unsilence

Tim-Luca Lagmöller

Nov 03, 2022

CONTENTS

CHAPTER

ONE

CONTENTS

1.1 Getting started

1.1.1 Installation

To install unsilence as a command line tool, you can use pipx.

```
# Installing pipx
$ pip install pipx

# Installing Unsilence as Command Line Software
$ pipx install unsilence

# If pipx asks you to, you also need to execute the following line
# as well as close and reopen your terminal window
$ pipx ensurepath
```

If you just want to use it as a python library, you could install it using pip.

```
# Installing Unsilence as Command Line Software
$ pip install unsilence
```

To install the command line tool directly from the GitHub source, you can use this:

```
# Clone the repository (stable branch)
$ git clone -b master https://github.com/lagmoellertim/unsilence.git unsilence

# Change Directory
$ cd unsilence

# Install pip packages
$ pip install -r requirements.txt
$ pip install pipx

# If pipx asks you to, you also need to execute the following line
# as well as close and reopen your terminal window
$ pipx ensurepath

# Install unsilence package
$ pipx install .
```

To install the library from the GitHub source, you can use this:

```
# Clone the repository (stable branch)
$ git clone -b master https://github.com/lagmoellertim/unsilence.git unsilence

#Change Directory
$ cd unsilence

# Install pip packages
$ pip install -r requirements.txt

# Install unsilence package
$ python3 setup.py install
```

1.2 API Reference

This page contains auto-generated API reference documentation[1].

1.2.1 unsilence

Subpackages

unsilence.command_line

Submodules

unsilence.command_line.ChoiceDialog

Module Contents

Functions

choice_dialog(console, message[, default])	A simple yes/no console dialog option

unsilence.command_line.ChoiceDialog.**choice_dialog**(*console: rich.console.Console*, *message: str*, *default: bool = None*)

A simple yes/no console dialog option :param console: rich.Console Instance :param message: Message should be asked :param default: Default value when enter is pressed without an input (None, True, False) :return: Answer (True, False)

[1] Created with sphinx-autoapi

unsilence.command_line.EntryPoint

Module Contents

Functions

main()	Entry Point if this script is run as a script instead of a library
run()	Run the Console Interface for Unsilence

unsilence.command_line.EntryPoint.**main**()

Entry Point if this script is run as a script instead of a library :return: None

unsilence.command_line.EntryPoint.**run**()

Run the Console Interface for Unsilence :return: None

unsilence.command_line.ParseArguments

Module Contents

Functions

convert_to_path([should_exist, should_parents_exist])	Sets options for the nested class
number_bigger_than_zero(s)	Returns the Number representation of s if it is bigger than zero, else an error occurs
parse_arguments()	Parses console arguments for the Unsilence Console Interface

unsilence.command_line.ParseArguments.**convert_to_path**(*should_exist=True*, *should_parents_exist=True*)

Sets options for the nested class :param should_exist: If the file needs to exists :param should_parents_exist: If the files parents need to exist :return: Nested Handle Function

unsilence.command_line.ParseArguments.**number_bigger_than_zero**(*s*)

Returns the Number representation of s if it is bigger than zero, else an error occurs :param s: Input string :return: Integer or None

unsilence.command_line.ParseArguments.**parse_arguments**()

Parses console arguments for the Unsilence Console Interface :return: List of Console Line Arguments

unsilence.command_line.PrettyTimeEstimate

Module Contents

Functions

format_timedelta(seconds)	Generates a pretty time representation for seconds (format: hour:minute:second)
pretty_time_estimate(time_data)	Generates a rich.table.Table object from the time_data dict (from lib.Intervals.TimeCalculations.calculate_time)

unsilence.command_line.PrettyTimeEstimate.**format_timedelta**(*seconds*)

Generates a pretty time representation for seconds (format: hour:minute:second) :param seconds: Amount of seconds (can be negative) :return: String representation

unsilence.command_line.PrettyTimeEstimate.**pretty_time_estimate**(*time_data: dict*)

