turkleton Documentation

Release 1.2.1

Eric Scrivner

July 28, 2015

Contents

Contents:

CHAPTER 1

turkleton

Dead simple Python interfaces for Amazon Mechanical Turk.

- Free software: BSD license
- Documentation: https://turkleton.readthedocs.org.

1.1 Installation

Simply use pip to download the package from PyPI

```
$ pip install turkleton
```

1.2 Features

Turkleton aims to leverage the expressive powers of Python to make using Mechanical Turk easier. The highlights are:

- Simple interface to define human intelligence tasks (HITs).
- Define schemas for your results before downloading them.
- Easily upload tasks in batches.
- Easily download and validate assignments.

1.3 Examples

Some basic terminology is required to get up and running with Turkleton.

A Task is a Human Intelligence Task (HIT). To get started with Turkleton you should first create a layout for your task in Mechanical Turk. You then provide your layout ID to turkleton as part of your task definition.

Assignments contain the answers given by a turker to the questions in your task. An assignment defines the schema for the answers. Turkleton then uses your assignment to parse and validate the answers it receives.

1.3.1 Setting Up Your Connection

The first thing you need to do is setup your connection to Mechanical Turk.

Turkleton uses a per-process global connection. You should always initialize it before you attempt to upload or download anything. You initialize it like so:

```
from turkleton import connection
connection.setup(AWS_ACCESS_KEY, AWS_SECRET_ACCESS_KEY)
```

That's it!

1.3.2 Creating A Task And Uploading It

Once you've created your layout on Mechanical Turk you can create HITs by defining a task in Turkleton.

To define a HIT you create a Task representing the template of the assignment you want a worker to complete. For example:

```
import datetime

from turkleton.assignment import task

class MyTask(task.BaseTask):
    __layout_id__ = 'MY LAYOUT ID'
    __reward__ = 0.25
    __title__ = 'Guess How Old From Picture'
    __description__ = 'Look at a picture and guess how old the person is.'
    __keywords__ = ['image', 'categorization']
    __time_per_assignment__ = datetime.timedelta(minutes=5)
```

Now that you've defined your task you can easily upload HITs as follows:

```
task = MyTask(image_url='http://test.com/img.png', first_guess='29')
hit = task.upload(batch_id='1234')
```

This will create a new assignment from the task template and upload it to Mechanical Turk. The variables image_url and first_guess in your template will contain the values given. The optional batch_id parameter allows you to set the requester annotation for the task to an arbitrary string. This is useful when you've uploaded more than one task in a batch. In the evaluation phase you can filter which assignments are downloaded by a given batch id.

1.3.3 Uploading Multiple Tasks

Usually you want to upload more than a one task. Turkleton provides two methods for easily doing this.

The first method uses the create_and_upload method on your Task as follows:

```
for image_url in all_image_urls:
    MyTask.create_and_upload(
        image_url=image_url, first_guess='29', batch_id='1234'
    )
```

It is often convenient to only set the batch id once. The task.batched_upload context manager is providing to make this approach easy as well:

```
with task.batched_upload(batch_id='1234')
    for image_url in all_image_urls:
       MyTask.create_and_upload(image_url=image_url, first_guess='29')
```

Every task you upload within the context will be automatically given the specified batch id.

1.3.4 Downloading The Results

When you want to download your results you'll need to define an assignment. The assignment defines the types of values you expect to get. These are used to automatically parse and type cast your answers so you can just deal with evaluating the results.

You can define a simple task for categorizing an image as follows:

```
from turkleton.assignment import assignment
from turkleton.assignment import answers

class MyAssignment(assignment.BaseAssignment):
    categories = answers.MultiChoiceAnswer(question_name='Categories')
    notes = answers.TextAnswer(question_name='AdditionalNotes', default='')
    does_not_match_any = answers.BooleanAnswer(
        question_name='DoesNotMatchAnyCategories', default=False
    )
```

You can then download all of the HITs in a given batch as follows:

```
from turkleton.assignment import hit
reviewable_hits = hit.get_reviewable_by_batch_id('1234')
```

Each HIT may have multiple assignments associated with it. This is the case if the __assignments_per_hit__ attribute in your task contains a number greater than 1.

Now that you have the HITs you can download all the assignments, review them, and dispose of the HIT as follows:

```
for each in MyAssignment.get_by_hit_id(hit.hit_id):
    print('{} - {} - {}'.format(each.categories, each.notes, each.does_not_match_any))
    if is_valid_assignment(each):
        each.accept('Good job!')
    else:
        each.reject('Assignment does not follow instructions.')
hit.dispose()
```

CHAPTER 2

Installation

At the command line:

```
$ easy_install turkleton
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv turkleton
$ pip install turkleton
```

CHAPTER 3

Usage

To use turkleton in a project:

```
import turkleton
```

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/etscrivner/turkleton/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

turkleton could always use more documentation, whether as part of the official turkleton 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/etscrivner/turkleton/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 *turkleton* for local development.

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

```
$ git clone git@github.com:your_name_here/turkleton.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 turkleton
$ cd turkleton/
$ 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 turkleton 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, 3.3, and 3.4, and for PyPy. Check https://travis-ci.org/etscrivner/turkleton/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_turkleton
```

CHAPTER 5

Credits

5.1 Development Lead

- Eric Scrivner <eric.t.scrivner@gmail.com>

5.2 Contributors

None yet. Why not be the first?

CHAPTER 6

History

1.2.1 (2015-06-15)

- Bugfix, error when retrieving hits by batch id

CHAPTER 8

1.2.0 (2015-06-11)

- More answer types
- Bugfix where answers retained single value

1.1.0 (2015-06-06)

- Improvements to connection management
- More convenient syntax for uploading batches

CHAPTER 10

1.0.0 (2015-06-05)

- Major version revisions and updates

CHAPTER 11

0.1.0 (2015-01-11)

- First release on PyPI.

CHAPTER 12

Indices and tables

- genindex
- modindex
- search