medleydb Documentation

Release 1.3.4

Rachel Bittner

Oct 11, 2018

Contents

This library contains tools for working with the MedleyDB dataset. For more information on the MedleyDB dataset, visit http://medleydb.weebly.com/.

CHAPTER 1

Examples

1.1 Basic Usage

```
import medleydb as mdb

# Load all multitracks
mtrack_generator = mdb.load_all_multitracks()
for mtrack in mtrack_generator:
    print(mtrack.track_id)
    # do stuff

# Load specific multitracks
mtrack1 = mdb.MultiTrack('LizNelson_Rainfall')
mtrack2 = mdb.MultiTrack('Phoenix_ScotchMorris')

# Look at some attributes
mtrack1.has_bleed
> False
mtrack2.has_bleed
> True
mtrack1.artist
> LizNelson
mtrack2.artist
> Phoenix
mtrack1.is_instrumental
> False
mtrack2.is_instrumental
> True
mtrack1.stem_instruments
> ['acoustic guitar',
   'clean electric guitar',
   'female singer',
   'female singer',
   'female singer']
```

(continues on next page)

(continued from previous page)

```
mtrack1.melody1_annotation[:2]
> [[0.0, 0.0], [0.005804988662131519, 0.0]]

# Attributes of a stem
example_stem = mtrack1.stems[1]
example_stem.instrument
> 'female singer'
example_stem.f0_type
> 'm'
example_stem.pitch_annotation[:2]
> [[14.622766439, 167.49], [14.628571428, 166.969]]
example_stem.component
> 'melody'
```

1.2 Loading a subset of the dataset

```
import medleydb as mdb
track_ids = ['MusicDelta_Rock', 'MusicDelta_Reggae', 'MusicDelta_Disco']
dataset_subset = mdb.load_multitracks(track_ids)
```

1.3 Getting stems matching an instrument label

```
import medleydb as mdb

dataset = mdb.load_all_multitracks()
clarinet_files = mdb.get_files_for_instrument('clarinet')

# get all valid instrument labels
instruments = mdb.get_valid_instrument_labels()
print instruments
```

CHAPTER 2

API Reference

2.1 Multitrack

Class definitions for MedleyDB multitracks.

class medleydb.multitrack.**MultiTrack**(*track_id*)

MultiTrack Class definition.

This class loads all available metadata, annotations, and filepaths for a given multitrack directory.

Parameters

track_id [str] Track id in format 'Artist_Title'.

Examples

```
>>> mtrack = Multitrack('LizNelson_Rainfall')
>>> another_mtrack = Multitrack('ArtistName_TrackTitle')
```

Attributes

artist [str] The artist of the multitrack

title [str] The title of the multitrack

track_id [str] The unique identifier of the multitrack. In the form 'Artist_Title'

annotation_dir [str] Path to multitrack's annotation directory

audio_path [str] Path to multitrack's top level audio directory

mix_path [str] Path to multitrack's mix file.

melody_rankings [dictionary] Dictionary of melody rankings keyed by stem id

melody1_fpath [str] Path to melody 1 annotation file

melody2_fpath [str] Path to melody 2 annotation file

melody3_fpath [str] Path to melody 3 annotation file

melody_intervals_fpath [str] Path to melody intervals file

melody_rankings_fpath [str] Path to melody rankings file

activation_conf_fpath [str] Path to original activation confidence file

activation_conf_v2_fpath [str] Path to version 2 activation confidence file

source_id_fpath [str] Path to source id file

mixing_coefficients [dictionary] Dictionary of mixing weights keyed by stem id

stems [dictionary] Dictionary of stem Track objects keyed by stem id

raw_audio [dictionary] Dictionary of dictionaries keyed by stem id

stem_instruments [list] List of stem instrument labels

raw_instruments [list] List of raw audio instrument labels

duration [float or None] float: Duration of the mix

is_excerpt [bool] True if multitrack is an excerpt

has_bleed [bool] True if multitrack has bleed

is_instrumental [bool] True if multitrack is instrumental

origin [str] Origin of multitrack

genre [str] Genre of multitrack

metadata_version [str] Metadata version

has_melody [bool] True if multitrack has at least one melody stem

predominant_stem [Track or None] Track object for the predominant stem if availalbe, otherwise None

dataset_version [string] Iteration a multitrack came from. (E.g. "V1" for MedleyDB dataset_version 1, "V2" for MedleyDB dataset_version 2)

_stem_activations [np.array] Matrix of stem activations

_stem_activations_idx [dictionary] Dictionary mapping stem index to column of the stem_activations matrix

_meta_path [str] Path to metadata file.

