Talkey Documentation

Release 0.1.2

Nickolas Grigoriadis

January 29, 2017

Contents

Simple Text-To-Speech (TTS) interface library with multi-language and multi-engine support.

Contents:

CHAPTER 1

Getting Started

1.1 Installation

Install from pypi:

```
pip install talkey
```

1.1.1 Engines supported

By default it will try to locate and use the local instances of the following TTS engines:

- Flite
- SVOX Pico
- Festival
- eSpeak
- mbrola via eSpeak

Installing one or more of those engines should allow the libary to function and generate speech.

It also supports the following networked TTS Engines:

- MaryTTS (needs hosting)
- Google TTS (cloud hosted) Requires:

```
        pip install gTTS
```

1.2 Usage

At its simplest use case:

```
import talkey
tts = talkey.Talkey()
tts.say('Old McDonald had a farm')
```

If you get a `talkey.base.TTSError: No supported languages` error, it means that you don't have a supported TTS engine installed. Please see below.

1.2.1 Simple configuration

For best results you should configure it:

```
import talkey
tts = talkey.Talkey(
    preferred_languages = ['en', 'af', 'el', 'fr'],
    espeak = {
        'languages': {
            'en': {
                'voice': 'english-mb-en1',
                'words_per_minute': 130
            },
        }
    })
tts.say('Old McDonald had a farm')
```

1.3 Installing TTS engines

1.3.1 Ubuntu/Debian:

For festival:

```
sudo apt-get install festival
```

For flite:

```
sudo apt-get install flite
```

For SVOX Pico:

```
sudo apt-get install libttspico-utils
```

For eSpeak:

```
sudo apt-get install espeak
```

For mbrola and en1 voice (example, there are many other mbrola- packages):

```
sudo apt-get install mbrola-en1
```

1.3.2 Windows:

Install eSpeak:

> Go to http://espeak.sourceforge.net/download.html and download and install `setup_espeak-<version>.exe`

For mbrola and its voices:

> Go to http://espeak.sourceforge.net/mbrola.html and download and install `MbrolaTools<version>.exe` and follow directions to install voices from http://www.tcts.fpms.ac.be/synthesis/mbrola/mbrcopybin.html

For google TTS:

install python package gTTS

Download ffmpeg from http://ffmpeg.zeranoe.com/builds/

Extract with 7Zip, and add the \bin folder to the PATH.

e.g.: extract to C:\ffmpeg and add C:\ffmpeg\bin to the PATH

(In cmd.exe you should be able to just run ffmpeg and see it showing information, then it is working right)

CHAPTER 2

Usage

2.1 talkey module:

`talkey.`**`enumerate_engines`**`()`
Returns list of engine SLUGs in order of preference

`talkey.`**`create_engine`**`(`*engine*, *options=None*, *defaults=None*`)`
Creates an instance of an engine. There is a two-stage instantiation process with engines.

1. **`options:`** The keyword options to instantiate the engine class
2. **`defaults:`** The default configuration for the engine (options often depends on instantiated TTS engine)

class `talkey.`**`Talkey`**`(`*preferred_languages=None*, *preferred_factor=80.0*, *engine_preference=None*, ***config*`)`
Manages engines and allows multi-lingual say()

`preferred_languages` A list of languages that are weighted in preference. This is a weighting to assist the detection of language by classify().

`preferred_factor` The weighting factor to prefer the `preferred_languages` list. Higher number skews towards preference.

`engine_preference` Specify preferred engines in order of preference.

`**config` Engine-specfic configuration, e.g.:

```
        # Key is the engine SLUG, in this case ``espeak``
        espeak={
            # Specify the engine options:
            'options': {
                'enabled': True,
            },

            # Specify some default voice options
            'defaults': {
                    'words_per_minute': 150,
                    'variant': 'f4',
            },

            # Here you specify language-specific voice options
            # e.g. for english we prefer the mbrola en1 voice
            'languages': {
                'en': {
                    'voice': 'english-mb-en1',
```

```
                    'words_per_minute': 130
                },
            }
        }
```

classify(*txt*)
Classifies text by language. Uses preferred_languages weighting.

get_engine_for_lang(*lang*)
Determines the preferred engine/voice for a language.

say(*txt*, *lang=None*)
Says the text.

if `lang` is `None`, then uses `classify()` to detect language.

exception `talkey.`**TTSError**(*error*, *valid_set=None*)
The exception that Talkey will throw if any error occurs.

2.2 Engine options:

2.2.1 espeak:

class `talkey.engines.`**EspeakTTS**(***_options*)
Uses the eSpeak speech synthesizer.

Requires `espeak` and optionally `mbrola` to be available.

Initialization options:

enabled Is enabled?

type bool

default True

espeak eSpeak executable path

type exec

default ['espeak', 'c:\Program Files\eSpeak\command_line\espeak.exe']

mbrola mbrola executable path

type exec

default mbrola

mbrola_voices mbrola voices path

type str

default /usr/share/mbrola

passable_only Only allow languages of passable quality, as per http://espeak.sourceforge.net/languages.html

type bool

default True

2.2.2 festival:

class `talkey.engines.`**`FestivalTTS`**`(`***_options*`)`
Uses the festival speech synthesizer.

