

---

## aeneas

**aeneas** is a Python/C library and a set of tools to automagically synchronize audio and text (aka forced alignment).

- Version: 1.7.3
- Date: 2017-03-15
- Developed by: ReadBeyond
- Lead Developer: Alberto Pettarin
- License: the GNU Affero General Public License Version 3 (AGPL v3)
- Contact: [aeneas@readbeyond.it](mailto:aeneas@readbeyond.it)
- Quick Links: [Home](#) - [GitHub](#) - [PyPI](#) - [Docs](#) - [Tutorial](#) - [Benchmark](#) - [Mailing List](#) - [Web App](#)

### Goal

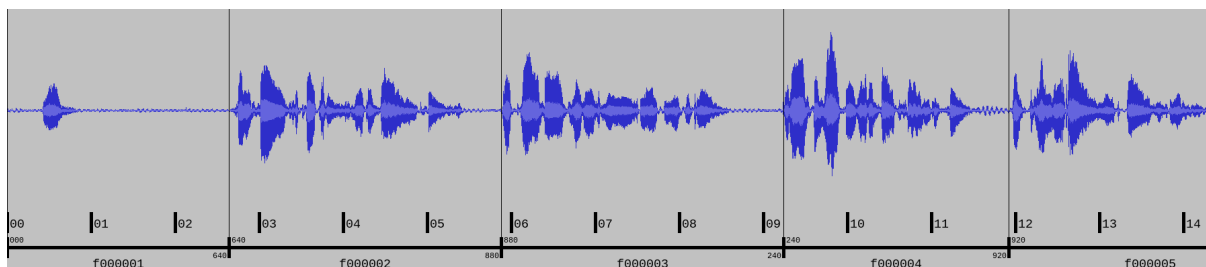
**aeneas** automatically generates a **synchronization map** between a list of text fragments and an audio file containing the narration of the text. In computer science this task is known as (automatically computing a) **forced alignment**.

For example, given this text file and this audio file, **aeneas** determines, for each fragment, the corresponding time interval in the audio file:

1	1		=> [00:00:00.000,
		00:00:02.640]	
2	From fairest creatures we desire increase,		=> [00:00:02.640,
		00:00:05.880]	
3	That thereby beauty's rose might never die,		=> [00:00:05.880,
		00:00:09.240]	
4	But as the ripper should by time decease,		=> [00:00:09.240,
		00:00:11.920]	
5	His tender heir might bear his memory:		=> [00:00:11.920,
		00:00:15.280]	
6	But thou contracted to thine own bright eyes,		=> [00:00:15.280,
		00:00:18.800]	
7	Feed'st thy light's flame with self-substantial fuel,		=> [00:00:18.800,
		00:00:22.760]	
8	Making a famine where abundance lies,		=> [00:00:22.760,
		00:00:25.680]	
9	Thy self thy foe, to thy sweet self too cruel:		=> [00:00:25.680,
		00:00:31.240]	
10	Thou that art now the world's fresh ornament,		=> [00:00:31.240,
		00:00:34.400]	
11	And only herald to the gaudy spring,		=> [00:00:34.400,
		00:00:36.920]	

---

12	Within thine own bud buriest thy content,	=> [00:00:36.920,
	00:00:40.640]	
13	And tender churl mak'st waste in niggarding:	=> [00:00:40.640,
	00:00:43.640]	
14	Pity the world, or else this glutton be,	=> [00:00:43.640,
	00:00:48.080]	
15	To eat the world's due, by the grave and thee.	=> [00:00:48.080,
	00:00:53.240]	



This synchronization map can be output to file in several formats, depending on its application:

- research: Audacity (AUD), ELAN (EAF), TextGrid;
- digital publishing: SMIL for EPUB 3;
- closed captioning: SubRip (SRT), SubViewer (SBV/SUB), TTML, WebVTT (VTT);
- Web: JSON;
- further processing: CSV, SSV, TSV, TXT, XML.

