ECS7028U/P: DATA SEMANTICS

UNIVERSITY ONTOLOGY DESIGN REPORT-final coursework

Student Name: Sunantha Kannan

Student ID Number: 230841125

Table of Contents

1. BASIC TASK:

1.1. Introduction

The goal of this ontology is to store all kinds of information about universities and the things that are connected to them, like chancellors, presidents, types of universities, and the states where they are located. This ontology tries to make administrative tasks, academic research, and data-driven decision-making easier in the higher education domain by capturing the most important relationships and properties of these entities. This ontology offers a comprehensive toolset for evaluating university characteristics. This allows prospective students and educational researchers to assess and compare universities effectively, facilitating informed decision-making based on a wide range of academic and administrative criteria.

Main Classes of the University Ontology Include:

1. **University**: Represents individual higher education institutions, detailing their names, student populations, faculty sizes, and more.

2. **Chancellor**: Represents the senior administrative leaders of universities, capturing detailed personal information such as name, birth date, and birthplace.

3. **President**: Represents the chief executive officers of universities, detailing similar personal and professional attributes as chancellors.

4. **State**: Represents the geographical entities where universities are located, detailing state names and their capitals.

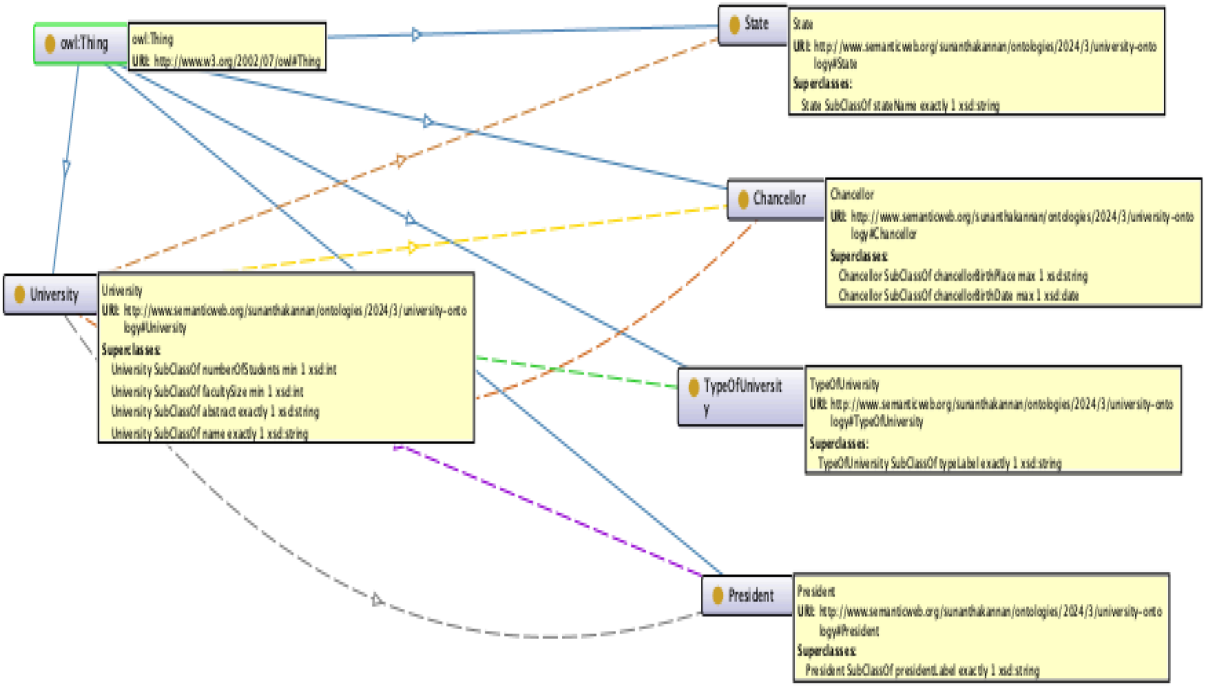
5. **TypeOfUniversity**: Represents different classifications of universities, such as public, private, or community colleges, each characterized by specific attributes.

The university ontology employs object properties such as **hasChancellor**, **hasPresident**, **isLocatedIn**, and **isOfType** to model the relationships between universities and their associated entities like chancellors, presidents, and geographic locations. These properties facilitate the linking of universities to their administrative leaders and their classification by type and state. Additionally, it utilizes data properties such as **name**, **numberOfStudents**, **facultySize**, **numberOfSocialMediaFollowers etc..** to provide detailed descriptive and quantitative information about each university This structured approach enhances the ontology's capability to support detailed organizational representation and complex querying.

1.2. Ontology Design

We used OWL2 for our ontology because it supports Description Logic and is popular in the Semantic Web. The T-Box, which defines the ontology's taxonomy and structure, was created using the popular Protégé ontology editor.

See the Onto Graph below.



1.2.1 Class Hierarchy

The university ontology features a structured class hierarchy that includes key entities such as **University**, **Chancellor**, **President**, **State**, and **TypeOfUniversity**. This hierarchy systematically organizes the different aspects of a university, from administrative roles to geographical locations. Each class is designed to encapsulate specific attributes and relationships, ensuring a comprehensive representation of the university ecosystem within the ontology.

1.2.2. Object Properties

Object properties connect two individuals (a subject and object) with a predicate. Domain represents possible subjects of a property. Range represents possible objects of a property.

- **hasChancellor (Domain: University, Range: Chancellor)**: Connects a university to its chancellor.
    - Inverse: **chancellorOf**
- **hasPresident (Domain: University, Range: President)**: Connects a university to its president.
    - Inverse: **isPresidentOf**

- **isLocatedIn (Domain: University, Range: State)**: Associates a university with the state where it is located.

- **isOfType (Domain: University, Range: TypeOfUniversity)**: Categorizes universities into different types based on their governance or educational focus.

1.2.3. Object Properties Characteristics

The property behaviour used to represent understanding of the relationships between their subjects and objects.

Functional and Inverse Functional Properties:

- Functional Properties are not used for hasChancellor and hasPresident because these roles can be held by the same individual at multiple universities simultaneously. Applying a functional restriction would imply that each university can have only one unique chancellor or president, which is not possible.
- Inverse Functional Properties are also not suitable for chancellorOf and isPresidentOf as these would restrict each chancellor and president to be linked to only one university, contradicting real-world scenarios where officials might serve multiple institutions.

Symmetric, Asymmetric, Reflexive, and Irreflexive Properties:

- Symmetric and Asymmetric Properties do not apply since the relationships (university to chancellor/president) are inherently directional; a university appoints a chancellor but not vice versa.
- Reflexive and Irreflexive Properties are irrelevant as it does not logically apply for a university to hold a position or title like chancellor or president within itself.

Transitive Property:

- The Transitive Property is not applicable to properties like hasChancellor or hasPresident because the roles do not logically support the transitive relational concept (i.e., the relationships don't extend beyond a direct link between two entities)

Why Only "Inverse Of" is Used:

- **Inverse Of:** Using "inverse of" relationships (hasChancellor is the inverse of chancellorOf, and similarly for hasPresident and isPresidentOf) is beneficial for maintaining bidirectional consistency. This means if a university is linked to a chancellor through hasChancellor, the chancellor is automatically linked back to the university through chancellorOf. This aids in querying the data from both directions efficiently without redundancy.
- Inverse of does not impose constraints on the uniqueness of the relationships, making it suitable for situations where officials may hold multiple appointments.

1.2.4. Data Properties

The predicate connects a single subject with some form of attribute data. Data properties have defined datatypes including string, integer, date, datetime, literal, Boolean etc.The below data properties have been chosen to gather required information related to university.

