THE SEMANTIC WEB - ECS 735P

Final Coursework

TANISHQ VERMA - 210642458

Abstract

The project is on movies and tv series which consist of different types of classes, sub-classes, data properties and object properties. The first step of was creating an ontology which contains classes, sub-classes, data properties and object properties related to the data set we choose. I further need to choose the data set and according to the data set I create the ontology for it. For the basic task the data source is **dbpedia** through which populating data I created a query file to run query on it. For the bonus task 1, the data source I choose is **dbkwiki**. After populating data, we further create a python to query on it. Finally for the bonus 2, I added all the relevant SWRL rule on the ontology. As defines in the Coursework specification the software we used is 'Protégé'.

Ontology –

Our ontology consists of class hierarchy which contains different types of classes and sub-classes. To illustrate, in my ontology some of the classes are Language, country and further some classes are People which have different types of sub classes such as Directors, Producers, Editors, Cast members and others. Further there is a class with format which contains sub classes such as Picture format, camera format, genre and other. All the data is related to the movies and tv series data that is present in our data source. To the best of my understanding, according to the data, I created an ontology.

Classes created are:

- Country
- Creative_Work
- Formats
- Language
- Media_Rights
- People

Sub classes are:

- Creative_Work
- Media_Rights
- Formats
- People

Sub-sub class is:

- Composer

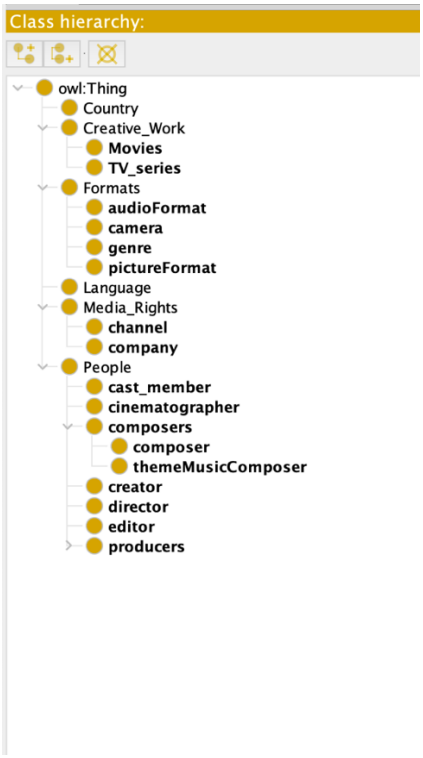


Fig 1 – Class hierarchy

Once we create the class hierarchy, we further need to create object property hierarchy. Object property defines the relationship between different individuals or instances. Some of the object property that I have created are creator of, director of etc.

Fig 2 – Object Property Hierarchy

After creating the object property, we need to create the data property for the individuals that we will be adding. Some of the birth place, birth date, budget, box office and other. Data property defines the data for the individuals or instances. For example, an actor working in a movie must have a Birthdate, name, age etc.

Fig 3 – Data Property hierarchy

After creating the class hierarchy, object property and data property, our ontology creating part is completed.

Complete Visualization of the ontology using owl viz window of ‘Protégé’:

Fig 4 – Asserted Hierarchy for Ontology

Data Source

The next part of the project is to populate the ontology with the data source that we have decided. For this project, the data source from **‘dbpedia.org’**. The most tedious task was to find the right data for the project since most of the data source, the endpoint was down. I finally decided to get data from dbpedia. The data source contains a lot of information for different types of movies and tv series. There information contains the summary, box office, budget, cast members, directors, producers, editors, genre, run time, release date, country, language and other. The data source was a good once. For us to populate the data on the ontology, we need the appropriate prefix as well as the endpoint. For the bonus task 1, the data source, I choose was the **‘dbkwiki’** and populated the existing owl file with it.

Basic Task

I finally managed to get the end point and the prefix for writing the query so that the python file will be able to populate the data. I managed to load a lot of data to my owl file ‘final report basic’.

Fig 5 – Individual after populating data

Some of the difficulties –

1. Finding the endpoint and the prefix for the data source that we have selected.
2. Updating the python version and importing the relevant libraries
3. Constructing the queries and parsing it to a new owl file. Unfortunately, sometimes even a small error in the query could lead to not parse the data, so I had to use an apache jena fuseki to lead data and run queries and see the output.
4. Since I wanted to add the data for both movies and tv series, I created two queries and applies both queries once after the other so that data for both gets populated.
5. Further I wanted to add as much data as possible, so my query became big and complex due to the where clause.

