Rapport - Project: An Intelligent Web Application with Semantic Technology

Introduction :

Le but principal du projet est de **créer une application** pratique en utilisant ce que nous avons appris jusqu'à présent. Pour cela, nous avons commencé par choisir un sujet intéressant et formuler une question à laquelle nous pouvions répondre avec des données existantes.

Nous nous sommes également inspirés de nos expériences précédentes, où nous avions utilisé des requêtes spéciales pour obtenir des informations à partir de bases de données liées, comme DBpedia ou Wikidata dans notre cas. Ces expériences nous ont aidés à anticiper les difficultés potentielles liées à l'utilisation de telles données.

Malgré quelques difficultés rencontrées, nous sommes très satisfaits des résultats obtenus, qui montrent à quel point les techniques du Web sémantique peuvent être efficaces pour résoudre des problèmes complexes. Ce rapport détaillera nos étapes, les difficultés rencontrées et les résultats de notre exploration passionnante du monde du Web sémantique.

Find an interesting question to answer

Dans notre processus, la partie la plus significative et la plus approfondie a été la sélection d'une question pertinente à explorer. Nous avons consacré du temps à choisir un domaine qui nous intéressait tous les deux, en effectuant un brainstorming approfondi. Sachant que pour notre projet de machine learning, nous avions choisi un sujet lié aux films, nous avons décidé de nous tourner vers le domaine de la musique pour ce projet-ci.

Après avoir examiné les différents dataset disponibles et les informations présente sur Wikidata, nous avons finalement opté pour la question suivante :

→ “**Parmi les albums les plus vendus des dernières années, quel est le label de musique qui vend en moyenne le plus d'albums depuis l'avènement du streaming aux États-Unis (à partir de 2014) ?"**

Explication de la question :

Un “label de musique” est une entreprise spécialisée dans la production, la promotion et la distribution de musique. Les labels signent des artistes et produisent leurs enregistrements, puis s'occupent de leur mise en marché et de leur distribution.
→ ils jouent un rôle clé dans l'industrie musicale en identifiant, développant et commercialisant les talents musicaux.

Depuis 2014, les "streaming" sont également pris en compte dans le calcul des ventes d'albums. Le "streaming" fait référence à la diffusion en continu de musique sur Internet, où les utilisateurs écoutent de la musique via des plateformes telles que Spotify, Apple Music ou YouTube. Avant 2014, les ventes d'albums étaient principalement basées sur les ventes physiques (CD, vinyles) et les téléchargements numériques. Cependant, avec l'essor du streaming à partir de 2014, les ventes d'albums incluent désormais les écoutes en continu, ce qui a considérablement modifié le paysage de l'industrie musicale en influençant les tendances de consommation et les classements des ventes d'albums.

Cette question nous a semblé à la fois intéressante et pertinente, nous permettant d'explorer les données disponibles et de tirer des conclusions significatives.

Dataset :

Dans le choix du dataset, notre objectif était de trouver un ensemble de données contenant au moins le **nom de l'album**, son **année de sortie** et les données sur les **ventes** (en quantité). Bien que le nom des labels de musique n'était pas nécessaire pour notre recherche initiale dans le dataset, ces données seront cruciales pour la liaison avec les données présentes sur Wikidata. En effet, nous pouvons utiliser les informations sur les ventes et les artistes de notre dataset pour récupérer la propriété "record label" (p:264) sur Wikidata, qui contient les noms des labels de musique associés à chaque album. Ainsi, même si les noms des labels ne figuraient pas initialement dans nos critères de sélection, ils deviennent indispensables pour enrichir nos données en les reliant à d'autres sources de données telles que Wikidata.

Ces critères étaient essentiels pour répondre à notre question de recherche. Après avoir exploré différentes sources de données, nous avons identifié un ensemble de données qui répondait à ces exigences et qui nous fournissait les informations nécessaires pour mener notre analyse.