Generates a rich.table.Table object from the time_data dict (from lib.Intervals.TimeCalculations.calculate_time) :param time_data: time_data dict (from lib.Intervals.TimeCalculations.calculate_time) :return: rich.table.Table object

unsilence.command_line.TerminalSupport

Module Contents

Functions

repair_console()

unsilence.command_line.TerminalSupport.**repair_console**()

unsilence.lib

Subpackages

unsilence.lib.detect_silence

Submodules

unsilence.lib.detect_silence.DetectSilence

Module Contents

Functions

detect_silence(input_file, **kwargs)	Detects silence in a file and outputs the intervals (silent/not silent) as a lib.Intervals.Intervals object

unsilence.lib.detect_silence.DetectSilence.**detect_silence**(*input_file: pathlib.Path*, ***kwargs*)

Detects silence in a file and outputs the intervals (silent/not silent) as a lib.Intervals.Intervals object :param input_file: File where silence should be detected :param kwargs: Various Parameters, see below :return: lib.Intervals.Intervals object

kwargs:

silence_level: Threshold of what should be classified as silent/audible (default -35) (in dB) silence_time_threshold: Resolution of the ffmpeg detection algorithm (default 0.5) (in seconds) short_interval_threshold : The shortest allowed interval length (default: 0.3) (in seconds) stretch_time: Time the interval should be enlarged/shrunken (default 0.25) (in seconds) on_silence_detect_progress_update: Function that should be called on progress update

(called like: func(current, total))

unsilence.lib.intervals

Submodules

unsilence.lib.intervals.Interval

Module Contents

Classes

Interval	Represents a section in time where the media file is either silent or audible

class unsilence.lib.intervals.Interval.**Interval**(*start=0*, *end=0*, *is_silent=False*)

Represents a section in time where the media file is either silent or audible

Initializes an Interval object :param start: Start time of the interval in seconds :param end: End time of the interval in seconds :param is_silent: Whether the interval is silent or not

property start

Get the start time :return: start time in seconds

property end

Get the end time :return: end time in seconds

property duration

Returns the duration of the interval :return: Duration of the interval

enlarge_audible_interval(*stretch_time*, *is_start_interval=False*, *is_end_interval=False*)

Enlarges/Shrinks the audio interval, based on if it is silent or not :param stretch_time: Time the interval should be enlarged/shrunken :param is_start_interval: Whether the current interval is at the start (should not enlarge/shrink) :param is_end_interval: Whether the current interval is at the end (should not enlarge/shrink) :return: None

copy()

Creates a deep copy of this Interval :return: Interval deepcopy

serialize()

Serializes the current interval into a dict format :return: serialized dict

static deserialize(*serialized_obj: dict*)

Deserializes a previously serializes Interval and generates a new Interval with this data :param serialized_obj: previously serializes Interval (type dict) :return: Interval

__repr__()

String representation :return: String representation

unsilence.lib.intervals.Intervals

Module Contents

Classes

Intervals	Collection of lib.Intervals.Interval

class unsilence.lib.intervals.Intervals.**Intervals**(*interval_list: list = None*)

Collection of lib.Intervals.Interval

Initializes a new Interval Collection :param interval_list: list of intervals, optional

property intervals

Returns the list of intervals :return:

add_interval(*interval*)

Adds an interval to the collection :param interval: interval to be added :return: None

optimize(*short_interval_threshold=0.3*, *stretch_time=0.25*)

Optimizes the Intervals to be a better fit for media cutting :param short_interval_threshold: The shortest allowed interval length (in seconds) :param stretch_time: The time that should be added/removed from a audible/silent interval :return: None

__combine_intervals(*short_interval_threshold*)

Combines multiple intervals in order to remove intervals smaller than a threshold :param short_interval_threshold: Threshold for the shortest allowed interval :return: None

__enlarge_audible_intervals(*stretch_time*)

Enlarges/Shrinks intervals based on if they are silent or audible :param stretch_time: Time the intervals should be enlarged/shrunken :return: None

remove_short_intervals_from_start(*audible_speed=1*, *silent_speed=2*)

