INF 558/CSCI 563: Building Knowledge Graphs

Homework 3: Entity Resolution & Knowledge Representation

Released: Jan 31th, 2020
Due: Feb 7th, 2020 @ 23:59

Ground Rules

This homework must be done individually. You can ask others for help with the tools, however, the submitted homework has to be your own work.

Summary

In this homework, you will link movies from the Internet Movie Database (IMDb) to the American Film Institute (AFI) and then represent the data using RDF.
The entity resolution task will be done using The Record Linkage ToolKit (RLTK), an open-source record linkage platform. You will use RDFLib, a Python library for working with RDF, for the task of knowledge representation. We provide a python notebook (`ER_KR.ipynb`), which contains instructions, code and descriptions on how to use the tools we mentioned.

Task 1: ER (5 points)

In this task, you are given a dataset of movies from IMDb (`imdb.jl`) and a dataset of movies in AFI (`afi.jl`). Your goal is to match records from these 2 datasets using entity linking methods. This means you need to figure out which pairs of movies in the two datasets are referring to the same movie.

IMDb and AFI datasets contain several attributes, some are unique for each dataset, some are present in both. For the task of linking, we are interested in looking at the 3 fields that are present in both datasets (**movie title/name**, **release date/year** and **genre**).

Task 1.1 (3 points)

For each attribute (total of 3): (1) Analyze the given data and choose string similarities that you think are appropriate for each. (2) Explain your choices in the report. (3) Implement a program (python method) that computes the field similarities between the records for the 2 datasets.

Notes:

- You can customize string similarity methods or change/clean attribute values if necessary. For example, you can choose the Levenshtein similarity method for the movie names, which you derive from the original attribute values.
- See the attached notebook for an example.

Task 1.2 (2 points)

Design a scoring function to combine your field similarities. Explain your choices of weights in the scoring function in the report. Implement a program that predicts the corresponding AFI movies for the IMDb movies using your scoring function.

Apply your program on the test set (`imdb.txt`). Set the value to `NULL` if there is no corresponding entry in the AFI dataset. Export your prediction to an output file (`Firstname_Lastname_hw03_imdb_afi_el.json`) with the following format (JSON):

```
[ { "imdb_movie": <imdb_url>, "afi_movie": <afi_url>}, ... ]
```

For example:

```
[
  {
    "imdb_movie": "https://www.imdb.com/title/tt0033467/",
    "afi_movie":  "https://catalog.afi.com/#dc440a1a7fa4a6bd30f183eded493ef2"
  },
  {
    "imdb_movie": "https://www.imdb.com/title/tt0108052/",
    "afi_movie":  "https://catalog.afi.com/#642a1d0b14872b56d8fde9228170da6f"
  }
]
```

Task 2: KR (5 points)

In this task, you will represent the movie data (after linking) using RDF. The ontology (vocabulary/schema) you will use is **schema.org**. As this ontology may not include all necessary classes and properties to model your data, you will need to extend the ontology with classes that you define on your own.

Task 2.1 (1 point)

Describe the model (in the report) you will use to generate the RDF data to describe the merged movie entry with all of the available attributes from the two sources you have matched.
There's a total of 13 attributes: title, release-date, certificate, runtime, genre, imdb-rating, imdb-metascore, imdb-votes, gross-income, producer, writer, cinematographer and production-company.
Use the appropriate classes and properties from **schema.org**. Define your own if you could not locate a suitable one in **schema.org**. Finalize the file describing your model (`model.ttl`) with the missing attributes and rename it to `Firstname_Lastname_hw03_model.ttl`.

Notes:

- As a starting point, you may want to use the class `` `https://schema.org/Movie` `` to represent a movie and the property `` `https://schema.org/datePublished` `` to represent a predicate that describes that movie's release time (as seen in `model.ttl`).
- The attribute 'production company' should not be referred to as a plain literal from the movie entry. Instead, create a local URL for each production company (as depicted in `model.ttl`, the movie's production company is an instance of a class, not a literal).

Task 2.2 (3 points)

Implement a program that uses the data from the 2 datasets and your results file from previous task (`Firstname_Lastname_hw03_imdb_afi_el.json`). The program should convert the combined movie data to RDF triples (in turtle format, `ttl`) using the model you defined in task 2.1, the generated file should be named `Firstname_Lastname_hw03_movie_triples.ttl`.

Notes:

- Use the IMDb URI as the identifier of the node (subject). You can discard the AFI URI
- See the attached notebook for an example of how create and generate RDF graph (triples) in `ttl` format.

Task 2.3 (1 point)

Choose two movie instances and a single production company instance (locate them in your `ttl` file, the two movies should refer to the same production company) and visualize the triple data in your report. Use the online tool at http://www.ldf.fi/service/rdf-grapher to visualize the triples in a graph. The result should look like what is shown in Figure 1 (the figure shows partial data, yours should be complete).

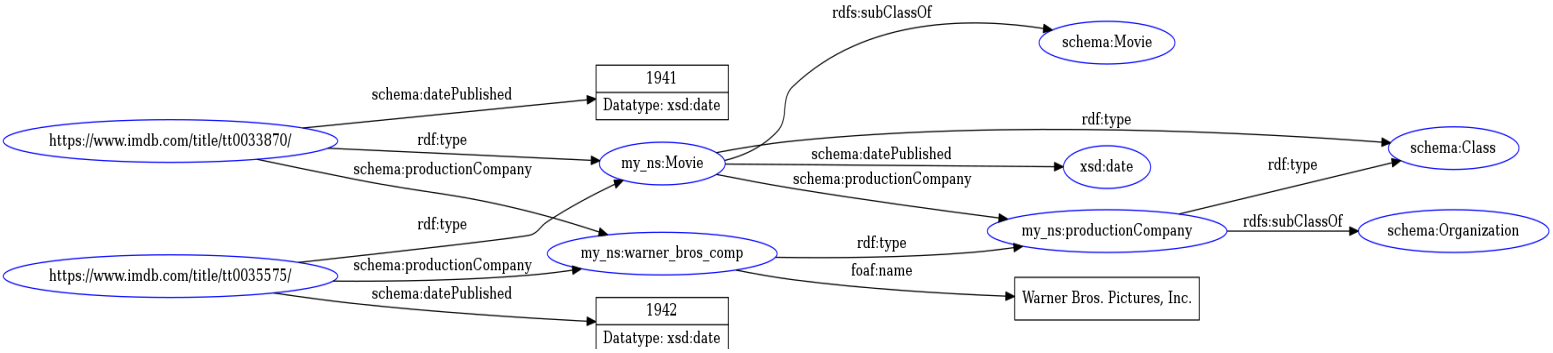

Figure 1: An example of a graph visualization

Submission Instructions

You must submit (via Blackboard) the following files/folders in a single `.zip` archive named `Firstname_Lastname_hw03.zip`:

- `Firstname_Lastname_hw03_report.pdf`: pdf file with your answers to Tasks 1 and 2
- `Firstname_Lastname_hw03_imdb_afi_el.json`: as described in Task 1.2
- `Firstname_Lastname_hw03_model.ttl`: as described in Task 2.1
- `Firstname_Lastname_hw03_movie_triples.ttl`: as described in Task 2.2
- `source`: This folder includes all the code you wrote to accomplish Tasks 1 and 2