Ainsi le dataset que nous avons utilisé a été chargé depuis kaggle: top 10 albums by year . Il a ensuite été modifié pour retirer les colonnes qu'on utilise pas dans notre ontologies (Ranking,CDs,Tracks,Album Length,Hours,Minutes,Seconds) les valeurs nulles et les doublons en ne gardant que les albums publiés à partir de 2014 (ère du streaming) *cf* ***ressources/CSV/sriptCSV.py***.

Create an ontology to describe the domain

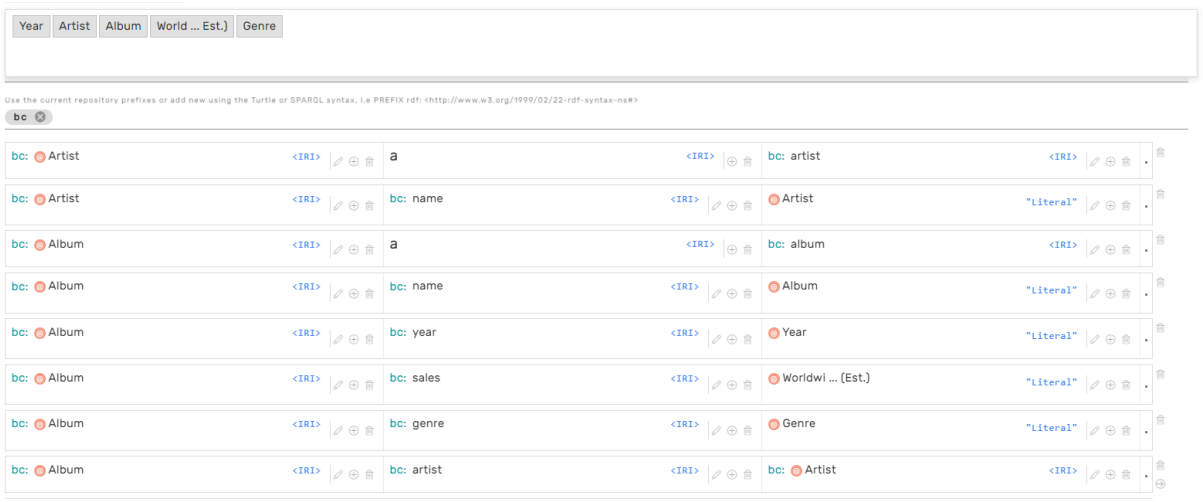
Ontology (*cf.* ***ressources/RDF/baradji-cannou.ttl***)
L'ontologie créée contient deux classe

- artist
    - name: string
- album
    - name :string
    - sales :string
    - recordLabel :string
    - year :string
    - genre :string
    - artist: object

Nous avons structuré notre ontologie de cette manière pour plusieurs raisons. Bien que notre question nécessite uniquement les propriétés "**sales**", "**recordLabel**" et "**name**", nous avons décidé de conserver d'autres éléments du dataset, tels que "**genre**" et "**year**". Cela nous permet d'avoir plus de flexibilité pour effectuer des requêtes spécialisées et d'approfondir notre analyse du dataset. De plus, en créant une classe distincte pour les "artistes" plutôt que de simplement stocker leur nom en tant que string dans la classe "album", nous nous assurons que notre ontologie est plus robuste et extensible. Cette approche nous permet d'ajouter d'autres propriétés aux artistes à l'avenir si nécessaire, tout en maintenant une structure plus propre et organisée pour notre ontologie.

Convert the tabular dataset to RDF and upload the RDF file to GraphDB

Le dataset a été transformé en un schéma rdf en utilisant ontotext Refine



Voir le fichier ressources/RDF/mapping_csv_to_rdf.json pour avoir les détails du mapping.

Le résultat se trouve dans **ressources/RDF/album.ttl**

Use SPARQL to link entities to the external knowledge graph.

