CITY
UNIVERSITY OF LONDON
EST 1894

IN3067/INM713 Semantic Web Technologies and Knowledge Graphs

Coursework Project (Part 2)

Ernesto Jiménez-Ruiz

Academic course: 2023-2024
Updated: February 5, 2024

1 Introduction

DEADLINE: Sunday, 12 May 2024, 5:00 PM

The second part of the module's assignment consists of a hands-on coursework, that combines some of the theoretical and practical aspects learnt during the lectures and laboratory sessions. **This assignment constitutes 80% of the total mark for the IN3067/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 Part 2 of the 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 (*i.e.*, a Knowledge Graph). A target ontology for the transformation is given.

The coursework involves the design and development of **software components** (in Java and/or Python).[1]

1.1 Working in pairs or individually

You can work individually or team-up with one of your peers. Given the experience from previous years, working in pairs will enhance the learning process as the module brings together students with different skills and backgrounds.

You should let me know by **March 17** if you are planning to work individually or in pairs (together with the name of the members) for Part 2 of the coursework. It is desired that pairs from Part 1 are preserved. **Use the form in moodle**. Groups formed after the deadline may not be accepted.

1.2 Extenuating Circumstances

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

1.3 Plagiarism

We encourage group discussions beyond the formed pairs about the details and solutions of the lab sessions. The coursework, however, should be conducted by the formed pairs or individually. When submitting material in moodle a plagiarism check is performed checking submissions from previous, the current year and external resources.

Note that working in pairs means that you work "together" on all tasks. You can take the lead in some tasks, while your partner leads others; but both of you must have a good understanding of all tasks. The most important outcome is that you learn and be able to independently propose interesting (not necessarily perfect) solutions.

[1] Python notebooks are also allowed.

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. The weight for each task is as follows:

- Coursework Part 1 (previous assignment about ontology modelling): **20%**
- **Coursework Part 2** (described in this document):
  - Tabular Data to Knowledge Graph: **40%**
  - SPARQL and Reasoning: **20%**
  - Ontology Alignment: **10%**
  - Ontology Embeddings: **10%**

The percentages in each subsection refer to the internal weights within each task. For example, **Subtask RDF.3** has an internal task weight of 50%, and an overall weight of 20% (*i.e.*, $0.4 \times 0.5 = 0.2 \rightarrow 20\%$).

2.1 Dataset and Ontology

Dataset. As in Part 1, the dataset for the coursework is based on the kaggle dataset about *Pizza Restaurants and the Pizza They Sell.*[2] The dataset version to be used in Part 2 of the coursework **contains 500 data rows** and is available in Moodle. We will refer to this dataset as `cw_data`.

Ontology. I have created the ontology `pizza-restaurants-ontology.ttl` as a model solution for Part 1 (***to be released on March 4***). In Part 2 we are using this ontology as input vocabulary to create the RDF triples. In this way everyone is using the **same ontology**. We will refer to this ontology as `cw_onto`.

Example instances. I have also created a few instances as example in `cw_onto`. For ingredients, for simplicity, I do not differentiate among the "same" ingredient used in different restaurants, I just added a generic instance (*e.g.*, `cw:mozzarella` that belongs to the class `cw:Mozzarella`). One could potentially make the difference. The class in any case is the one that represents the general concept. For pizzas I do prefer to differentiate between "similar" pizzas served in different restaurants. A *margherita pizza* served in *restaurant X* is different from a *margherita pizza* in *restaurant Y*.

2.2 Tabular Data to Knowledge Graph (Task RDF)

This task aims at (**automatically**) transforming `cw_data` into RDF triples using your favourite programming language. Use `cw_onto` to guide the transformation and as

[2]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.

target vocabulary. To build the RDF triples, create new instances and use the entities in `cw_onto` (*i.e.*, classes, object properties, data properties and instances). For example:[3]

- `cw:little_Pizza_Paradise_Bend rdf:type cw:Pizzeria`
- `cw:little_Pizza_Paradise_Bend cw:serves cw:bianca_pizza_lpp`
- `cw:bianca_pizza_lpp rdf:type cw:PizzaBianca`

Save the RDF data into `turtle format (.ttl).`

Tip 1: *As in lab session 5 you will need to define some "manual" assumptions about column types and relationships among columns (i.e., mapping/link to the ontology vocabulary), then the transformation to RDF triples will be automatic.*

Tip 2: *You will need to apply some basic text processing and entity recognition based on the provided ontology vocabulary.* For example the column "menu item" contains the name of the pizza but it may also include additional information about the type of pizza. Similarly, the column "item description" include mentions of ingredients.

