Covid-19 Data Pipeline

Final report Knowledge and Data 2024 - Group [77]

Matúš Hába - 2813195
Nikolai Bachkaikine - 2809150
Wiktor Jurkiewicz - 2812471
Jakub Kuwałek - 2796409

Date: [28-10-2024]

1 Introduction

1.1 Goal, motivation and users

The goal of our data pipeline is to accomplish 4 key aspects. The first being the creation of a core supportive ontology that can integrate RDF and non RDF sources easily. The ontology creates a framework for anyone to work with the basic concepts when it comes to capturing an epidemic in populated geographical areas over a period of time.

The second aspect of our data pipeline is the quantization of prediction accuracy of a logistic-growth model, when working with the covid-19 historical data.

The third aspect would be a visualization of covid-19 spread in the Netherlands using an interactive graph.

The fourth aspect is interconnectivity between all the elements meaning there will be one notebook from which data will flow from one source to another creating the ontology, predictions and visualizations in one go.

Our pipeline addresses two main problems. First, it provides an accessible and well-structured infectious disease ontology for academic research and integration with other ontologies. Second, it offers insights into model accuracy and the interaction between machine learning and ontology-based data, allowing for better understanding and evaluation of predictive models in the context of infectious diseases.

This project helps stakeholders, like those in medical research and public health, by fulfilling their need for organized data. By creating an infectious disease ontology and providing insights into COVID-19 prediction models, stakeholders can use this structure to speed up their own research and decision-making. It makes building similar models for future diseases faster and easier, helping them respond more quickly when needed.
The final notebook should be one single pipeline that includes building of the ontology and transfer of ontology data to different cells for prediction and visualization.

1.2 Competency Questions

1) How well can a logistic growth function perform on a covid-19 dataset more precisely tested on the infectious population of Amsterdam in between the 10th of april 2020 and the 31st of march 2023?

This research question was chosen as we wanted to see how well machine learning models perform on covid-19 data sets providing us with insights into their real world performance.

2) How can we create an intuitive and visually attractive representation of covid-19 cases?

We chose this question as it is a crucial component for understanding what data we are dealing with. Understanding covid-19 cases via a visualization will help us connect with the data in a more meaningful way.

3) How can we create an universal and extensible ontology to model infectious diseases over time? How can this ontology be extended with other ontologies as well as external data sources?

This question allows us to think about the ontology as the common format between data collection and analysis/visualization. This model enables us to add new data sources without the need to change the way we retrieve the data for analysis.

2 External sources identified

2.1 Existing Ontologies identified

The first ontology we identified is the IDO ontology made by John Beverley, Shane Babcock and Lindsay Cowell. This ontology is centered around infectious diseases. This ontology, as the authors state, provides the following aspects : “At the core of the set is a general Infectious Disease Ontology (IDO-Core) of entities relevant to both biomedical and clinical aspects of most infectious diseases.” The biomedical and clinical aspects specifically may be useful for expanding our ontology as it will add more accurate information to it since we lack expertise in that domain.

The second ontology that we have decided to integrate our ontology with is the **Human Disease Ontology** (DO), which is accessible at this link. This ontology provides a detailed classification of human diseases, offering structured and standardized data on disease types and their characteristics. By linking to the Human Disease Ontology, we ensure our ontology aligns with widely accepted biomedical standards, enhancing interoperability across systems. The DO framework categorizes diseases based on various criteria such as causative agents, affected anatomical locations, and related medical conditions. COVID-19 is classified as a viral disease that falls under respiratory illnesses, particularly linked to viral pneumonia and lung disease. By adding this connection we achieved a more precise classification and organization of COVID-19, ensuring that our ontology reflects up-to-date, accurate medical knowledge. This approach not only improves the consistency of data representation but also makes it easier to share and integrate our ontology with other systems that follow similar ontological structures in medical research and clinical databases.

2.2 External SPARQL endpoints identified

The first data set that may be used is the wikidata covid-19 data set. This data set contains several elements that describe core aspects of the disease as well as death counts which may be useful for adding information to our ontology as. The second data set we found, DBpedia, includes geographical data of the Netherlands, concerning its cities and most importantly their total population, which will allow us to calculate the ratio of the infected and not infected.

In our search for data sources, we have identified and used wikidata as a reliable source of information about the population of cities and countries. For our project specifically we used the data concerning Dutch cities and the size of their individual populations. Wikidata also includes a label service, which allows us to label all cities in original dutch as well as english. The data was easily accessible, since wikidata offers a sparQL endpoint as well as GUI to interact with their knowledge graph.

2.3 External non-RDF Datasets identified

The first dataset we discovered does not contain a SPARQL endpoint but can be extracted via other means. It is present on kaggle. This data set contains a large amount of values which is what we need for neural network predictions.
The second dataset we might be using is the data set from the world health organization. This dataset may be used to verify values from the first dataset or be used in case of missing values. The data set seems to be credible, as the WHO is a good authority when it comes to epidemics.