Lors de l'exécution de nos requêtes plusieurs problèmes ont été rencontrés:

- ordonnance des requêtes : dans le processus de traitement des requêtes SPARQL, la partie concernant **les services externes, tels que Wikidata, est traitée en premier, suivie de la partie où on utilise nos propres propriétés spécifiques.** Cela signifie que notre requête est gérée de l'intérieur vers l'extérieur, ce qui peut entraîner des changements inattendus de données, comme cela s'est produit dans notre cas où **TOUS** les albums de wikidata étaient chargés, et non seulement ceux qui nous intéressaient spécifiquement. → Pour contrer ce problème nous avons rajouté des **filtre** au niveau du service qui permettent de **limiter** le nombre d'album charger depuis wikidata. Cette solution fonctionne mais n'est pas la plus optimale car un grand nombre d'albums inutiles est toujours chargé depuis wikidata. Nous n'avons pas trouvé de moyen pour que la partie SERVICE soit gérée **après** avoir récupéré toutes nos données local, ce qui nous aurait permis de charger dans le SERVICE **seulement** les albums qu'on aurait voulu et ainsi limiter les ressources et le temps de la requête (**environ 45 minutes pour insert**).

- données imbriquées: lorsqu'on veut récupérer une information qui n'est pas une donnée directe de l'objet de l'objet actuel mais se trouve dans un sous-objet, cela rajoute de la complexité à notre requête.
  ex:

```
prefix  bc: <https://ecampus.paris-saclay.fr/wod/baradji-cannou/>

select ?album ?artistName
where {
  ?album a bc:album;
     bc:artist ?artist.
  ?artist bc:name ?artistName.
}
```

  Ce type de recherche ajoute un niveau de complexité à notre requête. Sachant que ceci a été utilisé dans le service wikidata, le problème venait se couplé à celui énoncé juste avant.

- format et langues des données:
  - dataset: les valeurs numériques dans le dataset téléchargé sont au format US (2,000,000 au lieu de 2000000). Pour pouvoir les exploiter il faut les parser pour les convertir en nombre.
  - wikidata: les valeurs littérales renseignent aussi leur langue (ex: "mamady"@en). Pour faire le lien avec nos données locales, on ne peut pas les comparer directement. La solution a été, encore une fois, de rajouter des filtres sur la langue. Ainsi la langue n'est plus renseignée dans le résultat de la sous requête.

Le détail des requêtes se trouve dans le fichier **ressources/Queries/Queries.txt**.

Answer the question outlined in Step 1.

Voici la requête qui nous a permis de répondre à notre question énoncé précédemment :

```
PREFIX bc: <https://ecampus.paris-saclay.fr/wod/baradji-cannou/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?recordLabel
       (ROUND(SUM(xsd:integer(REPLACE(?sales, ",",
""")))/COUNT(?album)) as ?ratio )
       (COUNT(?album) as ?numberOfAlbums)
```

```
WHERE {
  ?album a bc:album;
          bc:recordLabel ?recordLabel;
          bc:sales ?sales.
}
GROUP BY ?recordLabel
ORDER BY DESC(?ratio)
```

Voici une partie du résultat que nous avons eu :

	recordLabel	ratio	numberOfAlbums
1	"XL Recordings"@en	"23000000"^^xsd:decimal	"1"^^xsd:integer
2	"RBMG Records"@en	"14000000"^^xsd:decimal	"1"^^xsd:integer
3	"Big Machine Records"@en	"10039320"^^xsd:decimal	"2"^^xsd:integer
4	"Def Jam Recordings"@en	"8525566"^^xsd:decimal	"2"^^xsd:integer
5	"Asylum Records"@en	"7991322"^^xsd:decimal	"2"^^xsd:integer
6	"Dreamus"@en	"6829450"^^xsd:decimal	"1"^^xsd:integer
7	"Capitol Records"@en	"6568176"^^xsd:decimal	"2"^^xsd:integer
8	"Fueled by Ramen"@en	"6500000"^^xsd:decimal	"1"^^xsd:integer
9	"Big Hit Music"@en	"6113233"^^xsd:decimal	"2"^^xsd:integer

On peut voir que le label : XL Recordings a un ratio de vente moyen de 23 000 000 d'exemplaires vendus par album (1 seul album dans l'ère du streaming)

