Utilisation des Datacube dans Dbnary

Requête dans une base de données RDF en utilisant SPARQL et visualisation graphique des données

Tristan Saminadayar

1 Introduction

Le but de la première parti de ce stage est de mettre en place différents outils pour visualiser les données de Dbnary [1]. Il y a donc deux phases pour cela ; une phase d'acquisition des données, cette phase sera généralisable à toute base de données disposant d'un point d'entrée SPARQL et utilisant le format datacube [2] et la phase d'exploitation des données qui elle n'est pas généralisable et fonctionne uniquement avec les différentes bases de données de Dbnary.

L'intégralité du code à été écrit en python (version 3.9) et une partie des scripts sont des jupyter notebooks et permettent de rendre les visualisation plus interactives grâce au module ipywidgets et au module bqplot. Le module principalement utilisé est le module SPARQLWrapper qui permet d'effectuer des requêtes SPARQL sur un serveur distant. Pour finir les données sont représentés dans un pandas DataFrame, module permettant facilement le maniement de gros jeux de données.

2 La requête SPARQL (SPARQL_query.py)

La classe présente à l'intérieur du fichier est une classe permettant de faire des requêtes SPARQL en se basant sur le module SPARQLWrapper, lors de l'appel de cette classe, si la requête est une requête SELECT, on va modifier la requête afin d'obtenir le nombre de lignes dans le résultat, en effet, dans certains cas, s'il y a trop de lignes dans la réponse le serveur ne va pas envoyer la totalité de la réponse. Dans ce cas, on va modifier la requête en ajoutant un OFFSET et un LIMIT pour récupérer les données en plusieurs fois et ne pas atteindre le maximum de ligne autorisée par le serveur dans ses réponses. De plus, la classe est compatible avec le widget *IntProgress* qui permet d'avoir une barre de chargement dans un notebook qui permet de visualiser l'état de la réception des réponses pour les requêtes qui dépassent le seuil du serveur. Pour finir, dans certains cas, la classe sera capable de détecter le fait que le résultat de la requête est incomplet dans le cas où le nombre maximum de retours aurait été dépassé ou lors d'un *execution timeout*

3 La détection et le téléchargement de datasets (Dataset_tools.py)

Cette étape se sépare en trois sous étapes :

1. Tout d'abord, on commence par demander au serveur l'ensemble des datasets disponible et leurs commentaires s'il y en a grâce à la requête :

```
SELECT DISTINCT ?dataset ?commentaire
  WHERE
  {
    ?dataset a <http://purl.org/linked-data/cube#DataSet>
    OPTIONAL
    {
      ?dataset <http://www.w3.org/2000/01/rdf-schema#comment> ?commentaire
    }
  }
```

2. Ensuite, on va chercher les différentes caractéristiques de chaque observation d'un dataset donnée :

```
SELECT ?property
  WHERE
  {
  { SELECT ?item
      WHERE
```

1. http://kaiko.getalp.org/about-dbnary/
2. Plus d'informations : https://www.w3.org/TR/vocab-data-cube/

```
6         {?item <http://purl.org/linked-data/cube#dataSet> <URI of dataset>}
7     LIMIT 1
8   }
9   ?item ?property ?value .
10  FILTER (?property not in (<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>))
11  }
```

3. Pour finir, on va télécharger le dataset voulue, mais avec uniquement les caractéristiques choisis :

```
SELECT ?name_property1 ... ?name_propertyN
  WHERE
  {
  ?o <http://purl.org/linked-data/cube#dataSet> <URI of dataset> .
  ?o <URI of property 1> ?name_property1 .
  ...
  ?o <URI of property N> ?name_propertyN .
  }
```

L'utilisation principale de ces fonctions est faite dans la visualisation (**Graphs.ipynb**) ou l'on fait uniquement l'étape 3 et dans le notebook qui permet d'obtenir un dataset choisis de façon interactive grâce au widgets (**Get_DataCube.ipynb**).

4 Exploitation des données (Graphs.ipynb)

Pour faire des graphiques interactifs, j'ai utilisé le module **bqplot** qui permet de faire des graphique pour lesquels on peut changer leurs données après l'affichage, il n'y a rien de particulièrement compliqué dans ce fichier, il faut juste faire attention sur certains points : dans certains datasets de Dbnary il faut faire un pivot d'une colonne en lignes, de plus il faut faire attention à ce que les dates soient bien considérées comme tel et non comme une simple valeur et plus généralement aux types des données et il faut faire attention à la porté des variable dans la gestion des événement des widgets.