## System Requirements, Supported Platforms and Installation

### System Requirements

1. a reasonably recent machine (recommended 4 GB RAM, 2 GHz 64bit CPU)
2. Python 2.7 (Linux, OS X, Windows) or 3.5 or later (Linux, OS X)
3. FFmpeg
4. eSpeak
5. Python packages [BeautifulSoup4](#), [lxml](#), and [numpy](#)
6. Python headers to compile the Python C/C++ extensions (optional but strongly recommended)
7. A shell supporting UTF-8 (optional but strongly recommended)

### Supported Platforms

**aeneas** has been developed and tested on **Debian 64bit**, with **Python 2.7** and **Python 3.5**, which are the **only supported platforms** at the moment. Nevertheless, **aeneas** has been confirmed to work on other Linux distributions, Mac OS X, and Windows. See the PLATFORMS file for details.

---

If installing **aeneas** natively on your OS proves difficult, you are strongly encouraged to use **aeneas-vagrant**, which provides **aeneas** inside a virtualized Debian image running under VirtualBox and Vagrant, which can be installed on any modern OS (Linux, Mac OS X, Windows).

## Installation

All-in-one installers are available for Mac OS X and Windows, and a Bash script for deb-based Linux distributions (Debian, Ubuntu) is provided in this repository. It is also possible to download a VirtualBox+Vagrant virtual machine. Please see the INSTALL file for detailed, step-by-step installation procedures for different operating systems.

The generic OS-independent procedure is simple:

1. **Install** Python (2.7.x preferred), FFmpeg, and eSpeak
2. Make sure the following **executables** can be called from your **shell**: `espeak`, `ffmpeg`, `ffprobe`, `pip`, and `python`
3. First install `numpy` with `pip` and then `aeneas` (this order is important):

```
1 pip install numpy
2 pip install aeneas
```

4. To **check** whether you installed **aeneas** correctly, run:

```
1 python -m aeneas.diagnostics
```

## Usage

1. Run without arguments to get the **usage message**:

```
1 python -m aeneas.tools.execute_task
2 python -m aeneas.tools.execute_job
```

You can also get a list of **live examples** that you can immediately run on your machine thanks to the included files:

```
1 python -m aeneas.tools.execute_task --examples
2 python -m aeneas.tools.execute_task --examples-all
```

2. To **compute a synchronization map** `map.json` for a pair (`audio.mp3`, `text.txt` in plain text format), you can run:

```
1 python -m aeneas.tools.execute_task \
2     audio.mp3 \
```

---

```
3 text.txt \  
4 "task_language=eng|os_task_file_format=json|is_text_type=plain  
  "\   
5 map.json
```

(The command has been split into lines with `\` for visual clarity; in production you can have the entire command on a single line and/or you can use shell variables.)

To **compute a synchronization map** `map.smil` for a pair (`audio.mp3`, `page.xhtml` containing fragments marked by `id` attributes like `f001`), you can run:

```
1 python -m aeneas.tools.execute_task \  
2 audio.mp3 \  
3 page.xhtml \  
4 "task_language=eng|os_task_file_format=smil|  
  os_task_file_smil_audio_ref=audio.mp3|  
  os_task_file_smil_page_ref=page.xhtml|is_text_type=unparsed  
  |is_text_unparsed_id_regex=f[0-9]+|is_text_unparsed_id_sort  
  =numeric" \  
5 map.smil
```

As you can see, the third argument (the *configuration string*) specifies the parameters controlling the I/O formats and the processing options for the task. Consult the documentation for details.

3. If you have several tasks to process, you can create a **job container** to batch process them:

```
1 python -m aeneas.tools.execute_job job.zip output_directory
```

File `job.zip` should contain a `config.txt` or `config.xml` configuration file, providing **aeneas** with all the information needed to parse the input assets and format the output sync map files. Consult the documentation for details.

The documentation contains a highly suggested tutorial which explains how to use the built-in command line tools.

## Documentation and Support

- Documentation: <http://www.readbeyond.it/aeneas/docs/>
- Command line tools tutorial: <http://www.readbeyond.it/aeneas/docs/clitutorial.html>
- Library tutorial: <http://www.readbeyond.it/aeneas/docs/libtutorial.html>
- Old, verbose tutorial: A Practical Introduction To The aeneas Package
- Mailing list: <https://groups.google.com/d/forum/aeneas-forced-alignment>
- Changelog: <http://www.readbeyond.it/aeneas/docs/changelog.html>
- High level description of how aeneas works: HOWITWORKS

- 
- Development history: HISTORY
  - Testing: TESTING
  - Benchmark suite: <https://readbeyond.github.io/aeneas-benchmark/>