Data Property Name	Domain	Range	Description
abstract	University	rdfs:literal	A brief description or abstract of the university.
address	University	rdfs:literal	The physical address of the university.
name	University	rdfs:literal	The name of the university.
numberOfStudents	University	xsd:integer	The number of students enrolled at the university.
facultySize	University	xsd:integer	The number of faculty members at the university.
numberOfSocialMediaFollowers	University	xsd:integer	The number of followers the university has on social media platforms.
chancellorBirthDate	Chancellor	(rdfs:Literal or xsd:date or xsd:dateTimeStamp)	The birth date of the chancellor.
chancellorBirthPlace	Chancellor	rdfs:literal	The place of birth of the chancellor.
chancellorName	Chancellor	rdfs:literal	The name of the chancellor.
typeLabel	TypeOfUniversity	rdfs:literal	A label describing the type of university (e.g., public, private, technical).
stateName	State	rdfs:literal	The name of the state where the university is located.

capitalName	State	rdfs:literal	The name of the capital of the state where the university is located.

1.3. Description Logic

DL allows us to define axioms, constraints, and restrictions between classes and properties, helping us detect and resolve potential inconsistencies or incorrect representations.

Description logic is a decidable fragment of predicate logic Addresses the expressivity-tractability tradeoff. Adequate representation & efficient inference. Description logic syntax is concise and intuitive for modeling and reasoning about knowledge

1.3.1 DL Axioms and Restrictions

1. Every University must have exactly one name.
    - **name exactly 1 xsd:string** (Domain: University)
    - **Explanation**: Each university must have exactly one name, ensuring each institution is uniquely identified by a single official name.
2. Every University must have exactly one abstract.
    - **abstract exactly 1 xsd:string** (Domain:University)
    - **Explanation**: Each university must have exactly one abstract or description, providing a brief overview of the university.
3. Every University can have at most one faculty size.
    - **facultySize max 1 xsd:int** (Domain: University)
    - **Explanation**: Each university must have exactly one name, ensuring each institution is uniquely identified by a single official name.
4. Every University must have atleast one student.
    - **numberOfStudents min 1 xsd:int** (Domain: University)
    - **Explanation**: Requires that a university has at least one student, defining the minimum population for it to be considered an active educational entity.
5. Chancellor can have atmost one birthplace.
    - **chancellorBirthPlace max 1 xsd:string** (Domain: Chancellor)
    - **Explanation**: Make sure that each chancellor can have at most one birthplace recorded, gives a singular value for this attribute.
6. Chancellor can have atmost one birth date.
    - **chancellorBirthDate max 1 xsd:date** (Domain: Chancellor)
    - **Explanation**: Limits the birth date of a chancellor to at most one value, ensuring no more than one date can be associated with each chancellor.
7. Every University must have a one president.
    - **presidentLabel exactly 1 xsd:string** (Domain: President)

- **Explanation**: Each president is uniquely identified by a single, definitive name within the ontology.

8. Every University must hold exactly one type of university.
    - **typeLabel exactly 1 xsd:string**(Domain: TypeOfUniversity)
    - **Explanation**: Every university type must have one descriptive label identifying it's kind of institution.
9. Every State must have exactly one name
    - **stateName exactly 1 xsd:string** (Domain: State)
    - **Explanation**: Each state is required to have exactly one name and to make sure that ontology state identification is clear.

1.4 Ontology Population and Querying

Ontology has been populated with data from **DBpedia**.

Source File: The dbpedia_data_load.py in .zip contains the SPARQL query that populates data into our Ontology file (dbpedia_loaded_university_ontology.owl).

SPARQL Query:

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX dbp: <http://dbpedia.org/property/>
PREFIX ex:
<http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/university
-ontology#>

CONSTRUCT {
   ?university rdf:type ex:University .
   ?university ex:name ?universityName .
   ?university ex:numberOfStudents ?numberOfStudents .
   ?university ex:abstract ?abstract .
   ?university ex:facultySize ?facultySize .

   ?university ex:hasPresident ?president .
   ?president  ex:isPresidentOf ?university .
   ?university ex:isLocatedIn ?state .
   ?university ex:isOfType ?type .
   ?university ex:hasChancellor ?chancellor .
   ?chancellor ex:chancellorOf ?university .

   ?chancellor rdf:type ex:Chancellor .
   ?chancellor ex:chancellorName ?chancellorName .
   ?chancellor ex:chancellorBirthDate ?chancellorBirthDate .
   ?chancellor ex:chancellorBirthPlace ?chancellorBirthPlace .

    ?state rdf:type ex:State .
```

```
    ?state ex:stateName ?stateName .
    ?state ex:capitalName ?capitalName .

    ?type rdf:type ex:TypeOfUniversity .
    ?type ex:typeLabel ?typeOfUniversity .

    ?president rdf:type ex:President .
    ?president ex:presidentLabel ?presidentLabel .
}

WHERE {
  ?university rdf:type dbo:University ;
              rdfs:label ?universityName ;
              dbo:chancellor ?chancellor ;
              dbo:state ?state ;
              dbo:president ?president ;
              dbo:type ?type .

  OPTIONAL { ?university dbo:numberOfStudents ?numberOfStudents. }
  OPTIONAL { ?university dbo:facultySize ?facultySize . }
  OPTIONAL { ?university dbo:abstract ?abstract . }

  ?chancellor rdfs:label ?chancellorName .
  OPTIONAL { ?chancellor dbp:birthDate ?chancellorBirthDate . }
  OPTIONAL { ?chancellor dbp:birthPlace ?chancellorBirthPlace . }

  ?state rdfs:label ?stateName .
  OPTIONAL { ?state dbp:capital ?capitalName . }

  ?president rdfs:label ?presidentLabel .

  ?type rdfs:label ?typeOfUniversity .

  FILTER (LANG(?chancellorName) = "en")
  FILTER (LANG(?universityName) = "en")
  FILTER (LANG(?abstract) = "en")
  FILTER (LANG(?typeOfUniversity) = "en")
  FILTER (LANG(?stateName) = "en")
  FILTER (LANG(?capitalName) = "en")
  FILTER (LANG(?presidentLabel) = "en")

