Connaissances et raisonnement

Création d'une ontologie

Sujet : Proposition d'une ontologie et d'un ensemble de règles, sur un sujet à déterminer, en vue d'applications potentielles qui seront précisées.

Lien du repository GitHub : https://github.com/romain-sen/CR_project

Table des matières

Cadrage du sujet

Dans un monde de plus en plus connecté et numérisé, l'accès à une multitude de données en ligne offre des possibilités sans précédent de recherche, d'analyse et de compréhension des phénomènes du monde réel. C'est dans ce contexte que s'inscrit notre projet, qui vise à construire notre ontologie avec nos propres données de suivi de séries télévisées. Le but étant ensuite d'exploiter les vastes ressources disponibles sur Internet afin de les combiner avec notre ontologie pour approfondir nos connaissances sur les séries que nous regardons.

Le but de ce projet est de synthétiser une base de connaissances ciblée sur les séries télévisées que nous suivons et d'approfondir notre suivi de nos séries grâce à une ontologie. Celle-ci doit pouvoir nous donner de nouveau point de vue sur nos données et être capable de se connecter à Wikidata pour trouver les informations qui lui manque. Nous visons à résoudre plusieurs questions clés :

- Comment avoir nos données de suivi de série ?
- Comment concevoir une ontologie qui reflète parfaitement les caractéristiques de nos données ?
- Comment exploiter au maximum notre ontologie ?
- Quelle méthode adopter pour collecter et préparer de manière optimale les données de Wikidata ?
- Quelles stratégies de requêtes SPARQL peuvent être employées pour filtrer des informations précises et significatives à partir de Wikidata ?

En abordant ces questions, le rapport couvrira la collecte et la préparation des données, la création d'une ontologie spécialisée, et l'analyse des données, en ajoutant celles de Wikidata via SPARQL. Cette démarche vise à établir une ressource informative et analytique de référence pour notre suivi personnel de série, enrichissant ainsi nos connaissances et notre analyse médiatique sur ce que nous regardons.

Choix de la donnée

Afin de réaliser une ontologie originale et intéressante, il est nécessaire de trouver de la donnée qui le soit aussi. Pour ce faire, nous avons fait des recherches pour trouver des jeux de données sur le thème que nous avons choisi : le cinéma. Cependant, le jeu de données de IMDB disponible publiquement a déjà été traité lors du TD 4 de ce cours. Nous avons donc opté pour une approche plus originale.

Avec les lois en vigueurs pour la protection de données et de la vie privées (RGPD), il est possible de demander à n'importe quelle entreprise d'obtenir toutes les données qui nous concernent personnellement.

Il existe une application du nom de 'TV Time' qui permet de suivre les films et séries que nous regardons. A chaque fois que l'on voit un film ou un épisode de série, on le note dans l'application. Voici une liste nous exhaustive des fonctionnalités de l'application :

- S'abonner aux séries qui nous intéressent et être avertit lorsque de nouveaux épisodes sortent.
- Réagir et noter sur cinq chaque films et épisodes que nous voyons.
- Commenter des films et épisodes.

Dans notre binôme, Romain utilise quotidiennement cette application. Il a donc une quantité de données intéressante qui lui est associée. Nous avons ainsi pu récupérer un jeu de donnée brut en faisant une demande particulière à l'entreprise en question. Nous récupérons un fichier compressé avec dedans une trentaine de fichier csv.

La donnée utilisée pour ce projet est donc brut et sortie d'une situation réelle. Elle n'est donc pas aussi propre que ce que nous pourrions trouver avec un jeu de données classique sur internet. Un autre point à souligner est que cette donnée n'est pas documentée. L'entreprise nous renvoie une trentaine de fichier csv qui représentent directement leur base de donnée. Il y a donc des noms de variables parfois incompréhensibles sans documentation, ou encore des données manquantes puisqu'il ne s'agit que de la donnée qui me concerne directement. En effet, nous avons la liste des séries que j'ai vu avec le nom de la série, mais aucunes informations sur la série en question, puisque celle-ci se trouve dans une autre table qui ne me concerne pas. L'entreprise n'a donc pas à nous envoyer ces données-là. C'est aussi le cas par exemple pour les émotions. Nous avons l'id de l'émotion en question, mais aucune information sur quelle émotion cela correspond.

Enfin, après analyse de nos données, les données concernant les filmes sont assez pauvres et plus difficile à exploiter que celles concernant les séries. C'est surement parce que TV Time a été créé dans un premier temps pour suivre les séries et non les films. Nous faisons donc le choix de ne traiter que les séries dans ce projet.

