NetworkX to Neo4j Documentation

Release 0.2.0

Rohit Aggarwal

January 26, 2017

Contents

Contents:

CHAPTER 1

NetworkX to Neo4j

A library to convert a NetworkX graph to a Neo4j graph.

Other information:

- Free software: MIT license.
- Documentation: http://neonx.rtfd.org.

1.1 Features

- Converts a NetworkX graph to a Geoff compatible string.
- Upload a NetworkX graph to Neo4j server using Requests. Ability to add Labels to created nodes in Neo4j 2.0.

CHAPTER 2

Installation

At the command line:

```
$ easy_install neonx
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv neonx
$ pip install neonx
```

CHAPTER 3

Usage

To use NetworkX to Neo4j in a project:

```
import neonx
```

To covert a Networkx graph to a Geoff string:

```
# create a Networkx graph
# LINKS_TO is the relatioship name between the nodes
data = neonx.get_geoff(graph, "LINKS_TO")
```

It is assumed that the all the properties for the nodes are edges are json encodable. If they are not, please extend JSONEncoder. For example, if you want to encode python date objects as well the usual types:

```
import json
import datetime

class DateEncoder(json.JSONEncoder):

    def default(self, o):
        if isinstance(o, datetime.date):
            return o.strftime('%Y-%m-%d')
        return json.JSONEncoder.default(self, o)

data = neonx.get_geoff(graph, "LINKS_TO", DateEncoder())
```

To upload the graph to neo4j server hosted on localhost:

```
results = neonx.write_to_neo("http://localhost:7474/db/data/", graph, 'LINKS_TO')
```

Again, it is assumed that the properties of the graph are json encodable. If not, please pass a custom encoder in a similar way to the example above.

In Neo4j 2.0, the concept of Labels was introduced. If you wish to add a label to the nodes created, just call the command with the label:

```
results = neonx.write_to_neo("http://localhost:7474/db/data/", graph, 'LINKS_TO', 'Person')
```

CHAPTER 4

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:

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/ducky427/neonx/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.

4.1.2 Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

4.1.3 Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

4.1.4 Write Documentation

NetworkX to Neo4j could always use more documentation, whether as part of the official NetworkX to Neo4j docs, in docstrings, or even on the web in blog posts, articles, and such.

4.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/ducky427/neonx/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 :)

4.2 Get Started!

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

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

```
    $ git clone git@github.com:your_name_here/neonx.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 neonx
    $ cd neonx/
    $ 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 neonx tests
$ python setup.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.

4.3 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 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/ducky427/neonx/pull_requests and make sure that the tests pass for all supported Python versions.

4.4 Tips

To run a subset of tests:

```
$ python -m unittest tests.test_neo
```

CHAPTER 5

Credits

5.1 Development Lead

- Rohit Aggarwal <rohit.aggarwal@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

6.1 0.2.0 (Unreleased)

- Added the ability to add a label to the Neo4j nodes created.

6.2 0.1.1 (2013-08-30)

- Fixed error if the Neo4j server dies crashes unexpectedly.
- Added package to Coveralls.

6.3 0.1.0 (2013-08-29)

- First release on PyPI.

CHAPTER 7

Indices and tables

- genindex
- modindex
- search