}
LIMIT 100
```

It loads existing RDF data, queries DBpedia using a SPARQL CONSTRUCT query, retrieves and processes RDdata,merges it with the existing graph, and saves the updated graph to dbpedia_loaded_university_ontology.owl

1.5. Justification, Explanation, and Validation of Ontological Modeling Decisions

Class Hierarchy and Properties:
The class hierarchy includes entities such as University, Chancellor, President, State, and University Type. Properties specific to each class capture essential details and define the relationships necessary to represent information about universities accurately.

Domain and Range Restrictions:
Domain and range restrictions are applied to properties to ensure that relationships and data values remain logical and appropriate across different classes. This helps in maintaining the accuracy and usability of the ontology data.

DL Axioms and Constraints:
Defining rules and limitations using Description Logic to govern how classes and properties interact. These rules ensure the ontology behaves as expected by defining permissible relationships and property values, thus preserving data integrity.

Ontology Population and Querying:
The ontology is populated with real world data from DBpedia, leveraging Python scripts for data extraction and SPARQL for querying. This process tests the ontology's ability to handle real-world data effectively and accomplish query requests, reflecting its utility in practical scenarios.

Validation:
Consistency checks and correctness tests were conducted using the Pellet Reasoner in Protégé 5.6.1. These tests are critical for ensuring that the ontology does not contain logical errors and that it functions as intended.

ONTOLOGY METRICS AFTER POPULATING DBPEDIA DATA

university-ontology (http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/university-ontology) : [/Users/sunanthakannan/Downloads/dbpedia_loaded_university_ontology.owl]

university-ontology (http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/university-ontology)

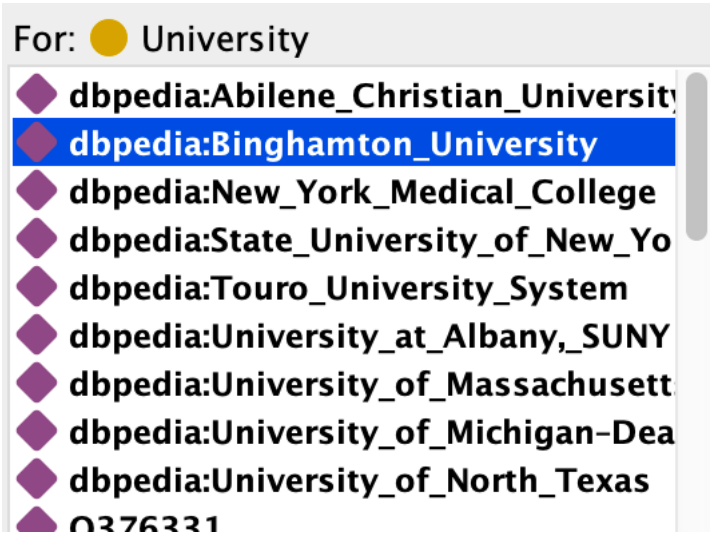
Ontology metrics:

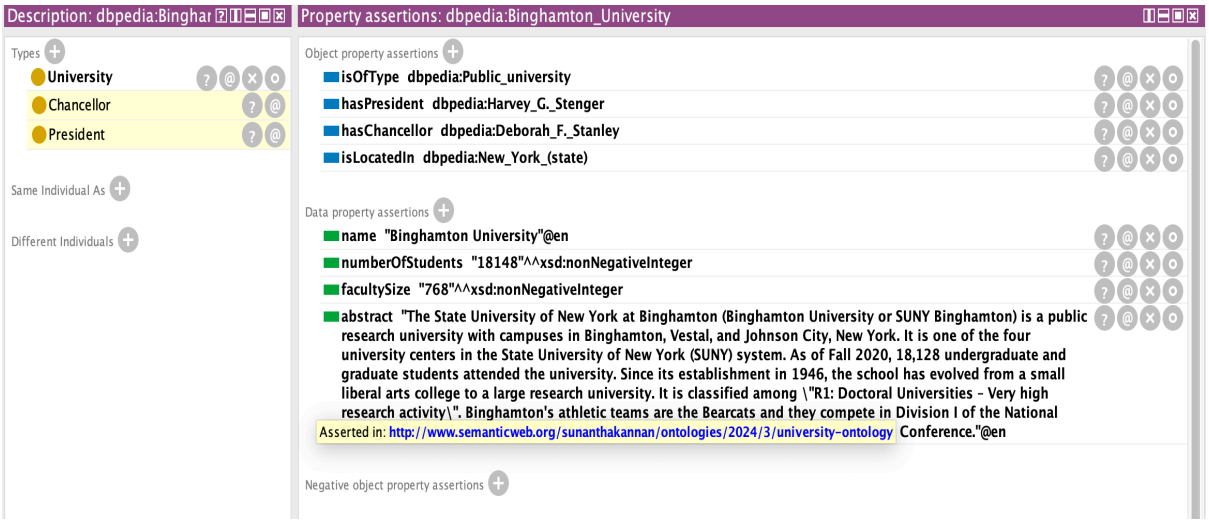
Metrics	
Axiom	253
Logical axiom count	225
Declaration axioms count	25
Class count	5
Object property count	7
Data property count	14
Individual count	30
Annotation Property count	1
Class axioms	
SubClassOf	9
EquivalentClasses	0
DisjointClasses	0
GCI count	0
Hidden GCI Count	0
Object property axioms	
SubObjectPropertyOf	6
EquivalentObjectProperties	0
InverseObjectProperties	2
DisjointObjectProperties	0
FunctionalObjectProperty	0
InverseFunctionalObjectProperty	0
TransitiveObjectProperty	0
SymmetricObjectProperty	0
AsymmetricObjectProperty	0
ReflexiveObjectProperty	0
IrrefexiveObjectProperty	0
ObjectPropertyDomain	6
ObjectPropertyRange	6
SubPropertyChainOf	0

Data property axioms	
SubDataPropertyOf	13
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	0
DataPropertyDomain	13
DataPropertyRange	13
Individual axioms	
ClassAssertion	31
ObjectPropertyAssertion	58
DataPropertyAssertion	68
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0
Annotation axioms	
AnnotationAssertion	3
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

Instance Description Section from DBPedia of one University:

The A BOX of this ontology defines specific universities, detailing their attributes such as location and student numbers, and links to their respective chancellors and presidents, embodying the real-world applications of the concepts described in the T BOX.





2. EXTRA TASK

2.1. Objective

The aim of this task is to integrate information from at least two distinct external data repositories and answers questions which cannot be answered by either remote knowledge base alone.

2.2. Methodology

In the previous task, DBpedia has been used to populate our ontology. To accomplish this task, Wikidata is used as other external source. The other data source is a CSV which is converted to RDF and loaded into the ontology.

2.3. Ontology Population from WikiData

WikiData is used as a second data source to populate the ontology by combining it with already DBPedia loaded data ontology.

In DBPedia, address and number of social media followers missing. I can be able to populate those fields from wikidata so that a person can know university address and also the popularity of the university based on the followers.

Source File:

The extratask1_wikidata_data_load.py in .zip contains the SPARQL query that populates data into our Ontology file (university_ontology_populated_data.owl).

SPARQL Query:

```
PREFIX wd: <http://www.wikidata.org/entity/>
PREFIX wdt: <http://www.wikidata.org/prop/direct/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX schema: <http://schema.org/>
PREFIX ex:
<http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/university
-ontology#>

CONSTRUCT {
  ?university rdf:type ex:University.
  ?university ex:name ?universityName.
  ?university ex:numberOfStudents ?numberOfStudents.
  ?university ex:abstract ?abstract.
  ?university ex:facultySize ?facultySize.
  ?university ex:address ?address.
  ?university ex:numberOfSocialMediaFollowers ?followers.
```

```
  ?university ex:hasPresident ?president.
  ?president ex:isPresident ?university.
  ?university ex:hasChancellor ?chancellor.
  ?chancellor ex:chancellorOf ?university.
  ?university ex:isLocatedIn ?location.
  ?university ex:isOfType ?type.


  ?president rdf:type ex:President.
  ?president ex:presidentLabel ?presidentName.

  ?chancellor rdf:type ex:Chancellor.
  ?chancellor ex:chancellorLabel ?chancellorName.
  ?chancellor ex:chancellorBirthDate ?chancellorBirthDate.
  ?chancellor ex:chancellorBirthPlace ?chancellorBirthPlace.

  ?location rdf:type ex:State.
  ?location ex:stateName ?stateName.
  ?location ex:capitalName ?capitalName.

  ?type rdf:type ex:TypeOfUniversity.
  ?type ex:typeLabel ?typeLabel.


