subliminal Documentation

Release 0.6.4

Antoine Bertin

October 29, 2013

CONTENTS

Release v0.6.4

Subliminal is a python library to search and download subtitles.

It uses video hashes and the powerful guessit library that extracts informations from filenames or filepaths to ensure you have the best subtitles. It also relies on enzyme to detect embedded subtitles and avoid duplicates.

CHAPTER

ONE

FEATURES

Multiple subtitles services are available:

- Addic7ed
- BierDopje
- OpenSubtitles
- SubsWiki
- Subtitulos
- TheSubDB
- TvSubtitles

You can use main subliminal's functions with a **file path**, a **file name** or a **folder path**.

1.1 CLI

Download english subtitles:

```
$ subliminal -l en The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4
**************************************************
Downloaded 1 subtitle(s) for 1 video(s)
The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.srt from opensubtitles
**************************************************
```

1.2 Module

List english subtitles:

```
>>> subliminal.list_subtitles('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4', ['en'])
```

1.3 Multi-threaded use

Use 4 workers to achieve the same result:

```
>>> with subliminal.Pool(4) as p:
...     p.list_subtitles('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4', ['en'])
```

CHAPTER

TWO

USER GUIDE

This part of the documentation details how to use subliminal for most common tasks

There are 4 different ways of using subliminal and each one is described in a dedicated section below.

First, here are some basics

2.1 Basics

2.1.1 Services

You can use subliminal with multiple services to get the best result. Current available services are available in the `subliminal.SERVICES` variable.

`subliminal.`**`SERVICES`** **= ['opensubtitles', 'bierdopje', 'subswiki', 'subtitulos', 'thesubdb', 'addic7ed', 'tvsubtitles']**
list() -> new empty list list(iterable) -> new list initialized from iterable's items

2.1.2 Languages

Subliminal supports multiple languages representations based on ISO-639 and ISO-3166. Any single ISO-639 string or combination of ISO-639 and ISO-3166 is acceptable. For example, you can use `pt-br` for Portuguese (Brazil) or `en` for English.

2.1.3 Paths

All paths parameters in subliminal most commont functions can be either *a file path*, *a file name* or a *folder path*

- File path (existing): hashes of the file will be computed and used during the search for services that supports this functionnality.
- File name (or non-existing file path): the guessit python library will be used to guess informations and a text-based search will be done with services.
- Folder path (containing video files): the given folder will be searched for video files using their `MIMETYPES` and/or `EXTENSIONS`. The default maximum depth to scan is 3

2.2 CLI

Subliminal is shipped with a Command Line Interface that allows you to download subtitles for one or more videos in a multithreaded way.

Note: The cache directory defaults to *~/.config/subliminal*. Even on Windows

2.2.1 Usage

You can have the documentation of the CLI using `subliminal --help`:

```
usage: subliminal [-h] [-l LG] [-s NAME] [-m] [-f] [-w N] [-a AGE] [-c]
                  [-q | -v] [--cache-dir DIR | --no-cache-dir] [--version]
                  PATH [PATH ...]

Subtitles, faster than your thoughts

positional arguments:
  PATH                  path to video file or folder

optional arguments:
  -h, --help            show this help message and exit
  -l LG, --language LG  wanted language (ISO 639-1)
  -s NAME, --service NAME
                        service to use
  -m, --multi           download multiple subtitle languages
  -f, --force           replace existing subtitle file
  -w N, --workers N     use N threads (default: 4)
  -a AGE, --age AGE     scan only for files newer or older (prefix with +)
                        than AGE (e.g. 12h, 1w2d, +3d6h)
  -c, --compatibility   try not to use unicode (use this if you have encoding
                        errors)
  -q, --quiet           disable output
  -v, --verbose         verbose output
  --cache-dir DIR       cache directory to use
  --no-cache-dir        do not use cache directory (some services may not
                        work)
  --version             show program's version number and exit
```

2.2.2 Cron job

This CLI is well suited for automatic subtitles downloads. For example, to download english and french subtitles for videos newer than one week under /path/to/videos/ each day at 1:00AM with a single worker, you can use the following crontab line:

```
0 1 * * * user /path/to/subliminal -m -l en -l fr -w 1 -a 1w -q /path/to/videos/
```

2.3 Simple module use

Subliminal comes with two basic functions to search and download subtitles. For example, you can do:

```
>>> subliminal.list_subtitles('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4', ['en'])
```

subliminal.**list_subtitles**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*)

