Package Index > **SpeechRecognition** > **3.4.6**

SpeechRecognition 3.4.6

Library for performing speech recognition, with support for several engines and APIs, online and offline.

Download
SpeechRecognition-3.4.6.tar.gz

pypi v3.4.6 status stable python 2.6, 2.7, 3.3, 3.4, 3.5 license BSD

Library for performing speech recognition, with support for several engines and APIs, online and offline.

Speech recognition engine/API support:

- **CMU Sphinx** (works offline)
- Google Speech Recognition
- **Wit.ai**
- **Microsoft Bing Voice Recognition**
- **api.ai**
- **IBM Speech to Text**

Quickstart: `pip install SpeechRecognition`. See the “Installing” section for more details.

To quickly try it out, run `python -m speech_recognition` after installing.

Project links:

- **PyPI**
- **Source code**
- **Issue tracker**

Library Reference

The **library reference** documents every publicly accessible object in the library. This document is also included under `reference/library-reference.rst`.

See **Notes on using PocketSphinx** for information about installing languages, compiling PocketSphinx, and building language packs from online resources. This document is also included under `reference/pocketsphinx.rst`.

Examples

See the `examples/` **directory** in the repository root for usage examples:

- **Recognize speech input from the microphone**
- **Transcribe an audio file**
- **Save audio data to an audio file**
- **Show extended recognition results**
- **Calibrate the recognizer energy threshold for ambient noise levels** (see `recognizer_instance.energy_threshold` for details)
- **Listening to a microphone in the background**

Installing

First, make sure you have all the requirements listed in the “Requirements” section.

The easiest way to install this is using `pip install SpeechRecognition`.

Otherwise, download the source distribution from **PyPI**, and extract the archive.

In the folder, run `python setup.py install`.

Requirements

To use all of the functionality of the library, you should have:

- **Python** 2.6, 2.7, or 3.3+ (required)
- **PyAudio** 0.2.9+ (required only if you need to use microphone input, `Microphone`)
- **PocketSphinx** (required only if you need to use the Sphinx recognizer, `recognizer_instance.recognize_sphinx`)
- **FLAC encoder** (required only if the system is not x86-based Windows/Linux/OS X)

The following requirements are optional, but can improve or extend functionality in some situations:

- On Python 2, and only on Python 2, some functions (like `recognizer_instance.recognize_bing`) will run slower if you do not have **Monotonic for Python 2** installed.
- If using CMU Sphinx, you may want to **install additional language packs** to support languages like International French or Mandarin Chinese.

The following sections go over the details of each requirement.

Python

The first software requirement is **Python 2.6, 2.7, or Python 3.3+**. This is required to use the library.

PyAudio (for microphone users)

PyAudio is required if and only if you want to use microphone input (`Microphone`). PyAudio version 0.2.9+ is required, as earlier versions have overflow issues with recording on certain machines.

If not installed, everything in the library will still work, except attempting to instantiate a `Microphone` object will throw an `AttributeError`.

The installation instructions are quite good as of PyAudio v0.2.9. For convenience, they are summarized below:

- On Windows, install PyAudio using **Pip**: execute `pip install pyaudio` in a terminal.
- On Debian-derived Linux distributions (like Ubuntu and Mint), install PyAudio using **APT**: execute `sudo apt-get install python-pyaudio python3-pyaudio` in a terminal.
    - If the version in the repositories is too old, install the latest release using Pip: execute `sudo apt-get install portaudio19-dev python-all-dev python3-all-dev && sudo pip install pyaudio` (replace `pip` with `pip3` if using Python 3).
- On OS X, install PortAudio using **Homebrew**: `brew install portaudio && sudo brew link portaudio`. Then, install PyAudio using **Pip**: `pip install pyaudio`.
- On other POSIX-based systems, install the `portaudio19-dev` and `python-all-dev` (or `python3-all-dev` if using Python 3) packages (or their closest equivalents) using a package manager of your choice, and then install PyAudio using **Pip**: `pip install pyaudio` (replace `pip` with `pip3` if using Python 3).

PyAudio **wheel packages** for 64-bit Python 2.7, 3.4, and 3.5 on Windows and Linux are included for convenience, under the `third-party/` **directory** in the repository root. To install, simply run `pip install wheel` followed by `pip install ./third-party/WHEEL_FILENAME` (replace `pip` with `pip3` if using Python 3) in the repository **root directory**.