Traitement de la donnée

Après une longue exploration des données récupérées afin de comprendre ce que nous avons et ce qui est pertinent pour notre projet, nous sélectionnons un total de six fichiers csv :

- emotions_episode_votes
- episode_comment
- followed_tv_show
- ratings_episode_votes
- rewatched_episode
- seen_episode

Nous devons maintenant traité nos données avant de pouvoir les utiliser pour notre projet.

Tout d'abord, nous supprimons toutes les lignes où le nom de la série est vide dans « seen_episode ».

Ensuite, nous retirons les colonnes inutiles pour ne garder que celles qui nous intéressent et simplifier nos csv.

Nous procédons aussi à un formatage des données. En effet, dans plusieurs des csv, les dates sont aux formats « 2022-09-06 20:42:29 » mais dans certains, nous avons 2022-09-06 20:42:29 +0000 UTC. Cela prose problème par la suite pour caster les dates. Nous retirons donc le « +0000 UTC ».

Nous renommons les colonnes pour qu'elles soient toutes en minuscules, et que les noms de variables soient consistent (created en created_at ou encore id en comment_id).

Dans le fichier « episode_comment », nous faisons le choix de ne garder que les commentaires direct aux épisodes et de supprimer les commentaires fait sur d'autres commentaires. Pour cela, on filtre sur le champs « depth ».

Enfin, nous choisissons de fusionner « seen_episode » et « rewatched_episode » afin d'avoir un fichier final qui répertorie tous les épisodes vus et le nombre de fois qu'ils ont été revus.

Le code qui permet le traitement des données est retrouvables dans le notebook « data_cleaning.ipynb »

Création de l'ontologie

Pour créer l'ontologie, nous avons le choix entre utiliser protégé puis « ontotext refine » avant de charger tout cela dans GraphDB, ou créer l'ontologie en python en utilisant « owlready2 ».

Puisque nous avons déjà notre notebook pour prétraiter la données, et que le code est plus simple à partager et documenter que des manipulations sur des interfaces graphiques comme protégé, nous choisissons l'option avec owlready2 en python. Le code est disponible dans le notebook « ontologie_tv_time.ipynb ».

Pour notre schéma, nous nous basons sur les données que nous avons avec nos fichiers csv. Nous choisissons ainsi six classes qui nous semblent les plus pertinentes pour notre sujet. Nous avons donc :

- TVShow : c'est la classe qui représente une série. Elle contient le nom de la série.
- Episode : c'est la classe qui représente un épisode de série.
- User : c'est la classe qui représente l'utilisateur. Il n'y a dans ce jeu de données qu'un seul utilisateur (Romain) mais avec cette classe, il serait possible d'en avoir d'autres.
- Comment : c'est la classe qui représente les commentaires.
- Rating : c'est la classe qui permet de noter les épisodes avec une note sur 5.
- Emotion : c'est la classe qui permet de noter les épisodes avec une émotion.

Ensuite, nous définissons les propriétés. En voici une liste non exhaustive :

- aPourTitre : associe un titre a un TVShow
- aPourSaison, aPourNumero : information propre à un épisode pour l'identifier
- aPourAnnee, aPourMois : donne la date de quand l'épisode a été visionner
- aPourContenu : donne le contenu d'un commentaire
- nbLikes : donne le nombre de like qu'a reçu un commentaire
- aPourEmotion : donne la valeur (id) de l'émotion

Remarque : à cause de problème de parsing qui rendent l'ontologie inconsistente, nous avons enlever les dates complètes et extrait seulement l'année et le mois. Cela est lié au fait que **date** est dans le namespace xsd mais que owlready2 nous permet seulement de manipuler rdf et rdfs.

Enfin, nous définissons les relations de entre nos entités. En voici une liste non exhaustive :

- is_episode_of : un épisode appartient à une série
- user_posts_comment : un user peut écrire un commentaire. L'inverse de cette relation est posts_by qui dit qui a écrit le commentaire.
- comments_on : le commentaire est associé à un épisode.
- follows : un user peut suivre une série
- user_rates et rates_episodes : permet à un user de créer une note et de l'associé à un épisode.

Finalement, nous faisons tourner le raisonneur pellet et nous sauvegardons l'ontologie.

Aller plus loin

Créer des classes complexes

Maintenant que nous avons une ontologie avec toutes les données que nous avons récupérées par TV Time, nous pouvons créer des classes plus complexes afin d'interroger plus en profondeur nos données.