Subtask RDF.0 Before you start the transformation, extend `cw_data` with a few new entries in the CSV file (*e.g.*, 2 new restaurants selling 2 new pizzas each, with some defined ingredients). This task is mandatory, but it does not give points. Please mention in the report the created entries.

Subtask RDF.1 URI generation (**20%**). Use a good naming convention for the URIs of the created individuals.
Tip: *Different restaurants in different cities may share the same name. Similarly for Pizzas, a pizza bianca is served in both "Ciao Bella" and "Little Pizza Paradise" restaurants.*

Subtask RDF.2 RDF generation (**60%**). This task will be evaluated by the execution of a number of (blind) queries, which are expected to return results (not necessarily complete), over the generated data.[4] For example, *(i)* list American restaurants or *(ii)* list vegan pizzas.

Subtask RDF.3 Reuse URIs from state-of-the art knowledge graphs (**10%**). 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 an available KG (*e.g.*, `dbr:Chicago` instead of `cw:chicago`).

- **Subtask RDF.3.1** Use Google KG (**5%**).
- **Subtask RDF.3.2** Use Wikidata KG (**5%**).

In addition to the file with the generated data from task RDF.2, generate files for tasks RDF.3.1 and RDF.3.2.
Tip: *Communicate with the KG look-up services as we saw in the lab sessions.*

[3]`cw:` is the prefix for the namespace of `cw_onto` (`cw:Pizzeria` and `cw:PizzaBianca` are concepts in `cw_onto`). You can use a different namespace for the instances coming from `cw_data` (*e.g.*, cw:bianca_pizza_lpp).

[4]The queries are not known to the students.

Subtask RDF.4 Perform reasoning with `cw_onto` and the generated RDF data. Save the extended graph in `turtle format (.ttl)` (**10%**).
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).*[5]

Subtask RDF.5 (Optional) Exploit an external Knowledge Graph to perform disambiguation (*e.g.*, same city name in multiple states) and solve errors in the data (*e.g.*, wrong state names). (**extra 15%**)

2.3 SPARQL and Reasoning (Task SPARQL)

Write meaningful SPARQL queries according to the requirements in the following subtasks, and execute them over the extended graph after reasoning with both *(i)* `cw_onto` and *(ii)* the generated data in Task RDF. Return the results of each query as a *CSV file*. The created queries should be valid for your data, that is, they should return results. Please add the queries in the report and create a text file for each of the queries.

Subtask SPARQL.1 Create a query with at least two triple pattern and a FILTER. (**20%**).

Subtask SPARQL.2 Create a query that uses at least three triple pattern, a FILTER and a function. (**20%**).

Subtask SPARQL.3 Create a query that uses the Union graph pattern and SPARQL 1.1 negation. (**20%**).

Subtask SPARQL.4 Create a query that groups results, uses aggregates, and a filter over the aggregates. (**20%**).

Subtask SPARQL.5 Create a query (different from SPARQL.4) that groups results, uses aggregates, filters the results (over the aggregates) and orders the results according to two variables. (**20%**).

2.4 Ontology Alignment (Task OA)

Perform a basic alignment between the provided `pizza.owl` ontology in moodle and `cw_onto`. This alignment is important to perform SPARQL queries using the vocabulary of the `pizza.owl` ontology instead of the model ontology `cw_onto`.

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

Subtask OA.2 Compute the precision and recall of your mappings against the given reference mappings: `reference-mappings-pizza.ttl` (**10%**).

[5] *The reasoning results using the OWL-RL python library cannot be loaded in `Protégé`.*

Subtask OA.3 Perform reasoning with all the following sources in a single graph or RDF model: *(i)* `cw_onto`, *(ii)* the pizza.owl ontology, *(iii)* the computed alignment, and *(iv)* the generated data from `cw_data` (**10%**).
Tip: *Parse/load all files into a single graph or model.*

Subtask OA.4 Create a valid SPARQL query with at least two triple patterns that uses the vocabulary of `pizza.owl` an retrieves results from the generated data (*e.g.*, about restaurants and pizzas). Return the results of the query as a *CSV file*. (**30%**)

2.5 Ontology Embeddings (Task Vector)

This task consists in creating embeddings capturing the rich semantics of `cw_onto` and the generated data. We will use the tool OWL2Vec* (*to be released in lab 8*).[6] These embeddings can be used in a subsequent Machine Learning model that requires as input the encoding of the ontology entities. For example to perform advanced ontology alignment.