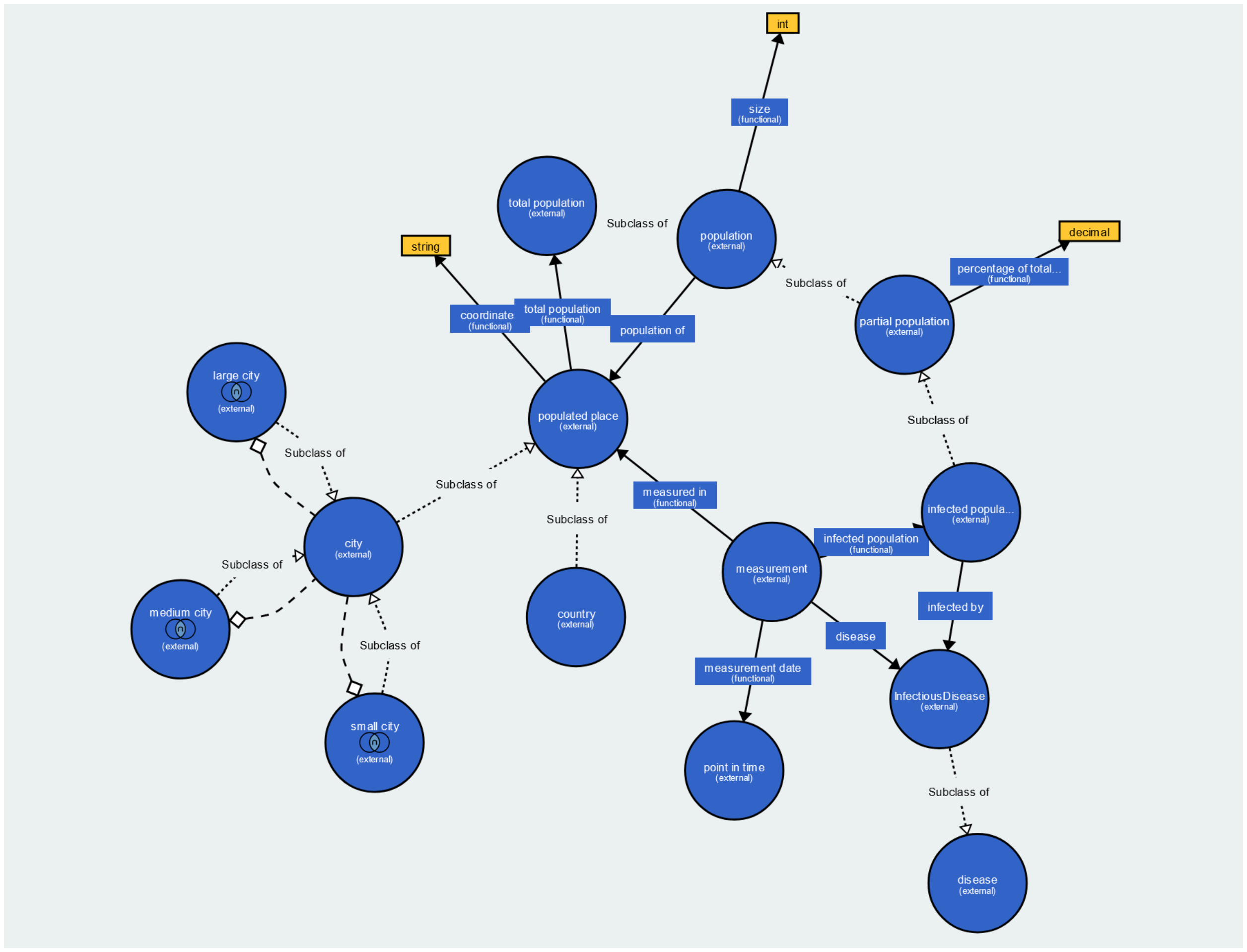
We ended up using a data set from the Rijksinstituut voor Volksgezondheid en Milieu, which contains detailed data about new cases reported every day from April 10th 2020 to March 31st 2023. This data is saved in a csv file and covers a great number of Dutch municipalities, including data about the province they are in etc. This data source is very reliable, since the data comes from the Dutch authorities, namely The National Institute for public health and the environment, which falls under the ministry of health, well-being and sport.

3 Design of Ontology

3.1 Methodology

The ontology is constructed and edited using an ontology editor named Protégé, which is a free, open-source software tool primarily used for building and managing ontologies. It allows users to create, edit, and visualize complex data structures that define the relationships between various concepts in a specific domain. Developed by Stanford University, Protégé supports OWL (Web Ontology Language), making it popular for knowledge representation in numerous fields. The main advantages that this editor uses are all parts of its environment, which make it easy to navigate an existing ontology, as well as keep track of classes, properties and their attributes (domain, range…) as well as class and property hierarchy. The software also offers an option to install extensions, which will be taken advantage of by using the Pellet reasoner to check knowledge graph consistency and infer not explicitly stated triples from sources like domain and range restrictions which will make our ontology less error prone, as well as faster to develop.

3.2 Conceptualisation



3.2.1 Classes*Include a table or list of classes and their meta-properties (disjointness, etc)*

PopulatedPlace
Description: Represents a place where at least one person lives.

Country
Subclass of: PopulatedPlace
Description: Represents a country.

City
Subclass of: PopulatedPlace
Description: Represents a city.

LargeCity
Subclass of: City

Disjoint with: SmallCity, MediumCity
Description: Represents a city with a population of at least 500,000.

MediumCity
Subclass of: City
Disjoint with: SmallCity, LargeCity
Description: Represents a city with a population between 100,000 (inclusive) and 500,000 (exclusive).

SmallCity
Subclass of: City
Disjoint with: MediumCity, LargeCity
Description: Represents a city with a population of less than 100,000.

Population
Description: Represents the population entity related to a city.

TotalPopulation
Subclass of: Population
Description: Represents the total population of a city.

PartialPopulation
Subclass of: Population
Description: Represents a part of a population.

InfectedPopulation
Subclass of: PartialPopulation
Description: Represents a proportion of a population infected by a disease at a certain point in time.

Disease
Description: Represents a disease.

InfectiousDisease
Subclass of: Disease
Description: Represents a disease that is infectious - can be contracted by being in contact with a person infected by the disease.

TimePoint
Description: Represents a specific point in time.

Measurement
Description: A class that relates time, disease, city and an infected population.