TVShowEngageUser

Nous définissons donc la classe TVShowEngageUser. Cette classe reprend les TVShow qui « engagent » un utilisateur. Pour engager un utilisateur, nous choisissons les critères suivants :

- L'utilisateur doit avoir commencé la série et vu au moins un épisode
- L'utilisateur doit avoir poster un commentaire sur un épisode de la série
- L'utilisateur doit suivre la série

Voici la règle que nous écrivons pour cette classe :

```
    rule1.set_as_rule('''
        has_started(?user, ?tv_show),
        has_watched(?user, ?episode),
        follows(?user, ?tv_show),
        user_posts_comment(?user, ?comment),
        comments_on(?comment, ?episode),
        is_episode_of(?episode, ?tv_show)
        -> TVShowEngageUser(?tv_show)
    ''')
```

Le résultat nous donne bien la liste des séries dont nous avons commenté au moins un épisode, que nous suivons et dont nous avons vu au moins un épisode.

TVShowInfluencer

Nous cherchons à connaître les TVShow pour lesquels Romain la somme des likes de tous les commentaires fais sur les épisodes de cette série est supérieure à 50. Nous considérons alors que Romain influence sur cette série. Pour ce faire, nous créons une nouvelle relation 'is_influenced_by' qui dit qu'un TVShow est influencé par un User. Ensuite, nous écrivons du code python afin créer les relations dans notre ontologie.

```
    for user in onto.User.instances():
        for tv_show in onto.TVShow.instances():
            sum_likes = 0
```

```
            for episode in user.has_watched:
                if tv_show in episode.is_episode_of and
episode.is_commented_on_by :
                    # print(f'{episode.is_commented_on_by}')
                    # print(f'User {user} has commented on
episode {episode} of TV show {tv_show}')
                    for comment in
episode.is_commented_on_by:
                        # print(f'{comment.nbLikes}')
                        sum_likes += comment.nbLikes
            if sum_likes > 50:
                tv_show.is_influenced_by.append(user)
```

Le résultat est très précis, en relevant donc deux séries ou la somme des likes sur les commentaires est supérieure à 50 : The Night Agent (2149 likes) et A Spy Among Friends (61 likes).

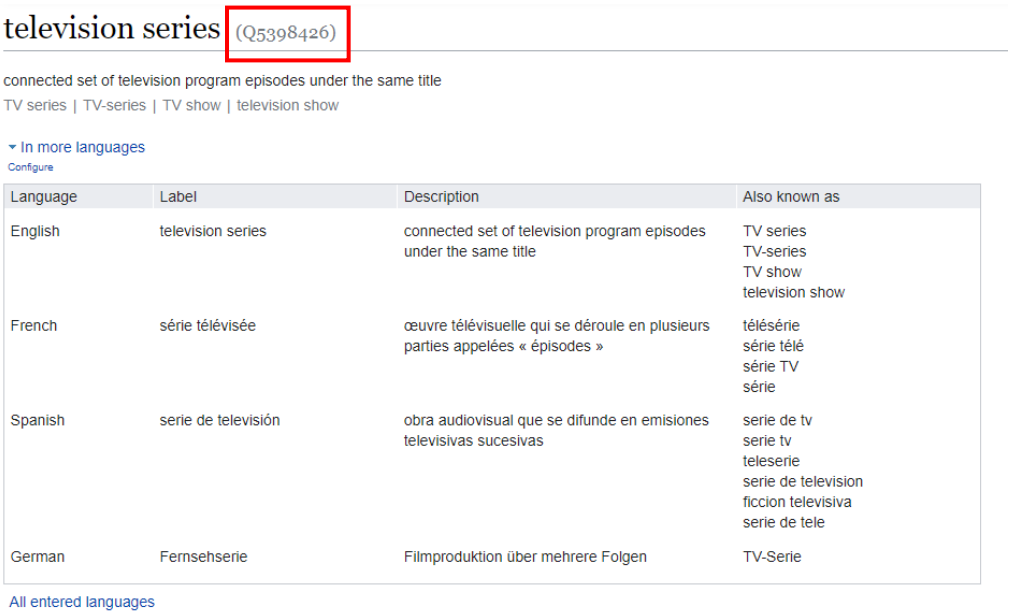
Faire des requêtes avec Wikidata

Wikidata

Wikidata est conçue pour collecter et organiser des données structurées afin de soutenir efficacement des projets tels que Wikipédia. Cette base de données stocke des informations sur une multiple de sujets : (individus, lieux, événements historiques, œuvres d'art, livres ...)