}
WHERE {
  ?university wdt:P31 wd:Q3918;
  OPTIONAL { ?university rdfs:label ?universityName FILTER
(LANG(?universityName) = "en") }
  OPTIONAL { ?university wdt:P2196 ?numberOfStudents }
  OPTIONAL { ?university schema:description ?abstract FILTER
(LANG(?abstract) = "en") }
  OPTIONAL { ?university wdt:P3342 ?facultySize }
  OPTIONAL { ?university wdt:P6375 ?address }
  OPTIONAL { ?university wdt:P8687 ?followers }
  OPTIONAL {
    ?university wdt:P35 ?president .
    ?president rdfs:label ?presidentName FILTER (LANG(?presidentName) =
"en")
  }

  OPTIONAL {
    ?university wdt:P35 ?chancellor .
    ?chancellor rdfs:label ?chancellorName FILTER
(LANG(?chancellorName) = "en")
    OPTIONAL { ?chancellor wdt:P569 ?chancellorBirthDate }
    OPTIONAL { ?chancellor wdt:P19 ?chancellorBirthPlace FILTER
(LANG(?chancellorBirthPlace) = "en") }
```

```
  }

  OPTIONAL {
    ?university wdt:P131 ?location .
    ?location wdt:P1448 ?stateName FILTER (LANG(?stateName) = "en")
    OPTIONAL {
      ?location wdt:P36 ?capital .
      ?capital rdfs:label ?capitalName FILTER (LANG(?capitalName) =
"en")
    }
  }

  OPTIONAL {
    ?university wdt:P31 ?typeEntity .
    ?typeEntity rdfs:label ?typeLabel FILTER (LANG(?typeLabel) = "en")
  }


}
LIMIT 100
```

Explanation of the mechanism:

SPARQL Endpoint Setup: Initializes a connection to the Wikidata SPARQL endpoint to execute queries.

SPARQL Query:Constructs an RDF graph using a CONSTRUCT query, retrieving details like university names, number of students, location, followers, president, administrative data etc...Maps Wikidata properties to custom ontology terms, enriching the ontology with detailed attributes of universities.

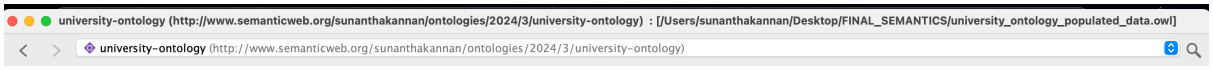
Execute and Fetch Data:Sends the query to Wikidata, receiving the data in RDF/XML format. Converts the returned data into an RDF graph.

Data Integration: Loads existing RDF data from a file and merges it with the newly fetched RDF data from Wikidata.This merged graph combines current and new data, maintaining a comprehensive dataset.

Save Updated Ontology: Serializes the updated RDF graph into an XML file, preserving the integrated data for further use(university_ontology_populated_data.owl).

ONTOLOGY METRICS:

ONTOLOGY METRICS AFTER WIKIDATA , DBPEDIA POPULATED ONTOLOGY



Ontology metrics:

Metrics

Axiom	759
Logical axiom count	601
Declaration axioms count	155
Class count	8
Object property count	7
Data property count	14
Individual count	126
Annotation Property count	4

Class axioms

SubClassOf	9
EquivalentClasses	0
DisjointClasses	0
GCI count	0
Hidden GCI Count	0

Object property axioms

SubObjectPropertyOf	6
EquivalentObjectProperties	0
InverseObjectProperties	2
DisjointObjectProperties	0
FunctionalObjectProperty	0
InverseFunctionalObjectProperty	0
TransitiveObjectProperty	0
SymmetricObjectProperty	0
AsymmetricObjectProperty	0
ReflexiveObjectProperty	0
IrrefexiveObjectProperty	0
ObjectPropertyDomain	6
ObjectPropertyRange	6
SubPropertyChainOf	0

Data property axioms

SubDataPropertyOf	13
EquivalentDataProperties	0
DisjointDataProperties	0
FunctionalDataProperty	0
DataPropertyDomain	13
DataPropertyRange	13

Individual axioms

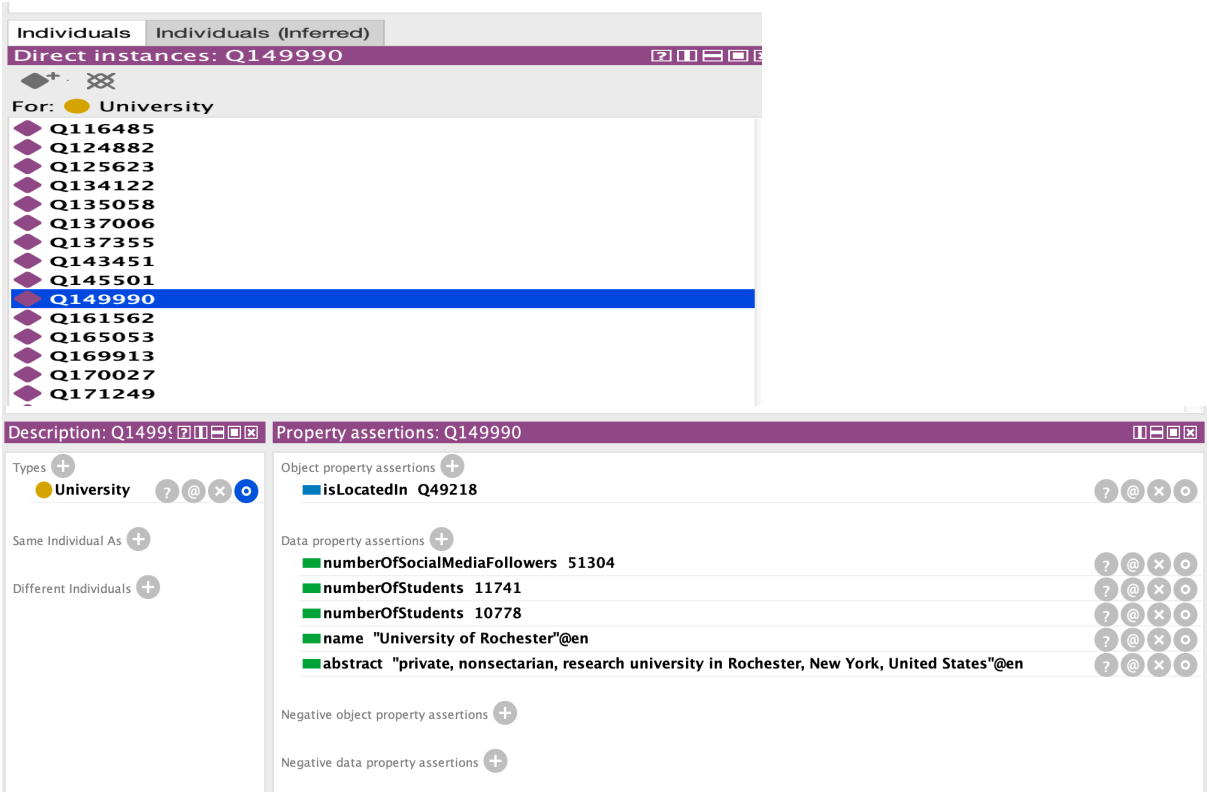
ClassAssertion	127
ObjectPropertyAssertion	64
DataPropertyAssertion	339
NegativeObjectPropertyAssertion	0
NegativeDataPropertyAssertion	0
SameIndividual	0
DifferentIndividuals	0

Annotation axioms

AnnotationAssertion	3
AnnotationPropertyDomain	0
AnnotationPropertyRangeOf	0

Instance Description Section from WIKIData of one University:

Populating extra fields from wikidata such as social media followers, University address which is not available in Dbpedia .



2.4. Integration of Non-Semantic Datasets

In this task, non-semantic data csv is created with required data to populate the ontology. A python script used to convert the CSV to RDF dumps. The RDF file is then imported directly into the active ontology.

Source File:

university_data.csv

extratask2_csv_data_load.py

university_populated_data.rdf

Source File:

```
import csv
from rdflib import Graph, Namespace, URIRef, Literal
from rdflib.namespace import RDF, XSD

# Define your ontology namespace
```

```
univ_ont =
Namespace("http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/
university-ontology#")

# Load the CSV file
with open('university_data.csv', newline='', encoding='utf-8') as
csvfile:
    reader = csv.DictReader(csvfile)
    g = Graph()

    for row in reader:
        # Create a new URI for each university
        university_uri = univ_ont[f"university/{row['name'].replace('
', '_')}"]
        g.add((university_uri, RDF.type, univ_ont.University))
        g.add((university_uri, univ_ont.name, Literal(row['name'])))
        g.add((university_uri, univ_ont.abstract,
Literal(row['abstract'])))
        g.add((university_uri, univ_ont.address,
Literal(row['address'])))
        g.add((university_uri, univ_ont.numberOfStudents,
Literal(row['numberOfStudents'], datatype=XSD.integer)))
        g.add((university_uri, univ_ont.numberOfSocialMediaFollowerspy,
Literal(row['numberOfSocialMediaFollowers'], datatype=XSD.integer)))
        g.add((university_uri, univ_ont.facultySize,
Literal(row['facultySize'], datatype=XSD.integer)))

