INM713 Semantic Web Technologies and Knowledge Graphs

Coursework Project

Ernesto Jiménez-Ruiz

Academic course: 2020-2021
Updated: March 10, 2021

1 Introduction

DEADLINE: Wednesday, 21 April 2021, 5:00 PM

The assessment consist of a hands-on coursework, applied to a potential Data Science scenario, that combines some of the theoretical and practical aspects learnt during the lectures and laboratory sessions. **This assignment constitutes 100% of the total mark for the INM713 module**.

A **knowledge scientist** is in charge of adding context to the data to make it more useful, accessible, clean, reliable and ready to be used by downstream analytics or AI-based systems. In this coursework you will adopt the role of a knowledge scientist to transform a given data set in tabular format (*e.g.*, CSV file) into semantic data.

The coursework involves the design and development of software components (in Java or Python). These software components, among other material, will be delivered together with a short report and a video describing the design and development choices.

1.1 Extenuating Circumstances

If you are not able to submit your coursework for any medical or personal reasons beyond your control, you should contact the programmes office and fill an extenuating circumstances form. Please let me know if you are applying for EC.

1.2 Plagiarism

We encourage group discussions about the details and solutions of the lab sessions. The coursework, however, is individual. The most important outcome is that you learn and be able to independently propose interesting (not necessarily perfect) solutions.

2 Coursework Tasks

You will not only be marked with respect to the final produced output; but also according to the implemented choices and the discussion around them. For example, the created ontology and data do not need to be perfect or exhaustive (*i.e.*, covering all pieces of information), but they should be representative. The weight for each task is as follows:

- Ontology Modelling: **20%**
- Tabular Data to Knowledge Graph: **30%**
- SPARQL and Reasoning: **15%**
- Ontology Alignment: **20%**
- Knowledge Graph Embeddings: **10%**
- Clarity of report and video: **5%**

The percentages in each subsection refer to the internal weights within each task. For example, **Subtask OWL.1** has an internal task weight of 30%, and an overall weight of 6% (*i.e.*, $0.2 \times 0.3 = 0.06 \rightarrow 6\%$).

2.1 Dataset

The dataset for the coursework is based on the kaggle dataset about *Pizza Restaurants and the Pizza They Sell*.[1] The dataset version to be used is available in moodle. We will refer to this dataset as `Pizza_data`.

2.2 Ontology Modelling (Task OWL)

Create a small ontology that models the domain of `Pizza_data`. Use Protégé to develop the ontology. Save the ontology into `turtle` format (*e.g.*, `.ttl`) and `rdf/xml` format (*e.g.*, `.owl`). Guidelines:

Subtask OWL.1: Discussion and motivation of the different modelling choices in the report (**30%**).

Subtask OWL.2: Change namespace for the ontology *e.g.*, `http://www.city.ac.uk/ds/inm713/your_name/` (**5%**).

Subtask OWL.3: Create a prefix (*e.g.*, your initials `ejr:`) for the defined namespace (**5%**).

Subtask OWL.4: The ontology should abstract knowledge about the domain, valid for the data at hand but potentially valid for other datasets (*e.g.*, tables). The ontology should, in principle, only contain concepts, object and data properties. ***Tip:*** *you should difference between the general concept* `Margherita_Pizza` *and the margherita pizza served at "Pizzeria da Mario"* (**10%**).

Subtask OWL.5: Organize classes into a hierarchy (**15%**).

Subtask OWL.6: Create local scope property restrictions or global scope ones (*i.e.*, domain and range) as necessary (**15%**).

Subtask OWL.7: Create appropriate property characteristics (**10%**).

Subtask OWL.8: Create a `label` (*e.g.*, name or synonym) and a `comment` (*e.g.*, meaning of the concept) for each created entity (**5%**).

Subtask OWL.9: Create or reuse an annotation property `creator` and annotate/indicate the ontology has been created by you (**5%**).

[1] Kaggle dataset about pizza restaurants: `https://www.kaggle.com/datafiniti/pizza-restaurants-and-the-pizza-they-sell`. This kaggle dataset is a subset of a dataset provided by Datafiniti's Business Database: `https://datafiniti.co/products/business-data/`

2.3 Tabular Data to Knowledge Graph (Task RDF)

Transform `Pizza_data` into RDF triples using your favourite programming language. Please document your code. Save the RDF data into `turtle format (.ttl)`. Guidelines:

Subtask RDF.1 Discuss in the report the different transformation choices and how entity resolution was treated (**30%**). As we saw in the module, there is not a unique way to transform the elements in a table into elements of a knowledge graph (*i.e.*, classes, object properties, data properties or instances). For example the column "menu item" contains the name of the pizza but it may also include additional information about toppings and type of pizza. ***Tip1:*** *You will need to apply some basic text processing and entity recognition based on the ontology vocabulary.* ***Tip2:*** *After processing the data, one may also need to extend the ontology with new elements.*

Subtask RDF.2 Create RDF triples (**30%**). Use the created ontology to guide the transformation *e.g.*, to link the data to the ontology (*e.g.*, via `rdf:type`) and to use the defined ontology properties (*e.g.*, `has_topping`) and concepts (`Margherita_Pizza`).

Subtask RDF.3 For the cells in the columns *city*, *country* and *state*; instead of creating new URIs (*e.g.*, new individuals) for the information in the table cells, reuse an entity URI from DBPedia, Wikidata or Google's Knowledge Graph (`http://dbpedia.org/resource/Los_Angeles`). ***Tip:*** *communicate with their respective look-up services as we saw in the lab session* (**25%**).

