

Constructing an RDF Knowledge Graph

Knowledge Representation & Reasoning – Mod. 2 Artificial Intelligence 2022-2023 Final Project

NTEGANO Bahenda Yvon Dylan

1 Datasets and Task

The project was based in RDFLib, a python library for implementing RDF, and basically we had to create an RDF knowledge graph from structured data and work on it.

We create the KG by adding triples from data that was provided as two csv files which we joined together in a single panda dataframe as it would be easier to work with.

We had to enrich our data in the KG using data from DBpedia and also unstructured data from Wikipedia using a python wrapper for Wikipedia's API. We also added an ontology and a taxonomy which were useful later in the project for making inferences and the queries.

We used RDF/RDFS concepts and syntax with writing the data in the KG as triples. We also used SPARQL to query the graph or DBpedia. And also OWL for materializing the inferences.

2 Workflow

Firstly, I had to join the two pandas dataframes into one dataframe using the merge module of the panda library and I selected some of

the columns of the combined data frames and left some out to have a manageable data frame with low redundancy as well.

To create the RDF KG I made a loop to iterate through each row of the data frame and read some of the data like album, artist, genre and others and I append triples, like (dbr:album, dbo:artist, dbr:artist) , to a list. Then simply I iterate through that list and add each triple to my RDF KG.

Because of the different structures of the data you have to replace some symbols so it is easily read and written to the graph.

Secondly, I had to integrate my data with DBpedia's data. I used a SPARQL query to search through DBpedia more data about every band, like the band members, their birthplaces and the band's hometown. The query results were then converted into a pandas dataframe, to be easily manipulated, from which iterated through each row and added triples based on the given results. Some results were a bit unusual because of the strange syntax of the names so I used try... except... method to deal with them. Then all the data from DBpedia was added to the KG as well.

For the third, the word recognition system was simply a list and an if condition to check if a word in the summary is in my list. I iterated through the KG and created a list of all the members, then I iterated through that list to get their roles and add the triple to a list so as to add it to the knowledge graph later. Wikipedia is one of the ways of gathering information from unstructured data.

The fourth which is about creating an ontology and a taxonomy for genre. We add triples to define some classes and also the relation between the properties by specifying the domain and range for the properties that were used. With a taxonomy of this form it can later be used to make inferences and expand our KG.

The fifth was to simply materialize the inferences and expand our graph. There were some inferences made which were to be expected, we could see this from the SPARQL query.

The last part consisted of querying our KG using SPARQL. The intention of this was to highlight the information integrated from DBpedia and also demonstrate the inferences that had been made from the previous point.


3 RDF Knowledge Graph

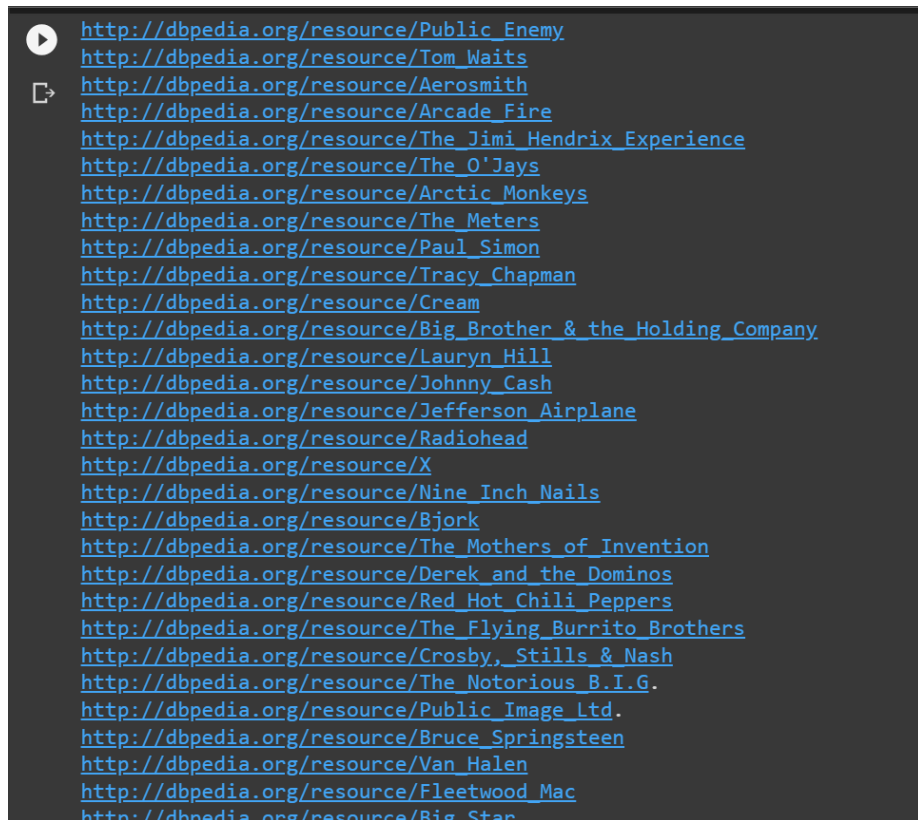
The Knowledge Graph is primarily based on triples describing the albums like its label, artist, its band members, its genres and the album descriptors.