        # President information
        president_uri =
univ_ont[f"president/{row['presidentLabel'].replace(' ', '_')}"]
        g.add((president_uri, RDF.type, univ_ont.President))
        g.add((president_uri, univ_ont.presidentName,
Literal(row['presidentLabel'])))
        g.add((university_uri, univ_ont.hasPresident, president_uri))

        # Chancellor information
        chancellor_uri =
univ_ont[f"chancellor/{row['chancellorName'].replace(' ', '_')}"]
        g.add((chancellor_uri, RDF.type, univ_ont.Chancellor))
        g.add((chancellor_uri, univ_ont.chancellorName,
Literal(row['chancellorName'])))
        g.add((chancellor_uri, univ_ont.chancellorBirthDate,
Literal(row['chancellorBirthDate'], datatype=XSD.date)))
        g.add((chancellor_uri, univ_ont.chancellorBirthPlace,
Literal(row['chancellorBirthPlace'])))
        g.add((university_uri, univ_ont.hasChancellor, chancellor_uri))

        # Location information
```

```
        state_uri = univ_ont[f"state/{row['stateName'].replace(' ',
'_')}"]
        g.add((state_uri, RDF.type, univ_ont.State))
        g.add((state_uri, univ_ont.stateName,
Literal(row['stateName'])))
        g.add((state_uri, univ_ont.capitalName,
Literal(row['capitalName'])))
        g.add((university_uri, univ_ont.isLocatedIn, state_uri))

        # University Type
        type_uri = univ_ont[f"type/{row['typeLabel'].replace(' ',
'_')}"]
        g.add((type_uri, RDF.type, univ_ont.TypeOfUniversity))
        g.add((type_uri, univ_ont.typeLabel,
Literal(row['typeLabel'])))
        g.add((university_uri, univ_ont.isOfType, type_uri))

    # Save the RDF data to a file
    g.serialize(destination="university_populated_data.rdf",
format="xml")
    print("RDF data generated and saved to 'university
populated_data.rdf'")
```

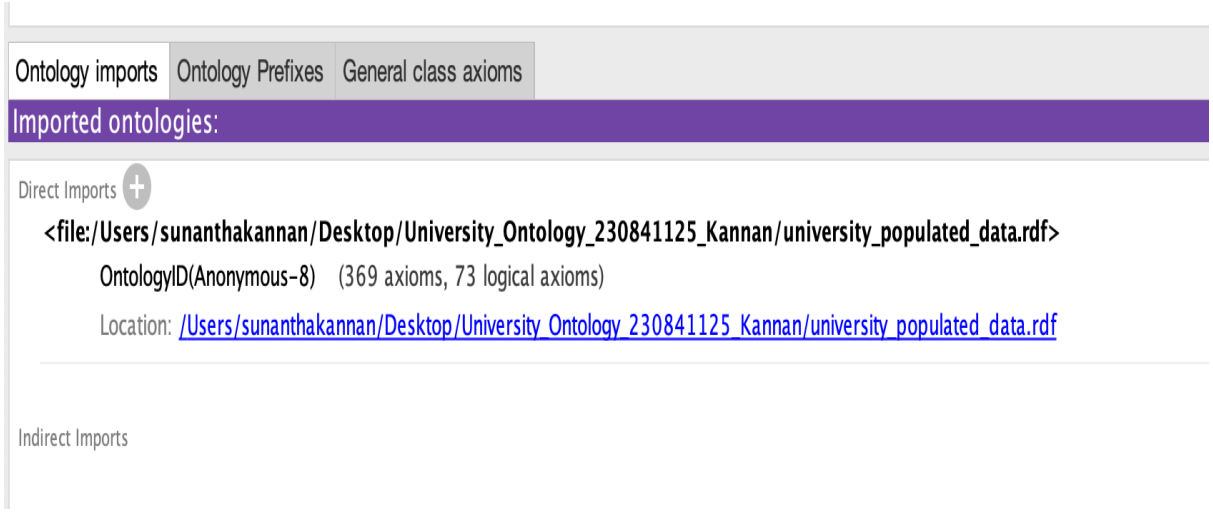
SAMPLE OUTPUT:

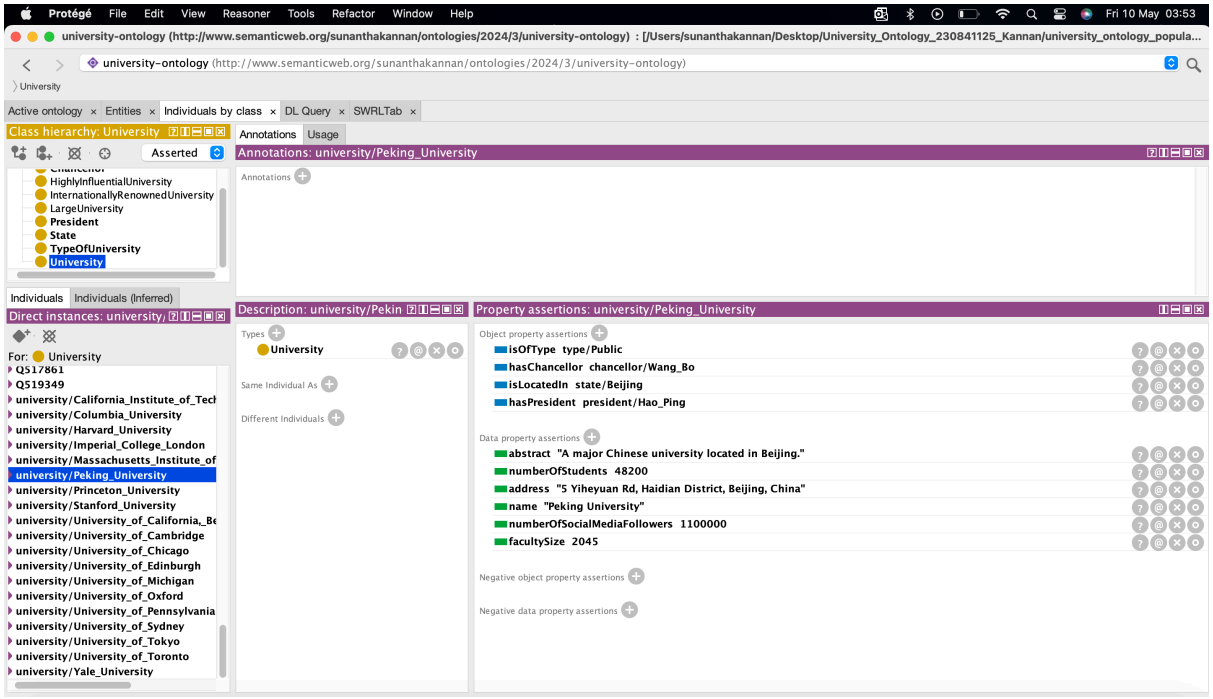
```
(venv) (base) sunanthakannan@192 University_Ontology_230841125_Kannan % python extratask2_csv_load_data.py
RDF data generated and saved to 'university_populated_data.rdf'
```

Imported generated RDF in active ontology:

NOTE: Just in case If instances are not visible under class after direct import of RDF file in Ontology. Keep it saved and launch it again. Data display will take place.

.





2.5. Querying Local Store

After integrating the data from wikidata, DBPedia and non semantic dataset csv into university ontology, querying locally make user can access information easily and the user should be able to execute any arbitrary query related to university details supported by university ontology

```
import rdflib

# Loaded the updated RDF graph
rdf_file = "university_ontology_populated_data.owl"
graph = rdflib.Graph()
graph.parse(rdf_file, format="xml")

