General harvester for Ricgraph (Research in context graph)

Rik D.T. Janssen, June 24, 2025

Introduction

In this project, the aim is to design and program in Python a general harvester for Ricgraph. A harvester gets information from source systems and stores this information in Ricgraph. This new harvester should be easy to maintain and easy to extend for new harvest sources, such as e.g. Zenodo, ORCID, Scopus, Lens, OpenAIRE, DataCite Commons, GitHub (and other Gits).

Ricgraph (www.ricgraph.eu), also known as Research in context graph, enables the exploration of researchers, teams, their results, collaborations, skills, projects, and the relations between these items.

Ricgraph can store many types of items into a single graph. These items can be obtained from various systems and from multiple organizations. Ricgraph facilitates reasoning about these items because it infers new relations between items, relations that are not present in any of the separate source systems. It is flexible and extensible, and can be adapted to new application areas.

In this project, we apply Ricgraph to the application area research information. Research information is about anything related to research: research results, the persons in a research team, their collaborations, their skills, projects in which they have participated, as well as the relations between these entities. Examples of research results are publications, data sets, and software.

Ricgraph is open source software and can be found on GitHub (https://github.com/UtrechtUniversity/ricgraph).

Current situation

Ricgraph includes scripts to harvest research information from five source systems:

- OpenAlex: harvest_openalex_to_ricgraph.py.
- Pure: harvest_pure_to_ricgraph.py.
- Research Software Directory: harvest_rsd_to_ricgraph.py.
- Utrecht University staff pages: harvest_uustaffpages_to_ricgraph.py.
- Yoda: harvest_yoda_datacite_to_ricgraph.py.

The sources for these scripts are in https://github.com/UtrechtUniversity/ricgraph/tree/main/harvest, the documentation is in https://docs.ricgraph.eu/docs/ricgraph_harvest_scripts.html.

Every harvest script is structured as follows:

1. extract data from a source system;
2. (optional) process or combine (transform) the harvested data, e.g., combine a field with a first name and a field with a last name to one field representing a full name;
3. transform the data to item pairs, load in Ricgraph.

Since every script uses this structure, there is a lot of duplication in code, which is not considered to be a “good programming practice”. New harvest scripts for new sources are relatively easy to write, by copying one of the scripts and adapting it for the new harvest source. However, that generates even more code duplication, so this is not desirable.

Envisioned situation

A new general harvester that is easy to maintain and easy to extend for new harvest sources. That is, easy to extend for persons that know how to program in Python.

A possibility to implement this might be inheritance as used in object oriented environments. Another possibility a kind of “specification file” for every source system, that contains the things that are

different for each source system to be harvested, and a Python script that reads this specification file, and then does the three steps mentioned in the previous section. It is part of the design phase of the general harvester to devise an attractive and “future proof” solution, in “good programming practice”, in maintainability, and in ease of writing scripts to harvest new sources.

The harvest scripts for the five source systems in section “Current situation” vary in how complicated they are. The harvest script for the Research Software Directory is the easiest, followed by the harvest script for OpenAlex. The harvest script for Pure is the most complicated, since it harvests several types of items (persons, organizations, research outputs, projects) which are dependent on each other, e.g.:

- Persons, projects and research results are connected to organizations.
- Organizations have (sub-)organizations.
- Research results have external authors and author collaborations which are only found during parsing of the research results (and not during parsing of persons).
- Projects are connected to almost everything.

The UU staff pages harvest script has a special structure because of its API interface, and the Yoda harvest script harvests XML instead of JSON.

In this project, an incremental approach for design and implementation is used, starting with the Research Software Directory or OpenAlex, followed by Pure, followed by the UU staff pages and Yoda.

The following is a list of deliverables. There is no implied order. It is to be expected that many of these activities are intertwined.

- Web review on possible existing design approaches.
- Web review on possible existing solutions.
- Design & implementation in Python. Documentation.
- For the chosen design, creating scripts/specification files/whatever for each of the five source systems.
- Testing. This is done by a comparison of the harvesting results of each of the five current harvest scripts, compared to that of the general harvester for the same harvest source.
- Bonus: creating scripts/specification files/whatever for one or more new source systems.