After writing the queries and parsing the whole date to our ontology and creating a new owl file ‘final report basic’, I managed to load all of the data. I create basic.py file which would take the ‘final report.owl’ file parse the data and create a new file called ‘final report basic.owl’.

The next step for the basic task was to write the query on the new created ‘final report basic.owl’ file. Again, I wrote queries for both the movies and tv series. I applied both the queries one after the other and got the output for it by using my query basic.py file. The queries produce the name, box office, budget and director and cast member for both movies and tv series.

Fig 6 – Output for Basic Task

I was successfully able to populate the data for both movies and tv series and further was also able to query the data for both the movies and tv series.

To understand more about the ontology after populating the data:

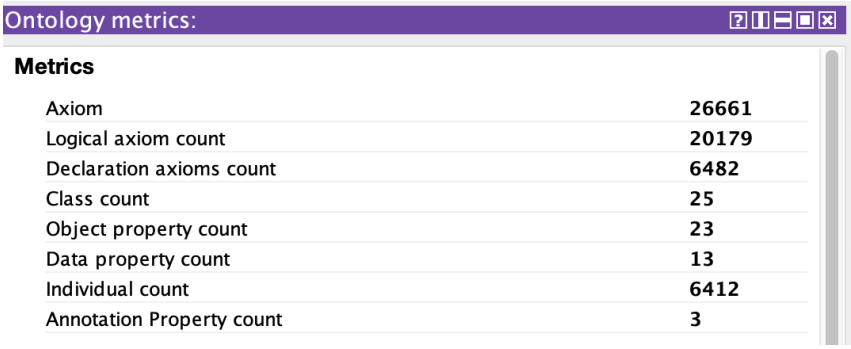


Fig 7 – Ontology Metrices

After adding the data, to understand more about individual or instances property assertion:

Fig 8 – Property assertion

Bonus Task 1

For the first bonus task, since we need to populate more data from a different source, I need to look for a data source. For this task, the data source is ‘dbkwiki’. I got the working endpoint and the prefix for the data.

In this task, I re-populated more data to my already existing owl file ‘final report basic.owl’ and created a new owl file ‘final report bonus.owl’. Since I already had a lot of data, I added more data for TV series and Movies but gave a limit on the query so as the file does not become big. I limited adding only 5 movies and 5 tv shows to see if the query is working.

Fig 9 – Data populated from dbkwiki

Once the data gets populated on the new ‘final report bonus.owl’ file using the bonus.py, I created a new python file for query the bonus file. I created the ‘query bonus.py’ which create query using the ‘final report bonus.owl’. The query generated will get the name of the TV series and movies using the regex file which contains ‘r’ and ‘i’ letter in them. The output will contain the name along with the budget and the release date of the tv series and the movies.

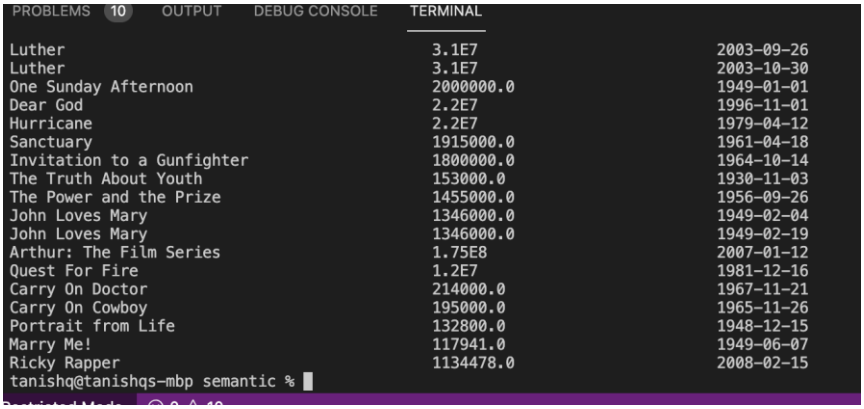


Fig 10 – Output for the Bonus Task 1

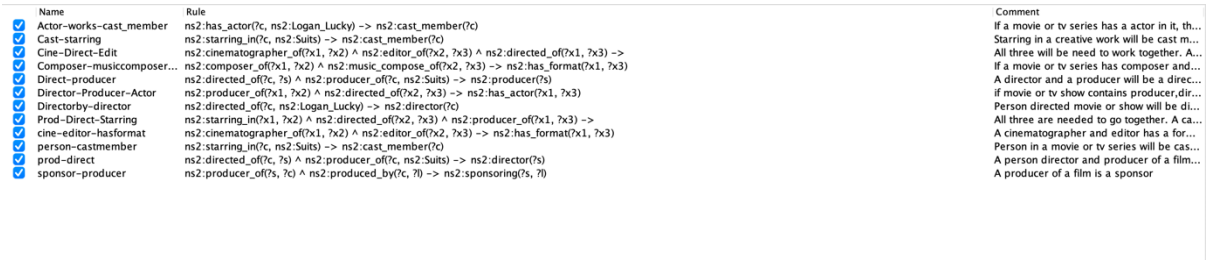
Bonus Task 2

For the second bonus task, we need to apply SWRL rule on object as well as data property. SWRL or Semantic Web Rule Language is a language in semantic web which can be used to express semantic and logic rules.