# Defining a SPARQL query to check the integrated data
```

```
query = """
PREFIX ex:
<http://www.semanticweb.org/sunanthakannan/ontologies/2024/3/university
-ontology#>
SELECT ?university ?name ?numberOfStudents ?abstract ?facultySize
?address ?followers ?presidentName ?chancellorName
WHERE {
    ?university rdf:type ex:University ;
                ex:name ?name .
    OPTIONAL { ?university ex:numberOfStudents ?numberOfStudents . }
    OPTIONAL { ?university ex:abstract ?abstract . }
    OPTIONAL { ?university ex:facultySize ?facultySize . }
    OPTIONAL { ?university ex:address ?address . }
    OPTIONAL { ?university ex:numberOfSocialMediaFollowers ?followers .
}

    OPTIONAL {
        ?university ex:hasPresident ?president .
        ?president ex:presidentLabel ?presidentName .
    }

    OPTIONAL {
        ?university ex:hasChancellor ?chancellor .
        ?chancellor ex:chancellorLabel ?chancellorName .
    }
}
LIMIT 50
"""

# Executing the query
results = graph.query(query)
print(f"Total records found: {len(results)}")
for row in results:
    print("University URI:", row["university"])
    print("Name:", row["name"])
    print("Number of Students:", row.get("numberOfStudents", "Not
provided"))
    print("Abstract:", row.get("abstract", "Not provided"))
    print("Faculty Size:", row.get("facultySize", "Not provided"))
    print("Address:", row.get("address", "Not provided"))
    print("Social Media Followers:", row.get("followers", "Not
provided"))
    print("President Name:", row.get("presidentName", "Not provided"))
    print("Chancellor Name:", row.get("chancellorName", "Not
provided"))
    print()
```

SAMPLE OUTPUT:

```
(venv) (base) sunanthakannan@192 University_Ontology_230841125_Kannan % python query_local_store.py
Total records found: 50
University URI: http://dbpedia.org/resource/Abilene_Christian_University
Name: Abilene Christian University
Number of Students: 5731
Abstract: Abilene Christian University (ACU) is a private Christian university in Abilene, Texas. It was founded in 1906 as Childers Classical I
nstitute. ACU is one of the largest private universities in the Southwestern United States and has one of the 200 largest university endowments
in the United States. Affiliated with Churches of Christ, the university is nationally recognized for excellence in service learning, undergradu
ate research, and undergraduate teaching.
Faculty Size: 200
Address: None
Social Media Followers: None
President Name: Phil Schubert
Chancellor Name: None

University URI: http://dbpedia.org/resource/Binghamton_University
Name: Binghamton University
Number of Students: 18148
Abstract: The State University of New York at Binghamton (Binghamton University or SUNY Binghamton) is a public research university with campuse
s in Binghamton, Vestal, and Johnson City, New York. It is one of the four university centers in the State University of New York (SUNY) system.
 As of Fall 2020, 18,128 undergraduate and graduate students attended the university. Since its establishment in 1946, the school has evolved fr
om a small liberal arts college to a large research university. It is classified among "R1: Doctoral Universities – Very high research activity"
. Binghamton's athletic teams are the Bearcats and they compete in Division I of the National Collegiate Athletic Association (NCAA). The Bearca
ts are members of the America East Conference.
Faculty Size: 768
Address: None
Social Media Followers: None
President Name: Harvey G. Stenger
Chancellor Name: None

University URI: http://dbpedia.org/resource/New_York_Medical_College
Name: New York Medical College
Number of Students: 1660
Abstract: New York Medical College (NYMC or New York Med) is a private medical school in Valhalla, New York. Founded in 1860, it is a member of
the Touro College and University System. NYMC offers advanced degrees through its three schools: the School of Medicine (SOM), the Graduate Scho
ol of Biomedical Sciences (GSBMS) and the School of Health Sciences and Practice (SHSP). Total enrollment is 1,660 students (including 774 medic
al students) in addition to 800 residents and clinical fellows. NYMC employs 1,350 full-time faculty members and 1,450 part-time and voluntary f
aculty. The university has more than 12,000 alumni active in medical practice, healthcare administration, public health, teaching and research.
Part of the Touro College and University System since 2011, New York Medical College is located on a shared suburban 600-acre campus with its ac
ademic medical center, Westchester Medical Center (WMC) and the Maria Fareri Children's Hospital. Many of NYMC's faculty provide patient care, t
each, and conduct research at WMC. New York Medical College's university hospital, Metropolitan Hospital Center, in the Upper East side neighbor
hood of Yorkville and East Harlem in Manhattan, has been affiliated with NYMC since it was founded in 1875, representing the oldest partnership
between a hospital and a private medical school in the United States. Metropolitan is part of the New York City Health and Hospitals Corporation
 (HHC), the largest municipal hospital and healthcare system in the country. With a network of 20+ affiliated hospitals in New York, New Jersey,
 Connecticut and West Virginia, NYMC's hospital affiliations include large urban medical centers, small suburban clinics, rural medical centers
and high-tech regional tertiary care facilities, where medical students and residents are afforded a wide variety of clinical training opportuni
ties.
Faculty Size: 3000
Address: None
Social Media Followers: None
President Name: Alan Kadish
```

```
University URI: http://www.wikidata.org/entity/Q376331
Name: Nordic Summer University
Number of Students: None
Abstract: university
Faculty Size: None
Address: None
Social Media Followers: None
President Name: None
Chancellor Name: None

University URI: http://www.wikidata.org/entity/Q378014
Name: University of al-Qarawiyyin
Number of Students: None
Abstract: University in Fez, Morocco
Faculty Size: None
Address: None
Social Media Followers: None
President Name: None
Chancellor Name: None

University URI: http://www.wikidata.org/entity/Q382711
Name: Sanyo-Onoda City University
Number of Students: None
Abstract: higher education institution in Yamaguchi Prefecture, Japan
Faculty Size: None
Address: None
Social Media Followers: None
President Name: None
Chancellor Name: None

University URI: http://www.wikidata.org/entity/Q386203
Name: Ōtani University
Number of Students: None
Abstract: university in Kyoto, Japan
Faculty Size: None
Address: None
Social Media Followers: 2437
President Name: None
Chancellor Name: None

