pywebcat Documentation

Tomas Beuzen

Jun 12, 2020

Contents:

Python tool for working with the NOAA NOS Web Camera Applications Testbed (WebCAT).

To get started, check out the sections below:

CHAPTER 1

Background

This package provides a Pythonic way to interface with the NOAA National Ocean Service Web Camera Applications Testbed (WebCAT). The real-time data is hosted on the SECOORA site (Southeast Coastal Ocean Observing Regional Association), on the dedicated WebCAT page. Historic data can also be accessed by retrieving files using specific HTTP requests (using a pattern described on the WebCAT page).

More details about WebCAT are available in this Open Access paper:

> Dusek, G., Hernandez, D., Willis, M., Brown, J. A., Long, J. W., Porter, D. E., & Vance, T. C. (2019). WebCAT: Piloting the development of a web camera coastal observing network for diverse applications. Frontiers in Marine Science, 6, 353, 25 June 2019 | https://doi.org/10.3389/fmars.2019.00353

CHAPTER 2

Installation

2.1 Stable release

To install *pywebcat*, run this command in your terminal:

```
$ pip install pywebcat
```

This is the preferred method to install *pywebcat*, 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.

2.2 From sources

The sources for *pywebcat* can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git@github.com:UNCG-DAISY/WebCAT-Utilities.git
```

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

```
$ poetry install
```

CHAPTER 3

Usage

3.1 CLI

A key workflow for utilising WebCAT videos is to split videos into frames for further analysis. As a result, this functionality is exposed as a CLI with the command line argument *pywebcat*.

```
$ pywebcat --help

usage: webcat_utils.py [-h] -dir DIRECTORY -s STATION [STATION ...] -y YEAR
                       [YEAR ...] -m MONTH [MONTH ...] -d DAY [DAY ...] -t
                       TIME [TIME ...] [-i INTERVAL] [-n] [-v]

CLI for saving frames of webCAT video(s).

optional arguments:
  -h, --help            show this help message and exit
  -i INTERVAL, --interval INTERVAL
                        Interval in seconds between video frames to save
                        (default: 10).
  -n, --no_meta         Don't save .csv file of metadata of saved video
                        frames.
  -v, --verbose         Print program status.

required arguments:
  -dir DIRECTORY, --directory DIRECTORY
                        Absolute path of directory to save frames in.
  -s STATION [STATION ...], --station STATION [STATION ...]
                        The station name, e.g., buxtoncoastalcam.
  -y YEAR [YEAR ...], --year YEAR [YEAR ...]
                        The video year(s), e.g., 2019 2020.
  -m MONTH [MONTH ...], --month MONTH [MONTH ...]
                        The video month(s), e.g., 9 10 11.
  -d DAY [DAY ...], --day DAY [DAY ...]
                        The video day(s) e.g., 1 11 21.
```

(continues on next page)

(continued from previous page)

```
  -t TIME [TIME ...], --time TIME [TIME ...]
                        The video time(s), e.g., 1000 1330 1510.
```

The CLI facilitates efficiently looping over input arguments to locate one or more WebCAT videos and split them into a desired number of frames which are then saved locally. Here's an example looping over several videos from the Buxton coastal camera on 13th Nov 2019 at different times (a 100 second interval is specified for saving the frames and verbosity is turned on):

```
$ pywebcat -dir /Users/tbeuzen/Downloads -s buxtoncoastalcam -y 2019 -m 11 -d 13 -t␣
↪1000 1210 1530 -i 100 -v
Saving frames of buxtoncoastalcam_2019_11_13_1000...
100%|| 62/62 [00:14<00:00,  4.22it/s]
Saving frames of buxtoncoastalcam_2019_11_13_1210...
100%|| 62/62 [00:13<00:00,  4.51it/s]
Saving frames of buxtoncoastalcam_2019_11_13_1530...
100%|| 62/62 [00:14<00:00,  4.38it/s]
```

The resultant directory structure looks like:

```
Users/tbeuzen/Downloads
                └── buxtoncoastalcam
                    ├── buxtoncoastalcam_2019_11_13_1000
                    │   ├── buxtoncoastalcam_2019_11_13_1000.csv
                    │   └── jpg
                    │       ├── frame_0.jpg
                    │       ├── frame_280.jpg
                    │       ├── ...
                    ├── buxtoncoastalcam_2019_11_13_1210
                    │   ├── ...
                    └── buxtoncoastalcam_2019_11_13_1530
                        ├── ...
```

The outputted .csv file contains metadata for the saved frames:

url	name	frame	path
http://webcat-video.axds.co/buxtoncoastalcam/raw/2019/2019_11/2019_11_13/buxtoncoastalcam.2019-11-13_1000.mp4	buxtoncoastalcam_2019_11_13_1000	0	/Users/tbeuzen/Downloads/buxtoncoastalcam/
http://webcat-video.axds.co/buxtoncoastalcam/raw/2019/2019_11/2019_11_13/buxtoncoastalcam.2019-11-13_1210.mp4	buxtoncoastalcam_2019_11_13_1210	280	/Users/tbeuzen/Downloads/buxtoncoastalcam/
...	...	...	...

3.2 Module

The pywebcat utilities can also be imported through the *pywebcat.utils* for use in other libraries or workflows. See the [demo Jupyter notebook](notebooks/pywebcat_demo.ipynb) for a worked example.

```
from pywebcat.utils import WebCAT
wc = WebCAT()
wc.generate_url("buxtoncoastalcam", 2019, 11, 13, 1000)  # create the video url

# attributes
wc.url      # the created url
wc.name     # unique name for the video object
wc.width    # frame width in pixels
wc.height   # frame height in pixels
wc.frames   # total frames in video
wc.fps      # frames per second

# methods
wc.download_url()         # download the video at the url
wc.save_frames()          # save video frames as .jpg
wc.plot_frames()          # plot select video frames
wc.plot_average_frame()   # plot time-averaged frame
```

CHAPTER 4

Contributing

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

4.1 Types of Contributions

4.1.1 Report Bugs

Report bugs at https://github.com/UNCG-DAISY/WebCAT-Utilities/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" and "help wanted" is open to whoever wants to implement it.

4.1.3 Implement Features

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

4.1.4 Write Documentation

foocattb could always use more documentation, whether as part of the official foocattb 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/UNCG-DAISY/WebCAT-Utilities/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 *pywebcat* for local development.

1. Fork the *WebCAT-Utilities* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:UNCG-DAISY/WebCAT-Utilities.git
```

3. Create a branch for local development:

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

Now you can make your changes locally.
```

4. 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
```

5. Submit a pull request by e.g., using 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.md.
3. The pull request should pass GitHub actions workflows.

4.4 Code of Conduct

Please note that the WebCAT-Utilities project is released with a Contributor Code of Conduct. By contributing to this project you agree to abide by its terms.

CHAPTER 5

Code of Conduct

5.1 Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

5.2 Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

5.3 Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

5.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

5.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

5.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant homepage, version 1.4.

CHAPTER 6

Credits

6.1 Development Lead

- Tomas Beuzen
- Evan Goldstein

6.2 Contributors

- None yet. Why not be the first?

CHAPTER 7

Indices

- genindex
- modindex