Subtask RDF.4 Correctness of the code and code documentation (**15%**).

2.4 SPARQL and Reasoning (Task SPARQL)

Write SPARQL queries, according to the requirements in the following subtasks, and execute them over the created ontology and the generated data.

Subtask SPARQL.1 Perform reasoning with the created ontology and the generated data.[2] Save the extended graph in `turtle format (.ttl)` (**10%**).

Subtask SPARQL.2 Return all the details of the restaurants that sell pizzas without tomate (*i.e.*, pizza bianca). Return the results as a `CSV file` (**20%**).

Subtask SPARQL.3 Return the average prize of a Margherita pizza (**20%**).

Subtask SPARQL.4 Return number of restaurants by city, sorted by state and number of restaurants (**20%**).

Subtask SPARQL.5 Return the list of restaurants with missing postcode (**20%**).

Subtask SPARQL.6 Correctness of the queries and code, and documentation of the created SPARQL queries in the report (**10%**).

[2] **Tip:** When reasoning with data, using OWL 2 reasoning is expensive. Using an approximate reasoner is typically more suitable (*e.g.*, for OWL 2 RL or other subsets).

2.5 Ontology Alignment (Task OA)

Perform a basic alignment between the `pizza.owl` ontology and the created ontology. This alignment is important to perform SPARQL queries using the vocabulary of the `pizza.owl` ontology instead of the created ontology.

Subtask OA.1 Compute equivalences between the entities of the input ontologies. Save the equivalences as triples in `turtle format (.ttl)`. ***Tip:*** *use* `owl:equivalentClass` *and* `owl:equivalentProperty` *as predicates of the equivalence triples* (**50%**).

Subtask OA.2 Perform reasoning with *(i)* the created ontology, *(ii)* the pizza.owl ontology and *(iii)* the computed alignment (without the data) and list the number of unsatisfiable classes (*i.e.*, classes that have the empty interpretation). (**30%**)

Subtask OA.2.a If there are unsatisfiabilities, use Protégé to get an explanation and discuss the causes. You should load both ontologies and the alignment, and activate the reasoner in Protégé.

Subtask OA.2.b If there are not unsatisfiable classes, perform reasoning including the generated RDF data (*i.e.*, with *(i)* the created ontology, *(ii)* the generated data, *(iii)* the pizza.owl ontology and *(iv)* the computed alignment). Create a query to return the pizzas with type `pizza:MeatyPizza`. If no pizza is returned, discuss why this is the case (*e.g.*, missing alignment).

Subtask OA.3 Correctness of the code and discussion of the used alignment techniques (**20%**).

2.6 Ontology Embeddings (Task Vector)

This task consists in creating embeddings capturing the rich semantics of the created ontology and generated data. We will use the tool OWL2Vec*. These embeddings can be used in a subsequent Machine Learning model that requires as input the encoding of the ontology entities. In this coursework we are computing clusters of the ontology entities.

Subtask Vector.1 Run OWL2Vec* with the created ontology and generated data. Test three different configurations. Save the generated vectors in both binary and textual format (**20%**).

Subtask Vector.2 Select 5 pairs of entities (for any of the tested configurations) and discuss the similarity of their vectors (*e.g.*, compare the vectors of the concept `pizza:Margherita` and the word "pizza"). (**20%**).

Subtask Vector.3 Compute clusters (*e.g.*, using K-means) for the embeddings of the ontology concepts (*i.e.*, the URI embeddings). Test the algorithm with different number of clusters, visualise the clusters and discuss the results (**50%**).

Subtask Vector.4 Correctness and documentation of the codes (**10%**).

3 Submission Guidelines

You need to submit all the material in Moodle, in the dedicated "Submission Area". Please provide short but **meaningful names** to the files.

1. **Report** where you discuss the choices and results in the different coursework tasks. In **PDF format**, in principle no more than 10 pages (reasonable font size) with an appendix if additional space is necessary. The structure of the document is open, but the structure of the tasks and subtasks could be a good reference.

2. **Video** where you give an overview about the choices and results in the different coursework tasks. You can use slides as support. The video should be between 5 (preferred) and 10 minutes. In **mp4 format** or any other format that can be reproduced by a web browser.[3]

3. All the **produced code in a single PDF file**. This is a requirement for auditing purposes and to run TurnItIn on your code.

4. **Zip file** with the following folders and files:

    - **Code folder:** all your code and a 'readme' file with instructions about how to execute the different scripts.
    - **Ontology** in turtle or rdf/xml format.
    - **RDF data** in turtle format.
    - **Extended RDF data** after reasoning in turtle format.
    - **CSV file** created in SPARQL.2 Subtask.
    - **Ontology alignment** in turtle format.
    - **Ontology embeddings** in both binary and textual format.

3.1 Coding

The coursework can be implemented in both Python and Java. You can reuse material from the lab sessions as support and any other external resources/libraries as long as they are properly cited in the report.

We encourage the use of online code repositories like GitHub or GitLab. Google Colab can also be a potential alternative. In this case, *(i)* add a link to the repository in the report and *(ii)* make the repository public only after the submission deadline (alternatively you could also add me as a member of the private repository).

[3] To record the video, you can just start a Zoom or Teams call with just yourself, share the screen, and record the meeting. I can help on this if necessary.