According to the ontology, some of the SWRL rule are added on the 'final report.owl' file. In total I have added 12 SWRL rules which I found were most appropriate and relevant to the data. In addition, I have given comment for each of the SWRL rule to explain what each SWRL has.

1. ns2:has_actor(?c, ns2:Logan_Lucky) -> ns2:cast_member(?c) – This SWRL rule typically defines that if a movie has an actor or an actress, then they must be the cast member for the movie.
2. ns2:starring_in(?c, ns2:Suits) -> ns2:cast_member(?c) – This SWRL rule states that a person starring in a tv series then it must be the cast member of the tv series.
3. ns2:cinematographer_of(?x1, ?x2) ^ ns2:editor_of(?x2, ?x3) ^ ns2:directed_of(?x1, ?x3) -> - This SWRL rule depicts that a cinematographer and an editor must have a director as well. All and each depend on each other and must be together. It is valid to say that a movie or a tv show containing a cinematographer and an editor then it must also have a director to direct it.
4. ns2:composer_of(?x1, ?x2) ^ ns2:music_compose_of(?x2, ?x3) -> ns2:has_format(?x1, ?x3) – This SWRL rules say that a composer and a music composer must have a format to work on. The composer and the music compose must have a format be it a audio format, type of genre they are produce etc.
5. ns2:directed_of(?c, ?s) ^ ns2:producer_of(?c, ns2:Suits) -> ns2:producer(?s) – This SWRL illustrates that a director or a producer of a tv series or movies must be both the director and the producer.
6. ns2:producer_of(?x1, ?x2) ^ ns2:directed_of(?x2, ?x3) -> ns2:has_actor(?x1, ?x3) – This SWRL tells that a movie or a tv series having a director and a producer then it must also be having an actor to work with.
7. ns2:directed_of(?c, ns2:Logan_Lucky) -> ns2:director(?c) – This is a simple SWRL saying that a person directing a movie or tv series will be the director of it according to its role.
8. ns2:starring_in(?x1, ?x2) ^ ns2:directed_of(?x2, ?x3) ^ ns2:producer_of(?x1, ?x3) -> - This SWRL illustrates that starring, directors and produces must be together in order to work. Assume a movie having producer and director but no actor or vise versa.
9. ns2:cinematographer_of(?x1, ?x2) ^ ns2:editor_of(?x2, ?x3) -> ns2:has_format(?x1, ?x3) – This SWRL tells that a cinematographer and an editor must have a format. It is similar to the other SWRL defined above.
10. ns2:starring_in(?c, ns2:Suits) -> ns2:cast_member(?c) – This SWRL tells that a person starring in tv series or movie will be cast member of it.
11. ns2:directed_of(?c, ?s) ^ ns2:producer_of(?c, ns2:Suits) -> ns2:director(?s) – This SWRL is in relation to the above SWRL meaning that a person who is director and a producer will be both a director and a producer provided the person has both roles.
12. ns2:producer_of(?s, ?c) ^ ns2:produced_by(?c, ?l) -> ns2:sponsoring(?s, ?l) – This SWRL tells that a person who is producer of a company who is a producer is sponsoring the movie or tv series since all of the money is of the producers at stake.

These are the SWRL that are added in the ontology via the SWRL tab on Protégé.



