

# CS 5293, Spring 2020 Project 1

*Due 3/27 end of day*

## The Redactor

### Introduction

Whenever sensitive information is shared with the public, the data must go through a redaction process. That is, all sensitive names, places, and other sensitive information must be hidden. Documents such as police reports, court transcripts, and hospital records all containing sensitive information. Redacting this information is often expensive and time consuming.

### Task Overview

In this project, you will use your knowledge of Python and Text Analytics to design a system that accept plain text documents then detect and redact “sensitive” items. Below is an example execution of the program.

```
pipenv run python redactor.py --input '*.txt' \  
                                --input 'otherfiles/*.md' \  
                                --names --dates \  
                                --concept 'kids' \  
                                --output 'files/' \  
                                --stats stderr
```

Running the program with this command line argument should read all files given by the glob — in this case all the files ending in `.txt` in in the folder and also all files ending in `.md` from the folder called `otherfiles/`. All these files will be redacted by the program. The program will look to redact all names, dates, and phone numbers. Notice the flag `--concept`, this flag asks the system to redact all portions of text that have anything to do with a particular concept. In this case, all paragraphs or sentences that contain information about “kids” should be redacted. *It is up \*you\* to determine what represents a concept.* All the redacted files should be transformed to new files of the same name with the `.redacted` extension, and written to the folder described by `--output` flag. The final parameter, `--stats`, describes the file or

location to write the statistics of the redacted files. Below we discuss each of the parameter in additional detail.

## Parameters

### -input

This parameter takes a [glob](#) that represents the files that can be accepted. More than one input flag may be used to specify groups of files. If a file cannot be read or redacted an appropriate error message should be displayed to the user.

### -output

This flag should specify a directory to store all the redacted files. The redacted files, regardless of their input type should be written to text files. Each file should have the same name as the original file with the extension `.redacted` appended to the file name.

## Redaction flags

The redaction flags list the entity types that should be extracted from all the input documents. The list of flags you are required to implement are:

- `--names`
- `--genders`
- `--dates`

You are free to add your own! Note: gender should be any term that reveals the gender of a person (e.g. him, her, etc.). The other definitions of the rest of the terms should be straight forward. In your README.md discussion file clearly give the parameters you apply to each of the flags — be clear what constitutes an address and phone number. The redacted characters in the document should be replaced upon with a character of your choice. Some popular characters include the unicode [full block character](#) ■ (U+2588). You may choose to redact both words and the whitespaces between phrases. If you believe that one should also redact whitespaces between words (e.g. in a first and last name) please discuss why in your README.md.

### -concept

This flag, which can be repeated one or more times, takes one word or phrase that represents a concept. A concept is either an idea or theme. Any section of the input files that refer to this concept should be redacted. For example, if the given concept

word is prison, a sentence (or paragraph) either containing the word or similar concepts, such as jail or incarcerated, that *whole sentence* should be redacted. In your README.md file, make your definition of a concept clear. Also, clearly state how you create the context of a concept and justify your method. You may be creative here!

### —stats

Stats takes either the name of a file, or special files (`stderr`, `stdout`), and writes a summary of the redaction process. Some statistics to include are the types and counts of redacted terms and the statistics of each redacted file. Be sure to describe the format of your outfile to in your README file. Stats should help you while developing your code. Stats may also include the beginning and end index of each redacted item. It is up to you to design your stats output.

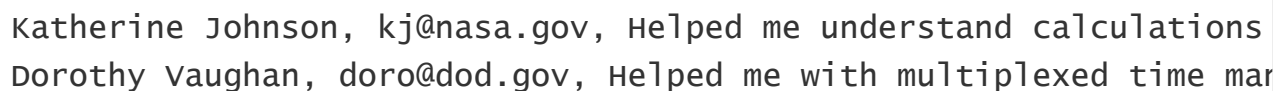
## Submission

### README.md

The README file name should be *uppercase* with an `.md` extension. You should write your name in it, and example of how to run it, and a list of any web or external resources that you used for help. The README file should also contain a list of any bugs or assumptions made while writing the program. Note that you should not be copying code from any website not provided by the instructor. You should include directions on how to install and use the code. You should describe any known bugs and cite any sources or people you used for help. **Be sure to include any assumptions you make for your solution.**