With the ontology and taxonomy I defined some classes like people, music album, music artist and so on and also described the domain and ranges for the properties I used like domain of the property “dbo:genre” is music album and its range is the class genre.

With the aforementioned additions I was able to make some inferences about the already existing data in the triples like the classes of some of the things I had.

The below query shows me every artist.

```
 for s, p, o in rdf_graph.triples((None, rdf.type, dbr.Artist)):  
    print(s)
```



```
▶ http://dbpedia.org/resource/Public_Enemy
↳ http://dbpedia.org/resource/Tom_Waits
http://dbpedia.org/resource/Aerosmith
http://dbpedia.org/resource/Arcade_Fire
http://dbpedia.org/resource/The_Jimi_Hendrix_Experience
http://dbpedia.org/resource/The_O'Jays
http://dbpedia.org/resource/Arctic_Monkeys
http://dbpedia.org/resource/The_Meters
http://dbpedia.org/resource/Paul_Simon
http://dbpedia.org/resource/Tracy_Chapman
http://dbpedia.org/resource/Cream
http://dbpedia.org/resource/Big_Brother_&_the_Holding_Company
http://dbpedia.org/resource/Lauryn_Hill
http://dbpedia.org/resource/Johnny_Cash
http://dbpedia.org/resource/Jefferson_Airplane
http://dbpedia.org/resource/Radiohead
http://dbpedia.org/resource/X
http://dbpedia.org/resource/Nine_Inch_Nails
http://dbpedia.org/resource/Bjork
http://dbpedia.org/resource/The_Mothers_of_Invention
http://dbpedia.org/resource/Derek_and_the_Dominos
http://dbpedia.org/resource/Red_Hot_Chili_Peppers
http://dbpedia.org/resource/The_Flying_Burrito_Brothers
http://dbpedia.org/resource/Crosby,_Stills_&_Nash
http://dbpedia.org/resource/The_Notorious_B.I.G.
http://dbpedia.org/resource/Public_Image_Ltd.
http://dbpedia.org/resource/Bruce_Springsteen
http://dbpedia.org/resource/Van_Halen
http://dbpedia.org/resource/Fleetwood_Mac
http://dbpedia.org/resource/Big_Starr
```

Note that I have both bands and musicians as artists, this is because in the original data the artist of an album could have been both a band or a single musician, however now the band members that were added from DBpedia and using the ontology described above we can infer that they are also artists like Paul Simon or Johnny Cash.

4 Conclusion

Due to the high amount of data I have to deal with, I can't be able to integrate the KG with more of DBpedia's data. The more albums they are, if I query DBpedia it will take too much time to run and I could have possibly been banned temporarily. It consumes a lot of memory and processing power.

It was not easy to map the large amounts of data from the pandas dataframe to RDF triples as it has to be done manually which is time-consuming and can involve so many errors.

We can expand it by using efficient conversion strategies like some optimized algorithms to improve the conversion process and map easily.