oc_graphenricher

Release 0.2.1

Gabriele Pisciotta

Jul 13, 2023

CONTENTS:

A tool to enrich any OCDM compliant Knowledge Graph, finding new identifiers and deduplicating entities.

CHAPTER

ONE

OC GRAPHENRICHER

A tool to enrich any OpenCitations Data Model (OCDM) compliant Knowledge Graph, finding new identifiers and deduplicating entities.

You can use integrate this package in your own python program or use it from the CLI.

1.1 License

Distributed under the ISC License. See *LICENSE* for more information.

1.2 Contact

Gabriele Pisciotta - ga.pisciotta@gmail.com

Project Link: https://github.com/opencitations/oc_graphenricher

1.3 Acknowledgements

This project has been developed as part of the Wikipedia Citations in Wikidata research project, under the supervision of prof. Silvio Peroni.

CHAPTER

TWO

HOW TO INSTALL

2.1 Installing from Pypi

To get the official and updated version of this package, follow these simple steps:

1. install python >= 3.8:

```
sudo apt install python3
```

2. Install oc_graphenricher via pip:

```
pip install oc-graphenricher
```

2.2 Installing from the sources

It's also possible to build the package from the sources. To do that, follow the following:

1. Having already installed python, you can also install GraphEnricher via cloning this repository:

```
git clone https://github.com/opencitations/oc_graphenricher`
cd ./oc_graphenricher
```

2. install poetry:

```
pip install poetry
```

3. install all the dependencies:

```
poetry install
```

4. build the package:

```
poetry build
```

5. install the package:

```
pip install ./dist/oc_graphenricher-<VERSION>.tar.gz
```

2.3 Run the tests

To run the tests (from the root of the project):

```
poetry run test
```

CHAPTER

THREE

TUTORIAL

The OC GraphEnricher is supposed to accept only graph set objects. To create one:

```
from oc_ocdm.reader import Reader
from oc_ocdm.graph import GraphSet
from rdflib import Graph

g = Graph()
g = g.parse('../data/test_dump.ttl', format='nt11')

reader = Reader()
g_set = GraphSet(base_iri='https://w3id.org/oc/meta/')
entities = reader.import_entities_from_graph(g_set, g, enable_validation=False,
↪ resp_agent='https://w3id.org/oc/meta/prov/pa/2')
```

3.1 Enrichment

At this point, to run the enrichment phase:

```
from oc_graphenricher.enricher import Enricher

enricher = GraphEnricher(g_set)
enricher.enrich()
```

You'll see the progress bar with an estimate of the time needed and the average time spent for each Bibliographic Resource (BR) enriched.

3.2 Deduplication

Then, having serialized the enriched graph set, and having read it again as the *g_set* object, to run the deduplication step do:

```
from oc_graphenricher.instancematching import InstanceMatching

matcher = InstanceMatching(g_set)
matcher.match()
```

The match method will run sequentially: - deduplication of Responsible Agents (RAs) - deduplication of Bibliographic Resources (BRs) - deduplication of Identifiers (IDs) - save to file

If you need to, you can also deduplicate one of those independently of each other.

To deduplicate Responsible Agents (RAs):

```
from oc_graphenricher.instancematching import InstanceMatching

matcher = InstanceMatching(g_set)
matcher.instance_matching_ra()
matcher.save()
```

To deduplicate Bibliographic Resources (BRs):

```
from oc_graphenricher.instancematching import InstanceMatching

matcher = InstanceMatching(g_set)
matcher.instance_matching_br()
matcher.save()
```

To deduplicate Identifiers (IDs):

```
from oc_graphenricher.instancematching import InstanceMatching

matcher = InstanceMatching(g_set)
matcher.instance_matching_id()
matcher.save()
```

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search