List subtitles in given paths according to the criteria

Parameters

- **paths** (*string or list*) – path(s) to video file or folder
- **languages** (list of `Language` or string) – languages to search for, in preferred order
- **services** (*list*) – services to use for the search, in preferred order
- **force** (*bool*) – force searching for subtitles even if some are detected
- **multi** (*bool*) – search multiple languages for the same video
- **cache_dir** (*string*) – path to the cache directory to use
- **max_depth** (*int*) – maximum depth for scanning entries
- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)

Returns found subtitles

Return type dict of `Video` => [`ResultSubtitle`]

Or even download missing subtitles for each episodes under the given folders in two different languages:

```
>>> subliminal.download_subtitles(['/mnt/videos/BBT/Season 05', '/mnt/videos/HIMYM/Season 07'],
...                               ['en', 'fr'], force=False, multi=True)
```

subliminal.**download_subtitles**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*, *order=None*)

Download subtitles in given paths according to the criteria

Parameters

- **paths** (*string or list*) – path(s) to video file or folder
- **languages** (list of `Language` or string) – languages to search for, in preferred order
- **services** (*list*) – services to use for the search, in preferred order
- **force** (*bool*) – force searching for subtitles even if some are detected
- **multi** (*bool*) – search multiple languages for the same video
- **cache_dir** (*string*) – path to the cache directory to use
- **max_depth** (*int*) – maximum depth for scanning entries
- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)
- **order** – preferred order for subtitles sorting

Returns downloaded subtitles

Return type dict of `Video` => [`ResultSubtitle`]

Note: If you use `multi=True`, `LANGUAGE_INDEX` has to be the first item of the `order` list or you might get unexpected results.

2.4 Multi-threaded module use

You can call the same functions on a `subliminal.Pool` object previously created with the appropriate number of workers.

class `subliminal.`**`Pool`**(*size*)

Pool of workers

`start`()

Start workers

`stop`()

Stop workers

`join`()

Join the task queue

`collect`()

Collect available results

Returns results of tasks

Return type list of `Task`

`list_subtitles`(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*)

See `subliminal.list_subtitles()`

`download_subtitles`(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*, *order=None*)

See `subliminal.download_subtitles()`

You have to call the `start()` method before any actions and `stop()` before exiting your program:

```
>>> p = subliminal.Pool(4)
... p.start()
... p.list_subtitles('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4', ['en'])
... p.stop()
```

To make the use of `Pool` easier, you can use the `with` statement that takes care of that for you:

```
>>> with subliminal.Pool(4) as p:
...     p.list_subtitles('The.Big.Bang.Theory.S05E18.HDTV.x264-LOL.mp4', ['en'])
```

CHAPTER

THREE

DEVELOPER GUIDE

This part of the documentation explains internal behavior of subliminal and its algorithms

This guide is going to explain the main logic of subliminal and detail every class or function.

3.1 Services

Subliminal aims at downloading subtitles. Over the web, one can find subtitles combining different websites but there is no guarantee of a perfect match. Even if OpenSubtitles has a gigantic subtitles database, you may not be able to find a subtitle on it but you will find it elsewhere, say BierDopje. Sometimes, it just takes some time before it shows up on a website even if already available on another, but you do not want to wait to watch the latest Big Bang Theory, right?

Given this, to be reliable, subliminal has to use different `services` and use a unified method to gather them all. The `ServiceBase` class will achieve this.

class `subliminal.services.`**`ServiceBase`**(*config=None*)

Service base class

Parameters **config** (`ServiceConfig`) – service configuration

`server_url` = **''**

URL to the service server

`user_agent` = **'subliminal v0.6'**

User Agent for any HTTP-based requests

`api_based` = **False**

Whether based on an API or not

`timeout` = **5**

Timeout for web requests

`languages` = **language_set([])**

`language_set` of available languages

`language_map` = **{}**

Map between language objects and language codes used in the service

`language_code` = **'alpha2'**

Default attribute of a `Language` to get with `get_code()`

`videos` = **[]**

Accepted video classes (`Episode`, `Movie`, `UnknownVideo`)