3.2.2 Properties

Include a table or list of properties and their meta-properties (transitive, etc)

populatedBy
Functional property
Domain: Populated place
Range: TotalPopulation
Description: Establishes a population as the total population of that city.

populationOf
Domain: Population
Range: PopulatedPlace
Description: Used to link a population to a city.

size
Functional property
Domain: Population
Range: xsd:int
Description: Represents the size of the population.

percentageOfPopulation
Functional property
Domain: PartialPopulation
Range: xsd:decimal
Description: Represents the proportion of the PartialPopulation in respect to the TotalPopulation of a City.

coordinates
Functional property
Domain: PopulatedPlace
Range: xsd:string
Description: A point describing the latitude and longitude of the center of a PopulatedPlace.

infectedBy
Domain: Infected population
Range: InfectiousDisease
Description: Links an infected population to an infectious disease.

measuredIn
Functional property
Domain: Measurement
Range: City
Description: Used to link a city to a measurement.

measuredDisease
Domain: Measurement
Range: InfectiousDisease
Description: Used to link data points to a certain time point.

measuredPopulation
Functional property
Domain: Measurement
Range: InfectedPopulation
Description: Used to link an infected population (of a certain size) to the measurement.

measuredAt
Functional property
Domain: Measurement
Range: TimePoint
Description: Used to link instances associated with a Measurement back to the TimePoint.

3.2.3 Restrictions

Include a table or list of the
OWL restrictions used

LargeCity
owl:equivalentTo City and population some (Population and size some xsd:int >300000, <=1000000)

MediumCity
owl:equivalentTo City and population some (Population and size some xsd:int >100000, <=300000)

SmallCity
owl:equivalentTo City and population some (Population and size some xsd:int <300000)

3.2.4 SWRL Rules

Taking advantage of the Pellet reasoner's capabilities, we have added the following rules which infer new triples in our ontology. These were often not possible to implement using the OWL ruleset, which doesn't for example include a rule for a 1 way inverse rule, or isn't capable of calculating new values based on existing data. The all parts of these generated triples are described in sections 3.2.1 - 3.2.3; this section only shows how they are generated by the reasoner.

populationOf

- populatedBy(?c, ?p) -> populationOf(?p, ?c)
- measuredPopulation(?dp, ?ip) ^ measuredIn(?dp, ?c) -> populationOf(?ip, ?c)

percentageOfPopulation

- PartialPopulation(?pp) ^ populationOf(?pp, ?c) ^ size(?pp, ?x) ^ populatedBy(?c, ?tp) ^ size(?tp, ?y) ^ swrlb:multiply(?temp, 100, ?x) ^ swrlb:divide(?percentage, ?temp, ?y) -> percentageOfPopulation(?pp, ?percentage)

infectedBy

- measuredPopulation(?dp, ?ip) ^ measuredDisease(?dp, ?d) -> infectedBy(?ip, ?d)

3.3 Formalization / Implementation

The final ontology looks similar to the original ontology, except it is connected to the Human Disease Ontology, as described in the following section in the paragraph about ontology extension. This ontology is very robust, since The human disease ontology provides a very specific and robust framework for disease classification

4 Data Integration and conversion

The process of creating the full knowledge graph containing data from all sources, as well as newly inferred triples is outlined in the pipeline using sections and subsections that have to follow a specific order. For the purposes of describing this process such that the description of its individual parts can be easily matched to the process in the pipeline we will refer to the parts of the code as subsections of section 1, as labeled in the jupyter notebook.