PocketSphinx-Python (for Sphinx users)

PocketSphinx-Python is **required if and only if you want to use the Sphinx recognizer** (`recognizer_instance.recognize_sphinx`).

PocketSphinx-Python **wheel packages** for 64-bit Python 2.7, 3.4, and 3.5 on Windows are included for convenience, under the `third-party/` **directory**. To install, simply run `pip install wheel` followed by `pip install ./third-party/WHEEL_FILENAME` (replace `pip` with `pip3` if using Python 3) in the SpeechRecognition folder.

On Linux and other POSIX systems (such as OS X), follow the instructions under “Building PocketSphinx-Python from source” in **Notes on using PocketSphinx** for installation instructions.

Note that the versions available in most package repositories are outdated and will not work with the bundled language data. Using the bundled wheel packages or building from source is recommended.

See **Notes on using PocketSphinx** for information about installing languages, compiling PocketSphinx, and building language packs from online resources. This document is also included under `reference/pocketsphinx.rst`.

FLAC (for some systems)

A **FLAC encoder** is required to encode the audio data to send to the API. If using Windows (x86 or x86-64), OS X (Intel Macs only, OS X 10.6 or higher), or Linux (x86 or x86-64), this is **already bundled with this library - you do not need to install anything**.

Otherwise, ensure that you have the `flac` command line tool, which is often available through the system package manager.

The included `flac-win32` executable is the **official FLAC 1.3.1 32-bit Windows binary**.

The included `flac-linux-x86` executable is built from the **FLAC 1.3.1 source code** with **Manylinux** to ensure that it’s compatible with a wide variety of distributions. The exact commands used are:

```
# download and extract the FLAC source code
wget http://downloads.xiph.org/releases/flac/flac-1.3.1.tar.xz
tar xf flac-1.3.1.tar.xz
sudo apt-get install --yes docker.io
sudo docker run --tty --interactive --rm --volume "$(pwd):/root" quay.io/pypa/manylinux1_i686:latest bash # download and start a shell inside the Many

# we're now in a Bash shell inside the Manylinux Docker image
cd /root/flac-1.3.1
./configure LDFLAGS=-static # compiler flags to make a static build
make
exit # return to the original shell
```

The resulting executable can then be found at `./flac-1.3.1/src/flac` relative to the working directory. A copy of the source code can also be found at `third-party/flac-1.3.1.tar.xz`. The build should be bit-for-bit reproducible.

The included `flac-mac` executable is extracted from **xACT 2.37**, which is a frontend for FLAC that conveniently includes binaries for all of its encoders. Specifically, it is a copy of `xACT 2.37/xACT.app/Contents/Resources/flac` in `xACT2.37.zip`.

Monotonic for Python 2 (for faster operations in some functions on Python 2)

On Python 2, and only on Python 2, if you do not install the **Monotonic for Python 2** library, some functions will run slower than they otherwise could (though everything will still work correctly).

On Python 3, that library’s functionality is built into the Python standard library, which makes it unnecessary.

This is because monotonic time is necessary to handle cache expiry properly in the face of system time changes and other time-related issues. If monotonic time functionality is not available, then things like access token requests will not be cached.

To install, use **Pip**: execute `pip install monotonic` in a terminal.

Troubleshooting

The recognizer tries to recognize speech even when I’m not speaking.

Try increasing the `recognizer_instance.energy_threshold` property. This is basically how sensitive the recognizer is to when recognition should start. Higher values mean that it will be less sensitive, which is useful if you are in a loud room.

This value depends entirely on your microphone or audio data. There is no one-size-fits-all value, but good values typically range from 50 to 4000.

The recognizer can’t recognize speech right after it starts listening for the first time.

The `recognizer_instance.energy_threshold` property is probably set to a value that is too high to start off with, and then being adjusted lower automatically by dynamic energy threshold adjustment. Before it is at a good level, the energy threshold is so high that speech is just considered ambient noise.

The solution is to decrease this threshold, or call `recognizer_instance.adjust_for_ambient_noise` beforehand, which will set the threshold to a good value automatically.

