PhD course on Knowledge Graphs in the era of Large Language Models

Setting-up the Infrastructure

Ernesto Jiménez-Ruiz

April 2024
Updated: April 19, 2024

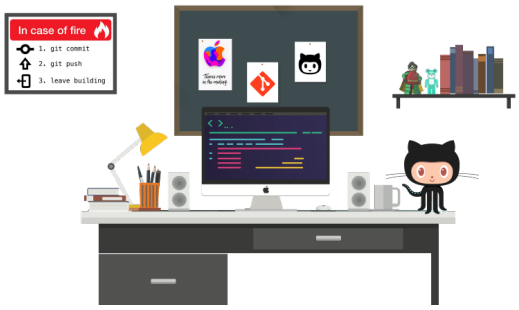
Contents

In this course we will use Python as programming language for the practical sessions. The goal of this documents is to start setting up the coding environment and the ontology editor Protégé.

1 Git Repository

I have created the following GitHub repository to include the material and the support codes for the course:

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



1.1 Accessing the example codes

In order to download the codes in the GitHub repository, you can (1) use Git and clone/-fork the repository (see instructions in Section 1.2), or (2) directly download the full repository as a ZIP file. Option (1) will allow you to be up-to-date with my solutions and additional material without downloading the whole repository; option (2) is simpler, but you will need to download the whole repository every time there is a change.

1.2 Getting around with Git

If you decide to go with Option (2), you can ignore this section.

Create a GitHub account. If you don not have already a GitHub account, you can set up a free account by visiting `https://github.com/`. GitHub is a public commercial widely service owned by Microsoft. Gitlab is an alternative solution (`https://about.gitlab.com/`). You can create both public (like the ones in `https://github.com/city-knowledge-graphs`) and private repositories.

Git tutorials. The underlying technology for distributed version control/collaboration is called *git*. You can find several tutorial on the Web, for example:

- `https://www.w3schools.com/git/`
- `https://www.youtube.com/watch?v=SWYqp7iY_Tc`
- `https://www.freecodecamp.org/news/learn-the-basics-of-git-in-under-10-minutes-da548267cc91/`

Desktop application. Most tutorials refer to the command line for git. Using the command-line or terminal window is typically natural for macOS and Linux users, but may not be that common for Windows user. GitHub Desktop (`https://desktop.github.com/`) may be an easy solution to manage your git repositories. It is only available for macOS and Windows users. It comes with documentation (`https://docs.github.com/en/desktop`)

Command-line. Check if git is installed in your computer (*e.g.*, type in the command line `git --help`). You can download/install git from `https://git-scm.com/downloads`).

`clone` is the command to clone and download a git repository. First select the folder where you would like to download the codes and then run the command:

- Java: `git clone https://github.com/city-knowledge-graphs/java-2024.git`
- Python: `git clone https://github.com/city-knowledge-graphs/python-2024.git`

To update the local copy with new remote changes (*e.g.*, codes of future laboratory sessions) use the command pull: `git pull` from the respective local repository folders. To avoid clashes with the changes on the main branch it is suggested to add generated codes and data in the folder `student-codes-data`.

2 Python libraries

We will use Python 3.10 (or newer). Check if you have python installed in your computer with (`python --version` or `python3 --version`. Otherwise install from `https://www.python.org/downloads/`

I use Eclipse as Python editor, but Spyder IDE (`https://www.spyder-ide.org/`) and Visual Studio (`https://visualstudio.microsoft.com/`) is typically the choice for many Python developers.

In order to deal with the required libraries and dependencies for each lab sessions it is recommended to use a virtual environment. I typically use `pip` (`https://pip.pypa.io/en/stable/installation/`) and `venv` (`https://docs.python.org/3/library/venv.html`), but there are alternatives like `conda` (`https://docs.conda.io/`).