_stem_dir_path [str] Path to multitrack's stem file directory

_raw_dir_path [str] Path to multitrack's raw file directory

_stem_fmt [str] Format of stem file basenames

_raw_fmt [str] format of raw file basenames

_metadata [dict] dictionary of data loaded from metadata file

_melody1_annotation [np.array or None] Melody 1 annotation if exists, otherwise None

_melody2_annotation [np.array or None] Melody 2 annotation if exists, otherwise None

_melody3_annotation [np.array or None] Melody 3 annotation if exists, otherwise None

Methods

activation_conf_from_stem(stem_idx[, version])	Get activation confidence from given stem.
bass_stems()	Get list of stems that contain bass.
melody_stems()	Get list of stems that contain melody.
num_raw()	Number of raw audio files.
num_stems()	Number of stems.
raw_filepaths()	Get list of filepaths to raw audio files.
stem_filepaths()	Get list of filepaths to stem files.

activation_conf_from_stem(*stem_idx*, *version=None*)
Get activation confidence from given stem.

Parameters

stem_idx [int] stem index (eg. 2 for stem S02)

version [str] If 'v2', uses the version 2 annotations. Otherwise uses acitvation annotations from the original release.

Returns

activation_confidence [list] List of time, activation confidence pairs

bass_stems()
Get list of stems that contain bass.

Returns

bass_stems: list List of Track objects where component='bass'.

duration
float: Duration of the mix

melody1_annotation
np.array: Melody 1 annotation.

melody2_annotation
np.array: Melody 2 annotation.

melody3_annotation
np.array: Melody 3 annotation.

melody_stems()
Get list of stems that contain melody.

Returns

melody_stems [list] List of Track objects where component='melody'.

num_raw()
Number of raw audio files.

Returns

n_raw [int] Number of raw audio files.

num_stems()
Number of stems.

Returns

n_stems [int] Number of stems.

raw_filepaths()
Get list of filepaths to raw audio files.

Returns

raw_fpaths [list] List of filepaths to raw audio files.

stem_activations
np.array: Matrix of stem activations

stem_activations_idx
dictionary : Dictionary mapping stem index to column of the stem_activations matrix.

stem_activations_idx_v2
dictionary : Dictionary mapping stem index to column of the stem_activations matrix. (annotations version 2)

stem_activations_v2
np.array: Matrix of stem activations (annotations version 2)

stem_filepaths()
Get list of filepaths to stem files.

Returns

stem_fpaths [list] List of filepaths to stems.

class medleydb.multitrack.**Track**(*instrument*, *audio_path*, *stem_idx*, *mix_path*, *file_id=None*, *raw_idx=None*, *component=''*, *ranking=None*, *mixing_coefficient=None*)
Track class definition. Used for stems and for raw audio tracks.

Parameters

instrument [list of str] The track's instrument label(s).

audio_path [str] Path to corresponding audio file.

stem_idx [int or str] stem index, either as int or str For ArtistName_TrackTitle_STEM_05.wav, either 5 or 'S05'

mix_path [str] Path to corresponding mix audio file.

file_id [str or None, default=None] The file's basename - e.g. Artist_Track_STEM_01

raw_idx [int str or None, default=None] Raw index, either as int or str For ArtistName_TrackTitle_RAW_05_02.wav, either 2 or 'R02'

component [str, default=''] stem's component label, if exists.

ranking [int or None, default=None] The Track's melodic ranking

mixing_coefficient [float or None, default=None] The Tracks's mixing coefficient

Attributes

instrument [str] The track's instrument label

f0_type [str]

The track's f0 type. One of

- 'm' for monophonic sources
- 'p' for polyphonic sources

- 'u' for unpitched sources

audio_path [str] Path to corresponding audio file

component [str or None] The Track's component label, if exists E.g. 'melody', 'bass'

ranking [int or None] The Track's melodic ranking, if exists

stem_idx [int] The Track's stem index

raw_idx [int or None] The Track's raw index, if exists

mixing_coefficient [float or None] The Tracks's mixing coefficient, if exists

duration [float or None] float: duration of audio file

mix_path [str] The path to the track's corresponding mix

pitch_path [str] The path to the track's pitch annotation

pitch_annotation [list or None] list: List of pairs of time (seconds), frequency (Hz)

_pitch_annotation [list or None] List of pairs of time (seconds), frequency (Hz)

duration
float: duration of audio file

pitch_annotation
list: List of pairs of time (seconds), frequency (Hz)

pitch_estimate_pyin
list: List of pairs of time (seconds), frequency (Hz)

medleydb.multitrack.**format_index**(*index*)
Load stem or raw index. Reformat if in string form.

Parameters

index [int or str] Index in string or integer form. E.g. any of 1 or 'S01' or 'R01'

Returns

formatted_index [int] Index in integer form

medleydb.multitrack.**get_dataset_version**(*track_id*)
Get the dataset version a track id appears in.