The recognizer doesn't understand my particular language/dialect.

Try setting the recognition language to your language/dialect. To do this, see the documentation for `recognizer_instance.recognize_sphinx`, `recognizer_instance.recognize_google`, `recognizer_instance.recognize_wit`, `recognizer_instance.recognize_bing`, `recognizer_instance.recognize_api`, and `recognizer_instance.recognize_ibm`.

For example, if your language/dialect is British English, it is better to use `"en-GB"` as the language rather than `"en-US"`.

The code examples throw `UnicodeEncodeError: 'ascii' codec can't encode character` when run.

When you're using Python 2, and your language uses non-ASCII characters, and the terminal or file-like object you're printing to only supports ASCII, an error is thrown when trying to write non-ASCII characters.

This is because in Python 2, `recognizer_instance.recognize_sphinx`, `recognizer_instance.recognize_google`, `recognizer_instance.recognize_wit`, `recognizer_instance.recognize_bing`, `recognizer_instance.recognize_api`, and `recognizer_instance.recognize_ibm` return unicode strings (`u"something"`) rather than byte strings (`"something"`). In Python 3, all strings are unicode strings.

To make printing of unicode strings work in Python 2 as well, replace all print statements in your code of the following form:

```
print SOME_UNICODE_STRING
```

With the following:

```
print SOME_UNICODE_STRING.encode("utf8")
```

This change, however, will prevent the code from working in Python 3.

The program doesn't run when compiled with PyInstaller.

As of PyInstaller version 3.0, SpeechRecognition is supported out of the box. If you're getting weird issues when compiling your program using PyInstaller, simply update PyInstaller.

You can easily do this by running `pip install --upgrade pyinstaller`.

On Ubuntu/Debian, I get errors like "jack server is not running or cannot be started" or "Cannot lock down [...] byte memory area (Cannot allocate memory)".

The Linux audio stack is pretty fickle. There are a few things that can cause these issues.

First, make sure JACK is installed - to install it, run `sudo apt-get install multimedia-jack`

You will then want to configure the JACK daemon correctly to avoid that "Cannot allocate memory" error. Run `sudo dpkg-reconfigure -p high jackd2` and select "Yes" to do so.

Now, you will want to make sure your current user is in the `audio` group. You can add your current user to this group by running `sudo adduser $(whoami) audio`.

Unfortunately, these changes will require you to reboot before they take effect.

After rebooting, run `pulseaudio --kill`, followed by `jack_control start`, to fix the "jack server is not running or cannot be started" error.

On Ubuntu/Debian, I get annoying output in the terminal saying things like "bt_audio_service_open: [...] Connection refused" and various others.

The "bt_audio_service_open" error means that you have a Bluetooth audio device, but as a physical device is not currently connected, we can't actually use it - if you're not using a Bluetooth microphone, then this can be safely ignored. If you are, and audio isn't working, then double check to make sure your microphone is actually connected. There does not seem to be a simple way to disable these messages.

For errors of the form "ALSA lib [...] Unknown PCM", see **this StackOverflow answer**. Basically, to get rid of an error of the form "Unknown PCM cards.pcm.rear", simply comment out `pcm.rear cards.pcm.rear` in `/usr/share/alsa/alsa.conf`, `~/.asoundrc`, and `/etc/asound.conf`.

On OS X, I get a `ChildProcessError` saying that it couldn't find the system FLAC converter, even though it's installed.

Installing **FLAC for OS X** directly from the source code will not work, since it doesn't correctly add the executables to the search path.

Installing FLAC using **Homebrew** ensures that the search path is correctly updated. First, ensure you have Homebrew, then run `brew install flac` to install the necessary files.

Developing

To hack on this library, first make sure you have all the requirements listed in the "Requirements" section.

- Most of the library code lives in `speech_recognition/__init__.py`.
- Examples live under the `examples/` **directory**, and the demo script lives in `speech_recognition/__main__.py`.
- The FLAC encoder binaries are in the `speech_recognition/` **directory**.
- Documentation can be found in the `reference/` **directory**.
- Third-party libraries, utilities, and reference material are in the `third-party/` **directory**.

To install/reinstall the library locally, run `python setup.py install` in the project **root directory**.