To set up a new environment:

```
python3 -m venv /home/ernesto/path_to_lab1_venv
```

To activate the environment, depends on the platform (`https://python.land/virtual-environments/virtualenv`). For example:

```
source /home/ernesto/path_to_lab1_venv/bin/activate
```

To deactivate a virtual environment: `deactivate`

The required libraries are in the GitHub repository: requirements.txt file (in GitHub). To install the libraries in the virtual environment:

```
pip3 install -r requirements.txt
```

Windows commands: using the command line in Windows is a bit different with respect to Linux and macOS. These are the typical commands:

- Creation of a virtual environment:

  ```
  py -m venv C:\Users\ernesto\path_to_env
  ```

- Activation of virtual environment (execute "activate" script):

  ```
  C:\Users\ernesto\path_to_env\Scripts\activate
  ```

- Installing pip and making sure pip, setuptools and wheels are up to date using:

  ```
  py -m pip install --upgrade pip setuptools wheel
  ```

- Installing requirements (requirements file in GitHub). Run one of the following:

  ```
  py -m install -r requirements.txt
  ```

  ```
  py -m pip install -r requirements.txt
  ```

2.1 Key libraries

- Jupyter notebook: Web-based application to execute (python) code together with comments.
  - Docs: `https://jupyter-notebook.readthedocs.io/en/latest/`
  - Example in lab1: `jupyter notebook lab1-notebook.ipynb`
- RDFlib: a library to manage RDF-based knowledge graphs.
  - Docs: `https://rdflib.readthedocs.io/en/stable/`
  - Examples: `https://github.com/RDFLib/rdflib`
- SPARQLWrapper: a python wrapper around a SPARQL service.
  - Docs and examples: `https://github.com/RDFLib/sparqlwrapper`
- Owlready: a package for ontology-oriented programming in Python.
  - Docs: `https://owlready2.readthedocs.io/en/latest/intro.html`
- OWL-RL: A simple library to perform reasoning on the OWL2 RL Profile.
  - Docs: `https://owl-rl.readthedocs.io/en/latest/`

3 Testing the infrastructure set-up

The codes provided in the folder *test* give an overview about some of the functionalities we will use in the course. They include methods for:

- Loading and ontology and printing its classes (*test-load-ontology.py* script/class).
- Loading and querying a local RDF knowledge graph (*test-load-RDFGraph.py* script).
- Querying a remote RDF knowledge graph via a SPARQL endpoint (*test-query-endpoint.py* script).
- Performing OWL reasoning within the RL profile (*test-owl-reasoning.py* script).
- Read tabular data from a CSV file (*test-open-csv.py* script).

Exercise 1.1: Check you can run the python scripts (or the `test-notebook.ipynb` Jupyter notebook), without compilation errors. At this stage you do not need to understand all the details, but you can start giving a look to the codes and get familiar with the libraries.

4 Protégé ontology editor

Protégé is an editor of OWL ontologies (`https://protege.stanford.edu/`). We will use the desktop version.

The pizza ontology and its tutorial are well-known resources in the Semantic Web community. They were developed for educational purposes by the University of Manchester. The tutorial have been recently updated by Michael DeBellis.

The pizza ontology and the tutorial are found at:

- `https://tinyurl.com/NewPizzaTutorialV3-2`
- `http://protege.stanford.edu/ontologies/pizza/pizza.owl`

Exercise 1.2: Install Protégé (Desktop version) and load the Pizza ontology (File → Open from URL: `https://protege.stanford.edu/ontologies/pizza/pizza.owl`).

Exercise 1.3: Take some time to explore the interface. You may want to check the first steps of the tutorial. Browse the class hierarchy, the property hierarchies and the individuals and note how the ontology describes the domain of pizzas. Check the documentation if necessary: `http://protegeproject.github.io/protege/`.

Exercise 1.4: Find `Margherita` and see how it is defined as a pizza with only cheese and tomato topping. Protégé uses OWL, and thus formal semantics to define the concepts of the domain. Look at the definition of `VegetarianPizza`. Is a `MargheritaPizza` a vegetarian pizza? Why / why not?