce aux deux propriétés **“a pour réalisateur”** (placée du côté du film) et **“est le réalisateur de”** (placée du côté du réalisateur).
Ces deux inférences sont inverses.

Le lieu de tournage d’un film peut être inféré par son association à un film grâce à la propriété **“a été réalisé à”** (placée du côté du film).

Enfin la propriété **“se situe dans”** permet d’inférer de façon transitive l’imbrication des lieux (Continent > Pays > Ville > Quartier > Adresse).

Grace a ces propriétés nous pouvons nous permettre de préciser la majeure partie des propriétés d’un objet dans les films, la majeure partie des propriétés étant liée à la classe Film. (Pour plus de précisions voir l’insertion de genre vis-à-vis de l’insertion de film disponible en annexe).

Alignement

Afin de compléter notre ontologie, nous avons choisi d'établir un alignement avec l'ontologie DBPedia qui nous a fourni toutes les informations nécessaires et relatives aux villes de France.

Dans un premier temps nous avons tenté de réaliser un alignement pur, en SPARQL, mais les résultats n'ont pas été à la hauteur de nos attentes, et face à la difficulté et avec le peu de temps que nous avions, nous avons choisi de nous y prendre différemment.

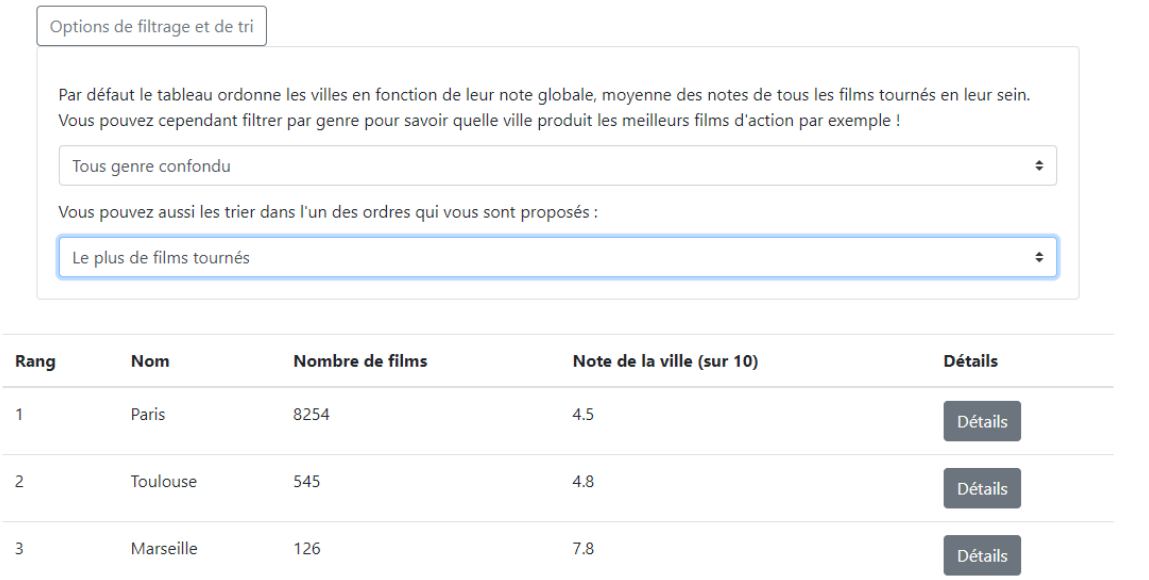
Nous avons donc récupéré les données de DBPedia, et, via un système de back-end, nous avons fusionné les données utiles de cette ontologie, avec les nôtres.

Bien que notre travail soit plus de la complétion de données qu'un alignement pur, nous avons réussi à fusionner les données des deux ontologies pour obtenir notre base d'information utilisable facilement par notre application web.