### COLLABORATORS file

This file should contain a comma separated list describing who you worked with and a small text description describing the nature of the collaboration. This information should be listed in three fields as in the example is below:



```
Katherine Johnson, kj@nasa.gov, Helped me understand calculations  
Dorothy Vaughan, doro@dod.gov, Helped me with multiplexed time mar
```

## Project Descriptions

Your code structure should be in a directory with something similar to the following format:

```
cs5293p20-project1/
├── COLLABORATORS
├── LICENSE
├── Pipfile
├── Pipfile.lock
├── README.md
├── project1
│   ├── __init__.py
│   ├── main.py
│   └── ...
├── docs/
├── setup.cfg
├── setup.py
└── tests/
    ├── test_names.py
    ├── test_genders.py
    └── ...
```

## setup.py

```
from setuptools import setup, find_packages

setup(
    name='project1',
    version='1.0',
    author='You Name',
    author_email='your ou email',
    packages=find_packages(exclude=('tests', 'docs')),
    setup_requires=['pytest-runner'],
    tests_require=['pytest']
)
```

Note, the `setup.cfg` file should have at least the following text inside:

```
[aliases]
test=pytest

[tool:pytest]
norecursedirs = .*, CVS, _darcs, {arch}, *.egg, venv
```

## Tests

The general rule is you should aim to have a test for each feature. Including tests help people understand how your code works, in addition to verifying assumptions during development. Tests should be runnable by using `pipenv run python -m pytest`. You should discuss your tests in you README.

## Extra links and Notes

It is expected that you will use nltk to complete this assignment. However, you are welcomed to use other popular APIs. Some of these API's a larger instance may require a larger instance, please let us know if you plan to do this. Also, some APIs require specialised keys, please let the TA know how you plan to use your keys. Below is the information for using Google tools.

Creating API Keys <https://cloud.google.com/docs/authentication/api-keys>

Google NLP <https://googlecloudplatform.github.io/google-cloud-python/latest/language/usage.html#annotate-text>

Using Google Natural Language Client [GCP](#)

[Spacy 2.0](#)

[NLTK](#)

## Create a repository for your project on your instance and GitHub

Create a private repository GitHub called `cs5293sp20-project1`

Add collaborators `cegme`, `kbanweer`, and `mghirsch42` by going to `Settings > Collaborators`.

Then go to your instance, create a new folder `/projects` if it is not already created, and clone the repository into that folder. For example:

You should regularly `git add`, `git commit -m`, and `git push origin master` your code changes to GitHub.

## Submitting your code

When ready to submit, create a tag on your repository using git tag on the latest commit:

```
git tag v1.0  
git push origin v1.0
```

The version v1.0 lets us know when and what version of code you would like us to grade. If you need to submit an updated version, you can use the tag v1.1. If you would like to submit a second version before the 24 hour deadline, use the tag v2.0.

If you need to update a tag, view the commands in the following [StackOverflow post](#).

## Deadline

Your submission/release is due on Friday 27th at 11:45pm. Submissions arriving between 11:45:01pm and 11:45pm on the following day will receive a 10% penalty; meaning these scores will only receive 90% of its value. Any later submissions will not receive credits for this project. You must have your instance running and code available otherwise you will not receive credit.

## Grading

Grades will be assessed according to the following distribution:

- 60%: Correctness.
  - This will be assessed by giving your code a range of inputs and checking the output.
  - Use the creation of tests to prove correctness.
- 40%: Documentation.
  - Your README file should fully explain your process for developing your code.
  - All other commands should be well-documented.

## Addendum

- 2020-03-12 Here is the GitHub's documentation for writing Markdown files <https://guides.github.com/features/mastering-markdown/>

[Back to Project List](#)

---