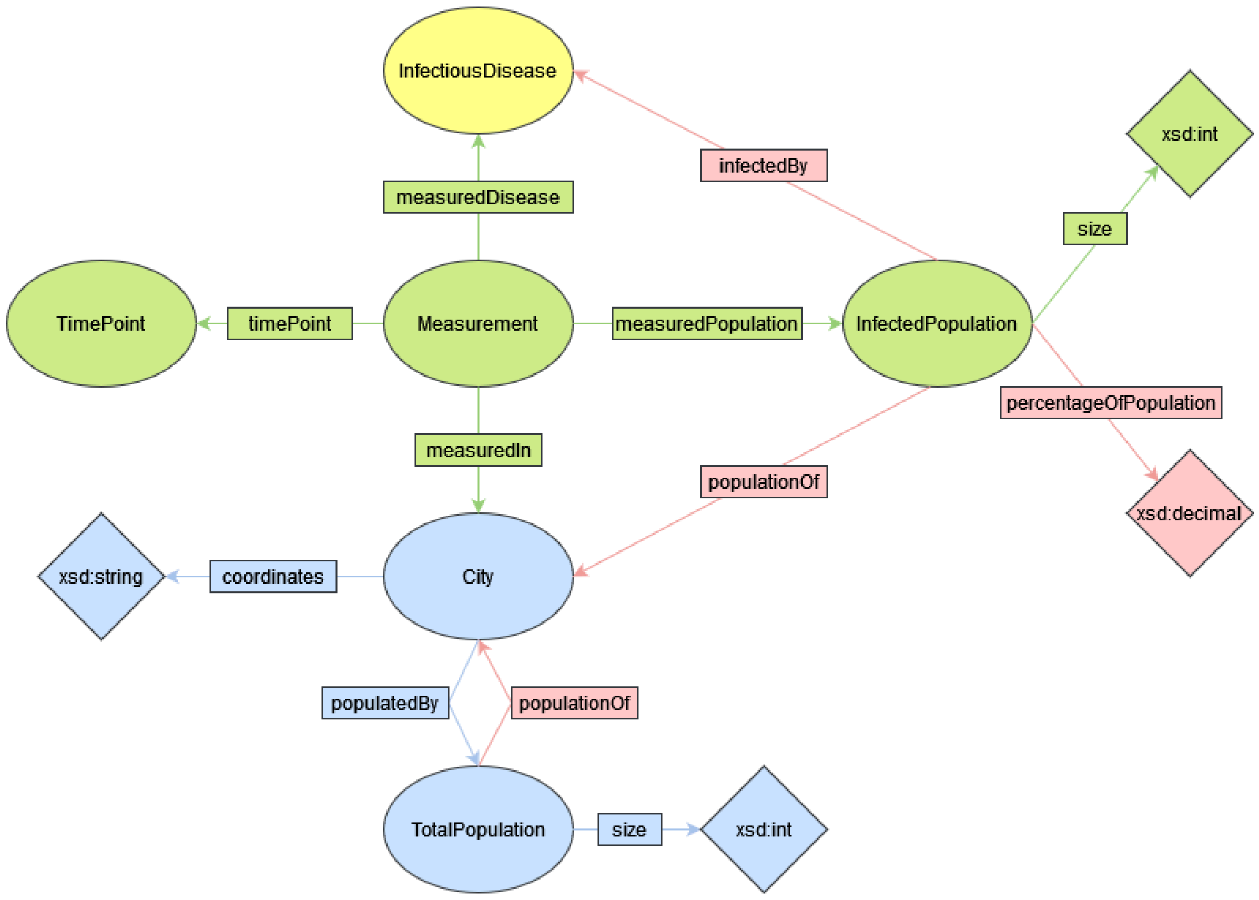


Figure 4.1

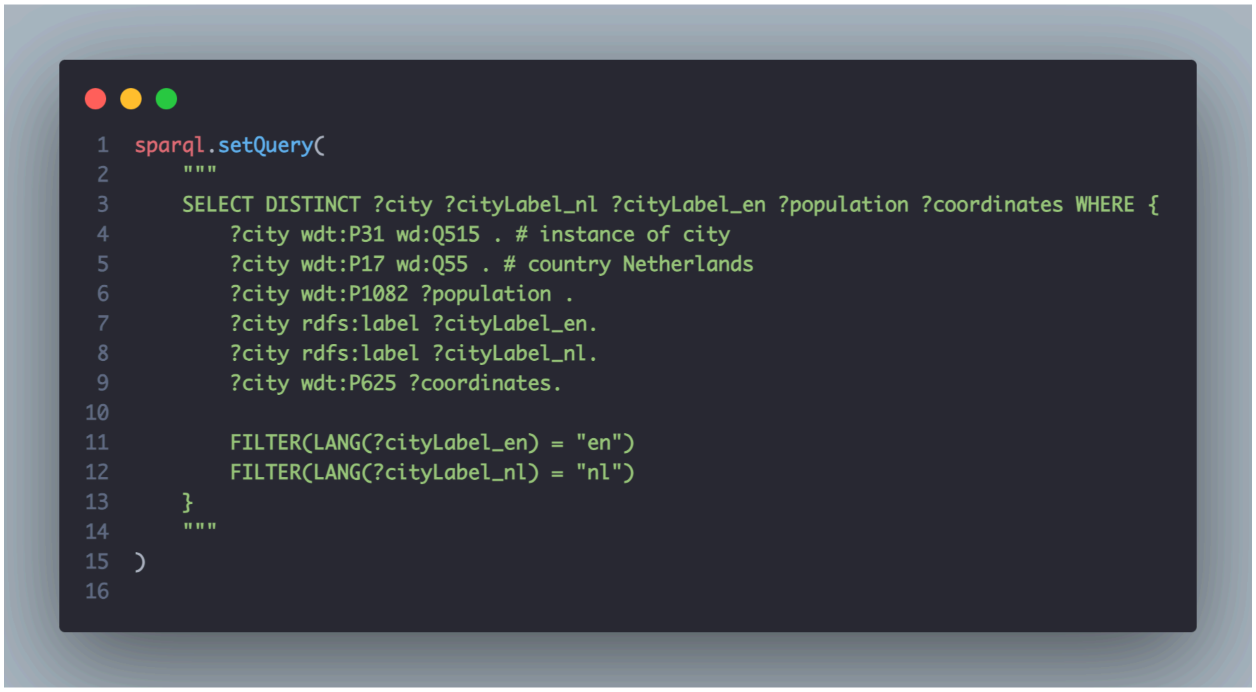
Figure 4.1 describes the relations between instances, which are instances of these classes (and their super classes). They are color-coded to show their origin as generated by different sections of the code.

Subsection 1.1 - Ontology extension

At first, an empty graph is created and the empty “epi” (as described in Section 3 of this report) ontology is imported. Then, the Human Disease Ontology (hdo) is loaded from the web using its iri, and is stripped of some annotation properties that do not apply to the merged ontology. Afterwards, the two knowledge graphs are merged together. The ontologies are linked using owl:equivalentClass properties, as explicitly stated in the code. Since we integrated the epi ontology with the hdo ontology, it is also possible to replace its iri with any iri of a hdo-based ontology or knowledge graph and potentially use this as a source of disease data, which allows for further analysis capabilities based on disease properties. Our ontology is merged using the equivalences of epi:Disease === hdo:DOID_4 and epi:InfectiousDisease === hdo:DOID_0050117.

Subsection 1.2 Infectious diseases (Yellow)
A new instance of epi:infectiousDisease class is created, and categorized based on the detailed hierarchy of the hdo ontology. This process has to be done manually (changed manually in the pipeline), since automatic categorization can result in incorrect classification of the disease, and is limited to knowledge captured in a data source. This way we allow the user to easily categorize even a newly discovered disease based on their knowledge. s.