L'application web

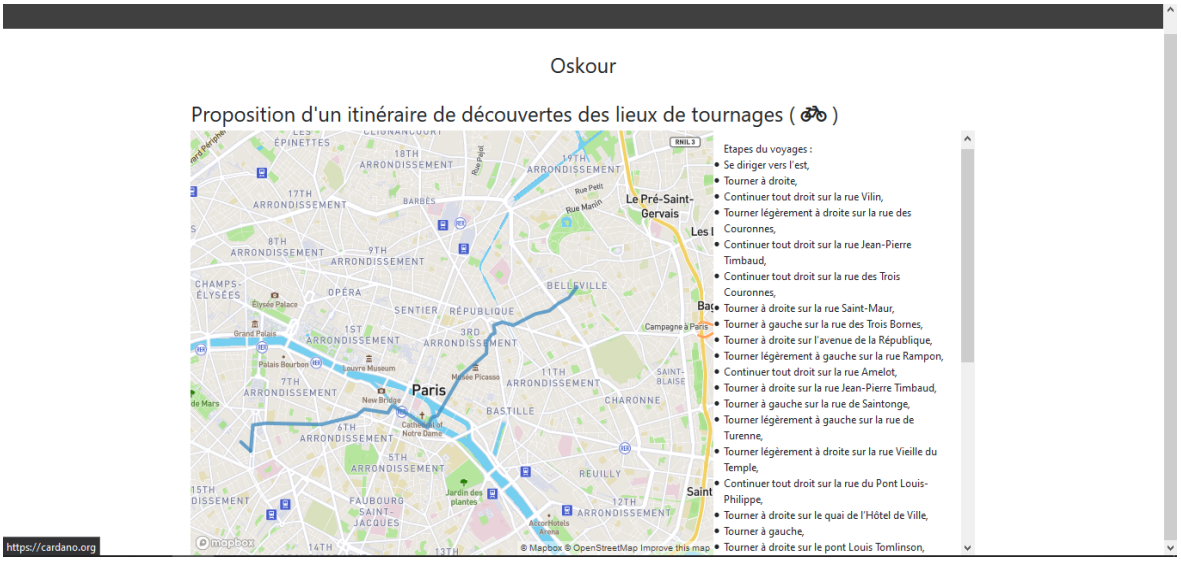
Vue de la page d'accueil de l'application

La page principale du site web (présentée ci-dessus) permet à l'utilisateur de s'informer sur la popularité globale des films tournés dans les différentes villes de France. Il peut filtrer par genre, et trier par ordre alphabétique, par nombre de films tournés, ou encore (et c'est ce qui est le plus intéressant) par la note moyenne globale issue de toutes les notes des films tournés dans la ville, comme le montre la capture d'écran ci-dessous :



Vue des options de filtrage et de tri

En cliquant sur le bouton ***Détails*** de la ville, l'utilisateur est redirigé sur une page dédiée à la ville, dans laquelle un descriptif de la ville se trouve, avec notamment une carte permettant de parcourir les sites de tournage des meilleurs films produits dans la ville. Nous avons réussi à intégrer la carte et à dessiner un tracé en fonction d'étapes :



Vue d'une page "Détails" consacrée à une ville

Bilan du travail réalisé

Le serveur node est fonctionnel, l'interface correspond a nos maquettes de départ, chacune des fonctionnalités marchent, néanmoins on peut facilement imaginer une version 2.0 de ce projet ou l'on pourrait lier chacunes de nos classes avec les classes de dbpedia, ce qui permettrait de pouvoir gérer plusieurs langues sur notres site et d'afficher les affiches de film ou les photos des lieux sur l'itinéraire pour se retrouver avec un rendu plus similaire à celui de la maquette.

La raison pour laquelle ces changements n'ont pas été implémentés dans cette version est le manque de temps.

Au final ce projet était très instructif et nous a permis d'appliquer les connaissances apprises en cours sur le web sémantique de manière ludique.

Annexes

Exemples de requêtes sparql :

Exemple d'insertion (Genre d'une vidéo) :

L'insertions des genres des films est réalisés de la manière suivante :

On convertit le csv au format json, grâce au package npm csvtojson, puis pour chaque ligne du tableau json obtenu on effectue le traitement suivant:
on sépare les genres afin d'obtenir un tableau de genre,
puis pour chaque élément de ce tableau on élimine les caractères spéciaux (accents ..), on groupe les genres similaires ensembles par exemple les genres non définis sont groupé sous la catégorie n/a, les genres music et musical sont groupé en un seul genre music. Enfin on ajoute chaque nouveau genre dans un tableau qui comporte la liste de tous les genres.