`require_video` = **False**

Whether the video has to exist or not

required_features = None
List of required features for BeautifulSoup

init()
Initialize connection

init_cache()
Initialize cache, make sure it is loaded from disk

terminate()
Terminate connection

get_code(*language*)
Get the service code for a `Language`

It uses the `language_map` and if there's no match, falls back on the `language_code` attribute of the given `Language`

get_language(*code*)
Get a `Language` from a service code

It uses the `language_map` and if there's no match, uses the given code as `language` parameter for the `Language` constructor

---

Note: A warning is emitted if the generated `Language` is "Undetermined"

---

query(**args*)
Make the actual query

list(*video*, *languages*)
List subtitles

As a service writer, you can either override this method or implement `list_checked()` instead to have the languages pre-filtered for you

list_checked(*video*, *languages*)
List subtitles without having to check parameters for validity

download(*subtitle*)
Download a subtitle

classmethod check_validity(*video*, *languages*)
Check for video and languages validity in the Service

Parameters

- **video** (`video`) – the video to check
- **languages** (`Language`) – languages to check

Return type bool

download_file(*url*, *filepath*)
Attempt to download a file and remove it in case of failure

Parameters

- **url** (*string*) – URL to download
- **filepath** (*string*) – destination path

download_zip_file(*url*, *filepath*)
Attempt to download a zip file and extract any subtitle file from it, if any. This cleans up after itself if anything fails.

Parameters

- **url** (*string*) – URL of the zip file to download
- **filepath** (*string*) – destination path for the subtitle

class `subliminal.services.`**`ServiceConfig`**(*multi=False*, *cache_dir=None*)

Configuration for any `Service`

Parameters

- **multi** (*bool*) – whether to download one subtitle per language or not
- **cache_dir** (*string*) – cache directory

3.2 Languages

To be able to support many languages, subliminal makes heavy use of ISO-3166 and ISO-639 in a dedicated module.

`subliminal.language.`**`COUNTRIES`**

ISO-3166-1 countries list from Debian package iso-codes 3.36-1. Each item of this list is a tuple like `('alpha2', 'alpha3', 'numeric', 'name')`

`subliminal.language.`**`LANGUAGES`**

ISO-639-2 languages list from the official source. Each item of this list is a tuple like `('alpha3', 'terminologic', 'alpha2', 'name', 'french_name')`

class `subliminal.language.`**`Country`**(*country*, *countries=None*)

Country according to ISO-3166

Parameters

- **country** (*string*) – country name, alpha2 code, alpha3 code or numeric code
- **countries** (see `COUNTRIES`) – all countries

class `subliminal.language.`**`Language`**(*language*, *country=None*, *languages=None*, *countries=None*, *strict=True*)

Language according to ISO-639

Parameters

- **language** (*string*) – language name (english or french), alpha2 code, alpha3 code, terminologic code or numeric code, eventually with a country
- **country** (`Country` or string) – country of the language
- **languages** (see `LANGUAGES`) – all languages
- **countries** (see `COUNTRIES`) – all countries
- **strict** (*bool*) – whether to raise a ValueError on unknown language or not

`Language` implements the inclusion test, with the `in` keyword:

```
>>> Language('pt-BR') in Language('pt')  # Portuguese (Brazil) is included in Portuguese
True
>>> Language('pt') in Language('pt-BR')  # Portuguese is not included in Portuguese (Brazil)
False
```

class `subliminal.language.`**`language_set`**(*iterable=None*, *languages=None*, *strict=True*)

Set of `Language` with some specificities.

Parameters

- **iterable** (iterable of `Languages` or string) – where to take elements from
- **languages** (see `LANGUAGES`) – all languages
- **strict** (*bool*) – whether to raise a ValueError on invalid language or not

The following redefinitions are meant to reflect the inclusion logic in `Language`

•Inclusion test, with the `in` keyword

•Intersection

•Substraction

Here is an illustration of the previous points:

```
>>> Language('en') in language_set(['en-US', 'en-CA'])
False
>>> Language('en-US') in language_set(['en', 'fr'])
True
>>> language_set(['en']) & language_set(['en-US', 'en-CA'])
language_set([Language(English, country=Canada), Language(English, country=United States)])
>>> language_set(['en-US', 'en-CA', 'fr']) - language_set(['en'])
language_set([Language(French)])
```

class `subliminal.language.`**`language_list`**(*iterable=None*, *languages=None*, *strict=True*)

List of `Language` with some specificities.