University URI: http://www.wikidata.org/entity/Q389852
Name: Vilnius Gediminas Technical University
Number of Students: None
Abstract: public university in Vilnius, Lithuania
Faculty Size: None
Address: None
Social Media Followers: None
President Name: None
Chancellor Name: None
```

2.6. Conclusion

Drawing a conclusion from an ontology that integrates data from diverse sources like DBpedia, Wikidata, and a non-semantic dataset (CSV).

The ontology provides detailed information about universities globally. This is due to the use of:**DBpedia**, which offers structured historical and contextual data.**Wikidata**, known for its up-to-date and editable content.**CSV Data**, which adds specific data that may not be available in other sources.

Integrating data from multiple sources enhances the detail and accuracy of your ontology: The variety of data enriches the information available.Cross-checking information from different sources helps correct errors, enhancing reliability.

The ontology is useful for several groups:Academic Researchers can study trends or compare educational data.**Prospective Students** can access detailed university information to help choose where to study.**Educational Administrators** can use the data for planning and improving educational offerings.

3. Advanced Task (20%)

3.1 Objective

Implementing SWRL rules and reasoners in university ontology populated data file, to express additional things that can be inferred from my dataset. Some inferences can be concluded from the ontology model itself, but others may not be expressible in the ontology language which requires a more functional representation.

3.2 Implemented SWRL Rules

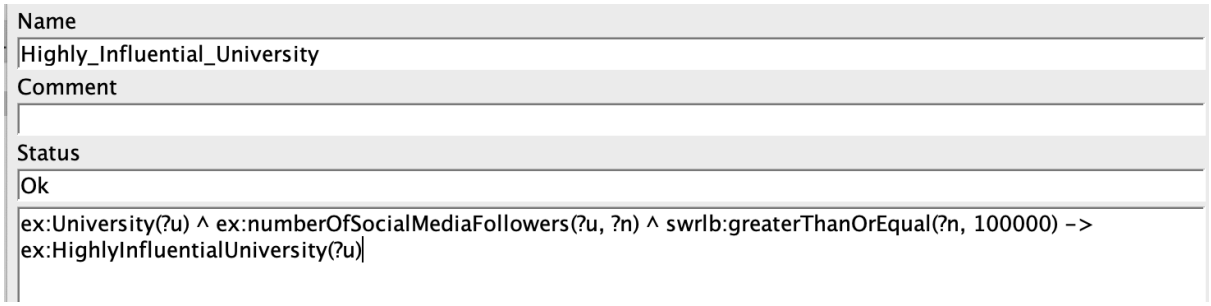
Pellet reasoner chosen here to make my model consistent after applying SWRL. (**Run the reasoner**)

3.2.1. Rule 1: Identifying universities with a significant social media following and classifying them as **'Highly Influential Universities'**.

Explanation:

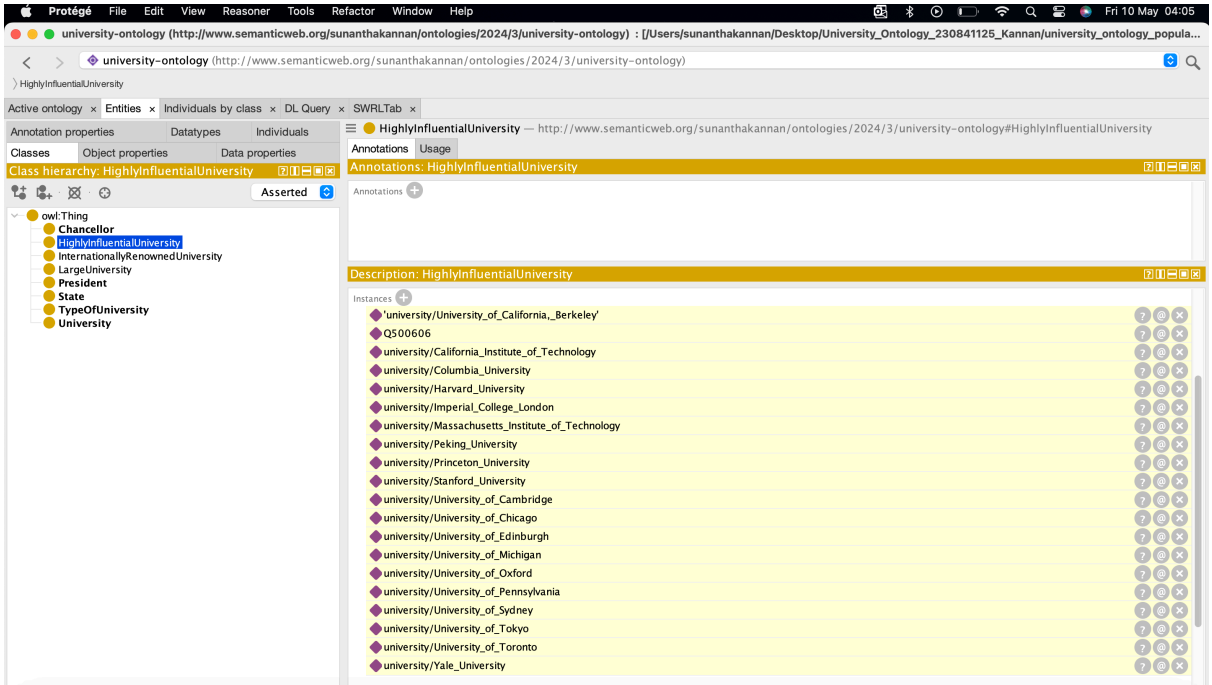
This rule filters universities that have a social media following of 100,000 or more and assigns them to a specific category, designated as 'Highly Influential Universities'. This categorization allows for easier organization, querying, and analysis of universities based on their influence and visibility on social media platforms.

SWRL Rule:



Output:

After applying the above rule, universities with more social media followings are identified and classified under the 'Highly Influential University' category. This provides the successful implementation of the rule and highlights universities that are particularly good in the social media platform, which influences prospective students, academic partnerships, and expanded community engagement.



3.2.2. Rule 2:

Identifying universities with large student bodies and substantial faculty size, classifying them as **'Internationally Renowned Universities'.**

Explanation:

This rule filters universities based on two key metrics: the number of students and the size of the faculty. It targets universities with more than 15,000 students and a faculty of over 1,000 members. Universities meeting these criteria are assigned to the category **'Internationally**

Renowned Universities'. This classification facilitates better organization, querying, and analysis of universities that are likely to have a significant impact and presence in the global academic community, based on their capacity and resources.

SWRL Rule:

Name

Internationally_Renowned_University

Comment

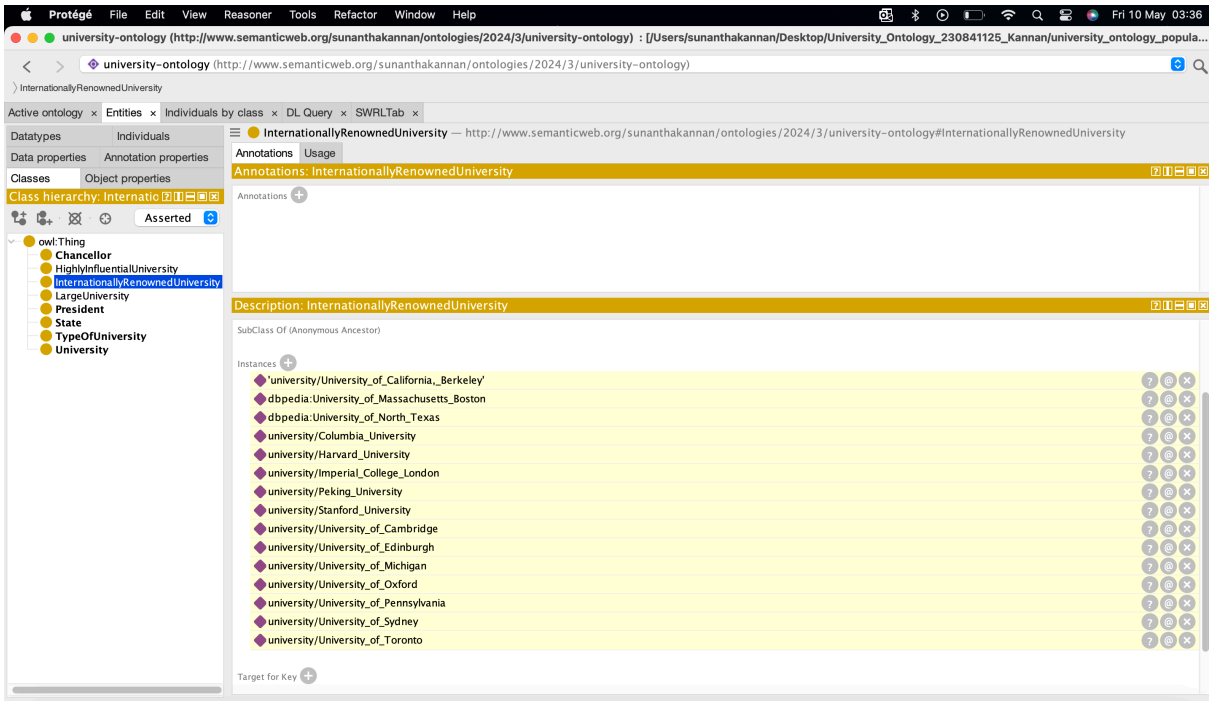
Status

Ok

