Web of Data Project

université PARIS-SACLAY

An Intelligent Web Application with Semantic Technology

In the previous weeks you have learned how to:

1. Describe data using RDF
2. Query that data using SPARQL
3. Describe a domain by encoding an ontology as RDFS and OWL
4. Integrate multiple datasets

The final project allows you to apply all of your newly acquired skills to build a fully functional Semantic Web Application.

NB: This is an individual assignment.

What do you have to do?

The project consists of two parts:

1. Design and implement a useful Semantic Web application that takes full advantage of the techniques taught in the course.
2. Document the process that led to the implementation, and present the results.

Pick a domain: think of an interesting question for which you believe (or know) there is data available. Be **creative**; it doesn't have to be about sports and population.

The work is divided into several steps:

1. Find an interesting question to answer. Answering this question requires integrating (i) a tabular dataset with (ii) an existing knowledge graph through its SPARQL endpoint. In the first hands-on session, we integrated a CSV file listing the summer Olympic medals won by each country with the Wikidata knowledge graph. The goal of that integration was to retrieve the population of each country from Wikidata in order to normalize the number of medals won by each country with respect to their population.

The only restrictions to consider when thinking of a question for your project are:
- You are not allowed to link countries
- You are not allowed to use the Olympics dataset
- You have to find a question that requires both the tabular dataset and the Wikidata knowledge graph to be answered (i.e., a question that cannot be answered when querying Wikidata by itself, or when querying the tabular dataset by itself).

We recommend the use of the Wikidata knowledge graph. Regarding the tabular file, there is no need to have more than 100 rows in the tabular dataset file. Otherwise, this might create some scalability issues during linking (you can manually remove some rows from the file you found).

2. Create an ontology to describe the domain. This ontology allows you to describe the information available in the tabular file, as well as the information that will be retrieved from the SPARQL endpoint. Although best practices encourage the re-use of existing available classes and properties, in this project, we ask you to create your own classes and properties (e.g., using the free ontology editor Protégé). We recommend the use of the following namespace for naming your classes and properties: https://ecampus.paris-saclay.fr/wod/[*your-last-name*]/. Make sure to use prefixes for simplifying the exploration of the data and writing of the queries before uploading the ontology to GraphDB.

3. Convert the tabular dataset to RDF and upload the RDF file to GraphDB. You can use the free GraphDB OntoRefine tool to convert the tabular dataset to RDF as we did in class. You are free as well to use other tools for this conversion step or even write your own script.

4. Use SPARQL to link entities to the external knowledge graph. Similarly to what we did in the course, you are asked to link the entities of the newly created RDF dataset (that you uploaded to GraphDB) to entities in the SPARQL endpoint (e.g., the Wikidata SPARQL endpoint). This linking step will allow you to retrieve the necessary information to answer the question that you outlined in the first step. Don't forget to test your linkage first using a CONSTRUCT query with a small LIMIT before transforming it to an INSERT query. Queries that we used in the last course are available in the course's GitHub repository: https://github.com/raadjoe/wod-2023-24

5. Answer the question outlined in Step 1. Write a SPARQL query to make use of the information available in the RDF dataset (that is the result of the conversion of the tabular dataset) and the information retrieved from the linkage step to answer the initial question outlined in Step 1.

6. Visualize the results of this query on an HTML page. You can use the HTML and JavaScript codes shared on GitHub to visualize the results of your question on an HTML page. You can make use of the different types of charts (pie-chart, bar-chart) that are available on angular-chart: https://jtblin.github.io/angular-chart.js/

It is not mandatory to use the HTML and JavaScript code provided on GitHub. You can implement this part using the programming language of your choice as long as it is well-documented. For example, RDFlib in Python, Jena or RDF4J in Java, dotNetRDF in C#, EasyRDF in PHP, N3.js in JavaScript, Ruby RDF in Ruby, etc.

7. Document the process that led to the implementation. We ask you to:

a. Motivate your choices of the datasets for answering this particular question that you outlined

b. Clearly document all the steps that are required for a user to replicate all the steps of the project:
   - Tabular file: its source, how to download it, and the changes you made to that file (if any)
   - The created ontology (in Turtle format)
   - RDF dataset: how the tabular file was converted and how to download the resulted Turtle file
   - The queries used for linking and answering the question
   - The code for visualizing the results of the query with its necessary dependent files and information such as the JavaScript files and the name of the GraphDB repository.

c. Record a video for showing a demonstration of your semantic web application (2 minutes max).

Be clear, structured and concise. The number of pages has no relevance at all.
The quality of your documentation (report + video) will significantly affect your grade.

Deadline for submitting the project: Friday January 12, 2024

How to submit the project: On eCampus (mini-project submission). Make sure that all the necessary documents, resources and queries are available in one compressed folder. Make sure that the documentation and the included files in the compressed file are enough for me to replicate the project (try to follow your own steps to see whether anything is missing).