Subtask Vector.1 Run OWL2Vec* with `cw_onto` and the generated data Task RDF (prior reasoning). Test two different configurations. Save the generated vectors in both binary and textual format (**40%**).
Tip: *It is easier if given to OWL2Vec* a single file. To do so just load both files (`cw_onto` and the generated data) into a single RDFlib Graph or Jena RDF model and save it into OWL format. Owlready seems to cope with Turtle format, but use OWL format in case it gives problems.*

Subtask Vector.2 Select 6 pairs of entities and discuss the similarity of their vectors (*e.g.*, compare the vectors of two restaurants for example). Try to select 3 pairs for which you expect a close similarity and 3 pairs for which you expect dissimilarity. Discuss the results within the report.

- **Subtask Vector.2.1**: Use the embeddings from Configuration 1 in Task Vector.1. (**30%**).
- **Subtask Vector.2.2**: Use the embeddings from Configuration 2 in Task Vector.1. (**30%**).

Subtask Vector.3 (Optional) Solve Subtask OA.1 using similarity among ontology embeddings calculated with OWL2Vec* (**extra 50%**).

3 Submission Guidelines

You need to submit all the material in Moodle, in the dedicated "Submission Area" (*i.e.*, *Coursework Assignment (Part 2) - 80%*). Provide short but **meaningful names** to the files. Please follow the instructions carefully, otherwise the submission may not be marked. Material to be submitted:

[6]OWL2Vec*: Embedding of OWL Ontologies. Mach Learn (2021): `https://rdcu.be/dxJbO`

1. Video in **mp4 format** or any other format that can be reproduced by a web browser. You can use slides as support. Please focus on interesting implementation choices and results in the different coursework tasks. The video should not be longer than 10 minutes (strict limit).[7]

2. Report in **PDF format**, no more than 5 pages (reasonable font size). The organisation of the document is open, but the structure of the tasks and subtasks could be a good reference. The report aims at complementing the video with additional information like the created SPARQL queries, the configurations and discussion in Task Vector, the precision/recall in Task OA or an interesting algorithm.

3. All the **produced code in a single PDF file**.[8] This is a requirement for auditing purposes and to run TurnItIn on your code. Please make sure the text in the generated PDF is "selectable" (not an image).

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.
    - **Task RDF**: Generated RDF data (turtle format), extended RDF data after reasoning (turtle format), and generated data reusing Google KG or Wikidata (turtle format).
    - **Task SPARQL**: SPARQL queries (one text file per query) and query results (one csv file per query).
    - **Task OA**: alignment (turtle format), SPARQL query (text file), extended RDF data after reasoning with the alignment (turtle format).
    - **Task Vector**: ontology embeddings files (both binary and textual format).

3.1 Individual work

Please submit the required files individually. You will need to submit 4 files with the content as described above:

(i) ErnestoVideo.mp4,

(ii) ErnestoReport.pdf,

(iii) ErnestoAllCode.pdf, and

(iv) ErnestoCompressedFile.zip.

[7]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.

[8]Copy and paste all your code into a text file and the generate a PDF file.

3.2 Work in pairs

You need to create two independent videos where each member describes the coursework. Both members need to submit material in moodle as follows:

- **Member 1:** Please submit the required files individually. You will need to submit 4 files with the content as described above:
  - *(i)* Member1NameVideo.mp4,
  - *(ii)* GroupNameReport.pdf,
  - *(iii)* GroupNameAllCode.pdf, and
  - *(iv)* GroupNameCompressedFile.zip.
- **Member 2:** Please **ONLY** submit your video file; *e.g.*, *Member2Name_Video.mp4*.

Each group decides internally who acts as Member 1 or Member 2.

3.3 Large files

If the video or the embeddings files exceed the 200Mb limit imposed by Moodle, please upload the file to One or Google drive and submit to Moodle a text file with the link to the large file (*i.e.*, *ErnestoFileWithLinkToALargeFile.txt*. These large uploads must also be done by the deadline.

3.4 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 and within the code**. It is important that the **code** is **well documented** and **easy to run**.

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).