Subsection 1.3 - Cities and Population (Blue)
This subsection is dedicated to extracting data from the Wikidata knowledge graph concerning cities in the Netherlands and their population. As stated on their website, “Wikidata is a free, collaborative, multilingual, secondary knowledge base, collecting structured data to provide support for Wikipedia, and to anyone in the world.” It is generally a reliable source of structured statistical data, such as population, and includes a number of the biggest dutch cities, which conveniently overlaps with the needs of this project. To extract data from this knowledge graph we used the following query:



```
sparql.setQuery(
    """
    SELECT DISTINCT ?city ?cityLabel_nl ?cityLabel_en ?population ?coordinates WHERE {
        ?city wdt:P31 wd:Q515 . # instance of city
        ?city wdt:P17 wd:Q55 . # country Netherlands
        ?city wdt:P1082 ?population .
        ?city rdfs:label ?cityLabel_en.
        ?city rdfs:label ?cityLabel_nl.
        ?city wdt:P625 ?coordinates.

        FILTER(LANG(?cityLabel_en) = "en")
        FILTER(LANG(?cityLabel_nl) = "nl")
    }
    """
)

```

This query outputs a city, with its name in Dutch and English, as well as its population and geographical coordinates. The query selects these based on wikidata properties that a city has.

Subsection 1.4 - Measurements (Green)
This section works with a Covid-19 report from the Dutch statistical office, that states how many new infections were reported every day from April 10th 2020 to March 31st 2023. This data is saved in a csv file and covers a great number of Dutch municipalities, including data about the province they are in etc. We imported the data using python csv module, and created new triples based on the epi ontology class hierarchy.

Subsection 1.6 - Pellet Reasoner (Red)
This section runs the pellet reasoner generating a new file which includes the complete graph with the inferred triples. The triples are inferred based on the ontology as described in section 3 of this report.

5 The resulting knowledge graph

The final knowledge graph consists of the extended ontology along with instances generated by our data pipeline. This graph includes details about cities and their respective populations, documented infectious disease(s), measurement instances, specific timepoints, and counts of infected populations. The infected population data reflects the cumulative number of individuals who were sick up until the specific date associated with each measurement instance. This structure allows for an integrated view of demographic, temporal, and health-related data, supporting comprehensive analysis of infection trends and population impact across different cities and time periods.

Our ontology is of moderate size, supported by several key statistics that highlight its structure and capacity. Firstly, in terms of class count, the Protégé software identifies exactly 18 790 classes, indicating the extensive range of categories and distinctions captured within the ontology. This substantial class count supports diverse knowledge representation and categorization.

Secondly, regarding object properties, our ontology includes 52 distinct object properties, enabling a robust and versatile framework for capturing the relationships between classes. This variety in object properties facilitates intricate connections and associations, allowing for detailed and nuanced representations of the domain.

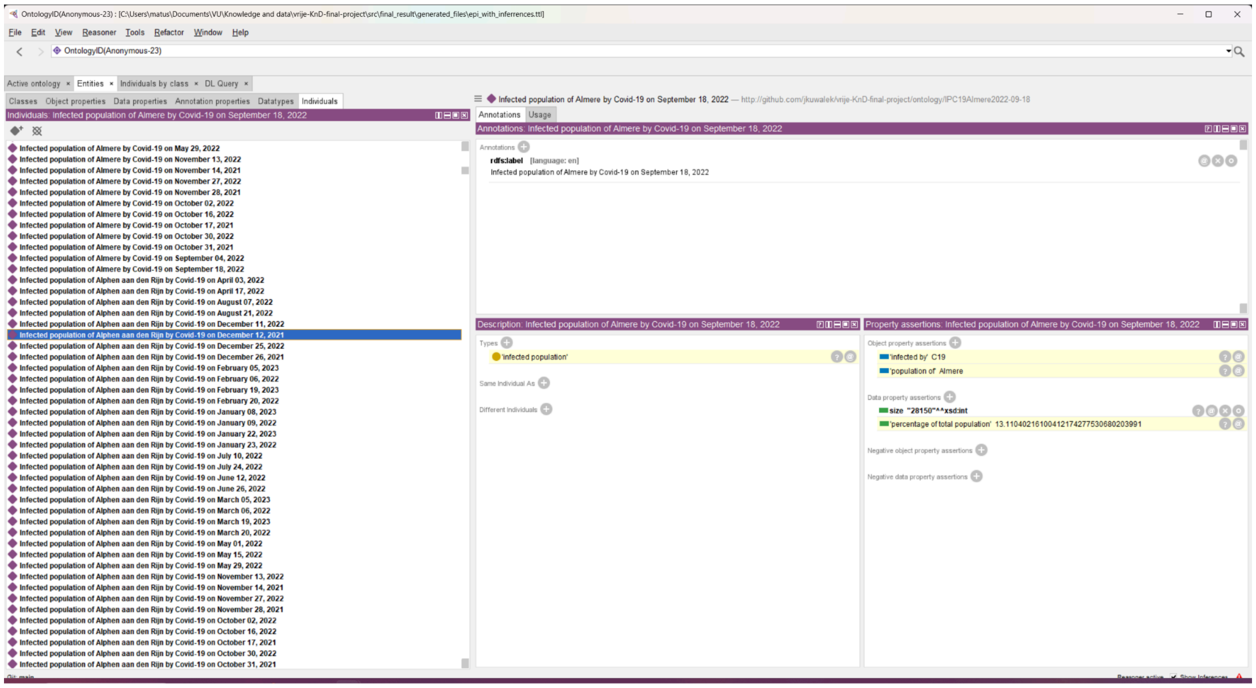
For data properties, we have deliberately limited our ontology to only 3 data properties. While this may seem minimal, the decision is purposeful. The large datasets utilized are capable of inferring additional information through these data properties, which aligns with our focus on efficiency and computational performance within Protégé. This careful selection enhances the capacity for logical inference and leverages Protégé's reasoning capabilities without unnecessary complexity.