Removes Intervals from start that are shorter than 0.5 seconds after speedup to avoid having a final output without an audio track :param audible_speed: The speed at which the audible intervals get played back at (float) :param silent_speed: The speed at which the silent intervals get played back at (float) :return: The new, possibly shorter, Intervals object

copy()

Creates a deep copy :return: Deep copy of Intervals

serialize()

Serializes this collection :return: Serialized list

static deserialize(*serialized_obj*)

Deserializes a previously serialized object and creates a new Instance from it :param serialized_obj: Serialized list :return: New instance of Intervals

__repr__()

String representation :return: String representation

unsilence.lib.intervals.TimeCalculations

Module Contents

Functions

calculate_time(intervals, silent_speed)	audible_speed,	Generates a time estimate on the time saved if the current speed settings get applied

unsilence.lib.intervals.TimeCalculations.**calculate_time**(*intervals: unsilence.lib.intervals.Intervals.Intervals*, *audible_speed: float*, *silent_speed: float*)

Generates a time estimate on the time saved if the current speed settings get applied :param intervals: Intervals which should be estimated (lib.Intervals.Intervals) :param audible_speed: The speed at which audible intervals should be played back at :param silent_speed: The speed at which silent intervals should be played back at :return: Time calculation dict

unsilence.lib.render_media

Submodules

unsilence.lib.render_media.MediaRenderer

Module Contents

Classes

MediaRenderer	The Media Renderer handles the rendering of Intervals objects, so it processes the complete video and concatenates

class unsilence.lib.render_media.MediaRenderer.**MediaRenderer**(*temp_path: pathlib.Path*)

The Media Renderer handles the rendering of Intervals objects, so it processes the complete video and concatenates the different intervals at the end

Initializes a new MediaRenderer Object :param temp_path: The temp path where all temporary files should be stored

render(*input_file: pathlib.Path*, *output_file: pathlib.Path*, *intervals:* unsilence.lib.intervals.Intervals.Intervals, ***kwargs*)

Renders an input_file and writes the final output to output_file :param input_file: The file that should be processed :param output_file: Where the processed file should be saved :param intervals: The Intervals that should be processed :param kwargs: Keyword Args, see below :return: None

kwargs:

audio_only: Whether the output should be audio only (bool) audible_speed: The speed at which the audible intervals get played back at (float) silent_speed: The speed at which the silent intervals get played back at (float) audible_volume: The volume at which the audible intervals get played back at (float) silent_volume: The volume at which the silent intervals get played back at (float) drop_corrupted_intervals: Whether corrupted video intervals should be discarded or tried to recover (bool) threads: Number of threads to render simultaneously (int > 0) on_render_progress_update: Function that should be called on render progress update

(called like: func(current, total))

on_concat_progress_update: Function that should be called on concat progress update

(called like: func(current, total))

static __concat_intervals(*file_list: list*, *concat_file: pathlib.Path*, *output_file: pathlib.Path*, *update_concat_progress*)

Concatenates all interval files to create a finished file :param file_list: List of interval files :param concat_file: Where the ffmpeg concat filter file should be saved :param output_file: Where the final output file should be saved :param update_concat_progress: A function that is called when a step is finished

(called like function(current, total))

Returns

None

unsilence.lib.render_media.RenderIntervalThread

Module Contents

Classes

RenderIntervalThread	Worker thread that can render/process intervals based on defined options

class unsilence.lib.render_media.RenderIntervalThread.**RenderIntervalThread**(*thread_id*, *input_file: pathlib.Path*, *render_options: types.SimpleNamespace*, *task_queue: queue.Queue*, *thread_lock: threading.Lock*, ***kwargs*)

Bases: threading.Thread

Worker thread that can render/process intervals based on defined options

Initializes a new Worker (is run in daemon mode) :param thread_id: ID of this thread :param input_file: The file the worker should work on :param render_options: The parameters on how the video should be processed, more details below :param task_queue: A queue object where the worker can get more tasks :param thread_lock: A thread lock object to acquire and release thread locks :param kwargs: Keyword Args, see below for more information

run()

Start the worker. Worker runs until stop() is called. It runs in a loop, takes a new task if available, and processes it :return: None

stop()

