scrubadub Documentation

Release 1.2.0

Dean Malmgren

Jun 16, 2017

Contents

Remove personally identifiable information from free text. Sometimes we have additional metadata about the people we wish to anonymize. Other times we don't. This package makes it easy to seamlessly scrub personal information from free text, without comprimising the privacy of the people we are trying to protect.

`scrubadub` currently supports removing:

- Names (proper nouns) via textblob
- Email addresses
- URLs
- Phone numbers via phonenumbers
- username / password combinations
- Skype usernames
- Social security numbers

CHAPTER 1

Quick start

Getting started with `scrubadub` is as easy as `pip install scrubadub` and incorporating it into your python scripts like this:

```
>>> import scrubadub

# John may be a cat, but he doesn't want other people to know it.
>>> text = u"John is a cat"

# Replace names with {{NAME}} placeholder. This is the scrubadub default
# because it maximally omits any information about people.
>>> scrubadub.clean(text)
u"{{NAME}} is a cat"

# Replace names with {{NAME-ID}} anonymous, but consistent IDs.
>>> scrubadub.clean(text, replace_with='identifier')
u"{{NAME-0}} is a cat"
>>> scrubadub.clean("John spoke with Doug.", replace_with='identifier')
u"{{NAME-0}} spoke with {{NAME-1}}."
```

There are many ways to tailor the behavior of `scrubadub` using different ‘‘Detector‘ and `Filth` classes <under_the_hood>‘. These *advanced techniques* allow users to fine-tune the manner in which `scrubadub` cleans dirty dirty text.

CHAPTER 2

Related work

`scrubadub` isn't the first package to attempt to remove personally identifiable information from free text. There are a handful of other projects out there that have very similar aims and which provide some inspiration for how `scrubadub` should work.

- MITRE gives the ability to replace names with a placeholder like `[NAME]` or alternatively replace names with fake names. last release in 8/2014. not on github. unclear what language although it looks like python. it is clear that the documentation sucks and is primarily intended for academic audiences (docs are in papers).
- physionet has a few deidentification packages that look pretty decent but are both written in perl and require advance knowledge of what you are trying to replace. Intended for HIPAA regulations. In particular, deid has some good lists of names that might be useful in spite of the fact it has 5k+ lines of gross perl.

Contents:

Advanced usage

By default, `scrubadub` aggressively removes content from text that may reveal personal identity, but there are certainly circumstances where you may want to customize the behavior of `scrubadub`. This section outlines a few of these use cases. If you don't see your particular use case here, please take a look *under the hood* and *contribute* it back to the documentation!

Suppressing a detector

In some instances, you may wish to suppress a particular detector from removing information. For example, if you have a specific reason to keep email addresses in the resulting output, you can disable the email address cleaning like this:

```
>>> import scrubadub
>>> scrubber = scrubadub.Scrubber()
>>> scrubber.remove_detector('email')
>>> text = u"contact Joe Duffy at joe@example.com"
```

```
>>> scrubadub.clean(text)
u"contact {{NAME}} {{NAME}} at joe@example.com"
```

Customizing filth markers

By default, `scrubadub` uses mustache notation to signify what has been removed from the dirty dirty text. This can be inconvenient in situations where you want to display the information differently. You can customize the mustache notation by changing the `prefix` and `suffix` in the `scrubadub.filth.base.Filth` object. For example, to bold all of the resulting text in HTML, you might want to do this:

```
>>> import scrubadub
>>> scrubadub.filth.base.Filth.prefix = u'<b>'
>>> scrubadub.filth.base.Filth.suffix = u'</b>'
>>> scrubber = scrubadub.Scrubber()
>>> scrubber.remove_detector('email')
>>> text = u"contact Joe Duffy at joe@example.com"
>>> scrubadub.clean(text)
u"contact <b>NAME</b> <b>NAME</b> at <b>EMAIL</b>"
```

Adding a new type of filth

It is quite common for particular use cases of `scrubadub` to require obfuscation of specific types of filth. If you run across something that is very general, please *contribute it back*! In the meantime, you can always add your own `Filth` and `Detectors` like this:

```
>>> import scrubadub
>>> class MyFilth(scrubadub.filth.base.Filth):
>>>     type = 'mine'
>>> class MyDetector(scrubadub.Detector.base.Detector):
>>>     filth_cls = MyFilth
>>>     def iter_filth(self, text):
>>>         # do something here
>>>         pass
>>> scrubber = scrubadub.Scrubber()
>>> scrubber.add_detector(MyDetector)
>>> text = u"My stuff can be found there"
>>> scrubadub.clean(text)
u"{{MINE}} can be found there."
```

Customizing the cleaned text

Under the hood

`scrubadub` consists of three separate components:

- `Filth` objects are used to identify specific parts of a piece of dirty dirty text that contain sensitive information and they are responsible for deciding how the resulting information should be replaced in the cleaned text.
- `Detector` objects are used to detect specific types of `Filth`.
- The `Scrubber` is responsible for managing all of the `Detector` objects and resolving any conflicts that may arise between different `Detector` objects.