Parameters

- **iterable** (iterable of `Languages` or string) – where to take elements from
- **languages** (see `LANGUAGES`) – all languages
- **strict** (*bool*) – whether to raise a ValueError on invalid language or not

The following redefinitions are meant to reflect the inclusion logic in `Language`

•Inclusion test, with the `in` keyword

•Index

Here is an illustration of the previous points:

```
>>> Language('en') in language_list(['en-US', 'en-CA'])
False
>>> Language('en-US') in language_list(['en', 'fr-BE'])
True
>>> language_list(['en', 'fr-BE']).index(Language('en-US'))
0
```

3.3 Tasks

Subliminal is IO bound: it mostly waits for IO operations (web requests) to complete. Thus, subliminal is a good place for multi-threading. It works with atomic operations represented by a `Task` class which can be consumed with `consume_task()` but we'll see that later.

class `subliminal.tasks.`**`Task`**

Base class for tasks to use in subliminal

class `subliminal.tasks.`**`ListTask`**(*video*, *languages*, *service*, *config*)

List task used by the worker to search for subtitles

Parameters

- **video** (`Video`) – video to search subtitles for
- **languages** (*list*) – languages to search for
- **service** (*string*) – name of the service to use
- **config** (`ServiceConfig`) – configuration for the service

class `subliminal.tasks.`**`DownloadTask`**(*video*, *subtitles*)
Download task used by the worker to download subtitles

Parameters

- **video** (`Video`) – video to download subtitles for
- **subtitles** (list of `Subtitle`) – subtitles to download in order of preference

class `subliminal.tasks.`**`StopTask`**
Stop task that will stop the worker

3.4 Asynchronous

To consume those tasks in an asynchronous way without flooding services with requests, subliminal uses multiple instances of the `Worker` class that will consume the same task queue. Each worker will only create a single instance of each `service` and this save some initialization time. The `Pool` is here to instantiate and manage multiple workers at a time.

class `subliminal.async.`**`Worker`**(*tasks*, *results*)
Consume tasks and put the result in the queue

`terminate`()
Terminate instantiated services

class `subliminal.async.`**`Pool`**(*size*)
Pool of workers

`start`()
Start workers

`stop`()
Stop workers

`join`()
Join the task queue

`collect`()
Collect available results

Returns results of tasks

Return type list of `Task`

`list_subtitles`(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*)
See `subliminal.list_subtitles()`

`download_subtitles`(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*, *order=None*)
See `subliminal.download_subtitles()`

3.5 Core

The goal of subliminal's `core` module is to merge results from consumed tasks. Merging has to be intelligent and take user preferences into account. Core module is thus responsible for the computation of a `matching confidence` so the user knows the chances that the `ResultSubtitle` matches the `Video`

`subliminal.core.`**`create_list_tasks`**(*paths*, *languages*, *services*, *force*, *multi*, *cache_dir*, *max_depth*, *scan_filter*)

Create a list of `ListTask` from one or more paths using the given criteria

Parameters

- **paths** (*string or list*) – path(s) to video file or folder
- **languages** (*set*) – languages to search for
- **services** (*list*) – services to use for the search
- **force** (*bool*) – force searching for subtitles even if some are detected
- **multi** (*bool*) – search multiple languages for the same video
- **cache_dir** (*string*) – path to the cache directory to use
- **max_depth** (*int*) – maximum depth for scanning entries
- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)

Returns the created tasks

Return type list of `ListTask`

`subliminal.core.`**`create_download_tasks`**(*subtitles_by_video*, *languages*, *multi*)

Create a list of `DownloadTask` from a list results grouped by video

Parameters

- **subtitles_by_video** (dict of `Video` => [`Subtitle`]) – `ListTask` results with ordered subtitles
- **languages** (`language_list`) – languages in preferred order
- **multi** (*bool*) – download multiple languages for the same video

Returns the created tasks

Return type list of `DownloadTask`

`subliminal.core.`**`consume_task`**(*task*, *services=None*)

Consume a task. If the `services` parameter is given, the function will attempt to get the service from it. In case the service is not in `services`, it will be initialized and put in `services`

Parameters

- **task** (`ListTask` or `DownloadTask`) – task to consume
- **services** (*dict*) – mapping between the service name and an instance of this service

Returns the result of the task

Return type list of `ResultSubtitle`

`subliminal.core.`**`matching_confidence`**(*video*, *subtitle*)

Compute the probability (confidence) that the subtitle matches the video

Parameters

- **video** (`Video`) – video to match
- **subtitle** (`Subtitle`) – subtitle to match