Stops the worker after its current task is finished :return:

__render_interval(*interval_output_file: pathlib.Path*, *interval:* unsilence.lib.intervals.Interval.Interval, *apply_filter=True*, *drop_corrupted_intervals=False*, *minimum_interval_duration=0.25*)

Renders an interval with the given render options :param interval_output_file: Where the current output file should be saved :param interval: The current Interval that should be processed :param apply_filter: Whether the AV-Filter should be applied or if the media interval should be left untouched :param drop_corrupted_intervals: Whether to remove corrupted frames from the video or keep them in unedited :return: Whether it is corrupted or not

__generate_command(*interval_output_file: pathlib.Path*, *interval:* unsilence.lib.intervals.Interval.Interval, *apply_filter: bool*, *minimum_interval_duration: float*)

Generates the ffmpeg command to process the video :param interval_output_file: Where the media interval should be saved :param interval: The current interval :param apply_filter: Whether a filter should be applied or not :return: ffmpeg console command

static clamp_speed(*duration: float*, *speed: float*, *minimum_interval_duration=0.25*)

unsilence.lib.tools

Submodules

unsilence.lib.tools.ffmpeg_version

Module Contents

Functions

is_ffmpeg_usable()

unsilence.lib.tools.ffmpeg_version.**is_ffmpeg_usable**()

Submodules

unsilence.Unsilence

Module Contents

Classes

Unsilence	Unsilence Class to remove (or isolate or many other use cases) silence from audible video parts

class unsilence.Unsilence.**Unsilence**(*input_file: pathlib.Path*, *temp_dir: pathlib.Path = Path('.tmp')*)

Unsilence Class to remove (or isolate or many other use cases) silence from audible video parts

Parameters

- **input_file** (*Path*) – The file that should be processed
- **temp_dir** (*Path*) – The temp dir where temporary files can be saved

detect_silence(***kwargs*)

Detects silence of the file (Options can be specified in kwargs)

Parameters
****kwargs** – Remaining keyword arguments are passed to *detect_silence()*

Returns
A generated Intervals object

Return type
Intervals

set_intervals(*intervals:* unsilence.lib.intervals.Intervals.Intervals)

Set the intervals so that they do not need to be re-detected

Parameters
intervals (Intervals) – Intervals collection

Returns
None

get_intervals()

Get the current Intervals so they can be reused if wanted

Returns
Intervals collection

Return type
Intervals

estimate_time(*audible_speed: float = 6*, *silent_speed: float = 1*)

Estimates the time (savings) when the current options are applied to the intervals

Parameters

- **audible_speed** (*float*) – The speed at which the audible intervals get played back at
- **silent_speed** (*float*) – The speed at which the silent intervals get played back at

Raises
ValueError – If silence detection was never run

Returns
Dictionary of time information

Return type
dict

`render_media`(*output_file: pathlib.Path*, ***kwargs*)

Renders the current intervals with options specified in the kwargs

Parameters

- **output_file** (*Path*) – Where the final file should be saved at
- ****kwargs** – Remaining keyword arguments are passed to *render()*

Returns
None

`cleanup`()

Cleans up the temporary directories, called automatically when the program ends

Returns
None

`unsilence.__main__`

Package Contents

Classes

Unsilence	Unsilence Class to remove (or isolate or many other use cases) silence from audible video parts
Interval	Represents a section in time where the media file is either silent or audible
Intervals	Collection of lib.Intervals.Interval

`class unsilence.Unsilence`(*input_file: pathlib.Path*, *temp_dir: pathlib.Path = Path('.tmp')*)

Unsilence Class to remove (or isolate or many other use cases) silence from audible video parts

Parameters

- **input_file** (*Path*) – The file that should be processed
- **temp_dir** (*Path*) – The temp dir where temporary files can be saved

`detect_silence`(***kwargs*)

Detects silence of the file (Options can be specified in kwargs)

Parameters
****kwargs** – Remaining keyword arguments are passed to *detect_silence()*

Returns
A generated Intervals object

Return type
Intervals

`set_intervals`(*intervals:* unsilence.lib.intervals.Intervals.Intervals)