Releases are done by running either `build.sh` or `build.bat`. These are bash and batch scripts, respectively, that automatically build Python source packages and **Python Wheels**, then upload them to PyPI.

Features and bugfixes should be tested, at minimum, on Python 2.7 and a recent version of Python 3. It is highly recommended to test new features on Python 2.6, 2.7, 3.3, and the latest version of Python 3.

Authors

```
Uberi <azhang9@gmail.com> (Anthony Zhang)
bobsayshilol
arvindch <achembarpu@gmail.com> (Arvind Chembarpu)
kevinismith <kevin_i_smith@yahoo.com> (Kevin Smith)
haas85
DelightRun <changxu.mail@gmail.com>
maverickagm
kamushadenes <kamushadenes@hyadesinc.com> (Kamus Hadenes)
sbraden <braden.sarah@gmail.com> (Sarah Braden)
```

Please report bugs and suggestions at the **issue tracker**!

How to cite this library (APA style):

> Zhang, A. (2016). Speech Recognition (Version 3.4) [Software]. Available from **https://github.com/Uberi/speech_recognition#readme**.

How to cite this library (Chicago style):

> Zhang, Anthony. 2016. *Speech Recognition* (version 3.4).

Also check out the **Python Baidu Yuyin API**, which is based on an older version of this project, and adds support for **Baidu Yuyin**. Note that Baidu Yuyin is only available inside China.

License

Copyright 2014-2016 **Anthony Zhang (Uberi)**. The source code for this library is available online at **GitHub**.

SpeechRecognition is made available under the 3-clause BSD license. See `LICENSE.txt` in the project's **root directory** for more information.

For convenience, all the official distributions of SpeechRecognition already include a copy of the necessary copyright notices and licenses. In your project, you can simply **say that licensing information for SpeechRecognition can be found within the SpeechRecognition README, and make sure SpeechRecognition is visible to users if they wish to see it**.

SpeechRecognition distributes source code, binaries, and language files from **CMU Sphinx**. These files are BSD-licensed and redistributable as long as copyright notices are correctly retained. See `speech_recognition/pocketsphinx-data/*/LICENSE*.txt` and `third-party/LICENSE-Sphinx.txt` for license details for individual parts.

SpeechRecognition distributes source code and binaries from **PyAudio**. These files are MIT-licensed and redistributable as long as copyright notices are correctly retained. See `third-party/LICENSE-PyAudio.txt` for license details.

SpeechRecognition distributes binaries from **FLAC** - `speech_recognition/flac-win32.exe`, `speech_recognition/flac-linux-x86`, and `speech_recognition/flac-mac`. These files are GPLv2-licensed and redistributable, as long as the terms of the GPL are satisfied. The FLAC binaries are an **aggregate** of **separate programs**, so these GPL restrictions do not apply to the library or your programs that use the library, only to FLAC itself. See `LICENSE-FLAC.txt` for license details.

File	Type	Py Version	Uploaded on	Size
SpeechRecognition-3.4.6.tar.gz (**md5**, **pgp**)	Source		2016-05-22	30MB

Author: Anthony Zhang (Uberi)
Home Page: https://github.com/Uberi/speech_recognition#readme
Bug Tracker: https://github.com/Uberi/speech_recognition/issues
Keywords: speech recognition voice google wit bing api ibm
License: BSD
Categories
- **Development Status :: 5 - Production/Stable**
- **Intended Audience :: Developers**
- **License :: OSI Approved :: BSD License**
- **Natural Language :: English**
- **Operating System :: MacOS :: MacOS X**
- **Operating System :: Microsoft :: Windows**
- **Operating System :: Other OS**
- **Operating System :: POSIX :: Linux**
- **Programming Language :: Python**
- **Programming Language :: Python :: 2**
- **Programming Language :: Python :: 2.6**
- **Programming Language :: Python :: 2.7**
- **Programming Language :: Python :: 3**
- **Programming Language :: Python :: 3.3**
- **Programming Language :: Python :: 3.4**
- **Programming Language :: Python :: 3.5**
- **Topic :: Multimedia :: Sound/Audio :: Speech**
- **Topic :: Software Development :: Libraries :: Python Modules**

Package Index Owner: Anthony.Zhang
DOAP record: SpeechRecognition-3.4.6.xml