```
ex:University(?u) ^ ex:numberOfStudents(?u, ?n) ^ ex:facultySize(?u, ?f) ^ swrlb:greaterThan(?n, 15000) ^
swrlb:greaterThan(?f, 1000) -> ex:InternationallyRenownedUniversity(?u)
```

Output:

After implementing this rule, the below universities that are large in both student population and faculty numbers are successfully identified and classified under the 'Internationally Renowned University' category which underscores the capabilities of these institutions to offer diverse programs and engage in extensive research activities, factors that contribute to their international renown.



3.2.3. Rule 3: Identifying universities with very large student bodies, classifying them as **'Large Universities'.**

Explanation:

The rule identifies universities with over 20,000 students as 'Large Universities'. This classification underscores their scale, reflecting their ability to accommodate a large student population.

SWRL Rule:

Name

Large_University

Comment

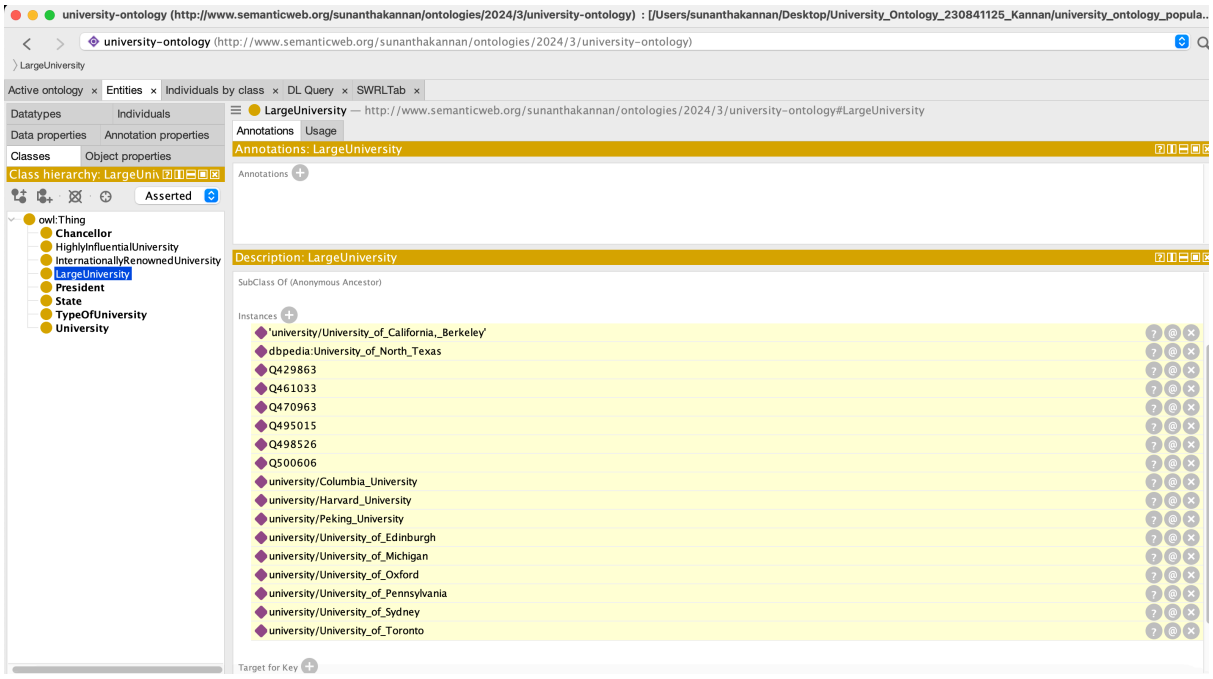
Status

Ok

```
ex:University(?u) ^ ex:numberOfStudents(?u, ?n) ^ swrlb:greaterThan(?n, 20000) -> ex:LargeUniversity(?u)
```

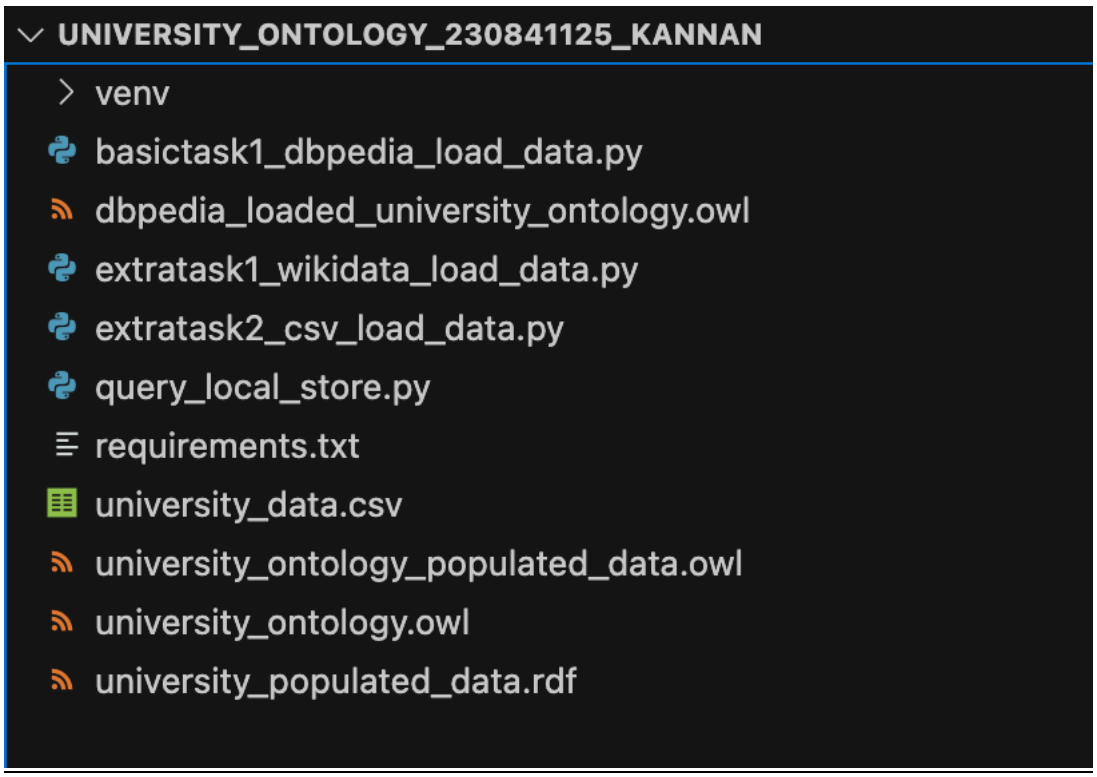
Output:

After applying this rule, universities with student populations exceeding 20,000 are successfully identified and classified under the 'Large University' category. This helps to identifying major educational institutions that play a significant role in accommodating a large number of students, often correlated with a broad curriculum and significant campus resources.



4.SOURCE CODE & EXECUTION

.zip file contains below files.



COMMANDS:

Go to the directory that contains all these files and run the following commands to setup virtualenv.

```
python3 -m venv venv
source venv/bin/activate
# Install requirements
pip install -r requirements.txt
```

To load data from DBpedia into our ontology. Run the following command.

```
python basictask1_dbpedia_load_data.py
```

To load data from WikiData into our ontology. Run the following command.

```
python extratask1_wikidata_load_data.py
```

To load data from CSV into our ontology. Run the following command.

```
python extratask2_csv_load_data .py
```

This code takes the csv file and converts it into an RDF file. This file has to be imported into the active ontology under direct imports. The reasoner requires synchronisation for the SWRL inferences to reflect in ontology.

5: REFERENCES

https://dbpedia.org/ontology/University

https://dbpedia.org/page/Virginia_Tech_Carilion_School_of_Medicine_and_Research_Institute

https://www.wikidata.org/wiki/Q3918

https://www.wikidata.org/wiki/Q487556

https://protege.stanford.edu/conference/2007/slides/08.01_OConnor.pdf