Building a graph with RDFlib for modeling knowledge and automate reasoning

Manuel Dellabona
503086

Abstract

The final project of the Knowledge Representation and Reasoning course consisted in the creation of a RDF knowledge graph from scratch, starting with tabular data. The project covered step by step all the topics concerning the creation of this type of graph, giving a more complete point of view about the application and the integration of all the lectures.

1 Goals

.The goals of this project can be summarized with the requests of all the steps:

1) Dealing with tabular data in python with the pandas library;
2) Learn the RDF grammar and its functions and apply those knowledges in python with RDFlib;
3) Create an ontology that fit to our data;
4) Create a Taxonomy;
5) Load the graph on a sparql endpoint with Docker and Virtuoso .
6) Materialize inferences in the endpoint's graph
7) Integrate our data with new ones from dbpedia

2 Dataset description

The dataset given describes a list of albums connected to their Release Date, Artist, album, genre, subgenre and Descriptors.

3 Methodology

3.1 Preliminaries

After having installed all the prerequisites and downloaded the data I figured out what my data needs to be in order to work with them. I merged the two databases integrating the information for the albums that appeared in both of them.

3.2 The graph

The datas are now preprocessed and ready to fill the graph; so I started declaring the graph and the prefixes. Then, in order to replenish the graph I have used a for loop that iterates for each element of the database and creates an rdf triple that will be added to the graph later.

3.3 The ontology

The ontology is composed by the set of declaration (with RDF triples) of the classes (s.a. Person) and the property (s.a. dbp.artist) with its domain and range that will play a role when i will be inferring triples with a reasoner.

3.4 The taxonomy

We create a taxonomy of the music genres presented in the database in order to represent the relation between a genre and its subgenres. To do this, i have setted a for loop that surf in list of all the genre and check if the string of a genre is contained in another, and creates an RDF triple of the type: (URIRef("http://example.org/)+gen[x],RDFS.subClassOf,URIref("http://example.org/")+genre).

3.5 Loading the graph on a sparql endpoint

The loading of the graph has been allowed by the creation of a virtual machine using Docker service and Virtuoso. After creating an empty graph on the endpoint I produced a query to insert data using the graph built on the notebook and its triples.

3.6 Materializing inferences

In order to expand the KG we can use a python library, Owlrl, that imports a reasoner that elaborate all the inferences that can be extracted from our KG and then, similarly as done before we create insert queries to update our graph on the endpoint. Then, in order to infer that, if a genre_y is a subclass of genre_x, and an album has genre_y, has also genre_y, I used an insert query that adds that to the graph.

3.7 Data integration with DBpedia

The integration of the data with DBpedia has been done in the following way:

a. With the update function I created an insert query to add to my local graph the triples I needed.
b. In order to check whether the album in my data exists on dbpedia i took the label that represents my titles and query dbpedia for the triples that have that same label and are an album.
c. Then I inserted in my graph the new data where the two labels correspond.
d. To conclude, I loaded the new triples on the sparql endpoint.

4 Faced problems

The strength of this project, as the capability to teach how to integrate many topics discussed and apply them on a simulated real field, led me to many problems produced by the several different tools that I had to use, so thanks to that I learnt how to solve this kind of problems with the help of the communities on the web.

5 Conclusions and future works

The resulting KG will be heavily focused on the albums because we had more data related to them, so if we consider the album as the focus we can see that they are connected to the data given at the start and the data extracted to dbpedia such as producer and studio.

The work can be easily expanded with more features and classes, perhaps we can consider to organize album by common characteristics.

One possible application of it could be a music adviser based on your preferred artist, genre or the sentiment described by the descriptors