Parameters

track_id [str] Track id

Returns

dataset_version [str] The dataset version string

medleydb.multitrack.**get_dict_leaves**(*dictionary*)
Get the set of all leaves of a dictionary.

Parameters

dictionary [dict] A dictionary or nested dictionary.

Returns

vals [set] Set of leaf values.

medleydb.multitrack.**get_duration**(*wave_fpath*)
Get the duration of a wave file, in seconds.

Parameters

wave_fpath [str] Wave file.

Returns

duration [float] Duration of wave file in seconds.

`medleydb.multitrack.get_f0_type(instrument)`

Get the f0 type of an instrument.

Parameters

instrument [str] Instrument label, e.g. 'flute'

Returns

f0_type [str]

The instrument's f0 type. One of

- 'm' for monophonic sources
- 'p' for polyphonic sources
- 'u' for unpitched sources

`medleydb.multitrack.get_valid_instrument_labels(taxonomy=<Mock name='mock()' id='140297779302864'>)`

Get set of valid instrument labels based on a taxonomy.

Parameters

taxonomy_file [str, default=INST_TAXONOMY] Path to instrument taxonomy file.

Returns

valid_instrument_labels [set] Set of valid instrument labels.

Examples

```
>>> valid_labels = get_valid_instrument_labels()
>>> my_valid_labels = get_valid_instrument_labels('my_taxonomy.yaml')
```

`medleydb.multitrack.is_valid_instrument(instrument)`

Test if an instrument is valid based on a taxonomy. This is case sensitive! Taxonomy instrument labels are all lowercase.

Parameters

instrument [str] Input instrument.

Returns

value [bool] True if instrument is valid.

Examples

```
>>> is_valid_instrument('clarinet')
True
>>> is_valid_instrument('Clarinet')
False
```

(continues on next page)

(continued from previous page)

```
>>> is_valid_instrument('mayonnaise')
False
```

medleydb.multitrack.**read_annotation_file**(*fpath*, *num_cols=None*, *header=False*)

Read an annotation file. The returned annotations can be directly converted to a numpy array, if desired.

When reading files generated by Tony, set num_cols=2. Annotation files created by Tony can contain a third column that sometimes has a value (e.g [2]) and sometimes does not. It isn't important for annotation and can be ignored.

Parameters

fpath [str] Path to annotation file.

num_cols [int or None, default=None] Number of columns to read. If specified, will only read the return num_cols columns of the annotation file.

Returns

annotation [list] List of rows of the annotation file.

header [list] Header row. Empty list if header=False.

Examples

```
>>> melody_fpath = 'ArtistName_TrackTitle_MELODY1.txt'
>>> pitch_fpath = 'my_tony_pitch_annotation.csv'
>>> melody_annotation, _ = read_annotation_file(melody_fpath)
>>> activation_annotation, header = read_annotation_file(
        actvation_fpath, header=True
    )
>>> pitch_annotation, _ = read_annotation_file(pitch_fpath, num_cols=2)
```

2.2 Mix

Functions for creating new mixes from medleydb multitracks.

medleydb.mix.**mix_melody_stems**(*mtrack*, *output_path*, *max_melody_stems=None*, *include_percussion=False*, *require_mono=False*)

Creates a mix using only the stems labeled as melody.

Parameters

mtrack [Multitrack] Multitrack object

output_path [str] Path to save output wav file.

max_melody_stems [int or None, default=None] The maximum number of melody stems to mix. If None, uses the number of melody stems in the mix.

include_percussion [bool, default=False] If true, adds percussion stems to the mix.

require_mono [bool, default=False] If true, only includes melody stems that are monophonic instruments.

Returns

melody_indices [list] List of selected melody indices.

stem_indices [list] List of stem indices used in mix.

medleydb.mix.**mix_mono_stems**(*mtrack*, *output_path*, *include_percussion=False*)

Creates a mix using only the stems that are monophonic. For example, in mix with piano, voice, and clarinet, the resulting mix would include only voice and clarinet.

Parameters

mtrack [Multitrack] Multitrack object

output_path [str] Path to save output wav file.

include_percussion [bool, default=False] If true, percussive instruments are included in the mix. If false, they are excluded.

Returns

mono_indices [list] List of stem indices containing monophonic instruments.

stem_indices [list] List of stem indices used in mix.

medleydb.mix.**mix_multitrack**(*mtrack*, *output_path*, *stem_indices=None*, *alternate_weights=None*, *alternate_files=None*, *additional_files=None*)

Mix the stems of a multitrack to create a new mix. Can optionally adjust the volume of stems and replace, remove, or add stems.

