WikiDataSets Documentation

Release 0.3.0

Armand Boschin

Apr 20, 2020

Tutorials:

CHAPTER 1

WikiDataSets

Breaking WikiData dumps into smaller knowledge graphs (e.g. graph of human entities).

- Free software: BSD license
- Documentation: https://wikidatasets.readthedocs.io.
- Paper: https://arxiv.org/abs/1906.04536
- Dataset downloads: here

1.1 Citations

If you find this code useful in your research, please consider citing our paper:

1.1.1 Human sub-graph

This is an example of how to build the sub-graph of all human entities from WikiData:

```
import pickle
from wikidatasets.processFunctions import get_subclasses, query_wikidata_dump, build_
↪dataset

path = 'humans/'  # this will contain the files output through the process
dump_path = 'latest-all.json.bz2'  # path to the bz2 dump file
n_lines = 56208653  # this can be an upper bound
test_entities = get_subclasses('Q5')
# Q5 refers to human : common name of Homo sapiens, unique extant species of the␣
↪genus Homo
```

(continues on next page)

(continued from previous page)

```
query_wikidata_dump(dump_path, path, n_lines,
                    test_entities=test_entities, collect_labels=True)

labels = pickle.load(open(path + 'labels.pkl', 'rb'))
build_dataset(path, labels)
```

1.1.2 Process Functions

Get subclasses

Get a list of WikiData IDs of entities which are subclasses of the subject.

wikidatasets.processFunctions.**get_subclasses**(*subject*)

Get a list of WikiData IDs of entities which are subclasses of the subject.

Parameters **subject** (*str*) – String describing the subject (e.g. 'Q5' for human).

Returns **result** – List of WikiData IDs of entities which are subclasses of the subject.

Return type list

Query wikidata dump

Go through a Wikidata dump. It can either collect entities that are instances of test entities or collect the dictionary of labels. It can also do both.

wikidatasets.processFunctions.**query_wikidata_dump**(*dump_path*, *path*, *n_lines*, *test_entities=None*, *collect_labels=False*)

This function goes through a Wikidata dump. It can either collect entities that are instances of *test_entities* or collect the dictionary of labels. It can also do both.

Parameters

- **dump_path** (*str*) – Path to the latest-all.json.bz2 file downloaded from https://dumps.wikimedia.org/wikidatawiki/entities/.
- **path** (*str*) – Path to where pickle files will be written.
- **n_lines** (*int*) – Number of lines of the dump. Fastest way I found was *$ bzgrep -c ".*" latest-all.json.bz2*. This can be an upper-bound as it is only used for displaying a progress bar.
- **test_entities** (*list*) – List of entities to check if a line is instance of. For each line (entity), we check if it as a fact of the type (id, query_rel, test_entity).
- **collect_labels** (*bool*) – Boolean indicating whether the labels dictionary should be collected.

Build dataset

Builds datasets from the pickle files produced by `query_wikidata_dump`.

wikidatasets.processFunctions.**build_dataset**(*path*, *labels*, *return_=False*, *dump_date='23rd April 2019'*)

Builds datasets from the pickle files produced by the query_wikidata_dump.

Parameters

- **path** (*str*) – Path to the directory where there should already be a pickles/ directory. In the latter directory, all the .pkl files will be concatenated into one dataset.
- **labels** (*dict*) – Dictionary collected by the query_wikidata_dump function when collect_labels is set to True.
- **return** (*bool*) – Boolean indicating if the built dataset should be returned on top of being written on disk.
- **dump_date** (*str*) – String indicating the date of the Wikidata dump used. It is used in the readme of the dataset.

Returns

- **edges** (*pandas.DataFrame*) – DataFrame containing the edges between entities of the graph.
- **attributes** (*pandas.DataFrame*) – DataFrame containing edges linking entities to their attributes.
- **entities** (*pandas.DataFrame*) – DataFrame containing a list of all entities & attributes with their Wikidata IDs and labels.
- **relations** (*pandas.DataFrame*) – DataFrame containing a list of all relations with their Wikidata IDs and labels.

1.1.3 Utilities

Load data and labels

Loads the edges and attributes files into Pandas dataframes and merges the labels of entities and relations to get.

wikidatasets.utils.**load_data_labels**(*path*, *attributes=False*, *return_dicts=False*)

This function loads the edges and attributes files into Pandas dataframes and merges the labels of entities and relations to get.

Parameters

- **path** (*str*) – Path to the directory containing the edges.txt, attributes.txt, entities.txt, relations.txt files.
- **attributes** (*bool*) – Boolean indicating if we should read the attributes files. If False, then the edges file is read.
- **return_dicts** (*bool*) – Boolean indicating if the entities and relations labels dictionaries should be returned.

Returns

- **df** (*pandas.DataFrame*) – DataFrame containing either the edges or the attributes depending on the value of *attributes*.
- **entities** (*pandas.DataFrame*) – DataFrame containing the list of all entities and wikidata IDs and labels.
- **relations** (*pandas.DataFrame*) – DataFrame containing the list of all relations and wikidata IDs and labels.

1.1.4 Installation

To install WikiDataSets, run this command in your terminal:

```
$ pip install wikidatasets
```

This is the preferred method to install WikiDataSets, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.1.5 Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/armand33/wikidatasets/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with "bug" and "help wanted" is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with "enhancement" and "help wanted" is open to whoever wants to implement it.

Write Documentation

WikiDataSets could always use more documentation, whether as part of the official WikiDataSets docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/armand33/wikidatasets/issues.

If you are proposing a feature:

- Explain in detail how it would work.

- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *wikidatasets* for local development.

1. Fork the *wikidatasets* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/wikidatasets.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv wikidatasets
$ cd wikidatasets/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 wikidatasets tests
$ python setup.py test or py.test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 3.6 and 3.7, and for PyPy. Check https://travis-ci.org/armand33/wikidatasets/pull_requests and make sure that the tests pass for all supported Python versions.

Deploying

A reminder for the maintainers on how to deploy. Make sure all your changes are committed (including an entry in HISTORY.rst). Then run:

```
$ bumpversion patch # possible: major / minor / patch
$ git push
$ git push --tags
```

Travis will then deploy to PyPI if tests pass.

1.1.6 Credits

Development Lead

- Armand Boschin <aboschin@enst.fr>

Contributors

None yet. Why not be the first?

1.1.7 History

0.3.0 (2020-04-20)

- Switched output format to clean tsv.

0.2.0 (2019-07-02)

- Added export of a nodes.txt to the build_dataset function.

0.1.0 (2019-07-01)

- First release on PyPI.

Index