Returns the matching probability

Return type float

`subliminal.core.`**`key_subtitles`**(*subtitle*, *video*, *languages*, *services*, *order*)

Create a key to sort subtitle using the given order

Parameters

- **subtitle** (`ResultSubtitle`) – subtitle to sort
- **video** (`Video`) – video to match
- **languages** (*list*) – languages in preferred order
- **services** (*list*) – services in preferred order
- **order** – preferred order for subtitles sorting

Returns a key ready to use for subtitles sorting

Return type int

`subliminal.core.`**`group_by_video`**(*list_results*)

Group the results of `ListTasks` into a dictionary of `Video` => `Subtitle`

Parameters **list_results** (list of result of `ListTask`) –

Returns subtitles grouped by videos

Return type dict of `Video` => [`Subtitle`]

3.6 Other objects

Subliminal uses some other self-explanatory functions and classes listed below.

3.6.1 Video

`subliminal.videos.`**`EXTENSIONS`** **= ['.avi', '.mkv', '.mpg', '.mp4', '.m4v', '.mov', '.ogm', '.ogv', '.wmv', '.divx', '.asf']**

Video extensions

`subliminal.videos.`**`MIMETYPES`** **= ['video/mpeg', 'video/mp4', 'video/quicktime', 'video/x-ms-wmv', 'video/x-msvideo', 'v**

Video mimetypes

class `subliminal.videos.`**`Video`**(*path*, *guess*, *imdbid=None*)

Base class for videos

Parameters

- **path** (*string*) – path
- **guess** (`Guess`) – guessed informations
- **imdbid** (*string*) – imdbid

classmethod **`from_path`**(*path*)

Create a `Video` subclass guessing all informations from the given path

Parameters **path** (*string*) – path

Returns video object

Return type `Episode` or `Movie` or `UnknownVideo`

exists
Whether the video exists or not

path
Path to the video

scan()
Scan and return associated subtitles

Returns associated subtitles

Return type list of `Subtitle`

class subliminal.videos.**Episode**(*path*, *series*, *season*, *episode*, *title=None*, *guess=None*, *tvdbid=None*, *imdbid=None*)
Episode `Video`

Parameters

- **path** (*string*) – path
- **series** (*string*) – series
- **season** (*int*) – season number
- **episode** (*int*) – episode number
- **title** (*string*) – title
- **guess** (`Guess`) – guessed informations
- **tvdbid** (*string*) – tvdbid
- **imdbid** (*string*) – imdbid

class subliminal.videos.**Movie**(*path*, *title*, *year=None*, *guess=None*, *imdbid=None*)
Movie `Video`

Parameters

- **path** (*string*) – path
- **title** (*string*) – title
- **year** (*int*) – year
- **guess** (`Guess`) – guessed informations
- **imdbid** (*string*) – imdbid

class subliminal.videos.**UnknownVideo**(*path*, *guess*, *imdbid=None*)
Unknown video

subliminal.videos.**scan**(*entry*, *max_depth=3*, *scan_filter=None*, *depth=0*)
Scan a path for videos and subtitles

Parameters

- **entry** (*string*) – path
- **max_depth** (*int*) – maximum folder depth
- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)

- **depth** (*int*) – starting depth

Returns found videos and subtitles

Return type list of (`Video`, [`Subtitle`])

`subliminal.videos.`**`hash_opensubtitles`**(*path*)
Compute a hash using OpenSubtitles' algorithm

Parameters **path** (*string*) – path

Returns hash

Return type string

`subliminal.videos.`**`hash_thesubdb`**(*path*)
Compute a hash using TheSubDB's algorithm

Parameters **path** (*string*) – path

Returns hash

Return type string

3.6.2 Subtitle

class `subliminal.subtitles.`**`Subtitle`**(*path*, *language*)
Base class for subtitles

Parameters

- **path** (*string*) – path to the subtitle
- **language** (`Language`) – language of the subtitle

`exists`
Whether the subtitle exists or not

class `subliminal.subtitles.`**`EmbeddedSubtitle`**(*path*, *language*, *track_id*)
Subtitle embedded in a container

Parameters

- **path** (*string*) – path to the subtitle
- **language** (`Language`) – language of the subtitle
- **track_id** (*int*) – id of the subtitle track in the container

class `subliminal.subtitles.`**`ExternalSubtitle`**(*path*, *language*)
Subtitle in a file next to the video file

classmethod **`from_path`**(*path*)
Create an `ExternalSubtitle` from path

class `subliminal.subtitles.`**`ResultSubtitle`**(*path*, *language*, *service*, *link*, *release=None*, *confidence=1*, *keywords=None*)
Subtitle found using `services`

