Supepgdedupeumentation

Release 0.2.1

DSaPP Researchers

May 03, 2017

Contents

Contents:

CHAPTER 1

pgdedupe

A work-in-progress to provide a standard interface for deduplication of large databases with custom pre-processing and post-processing steps.

- Free software: MIT license
- Documentation: https://pgdedupe.readthedocs.io.

Interface

This provides a simple command-line program, `pgdedupe`. Two configuration files specify the deduplication parameters and database connection settings. To run deduplication on a generated dataset, create a `database.yml` file that specifies the following parameters:

```
user:
password:
database:
host:
port:
```

You can now create a sample CSV file with:

```
$ python generate_fake_dataset.py --csv people.csv
creating people: 100%|| 9500/9500 [00:21<00:00, 445.38it/s]
adding twins: 100%|| 500/500 [00:00<00:00, 1854.72it/s]
writing csv:  47%|             | 4666/10000 [00:42<00:55, 96.28it/s]
```

Once complete, store this example dataset in a database with:

```
$ python test/initialize_db.py --db database.yml --csv people.csv
CREATE SCHEMA
DROP TABLE
CREATE TABLE
COPY 197617
```

```
ALTER TABLE
ALTER TABLE
UPDATE 197617
```

Now you can deduplicate this dataset. This will run dedupe as well as the custom pre-processing and post-processing steps as defined in config.yml:

```
$ pgdedupe --config config.yml --db database.yml
```

Custom pre- and post-processing

In addition to running a database-level deduplication with `dedupe`, this script adds custom pre- and post-processing steps to improve the run-time and results, making this a hybrid between fuzzy matching and record linkage.

- **Pre-processing:** Before running dedupe, this script does an exact-match deduplication. Some systems create many identical rows; this can make it challenging for dedupe to create an effective blocking strategy and generally makes the fuzzy matching much harder and time intensive.
- **Post-processing:** After running dedupe, this script does an optional exact-match merge across subsets of columns. For example, in some instances an exact match of just the last name and social security number are sufficient evidence that two clusters are indeed the same identity.

Further steps

This script was based upon and extended from the example in dedupe-examples. It would be nice to use this common interface across all database types, and potentially even allow reading from flat CSV files.

CHAPTER 2

Installation

Stable release

To install pgdedupe, run this command in your terminal:

```
$ pip install numpy && pip install pgdedupe
```

This is the preferred method to install pgdedupe, as it will always install the most recent stable release. Note that numpy is currently a pre-requisite for the installation process itself.

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

From sources

The sources for pgdedupe can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dssg/pgdedupe
```

Or download the tarball:

```
$ curl  -OL https://github.com/dssg/pgdedupe/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

Simply install with `pip install pgdedupe` and then run with a given configuration file and Postgres database credentials with:

> pgdedupe –config config.yaml –db database.yaml

See the example configuration file for a listing of the supported fields and their descriptions.

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:

Types of Contributions

Report Bugs

Report bugs at https://github.com/dssg/pgdedupe/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

More documentation is always helpful, whether as part of the official pgdedupe 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/dssg/pgdedupe/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 *pgdedupe* for local development.

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

```
$ git clone git@github.com:your_name_here/pgdedupe.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 pgdedupe
$ cd pgdedupe/
$ 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 pgdedupe 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 2.6, 2.7, 3.3, 3.4 and 3.5, and for PyPy. Check https://travis-ci.org/dssg/pgdedupe/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ py.test tests.test_pgdedupe
```

CHAPTER 5

Indices and tables

- genindex
- modindex
- search