Requires `festival` to be available.

Initialization options:

`enabled` Is enabled?

type bool

default True

`festival` Festival executable path

type str

default festival

2.2.3 flite:

class `talkey.engines.`**`FliteTTS`**`(`***_options*`)`
Uses the flite speech synthesizer.

Requires `flite` to be available.

Initialization options:

`enabled` Is enabled?

type bool

default True

`flite` FLite executable path

type str

default flite

2.2.4 pico:

class `talkey.engines.`**`PicoTTS`**`(`***_options*`)`
Uses the svox-pico-tts speech synthesizer.

Requires `pico2wave` to be available.

Initialization options:

`enabled` Is enabled?

type bool

default True

`pico2wave` pico2wave executable path

type str

default pico2wave

2.2.5 mary:

class `talkey.engines.`**`MaryTTS`**`(`*`**_options`*`)`

Uses the MARY Text-to-Speech System (MaryTTS) MaryTTS is an open-source, multilingual Text-to-Speech Synthesis platform written in Java. Please specify your own server instead of using the demonstration server (http://mary.dfki.de:59125/) to save bandwidth and to protect your privacy.

Initialization options:

`enabled` Is enabled?

type bool

default False

`host` Mary server address

type str

default 127.0.0.1

`port` Mary server port

type int

default 59125

min 1

max 65535

`scheme` HTTP schema

type enum

default http

values http, https

2.2.6 google:

class `talkey.engines.`**`GoogleTTS`**`(`*`**_options`*`)`

Uses the Google TTS online translator.

Requires module `gTTS` to be available.

Initialization options:

`enabled` Is enabled?

type bool

default False

2.3 Voice options:

2.3.1 generic:

`language` Language of voice

`voice` Specific voice to use

2.3.2 espeak:

Config options:

pitch_adjustment pitch_adjustment option

type int

default 50

min 0

max 99

variant variant option

type enum

default m3

values , croak, f1, f2, f3, f4, f5, klatt, klatt2, klatt3, klatt4, m1, m2, m3, m4, m5, m6, m7, whisper, whisperf

words_per_minute words_per_minute option

type int

default 150

min 80

max 450

CHAPTER 3

TTS Engines

3.1 Engine interface

class talkey.base.**AbstractTTSEngine**(***_options*)
Generic parent class for all speakers

SLUG = None
The SLUG is used to identify the engine as text

classmethod _get_init_options()
AbstractMethod: Returns dict of engine options

_get_languages()
AbstractMethod: Returns dict of supported languages and voices

_get_options()
AbstractMethod: Returns dict of voice options

_is_available()
AbstractMethod: Boolean on if engine is available

_say(*phrase*, *language*, *voice*, *voiceinfo*, *options*)
AbstractMethod: Let engin actually says the phrase

Phrase The text phrase to say

Language The requested language

Voice The requested voice

Voiceinfo Data about the requested voice

Options Extra options

configure(***_options*)
Sets language-specific configuration.

Raises TTSError on error.

configure_default(***_options*)
Sets default configuration.

Raises TTSError on error.

classmethod get_init_options()
Returns a dict describing the engine options.

Uses cls._get_init_options()

get_languages()
Returns dict of supported languages and voices.

Raises TTSError if not available.

get_options()
Returns dict of voice options.

Raises TTSError if not available.

is_available()
Boolean on if engine available.

Checks if enabled, can output audio and self._is_available()

play(*filename*, *translate=False*)
Plays the sounds.

Filename The input file name

Translate If True, it runs it through audioread which will translate from common compression formats to raw WAV.

say(*phrase*, ***_options*)
Says the phrase, optionally allows to select/override any voice options.

3.2 Creating your own engine

Subclass `talkey.base.AbstractTTSEngine`, and provide the abstract methods:

```
from talkey.base import AbstractTTSEngine

class SampleTTS(AbstractTTSEngine):
    SLUG = "sample"

    @classmethod
    def _get_init_options(cls):
        # Engine options
        return {
            'enabled': {
                'description': 'Disabled by default',
                'type': 'bool',
                'default': False,
            },
        }

    def _is_available(self):
        # Checks for engine availability/readiness
        return True

    def _get_options(self):
        # Same format as _get_init_options
        # This is the voice options
        return {
            'mooing': {
                'description': 'Cows sound effect',
                'type': 'bool',
                'default': False,
            },
```

```
        }

    def _get_languages(self):
        # Dict of languages containing voices
        return {
            'en': {
                'default': 'english',
                'voices': {
                    'english': {
                        # Any extra options describing this voice
                        # (for private use)
                    },
                    'cowlish': {
                        # Any extra options describing this voice
                        # (for private use)
                    }
                }
            },
            ...
        }

    def _say(self, phrase, language, voice, voiceinfo, options):
        # Actually run the phrase through the TTS Engine.
        # All parameters will be always provided for you
        ...
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Symbols

A

C

E

F

G

I

M

P

S

T