Parameters

- **path** (*string*) – path to the subtitle
- **language** (`Language`) – language of the subtitle
- **service** (*string*) – name of the service

- **link** (*string*) – download link for the subtitle
- **release** (*string*) – release name of the video
- **confidence** (*float*) – confidence that the subtitle matches the video according to the service
- **keywords** (*set*) – keywords that describe the subtitle

single

Whether this is a single subtitle or not. A single subtitle does not have a language indicator in its file name

Return type bool

subliminal.subtitles.**get_subtitle_path**(*video_path*, *language*, *multi*)

Create the subtitle path from the given video path using language if multi

Parameters

- **video_path** (*string*) – path to the video
- **language** (`Language`) – language of the subtitle
- **multi** (*bool*) – whether to use multi language naming or not

Returns path of the subtitle

Return type string

3.6.3 Utilities

subliminal.utils.**get_keywords**(*guess*)

Retrieve keywords from guessed informations

Parameters **guess** (`guessit.guess.Guess`) – guessed informations

Returns lower case alphanumeric keywords

Return type set

subliminal.utils.**split_keyword**(*keyword*)

Split a keyword in multiple ones on any non-alphanumeric character

Parameters **keyword** (*string*) – keyword

Returns keywords

Return type set

subliminal.utils.**to_unicode**(*data*)

Convert a basestring to unicode

Parameters **data** (*basestring*) – data to decode

Returns data as unicode

Return type unicode

3.6.4 Exceptions

exception subliminal.exceptions.**Error**

Base class for exceptions in subliminal

exception subliminal.exceptions.**ServiceError**

"Exception raised by services

exception `subliminal.exceptions.`**`DownloadFailedError`**

"Exception raised when a download task has failed in service

CHAPTER

FOUR

API DOCUMENTATION

Most common subliminal features are listed here

`subliminal.`**`list_subtitles`**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*)

List subtitles in given paths according to the criteria

Parameters

- **paths** (*string or list*) – path(s) to video file or folder
- **languages** (list of `Language` or string) – languages to search for, in preferred order
- **services** (*list*) – services to use for the search, in preferred order
- **force** (*bool*) – force searching for subtitles even if some are detected
- **multi** (*bool*) – search multiple languages for the same video
- **cache_dir** (*string*) – path to the cache directory to use
- **max_depth** (*int*) – maximum depth for scanning entries
- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)

Returns found subtitles

Return type dict of `Video` => [`ResultSubtitle`]

`subliminal.`**`download_subtitles`**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*, *order=None*)

Download subtitles in given paths according to the criteria

Parameters

- **paths** (*string or list*) – path(s) to video file or folder
- **languages** (list of `Language` or string) – languages to search for, in preferred order
- **services** (*list*) – services to use for the search, in preferred order
- **force** (*bool*) – force searching for subtitles even if some are detected
- **multi** (*bool*) – search multiple languages for the same video
- **cache_dir** (*string*) – path to the cache directory to use
- **max_depth** (*int*) – maximum depth for scanning entries

- **scan_filter** (*function*) – filter function that takes a path as argument and returns a boolean indicating whether it has to be filtered out (`True`) or not (`False`)
- **order** – preferred order for subtitles sorting

Returns downloaded subtitles

Return type dict of `Video` => [`ResultSubtitle`]

Note: If you use `multi=True`, `LANGUAGE_INDEX` has to be the first item of the `order` list or you might get unexpected results.

---

class `subliminal.`**`Pool`**(*size*)

Pool of workers

`Pool.`**`collect`**()

Collect available results

Returns results of tasks

Return type list of `Task`

`Pool.`**`download_subtitles`**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*, *order=None*)

See `subliminal.download_subtitles()`

`Pool.`**`join`**()

Join the task queue

`Pool.`**`list_subtitles`**(*paths*, *languages=None*, *services=None*, *force=True*, *multi=False*, *cache_dir=None*, *max_depth=3*, *scan_filter=None*)

See `subliminal.list_subtitles()`

`Pool.`**`start`**()

Start workers

`Pool.`**`stop`**()

Stop workers

PYTHON MODULE INDEX

S