Lastly, our ontology contains 3,261 individuals. These individuals are integral to the ontology's functionality, serving as concrete instances that link the abstract classes and properties to real-world data. This careful curation of individuals ensures that the ontology accurately aligns with the datasets it aims to model.

We had to limit the individual count in order to increase computational efficiency, in terms of both space and time complexity of the reasoner. Increasing the number of individuals did not lead to a noticeable gain in model precision; instead, it resulted in significant performance issues. Larger datasets caused excessive RAM usage, leading to system crashes during reasoning tasks. Given our time constraints, scaling up to a server infrastructure was impractical. By limiting the individual count, we achieved a balance that allows the reasoner to complete its tasks efficiently within 5 to 10 minutes on a standard modern machine, maintaining both usability and computational integrity.

6 Meaningful Inferences

During the analysis e.g. the model predictions we perform. We utilize the “percentage of total population” data property with a % value for inference. This percentage of total population is inferred by the reasoner using swrl rules. This inference is not only meaningful because we utilize swrl rules to perform calculations that can remain in the ontology natively but also due to the fact that the values used for this inference i.e. the total population of each city are queried from wikidata creating a data lakehouse in which we can create new rules and make new inferences if needed.



OntologyID(Anonymous-23) : [C:\Users\matus\Documents\VU\Knowledge and data\vrije-KnD-final-project\src\final_result\generated_files\epi_with_inferrences.ttl]

Individuals: Infected population of Almere by Covid-19 on September 18, 2022

Infected population of Almere by Covid-19 on May 29, 2022
Infected population of Almere by Covid-19 on November 13, 2022
Infected population of Almere by Covid-19 on November 14, 2021
Infected population of Almere by Covid-19 on November 27, 2022
Infected population of Almere by Covid-19 on November 28, 2021
Infected population of Almere by Covid-19 on October 02, 2022
Infected population of Almere by Covid-19 on October 16, 2022
Infected population of Almere by Covid-19 on October 17, 2021
Infected population of Almere by Covid-19 on October 30, 2022
Infected population of Almere by Covid-19 on October 31, 2021
Infected population of Almere by Covid-19 on September 04, 2022
Infected population of Almere by Covid-19 on September 18, 2022
Infected population of Alphen aan den Rijn by Covid-19 on April 03, 2022
Infected population of Alphen aan den Rijn by Covid-19 on April 17, 2022
Infected population of Alphen aan den Rijn by Covid-19 on August 07, 2022
Infected population of Alphen aan den Rijn by Covid-19 on August 21, 2022
Infected population of Alphen aan den Rijn by Covid-19 on December 11, 2022
Infected population of Alphen aan den Rijn by Covid-19 on December 12, 2021
Infected population of Alphen aan den Rijn by Covid-19 on December 25, 2022
Infected population of Alphen aan den Rijn by Covid-19 on December 26, 2021
Infected population of Alphen aan den Rijn by Covid-19 on February 05, 2023
Infected population of Alphen aan den Rijn by Covid-19 on February 06, 2022
Infected population of Alphen aan den Rijn by Covid-19 on February 19, 2023
Infected population of Alphen aan den Rijn by Covid-19 on February 20, 2022
Infected population of Alphen aan den Rijn by Covid-19 on January 08, 2023
Infected population of Alphen aan den Rijn by Covid-19 on January 09, 2022
Infected population of Alphen aan den Rijn by Covid-19 on January 22, 2023
Infected population of Alphen aan den Rijn by Covid-19 on January 23, 2022
Infected population of Alphen aan den Rijn by Covid-19 on July 10, 2022
Infected population of Alphen aan den Rijn by Covid-19 on July 24, 2022
Infected population of Alphen aan den Rijn by Covid-19 on June 12, 2022
Infected population of Alphen aan den Rijn by Covid-19 on June 26, 2022
Infected population of Alphen aan den Rijn by Covid-19 on March 05, 2023
Infected population of Alphen aan den Rijn by Covid-19 on March 06, 2022
Infected population of Alphen aan den Rijn by Covid-19 on March 19, 2023
Infected population of Alphen aan den Rijn by Covid-19 on March 20, 2022
Infected population of Alphen aan den Rijn by Covid-19 on May 01, 2022
Infected population of Alphen aan den Rijn by Covid-19 on May 15, 2022
Infected population of Alphen aan den Rijn by Covid-19 on May 29, 2022
Infected population of Alphen aan den Rijn by Covid-19 on November 13, 2022
Infected population of Alphen aan den Rijn by Covid-19 on November 14, 2021
Infected population of Alphen aan den Rijn by Covid-19 on November 27, 2022
Infected population of Alphen aan den Rijn by Covid-19 on November 28, 2021
Infected population of Alphen aan den Rijn by Covid-19 on October 02, 2022
Infected population of Alphen aan den Rijn by Covid-19 on October 16, 2022
Infected population of Alphen aan den Rijn by Covid-19 on October 17, 2021
Infected population of Alphen aan den Rijn by Covid-19 on October 30, 2022
Infected population of Alphen aan den Rijn by Covid-19 on October 31, 2021

Infected population of Almere by Covid-19 on September 18, 2022 — http://github.com/jkuwalek/vrije-KnD-final-project/ontology/IPC19Almere2022-09-18

Annotations: Infected population of Almere by Covid-19 on September 18, 2022

Annotations
rdfs:label [language: en]
Infected population of Almere by Covid-19 on September 18, 2022

Description: Infected population of Almere by Covid-19 on September 18, 2022

Types
'infected population'
Same Individual As
Different Individuals

Property assertions: Infected population of Almere by Covid-19 on September 18, 2022

