IN3067/INM713 Semantic Web Technologies and Knowledge Graphs

Laboratory 8: Ontology Alignment

Ernesto Jiménez-Ruiz

Academic course: 2023-2024
Updated: March 27, 2024

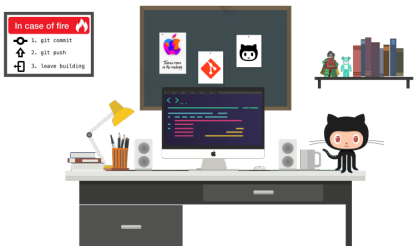
Contents

1 Git Repositories

Support codes for the laboratory sessions are available in *GitHub*. There are two repositories, one in Python and another in Java:

`https://github.com/city-knowledge-graphs`

For Java developers we use maven to deal with dependencies (see pom file). For Python developers there is always a `requirements.txt` within each lab folder with possibly new dependencies. It is recommended to use environments, but it is not strictly necessary.



2 Support Codes and Datasets

The GitHub repositories include support codes for this lab session including the ontologies (owl format) and reference alignments (ttl format). The reference alignment files can also serve as an example about how your computed mapping should look like.

In **Python** we will rely on the Owlready API (`https://owlready2.readthedocs.io/`)to load and process the ontologies. In **Java** we will continue with the Jena API (`https://jena.apache.org/documentation/ontology/`). The scripts `AccessEntityLabels.py` and `AccessEntityLabels.java` load an ontology, iterate over the classes and access the lexical information of the classes (*i.e.*, name/labels available as part of the URI or via the `rdfs:label` annotation).

In previous laboratory sessions, among other things we learnt how to compute lexical similarities among strings, and how to create RDF triples.

The reference scripts in Python: `CompareWithReference.py` and Java: `CompareWithReference.java` compute Precision and Recall given as input the system computed mappings and the reference mappings for the corresponding matching task.[1] For Python developers a Jupyter notebook is also provided.

Note that not all matching tasks have a reference alignment, specially those coming from real world applications. As described below, reference alignments are required for evaluation campaigns like the OAEI.

[1] `https://en.wikipedia.org/wiki/Precision_and_recall`

3 Ontology Alignment Evaluation

The Ontology Alignment Evaluation Initiative[2] (OAEI) is an international effort for the systematic evaluation of ontology alignment systems that has been running since 2004. The OAEI provides ontologies and reference alignment to drive a controlled evaluation among ontology alignment systems. In this lab session we are going to use two datasets from the OAEI.

Conference. This dataset aligns 7 ontologies from the same domain (conference organization): Cmt, ConfTool, Edas, Ekaw, Iasted, Sigkdd, Sofsem.

Anatomy. This evaluation dataset consists of finding an alignment between the Adult Mouse Anatomy (mouse) and a part of the NCI Thesaurus describing the human anatomy (human).

4 Tasks

Task 7.0 Basic ontology alignment can be reduced to comparing two list of elements.

1. Try to create a function that given two lists of elements, returns the list of elements in common (*i.e.*, equivalent). For example, given:
   `listA = ["pizza", "tomato sauce", "pepperoni", "restaurant"]`
   and
   `listB = ["pizza", "tomato", "peperone", "restaurant"]`
   return the `listC = ["pizza", "restaurant"]`.

2. Same as above but returning a list of "similar" elements. Recall the lexical similarity tools we used a few weeks back.

On the ontology setting you need to compare list of entities, and for each entity you know their URI and their label(s).

Task 7.1 Design and implement a (simple) lexical matcher that loads two ontologies and finds equivalence correspondences among their entities. Save the alignments as RDF triples in turtle format. For example:

```
cmt:Conference owl:equivalentClass confOf:Conference .
cmt:title owl:equivalentProperty confOf:hasTitle .
```

Tip 1: *align classes against classes and properties against properties. If there are instances, one should also target instances against instances using* `owl:sameAs` *(e.g.,* `city:ernesto owl:sameAs wd:ernesto`*.*
Tip 2: *Use lexical similarity and a threshold to decide which mappings are good enough to be added as output.*

Task 7.2 Apply your algorithm over the following pairs of ontologies from the OAEI's conference track:

[2]OAEI: `http://oaei.ontologymatching.org/`

- cmt.owl - ekaw.owl
- cmt.owl - confOf.owl
- confOf.owl - ekaw.owl

Give suitable names to the mapping files (*e.g.*, `ernesto-cmt-ekaw.ttl`). Use the provided script and reference alignments to compute the precision and recall of the mappings computed by your system.

Task 7.3 Submit your mappings via e-mail so that I can rank your contributions.

Task 7.4 Apply your algorithm over the OAEI's anatomy track and get the precision and recall of your mappings. Note that, unlike in the conference track, the ontologies in the anatomy track are relatively large and contain $2,744$ classes (mouse.owl) and $3,304$ classes (human.owl). If your algorithm computes a pair-wise comparison, this will imply $> 9,000,000$ lexical comparisons.[3]

Task 7.5 (Optional) Create an algorithm that *(i)* is scalable for large ontologies (*e.g.*, uses a lexical index) and *(ii)* involves not only the labels of the entities but also the structure of the ontology. This task gets closer to a potential **MSc project**.

[3]LogMap (`https://github.com/ernestojimenezruiz/logmap-matcher`) is a state-of-the-art system that can match the ontologies in the anatomy track in 7s while producing competitive results. Its simple lexical variant LogMapLt takes only 2s. Results: `http://oaei.ontologymatching.org/2021/results/anatomy/`