Name	Rule	Comment
Actor-works-cast_member	ns2:has_actor(?c, ns2:Logan_Lucky) -> ns2:cast_member(?c)	If a movie or tv series has a actor in it, th...
Cast-starring	ns2:starring_in(?c, ns2:Suits) -> ns2:cast_member(?c)	Starring in a creative work will be cast m...
Cine-Direct-Edit	ns2:cinematographer_of(?x1, ?x2) ^ ns2:editor_of(?x2, ?x3) ^ ns2:directed_of(?x1, ?x3) ->	All three will be need to work together. A...
Composer-musiccomposer...	ns2:composer_of(?x1, ?x2) ^ ns2:music_compose_of(?x2, ?x3) -> ns2:has_format(?x1, ?x3)	If a movie or tv series has composer and...
Direct-producer	ns2:directed_of(?c, ?s) ^ ns2:producer_of(?c, ns2:Suits) -> ns2:producer(?s)	A director and a producer will be a direc...
Director-Producer-Actor	ns2:producer_of(?x1, ?x2) ^ ns2:directed_of(?x2, ?x3) -> ns2:has_actor(?x1, ?x3)	if movie or tv show contains producer,dir...
Directorby-director	ns2:directed_of(?c, ns2:Logan_Lucky) -> ns2:director(?c)	Person directed movie or show will be di...
Prod-Direct-Starring	ns2:starring_in(?x1, ?x2) ^ ns2:directed_of(?x2, ?x3) ^ ns2:producer_of(?x1, ?x3) ->	All three are needed to go together. A ca...
cine-editor-hasformat	ns2:cinematographer_of(?x1, ?x2) ^ ns2:editor_of(?x2, ?x3) -> ns2:has_format(?x1, ?x3)	A cinematographer and editor has a for...
person-castmember	ns2:starring_in(?c, ns2:Suits) -> ns2:cast_member(?c)	Person in a movie or tv series will be cas...
prod-direct	ns2:directed_of(?c, ?s) ^ ns2:producer_of(?c, ns2:Suits) -> ns2:director(?s)	A person director and producer of a film...
sponsor-producer	ns2:producer_of(?s, ?c) ^ ns2:produced_by(?c, ?l) -> ns2:sponsoring(?s, ?l)	A producer of a film is a sponsor

Fig 11 – SWRL TAB

All of the SWRL have correct status and has been added SWRL rule and other relevant OWL knowledge to the rule engine. Drools has been completed and is running on the engine. Drools knowledge has been added to the ontology.

Contents of the zip file:

1. Final basic.owl – This is the ontology created based on the data set. It contains all the classes, sub-classes as well as the object properties and the data properties. Further, some of the data has been entered manually to understand the whole ontology. The ontology for the file has been defined at the beginning of the report. The bonus task 2 has been done in this file however it can be found in any owl file since all python file parse and generate a copy of it populating data to it.
2. Basic.py – This is the python file which contains two queries one for movies and the other for tv series. It uses the 'final basic.owl' file fetches the data using endpoint for both movies and tv series parse it to the new owl file. The name the newly generated owl file is 'final report basic.owl'.
3. Final report basic.owl – This is the owl file which is generated after running the basic.py file. This owl file has data from dbpedia and has data for both movies and tv series. The ontology metric for this file has been provided above.
4. Query_basic.py – This is the python file which uses the 'final report basic.owl' to generate the query for both the movies and tv series. Running the file in the vs code will generate the output for both the movies and well as tv series. Further, I have written two queries for this file as well where it run one query and then runs the second one as well one for movies and the other for tv series. The output has been provided above.
5. Bonus.py – This is the python file which uses the 'final report basic.owl' file fetches data using endpoint from dbkwiki and produces a new owl file. Like the previous basic.py, it has two queries one for tv series and the other for movies. Running this file will generate the 'final report bonus.owl' file. Since the previous owl file already have a lot of data, I have limited the query adding data to just 5, adding 5 data for movies and 5 data for tv series.
6. Final report bonus.owl – This is the owl file which is generated after running the bonus.py file. This owl file has data from both dbpedia as well as dbkwiki.

7. Query_bonus.py – This python file will run query and produce the output using the ‘final report bonus.owl’ file. Like other, it has two queries, and the output is provided above.

Rules for running the code:

1. First running the ‘basic.py’file. Changing the directory to the specific location and then command ‘ python3 basic.py’. This file will generate the ‘final report basic.owl’. **This process will take a lot of time because it is adding a lot of data to it.**
2. Once ‘final report basic.owl’ is generated, we need to run the ‘query_basic.py’ which will use the ‘final report basic.owl’ to print the query. The command is same as above ‘python3 query_basic.py’. If using the vs code, the output will generate in the terminal.
3. For the bonus task one, we need to run the ‘bonus.py’ which will use the ‘final report basic.owl’ and populate more data and will generate the ‘final report bonus.owl’. The command is ‘python3 bonus.py’
4. For the query of the bonus task, use the ‘query_bonus.py’ which will use the ‘final report bonus.owl’ to print the query. The command is the same ‘python3 query_bonus.py’.

Note: - Since I am populating a lot of data, some process especially during the basic.py may take some time. Also opening my owl file especially ‘final report basic.owl’ or ‘final report bonus.owl’ may have a lot of individuals and could take some time when opening using Protege.