Object property assertions
'infected by' C19
'population of' Almere
Data property assertions
size "28150"^^xsd:int
'percentage of total population' 13.11040216100412174277530680203991
Negative object property assertions
Negative data property assertions

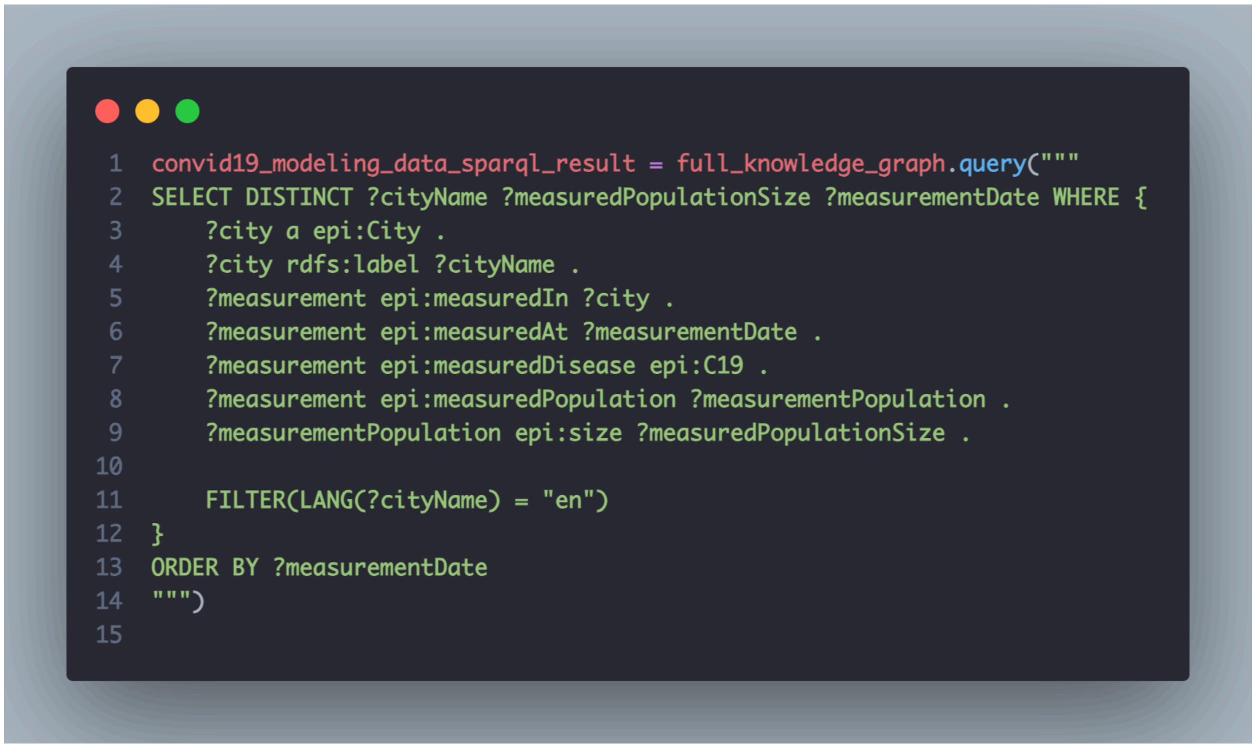
Figure 6.1

Figure 6.1 shows the pellet reasoner correctly inferring triples for an instance of InfectedPopulation, while correctly classifying it as such.

7 Relevant SPARQL queries

These SPARQL queries are designed to extract and organize COVID-19 data efficiently, each addressing distinct aspects of population measurements and geospatial data to support epidemic modeling and visualization.

Population and Measurement Query

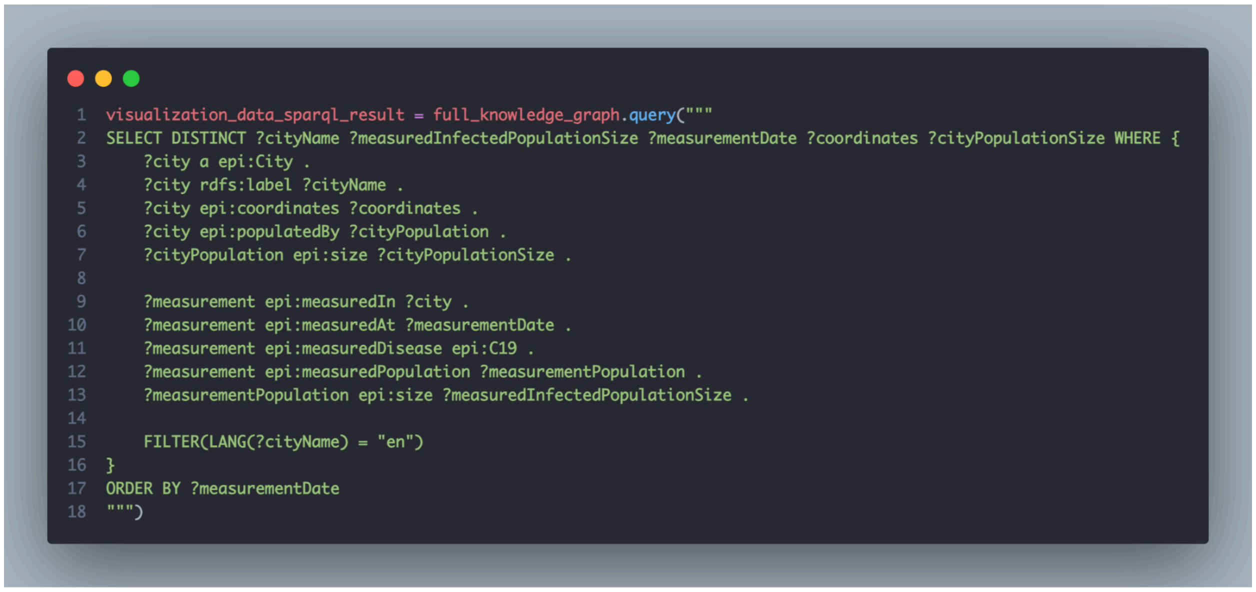


```
convid19_modeling_data_sparql_result = full_knowledge_graph.query("""
SELECT DISTINCT ?cityName ?measuredPopulationSize ?measurementDate WHERE {
    ?city a epi:City .
    ?city rdfs:label ?cityName .
    ?measurement epi:measuredIn ?city .
    ?measurement epi:measuredAt ?measurementDate .
    ?measurement epi:measuredDisease epi:C19 .
    ?measurement epi:measuredPopulation ?measurementPopulation .
    ?measurementPopulation epi:size ?measuredPopulationSize .

    FILTER(LANG(?cityName) = "en")
}
ORDER BY ?measurementDate
""")