Visualize the results of this query on an HTML page

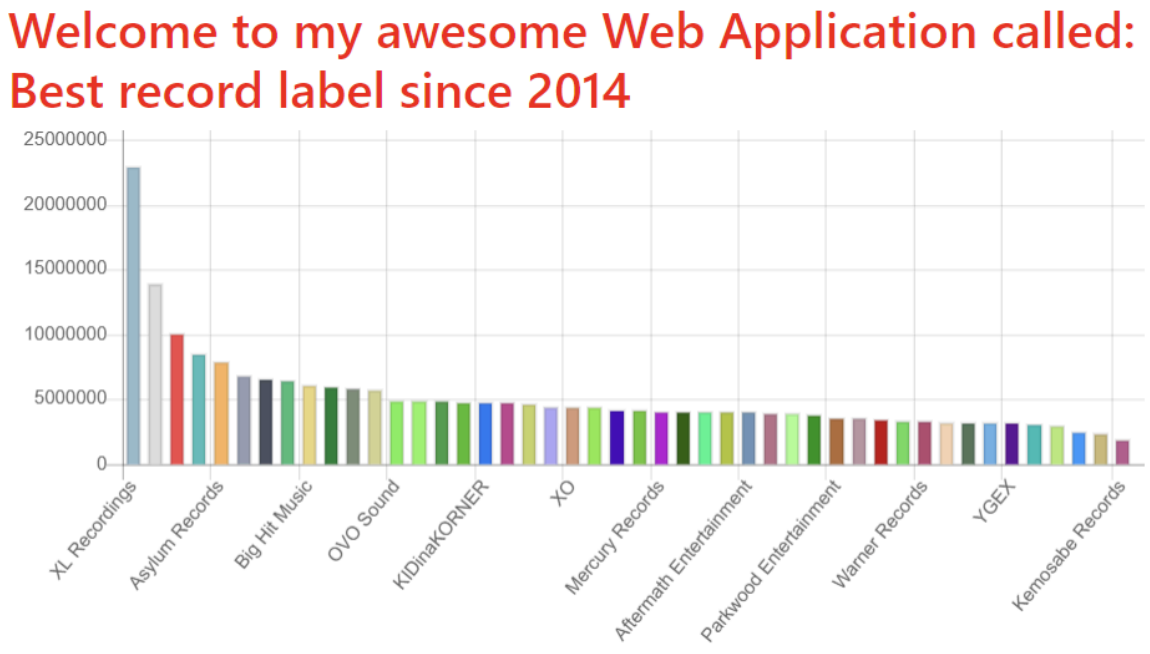
Nous avons utilisé le code HTML et JavaScript partagé sur GitHub comme base pour la visualisation des données. Ce code était initialement conçu pour afficher des graphiques circulaires (pie chart) à l'aide de la bibliothèque angular-chart.js. Cependant, afin de mieux répondre à nos besoins spécifiques, nous avons procédé à des modifications. Plutôt que d'utiliser un graphique circulaire, nous avons adapté l'application pour afficher un **histogramme (bar chart)**. Cette modification était plus appropriée pour la représentation visuelle de nos données, nous permettant ainsi de mieux analyser les relations entre les différentes données. En apportant ces ajustements, nous avons pu maximiser la pertinence et la clarté des informations tirées de la visualisation des données.

Vous pouvez tester vos propres requêtes. **Veillez à ce que le select retourne deux valeurs**. La première sera utilisée comme abscisse (label) dans le graph et la deuxième sera son ordonnée et doit être une valeur numérique.

```
PREFIX bc: <https://ecampus.paris-saclay.fr/wod/baradji-cannou/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

SELECT ?recordLabel  (ROUND(SUM(xsd:integer(REPLACE(?sales, ",",
"")))/COUNT(?album)) as ?ratio )
WHERE {
  ?album a bc:album;
          bc:recordLabel ?recordLabel;
          bc:sales ?sales.
}
GROUP BY ?recordLabel
ORDER BY DESC(?ratio)
```

Résultat :



Maintenant c'est à votre tour d'utiliser l'application

N'oubliez pas de lire le README.md et de regarder la vidéo de présentation