Acoustic Scene Classification Documentation

Release 0.3.1

Matthieu Berjon

Sep 07, 2017

Contents

Contents:

CHAPTER 1

Acoustic Scene Classification

Development	
Last release	
PyPI status	

Acoustic Scene Auditory (ASC) using Convolutional Neural Network (CNN) is a project being part of the Machine Learning Nanodegreen program given by Udacity. For a description of the proposal, you can refer to its web version.

Dataset

The dataset can be downloaded on the Zenodo server.

Features

- TODO

Credits

Project created by Matthieu Berjon and based on the work of Simone Battaglino, Ludovick Lepauloux and Nicholas Evans.

This package was created with Cookiecutter and the audreyr/cookiecutter-pypackage project template.

CHAPTER 2

Installation

Stable release

To install Acoustic Scene Classification, run this command in your terminal:

```
$ pip install asc
```

This is the preferred method to install Acoustic Scene Classification, 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.

In order to properly work you need to ensure that libsndfile is installed such as on Debian:

```
$ sudo apt-get install libsndfile
```

From sources

The sources for Acoustic Scene Classification can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/mattberjon/asc
```

Or download the tarball:

```
$ curl  -OL https://github.com/mattberjon/asc/tarball/master
```

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

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

CHAPTER 3

Usage

To use Acoustic Scene Classification in a project:

```
import asc
```

You can as well use it through a CLI:

$ asc –help

CHAPTER 4

Configuration

Parameters	Values	Description
audio.samplerate	(int) default=44100	Samplerate of the project in Hertz
path.tmp	(str) default='/tmp'	Path to the temporary download folder
path.data	(str) default='~/'	Path to the data folder
path.url_list	(str) no default value	Path to the file containing the URLs of the data to download

CHAPTER 5

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/mattberjon/asc-cnn/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

Acoustic Scene Classification could always use more documentation, whether as part of the official Acoustic Scene Classification 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/mattberjon/asc-cnn/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 *asc* for local development.

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

```
$ git clone git@github.com:your_name_here/asc-cnn.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 asc
$ cd asc/
$ 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 asc 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/mattberjon/asc-cnn/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_asc
```

CHAPTER 6

Credits

Development Lead

- Matthieu Berjon <matthieu@berjon.net>

Contributors

None yet. Why not be the first?

CHAPTER 7

History

0.3.1 (2017-08-22)

Added

- Issue #8: config file feature
- Issue #11: Samplerate setting up through CLI
- Issue #10: Spectrogram calculation
- Issue #24: Add a 'config' subcommand to the CLI
- Issue #25: Read a value from the config file
- Issue #21: Script to verify the samplerate sanity of the database
- Issue #30: Function to transform milliseconds to samples

Changed

- Issue #23: Update of the documentation for the installation process
- Issue #26: Update of the CLI subcommand 'getdata'
- Issue #32: Exception handling in config reading

Deprecated

Nothing.

Removed

Nothing.

Fixed

- Issue #9: trailing slash on data path
- Issue #13: Update of the config file parameters
- Issue #15: Crash when setting up the samplerate
- Issue #22: Update of the dependency list
- Issue #25: Update of the config file
- Update of docstrings
- Update of the documentation

Security

- Update of the Coveralls library from 1.1 to 1.2.0

0.2.3 (2017-08-07)

Added

- Travis config file
- pytest suite
- CLI tests
- issue #5: Package coverage for the development setup
- issue #6: adding of a *clear_zip()* to clean the archive files

Changed

- Python 3.3 testing removed

Deprecated

- Nothing

Removed

- Nothing

Fixed

- Issue #4: invalid functools dependency
- Issue #7: update of docstring *unzip_data()*

Security

- Nothing

0.2.2 (2017-07-31)

Added

- nothing

Changed

- Updated of the ChangeLog (HISTORY.rst)

Deprecated

- nothing

Removed

- nothing

Fixed

- nothing

Security

- nothing

0.2.1 (2017-07-31)

Added

- nothing

Changed

- nothing

Deprecated

- nothing

Removed

- nothing

Fixed

- unzip_data() url list issue
- download of temporary files in the right directory

Security

- nothing

0.2.0 (2017-07-31)

Added

- Adding of a documentation (with docstrings)
- CLI command to download and unzip data automatically
- creation of a python package
- configuration of Tox
- download() method in data class

Changed

- Use of RST instead of markdown for all the documentation
- development packages are now in requirements_dev.txt

Deprecated

- nothing

Removed

- nothing

Fixed

- source files satisfy PEP8
- bug fix on getdata cli

Security

- Update of all packages to their latest versions

0.1.0 (2017-07-25)

- First release as a package.

CHAPTER 8

asc package

Submodules

asc.asc module

asc.cli module

asc.data module

Data module

This module lets the program to download the data from the server.

class asc.data.**Data**

Bases: object

Data collection.

clear_zip(*url_list*, *tmp_dir*)

Clear the archives

Delete the downloaded archives.

Args: url_list (str): list of strings containing the urls tmp_dir (str): path where are store the archives

download(*url_list*, *dest_dir*)

Download data from a list of URLs

Download data from a list of URLs and display a progress according to the size of the file.

Args: url_list(list): list containing URLs of files to download. dest_dir (str): path to where download the data.

file_to_list(*filename*)

Parse a file and transform it into a list

Parse a file line by line to make a list with the urls contained inside.

Args: filename (str): path to the file containing the URLs.

unzip_data(*url_list*, *origin_dir*, *dest_dir*)
Unzip data files

Unzip data files given a list of file names, the path where they are store and where they will be unzipped.

Args: url_list (str): list of strings containing the urls origin_dir (str): directory where are stored the files dest_dir (str): directory where the files will be extracted

asc.utils module

asc.utils.**conf_param_extract**(*parameter*)
Extract the section and option given the parameter.

Extract the section and option given the parameter that is in the specific format section.option

Args: parameter (str): parameter with the format 'section.option'

Returns: The section and option

asc.utils.**ms2smp**(*ms*, *sample_rate*)
Milliseconds to samples converter.

Simple converter in order to compute the number of samples for a given time frame in milliseconds and the sampling rate.

Args:

ms (int): Number of milliseconds. sample_rate (int): Sampling rate in Hertz.

Returns: Return the number of sample (forced as an int).

asc.utils.**read_config**(*section*, *option*, *config_obj*)
Look for a given option in a config file.

If exists, return the value in a config file according to the section and option.

Args: section (str): section related to the option looked for. option (str): option related to the value looked for. config_obj (obj): configparser object.

Returns: value given for a specific tuple section/option.

Todo:

- Be able to cast the data into the right type.

asc.utils.**write_config**(*section*, *option*, *data*, *config_obj*, *config_file*)
Write/Update the configuration file

Write or update the configuration file according to the section or option provided.

Args: section (str): Name of the section option (str): Name of the option data (str): data related to the option to store config_obj (str): instance of the configuration object) config_file (str): instance of the configuration file where to save

the data

Returns: None

Todo: Need to cast the object to string before saving the data.

Module contents

Top-level package for Acoustic Scene Classification.

CHAPTER 9

asc

Python Module Index

a

Index