Une fois le tableau obtenu, on appelle pour chaque élément la fonction insert_genre qui retourne une string permettant d'insérer ce genre dans l'ontologie. Cette string est ajouté à la query globale qui sera envoyer au serveur fuseki via une requête http.

```
var json_datas = wdatas.json_datas;
var query = wdatas.query;

var tab_genre = []

json_datas.map(function(elt)
{
    var genres = elt["Genre"].split(",");
    for(var j = 0; j<genres.length; j++)
    {
        var genre = genres[j].toLowerCase().trim();
        genre = cleanUpSpecialChars(genre);

        if(genre == "" || genre == "n/a")
        {
            genre = "n/a";
        }

        if(genre == "musical")
        {
            genre = "music";
        }

        if(tab_genre.indexOf(genre) == -1)
        {
            tab_genre.push(genre);
        }
    }
```

Création du JSON

```
function insert_realisateur(nom_real, index)
{
    var str = " :realisateur"+index+" rdfs:label \""+nom_real+"\".\n"
    str += ":realisateur"+index+" a :realisateur .\n"
    return str;
}

function insert_genre(nom_genre, index)
{
    var str = " :genre"+index+" rdfs:label \""+nom_genre+"\".\n"
    str += ":genre"+index+" a :genre .\n"
    return str;
}
```

Fonctions d'insertion des données (réalisateurs et genres)

L'insertion des lieus, réalisateurs, et acteurs repose sur le meme principe. POur plus de détails consulter le fichier server_node/controllers/insert_datas.js

Exemple d'insertion d'un film :

L'insertion d'un film se fait en deux temps. On construit d'abord un dictionnaire de tous les films qui pour chaque film comporte les informations suivantes :

```
var tpm_film = {

    nom_film:       elt["nom_tournage"],
    types:          elt["type_tournage"],
    realisateur:    get_correspondant_index(tab_real_row, tab_real, "realisateur"),
    actors:         get_correspondant_index(tab_actors_row, tab_actors, "acteur"),
    genres:         get_correspondant_index(tab_genre_row, tab_genre, "genre"),
    duree:          elt["runtime"].replace(" min", ""),
    lieu_tournage:  get_correspondant_index(tab_adresse_row, tab_adresse, "adresse"),
    ratings:        elt["Rating"],
    nb_ratings:     elt["nb_votes"],
    annee_tournage: elt["annee_tournage"]

}

list_films[elt["nom_tournage"]] = tpm_film;
```

Structure d'un objet film

Enfin une fois le dictionnaire complet, on parcourt chaque film et on ajoute à la query principale le type du film, les acteurs du film, le genre du film ect…

```
for (var film in list_films)
{

    var type = list_films[film]["types"].indexOf("Série") != -1 ? "serie" : "film"
    query += ":video"+index_film  + " rdfs:label \"" + film + "\".\n" ;
    query += ":video"+index_film+" a :"+type+" .\n";

    for(var i = 0; i<list_films[film]["lieu_tournage"].length; i++)
    {
        query += ":video"+index_film+" :aEteRealiseA :"+list_films[film]["lieu_tournage"][i]+" .\n";
    }

    for(var i = 0; i<list_films[film]["actors"].length; i++)
    {
        query += ":video"+index_film+" :aPourActeur :"+list_films[film]["actors"][i]+" .\n";
    }

    for(var i = 0; i<list_films[film]["genres"].length; i++)
    {
        query += ":video"+index_film+" :aPourGenre :"+list_films[film]["genres"][i]+" .\n";
    }
```

Construction de la requête SPARQL pour un film

Au final on se retrouve avec une requête complète comme celle présente dans le fichier server_node/query_str.txt

Comment lancer l'application

- ouvrez un terminal
- cd path_to_application
- cd server_node
- npm install
- node app.js
- ouvrez un autre terminal et lancez fuseki sur le port 3030

- Ouvrez un navigateur sur la page localhost:3000/insert pour insérer les données du fichier transformed_datas.csv
- allez sur la page d'accueil : http://localhost:3000/home pour accéder à la liste des villes

  Félicitation, vous venez de lancer avec succès Oskour !
-