```

Figure 7.1

The first query, which can be seen in figure 7.1, is primarily designed to retrieve population measurements associated with specific dates across various cities. This query targets data that links population statistics to time-stamped records, enabling a temporal analysis of demographic changes or stability in populated areas over time.

Visualization Data with Coordinates Query



```
visualization_data_sparql_result = full_knowledge_graph.query("""
SELECT DISTINCT ?cityName ?measuredInfectedPopulationSize ?measurementDate ?coordinates ?cityPopulationSize WHERE {
    ?city a epi:City .
    ?city rdfs:label ?cityName .
    ?city epi:coordinates ?coordinates .
    ?city epi:populatedBy ?cityPopulation .
    ?cityPopulation epi:size ?cityPopulationSize .

    ?measurement epi:measuredIn ?city .
    ?measurement epi:measuredAt ?measurementDate .
    ?measurement epi:measuredDisease epi:C19 .
    ?measurement epi:measuredPopulation ?measurementPopulation .
    ?measurementPopulation epi:size ?measuredInfectedPopulationSize .

    FILTER(LANG(?cityName) = "en")
}
ORDER BY ?measurementDate
""")
```

Figure 7.2

The second query, seen in Figure 7.2, significantly enhances visualization capabilities by using geographic coordinates alongside COVID-19 infection data and city population metrics. By including this geospatial information, the dataset enables a comprehensive mapping of infection spread across various regions, aligning infection data with specific geographic locations. This integration allows for a more nuanced analysis, where demographic data, such as city population sizes, can be visually correlated with COVID-19 case counts in a single, cohesive dataset. Consequently, this enriched dataset supports deeper insights into infection patterns, population density effects, and potential hotspots, all within a spatial context.

8 Data Science pipeline

For the data science pipeline we decided to go through two steps which involve the creation of a prediction model and a heat map for intuitive visualization.

For the prediction model we inspired ourselves from one paper (Zhou & Gao, 2023). The paper presented several mathematical models for the prediction of various epidemic measures. For our case since we are working on the covid-19 pandemic in the Netherlands.We decided to implement a relatively simple model which is called the Generalised logistic function or Richard's growth model.

The mathematical function (“Logistic function”) looks like the following:

$$f(t;\theta_1,\theta_2,\theta_3,\xi) = \frac{\theta_1}{[1+\xi\exp(-\theta_2\cdot(t-\theta_3))]^{1/\xi}}$$

Where :

T1: final epidemic size

T2: the infection rate

T3: the lag phase

Epsilon: the carrying capacity

T: time.

In the notebook section titled “Covid-19 spread prediction” the code models COVID-19 infections in Amsterdam using a generalized logistic growth function (Richards model). It first filters the data for the city and calculates days since the start. The data is split into training (80%) and testing (20%) sets. The predictions are made based on the city's % of infected population by date. Using curve_fit, it optimizes parameters like final epidemic size and infection rate to fit the model to the training data. Predictions are then made on the test set, and the model's accuracy is evaluated with Mean Absolute Error (MAE). Finally, a plot compares training data, test data i.e. the data that the model is supposed to mimic and the model's predictions.

For the data pipeline, visualizations allows users to see daily infection rate via an interactive map with sliders and animations that was implemented using the Plotly library.

To address the first competency question, which examines the effectiveness of a logistic growth function on COVID-19 datasets, specifically focusing on a major urban area like Amsterdam, we evaluated our model's performance using the mean absolute error (MAE). In general, our model demonstrated good generalization across various cities, reflecting the typical logistic pattern observed in epidemics. However, for Amsterdam, the model slightly underperformed, resulting in a mean absolute error of 1.243.

The second competency question, achieving an intuitive and visually appealing interface, is fulfilled by the notebook’s animated map-based visualization. The interactive slider and "Play" button enable users to examine infection spread across specific dates smoothly, enhancing the user experience and making data interpretation straightforward and accessible.

The third competency question was addressed by first creating a core ontology that can map to any geographical area and then adding layers on top of it with external data source both rdf and non rdf from reputable datasets.

9 Contributions and Justification

Matúš created the ontology, and with some support from Wiktor as well as insights from the rest of the group merged and populated it. He also oversaw the reasoning process.

Nikolai worked on the prediction model pipeline as well as the visualization. He also ensured that several aspects were implemented correctly such as data parsing were correct.

Wiktor worked on connecting the pipeline as well as transforming the data from the ontology so it can work within the notebook for data prediction and for the visualizations.

Jakub created a data pipeline visualization based on the work of other group members.

The whole group contributed to the creation of this report, as well as creating the project outline.

Generative AI has been used in some sections of this report to enhance the text quality by ensuring correct grammar, as well as spelling and syntax.

Note:

References are added as hyperlinks for easy direct access.