Les informations dans Wikidata sont structurées en utilisant le format RDF, qui organise les données en triplets composés d'un sujet, d'un prédicat, et d'un objet. Cette structure permet de former des déclarations précises et interconnectées sur les entités concernées.

Les entités constituent les noyaux desquels s'articulent les données de Wikidata. Chaque entité, représentant un objet du monde réel ou un concept, se voit attribuer un identifiant unique. Ces entités peuvent être dotées de diverses propriétés qui en décrivent les caractéristiques essentielles, comme le titre d'une série télévisée, sa date de première diffusion, ses acteurs principaux, et plus encore. Ces liens sont établis à travers des propriétés spécifiques qui définissent les connexions. Par exemple, une propriété peut indiquer le type d'une entité (film, séries....) ou les acteurs participant à une œuvre cinématographique. Ces liens facilitent une compréhension approfondie des relations complexes entre les entités.



television series (Q5398426)

connected set of television program episodes under the same title

TV series | TV-series | TV show | television show

In more languages

Configure

Language	Label	Description	Also known as
English	television series	connected set of television program episodes under the same title	TV series TV-series TV show television show
French	série télévisée	œuvre télévisuelle qui se déroule en plusieurs parties appelées « épisodes »	télésérie série télé série TV série
Spanish	serie de televisión	obra audiovisual que se difunde en emisiones televisivas sucesivas	serie de tv serie tv teleserie serie de television ficcion televisiva serie de tele
German	Fernsehserie	Filmproduktion über mehrere Folgen	TV-Serie

All entered languages

SPARQL

Les requêtes SPARQL offrent une méthode puissante pour interroger les données de Wikidata, permettant aux utilisateurs d'extraire des informations spécifiques sur des sujets variés tels que, pour notre sujet, les séries télévisées, leurs genres, acteurs, réalisateurs ... Ces requêtes sont essentielles pour analyser et explorer les vastes données disponibles, offrant des insights sur des aspects tels que le pays d'origine, la langue originale, le nombre de saisons et d'épisodes, les diffuseurs, et les distinctions telles que les Oscars.

Dans le contexte de Wikidata, les requêtes SPARQL utilisent des préfixes et des identifiants spécifiques pour interroger efficacement la base de données. Ces éléments permettent de

préciser les données à récupérer et de structurer la requête de manière à obtenir des résultats pertinents. Ci-dessous, une explication des termes clés souvent utilisés dans les requêtes :

- wd: Ce préfixe est utilisé pour référencer des entités spécifiques dans Wikidata. Par exemple, wd:Q5398426 fait référence à l'entité de Wikidata pour "série télévisée". Chaque entité dans Wikidata possède un identifiant unique qui commence par "Q".
- wdt: Ce préfixe est utilisé pour les propriétés directes des entités. Par exemple, wdt:P31 fait référence à la propriété "instance de", utilisée pour spécifier le type d'une entité (comme une série télévisée, un film, etc.).
- wikibase:label Ce service est utilisé pour récupérer les étiquettes (labels) des entités, permettant d'obtenir le nom ou la désignation en langage naturel de l'entité. Cela rend les résultats de la requête plus lisibles et compréhensibles.

Ci-dessous un exemple d'une requête SparQL utilisée pour récupérer les genres des séries à partir de Wikidata :

Requête :

```
SELECT ?series ?seriesLabel ?genreLabel WHERE {
          ?series wdt:P31 wd:Q5398426.
          ?series wdt:P136 ?genre.
          SERVICE wikibase:label { bd:serviceParam wikibase:language "en". }
        }
```

Résultats :

	series	seriesLabel	genreLabel
0	http://www.wikidata.org/entity/Q494	Beakman's World	comedy television series
1	http://www.wikidata.org/entity/Q723	Rookie Blue	drama
2	http://www.wikidata.org/entity/Q723	Rookie Blue	police procedural
3	http://www.wikidata.org/entity/Q723	Rookie Blue	LGBTI+ related TV series
4	http://www.wikidata.org/entity/Q961	More Than Life at Stake	espionage television series

Toutes les autres requêtes ainsi que leurs résultats sont disponibles dans le notebook sparql.ipynb.

Le but de ces requêtes SPARQL est de les exécuter sur GraphDB, après avoir importé le fichier des ontologies. Cette démarche nous permettra d'exploiter les données structurées de Wikidata dans un environnement optimisé pour la gestion et l'analyse de données sémantiques complexes.