Parameters

mtrack [Multitrack] Multitrack object

output_path [str] Path to save output file.

stem_indices [list or None, default=None] stem indices to include in mix. If None, mixes all stems

alternate_weights [dict or None, default=None] Dictionary with stem indices as keys and mixing coefficients as values. Stem indices present that are not in this dictionary will use the default estimated mixing coefficient.

alternate_files [dict or None, default=None] Dictionary with stem indices as keys and filepaths as values. Audio file to use in place of original stem. Stem indices present that are not in this dictionary will use the original stems.

additional_files [list of tuple or None, default=None] List of tuples of (filepath, mixing_coefficient) pairs to additionally add to final mix.

Returns

filepaths [list] List of filepaths used in the mix

weights [list] List of weights used to mix filepaths

medleydb.mix.**mix_no_vocals**(*mtrack*, *output_path*)

Remixes a multitrack with anything type of vocals removed. If no vocals are present, the mix will be a simple weighted linear remix.

Parameters

mtrack [Multitrack] Multitrack object

output_path [str] Path to save output file.

Returns

stem_indices [list] List of stem indices used in mix.

medleydb.mix.**remix_vocals**(*mtrack*, *output_path*, *vocals_scale*)
Remixes a multitrack, changing the volume of the vocals.

Parameters

mtrack [Multitrack] Multitrack object

output_path [str] Path to save output wav file.

vocals_scale [float] The target scale factor for vocals. A value of 1 keeps the volume the same. Values above 1 increase the volume and below 1 decrease it.

Returns

alternate_weights [dict] Dictionary of vocal weights keyed by vocal stem index.

2.3 Utils

Utilities to navigate MedleyDB.

medleydb.utils.**artist_conditional_split**(*trackid_list=None*, *test_size=0.15*, *num_splits=5*, *random_state=None*, *artist_index=None*)
Create artist-conditional train-test splits. The same artist (as defined by the artist_index) cannot appear in both the training and testing set.

Parameters

trackid_list [list or None, default=None] List of trackids to use in train-test split. If None, uses all tracks

test_size [float, default=0.15] Fraction of tracks to use in test set. The test set will be as close as possible in size to this value, but it may not be exact due to the artist-conditional constraint.

num_splits [int, default=5] Number of random splits to create

random_state [int or None, default=None] A random state to optionally reproduce the same random split.

artist_index [dict or None, default=None] Dictionary mapping each track id in trackid_list to a string that uniquely identifies each artist. If None, uses the predefined index ARTIST_INDEX.

Returns

splits [list of dicts] List of length num_splits of train/test split dictionaries. Each dictionary has the keys 'train' and 'test', each which map to lists of trackids.

medleydb.utils.**get_files_for_instrument**(*instrument*, *multitrack_list=None*)
Get all (stem) files for a particular instrument from the dataset.

Parameters

instrument [str] Instrument files to extract.

multitrack_list [list of MultiTrack objects or None, default=None] List of MultiTrack objects. If None, uses all multitracks.

Returns

inst_list [list] List of filepaths corresponding to instrument label.

Examples

load drum set files from the full dataset: >>> drumset_files = get_files_for_instrument('drum set')

load violin files from a subset of the dataset: >>> track_list = ['ArtistName1_TrackName1', 'ArtistName2_TrackName2', 'ArtistName3_TrackName3'] >>> multitrack_subset = load_multitracks(track_list) >>> violin_files = get_files_for_instrument(

'violin', multitrack_subset

)

medleydb.utils.**load_all_multitracks**(*dataset_version=None*)

Load all multitracks in MEDLEYDB_PATH.

Parameters

dataset_version [list or None, default=None] List of dataset version ids. If None, uses version 1.

Returns

multitracks [list] List of multitrack objects.

Examples

```
>>> multitracks = load_all_multitracks()
>>> multitracks = load_all_multitracks(dataset_version=['V1', 'V2'])
```

medleydb.utils.**load_melody_multitracks**(*dataset_version=None*)

Load all multitracks that have melody annotations.

Returns

melody_multitracks [list] List of multitrack objects.

dataset_version [list or None, default=None] List of dataset version ids. If None, uses version 1.

Examples

```
>>> melody_multitracks = load_melody_multitracks()
>>> multitracks = load_melody_multitracks(dataset_version=['V2'])
```

medleydb.utils.**load_multitracks**(*track_list*)

Load a list of multitracks.

Parameters

track_list [list] List of track ids in format 'Artist_Title'

Returns

multitracks [dict] List of multitrack objects.

Examples

```
>>> track_list = ['ArtistName1_TrackName1',                    'ArtistName2_
↪TrackName2',                    'ArtistName3_TrackName3']
>>> multitracks = load_multitracks(track_list)
```

2.4 Generate Melody Annotations

CHAPTER 3

Contribute

- Issue Tracker
- Source Code

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

m

Index

T