Filth

Filth objects are responsible for marking particular sections of text as containing that type of filth. It is also responsible for knowing how it should be cleaned. Every type of `Filth` inherits from *scrubadub.filth.base.Filth*.

class `scrubadub.filth.base.`**`Filth`**(*beg=0*, *end=0*, *text=u''*)
Bases: `object`

This is the base class for all `Filth` that is detected in dirty dirty text.

`prefix` = **u'{{'**

`suffix` = **u'}}'**

`type` = **None**

`lookup` = **<scrubadub.utils.Lookup object>**

`placeholder`

`identifier`

`replace_with`(*replace_with='placeholder'*, ***kwargs*)

`merge`(*other_filth*)

There is also a convenience class for `RegexFilth`, which makes it easy to quickly remove new types of filth that can be identified from regular expressions:

class `scrubadub.filth.base.`**`RegexFilth`**(*match*)
Bases: *`scrubadub.filth.base.Filth`*

Convenience class for instantiating a `Filth` object from a regular expression match

`regex` = **None**

Detectors

`scrubadub` consists of several `Detector`'s, which are responsible for identifying and iterating over the `Filth` that can be found in a piece of text. Every type of `Filth` has a `Detector` that inherits from `scrubadub.detectors.base.Detector`:

class `scrubadub.detectors.base.`**`Detector`**
Bases: `object`

`filth_cls` = **None**

`iter_filth`(*text*)

For convenience, there is also a `RegexDetector`, which makes it easy to quickly add new types of `Filth` that can be identified from regular expressions:

class `scrubadub.detectors.base.`**`RegexDetector`**
Bases: *`scrubadub.detectors.base.Detector`*

`iter_filth`(*text*)

Scrubber

All of the `Detector`'s are managed by the `Scrubber`. The main job of the `Scrubber` is to handle situations in which the same section of text contains different types of `Filth`.

class scrubadub.scrubbers.**Scrubber**(**args*, ***kwargs*)
Bases: object

The Scrubber class is used to clean personal information out of dirty dirty text. It manages a set of Detector's that are each responsible for identifying their particular kind of Filth.

add_detector(*detector_cls*)
Add a Detector to scrubadub

remove_detector(*name*)
Remove a Detector from scrubadub

clean(*text*, ***kwargs*)
This is the master method that cleans all of the filth out of the dirty dirty text. All keyword arguments to this function are passed through to the Filth.replace_with method to fine-tune how the Filth is cleaned.

iter_filth(*text*)
Iterate over the different types of filth that can exist.

Contributing

The overarching goal of this project is to remove personally identifiable information from raw text as reliably as possible. In practice, this means that this project, by default, will preferentially be overly conservative in removing information that might be personally identifiable. As this project matures, I fully expect the project to become ever smarter about how it interprets and anonymizes raw text.

Regardless of which personal information is identified, this project is committed to being as agnostic about the manner in which the text is anonymized, so long as it is done with rigor and does not inadvertantly lead to improper anonymization. Replacing with placholders? Replacing with anonymous (but consistent) IDs? Replacing with random metadata? Other ideas? All should be supported to make this project as useful as possible to the people that need it.

Another important aspect of this project is that we want to have extremely good documentation and source code that is easy to read. If you notice a type-o, error, confusing statement etc, please fix it!

Quick start

1. Fork and clone the project:

```
git clone https://github.com/YOUR-USERNAME/scrubadub.git
```

2. Create a python virtual environment and install the requirements

```
mkvirtualenv scrubadub
pip install -r requirements/python-dev
```

3. Contribute! There are several open issues that provide good places to dig in. Check out the contribution guidelines and send pull requests; your help is greatly appreciated!

4. Run the test suite that is defined in .travis.yml to make sure everything is working properly

```
./tests/run.py
```

Current build status:

Change Log

This project uses semantic versioning to track version numbers, where backwards incompatible changes (highlighted in **bold**) bump the major version of the package.

latest changes in development for next release

1.2.0

- added python 3 compatability (#31 via @davidread)

1.1.1

- fixed `FilthMergeError` (#29 via @hugofvs)

1.1.0

- regular expression detection of Social Security Numbers (#17)
- Added functionality to keep `replace_with = "identifier"` (#21)
- several bug fixes, including:
  - inaccurate name detection (#19)

1.0.3

- minor change to force `Detector.filth_cls` to exist (#13)

1.0.1

- several bug fixes, including:
  - installation bug (#12)

1.0.0

- **major update to process Filth in parallel** (#11)

0.1.0

- added skype username scrubbing (#10)
- added username/password scrubbing (#4)
- added phone number scrubbing (#3)
- added URL scrubbing, including URL path removal (#2)
- make sure unicode is passed to `scrubadub` (#1)
- several bug fixes, including:

  - accuracy issues with things like “I can be reached at 312.456.8453” (#8)
  - accuracy issues with usernames that are email addresses (#9)

0.0.1

- initial release, ported from past projects

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Index