Set the intervals so that they do not need to be re-detected

Parameters
intervals (Intervals) – Intervals collection

Returns
None

`get_intervals`()

Get the current Intervals so they can be reused if wanted

Returns
Intervals collection

Return type
Intervals

`estimate_time`(*audible_speed: float = 6*, *silent_speed: float = 1*)

Estimates the time (savings) when the current options are applied to the intervals

Parameters

- **audible_speed** (*float*) – The speed at which the audible intervals get played back at
- **silent_speed** (*float*) – The speed at which the silent intervals get played back at

Raises
ValueError – If silence detection was never run

Returns
Dictionary of time information

Return type
dict

`render_media`(*output_file: pathlib.Path*, ***kwargs*)

Renders the current intervals with options specified in the kwargs

Parameters

- **output_file** (*Path*) – Where the final file should be saved at
- ****kwargs** – Remaining keyword arguments are passed to *render()*

Returns
None

`cleanup`()

Cleans up the temporary directories, called automatically when the program ends

Returns
None

`class unsilence.Interval`(*start=0*, *end=0*, *is_silent=False*)

Represents a section in time where the media file is either silent or audible

Initializes an Interval object :param start: Start time of the interval in seconds :param end: End time of the interval in seconds :param is_silent: Whether the interval is silent or not

`property start`

Get the start time :return: start time in seconds

property end

Get the end time :return: end time in seconds

property duration

Returns the duration of the interval :return: Duration of the interval

enlarge_audible_interval(*stretch_time*, *is_start_interval=False*, *is_end_interval=False*)

Enlarges/Shrinks the audio interval, based on if it is silent or not :param stretch_time: Time the interval should be enlarged/shrunken :param is_start_interval: Whether the current interval is at the start (should not enlarge/shrink) :param is_end_interval: Whether the current interval is at the end (should not enlarge/shrink) :return: None

copy()

Creates a deep copy of this Interval :return: Interval deepcopy

serialize()

Serializes the current interval into a dict format :return: serialized dict

static deserialize(*serialized_obj: dict*)

Deserializes a previously serializes Interval and generates a new Interval with this data :param serialized_obj: previously serializes Interval (type dict) :return: Interval

__repr__()

String representation :return: String representation

class unsilence.**Intervals**(*interval_list: list = None*)

Collection of lib.Intervals.Interval

Initializes a new Interval Collection :param interval_list: list of intervals, optional

property intervals

Returns the list of intervals :return:

add_interval(*interval*)

Adds an interval to the collection :param interval: interval to be added :return: None

optimize(*short_interval_threshold=0.3*, *stretch_time=0.25*)

Optimizes the Intervals to be a better fit for media cutting :param short_interval_threshold: The shortest allowed interval length (in seconds) :param stretch_time: The time that should be added/removed from a audible/silent interval :return: None

__combine_intervals(*short_interval_threshold*)

Combines multiple intervals in order to remove intervals smaller than a threshold :param short_interval_threshold: Threshold for the shortest allowed interval :return: None

__enlarge_audible_intervals(*stretch_time*)

Enlarges/Shrinks intervals based on if they are silent or audible :param stretch_time: Time the intervals should be enlarged/shrunken :return: None

remove_short_intervals_from_start(*audible_speed=1*, *silent_speed=2*)

Removes Intervals from start that are shorter than 0.5 seconds after speedup to avoid having a final output without an audio track :param audible_speed: The speed at which the audible intervals get played back at (float) :param silent_speed: The speed at which the silent intervals get played back at (float) :return: The new, possibly shorter, Intervals object

copy()

Creates a deep copy :return: Deep copy of Intervals

serialize()

Serializes this collection :return: Serialized list

static deserialize(*serialized_obj*)

Deserializes a previously serialized object and creates a new Instance from it :param serialized_obj: Serialized list :return: New instance of Intervals

__repr__()

String representation :return: String representation

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

CHAPTER

THREE

REFERENCES

PYTHON MODULE INDEX

u

INDEX

Symbols

A

C

D

E

F

U