## Supported Features

- Input text files in `parsed`, `plain`, `subtitles`, or `unparsed` (XML) format
- Multilevel input text files in `mplain` and `munparsed` (XML) format
- Text extraction from XML (e.g., XHTML) files using `id` and `class` attributes
- Arbitrary text fragment granularity (single word, subphrase, phrase, paragraph, etc.)
- Input audio file formats: all those readable by `ffmpeg`
- Output sync map formats: AUD, CSV, EAF, JSON, SMIL, SRT, SSV, SUB, TEXTGRID, TSV, TTML, TXT, VTT, XML
- Confirmed working on 38 languages: AFR, ARA, BUL, CAT, CYM, CES, DAN, DEU, ELL, ENG, EPO, EST, FAS, FIN, FRA, GLE, GRC, HRV, HUN, ISL, ITA, JPN, LAT, LAV, LIT, NLD, NOR, RON, RUS, POL, POR, SLK, SPA, SRP, SWA, SWE, TUR, UKR
- MFCC and DTW computed via Python C extensions to reduce the processing time
- Several built-in TTS engine wrappers: AWS Polly TTS API, eSpeak (default), eSpeak-ng, Festival, MacOS (via say), Nuance TTS API
- Default TTS (eSpeak) called via a Python C extension for fast audio synthesis
- Possibility of running a custom, user-provided TTS engine Python wrapper (e.g., included example for speect)
- Batch processing of multiple audio/text pairs
- Download audio from a YouTube video
- In multilevel mode, recursive alignment from paragraph to sentence to word level
- In multilevel mode, MFCC resolution, MFCC masking, DTW margin, and TTS engine can be specified for each level independently
- Robust against misspelled/mispronounced words, local rearrangements of words, background noise/sporadic spikes
- Adjustable splitting times, including a max character/second constraint for CC applications
- Automated detection of audio head/tail
- Output an HTML file for fine tuning the sync map manually (`finetuneas` project)
- Execution parameters tunable at runtime
- Code suitable for Web app deployment (e.g., on-demand cloud computing instances)
- Extensive test suite including 1,200+ unit/integration/performance tests, that run and must pass before each release

---

## Limitations and Missing Features

- Audio should match the text: large portions of spurious text or audio might produce a wrong sync map
- Audio is assumed to be spoken: not suitable for song captioning, YMMV for CC applications
- No protection against memory swapping: be sure your amount of RAM is adequate for the maximum duration of a single audio file (e.g., 4 GB RAM => max 2h audio; 16 GB RAM => max 10h audio)
- Open issues

## A Note on Word-Level Alignment

A significant number of users runs **aeneas** to align audio and text at word-level (i.e., each fragment is a word). Although **aeneas** was not designed with word-level alignment in mind and the results might be inferior to ASR-based forced aligners for languages with good ASR models, **aeneas** offers some options to improve the quality of the alignment at word-level:

- multilevel text (since v1.5.1),
- MFCC nonspeech masking (since v1.7.0, disabled by default),
- use better TTS engines, like Festival or AWS/Nuance TTS API (since v1.5.0).

If you use the `aeneas.tools.execute_task` command line tool, you can add `--presets-word` switch to enable MFCC nonspeech masking, for example:

```
1 $ python -m aeneas.tools.execute_task --example-words --presets-word
2 $ python -m aeneas.tools.execute_task --example-words-multilevel --
  presets-word
```

If you use **aeneas** as a library, just set the appropriate `RuntimeConfiguration` parameters. Please see the command line tutorial for details.

## License

**aeneas** is released under the terms of the GNU Affero General Public License Version 3. See the LICENSE file for details.

Licenses for third party code and files included in **aeneas** can be found in the licenses directory.

No copy rights were harmed in the making of this project.

---

## Supporting and Contributing

### Sponsors

- **July 2015:** Michele Gianella generously supported the development of the boundary adjustment code (v1.0.4)
- **August 2015:** Michele Gianella partially sponsored the port of the MFCC/DTW code to C (v1.1.0)
- **September 2015:** friends in West Africa partially sponsored the development of the head/tail detection code (v1.2.0)
- **October 2015:** an anonymous donation sponsored the development of the “YouTube downloader” option (v1.3.0)
- **April 2016:** the Fruch Foundation kindly sponsored the development and documentation of v1.5.0
- **December 2016:** the Centro Internazionale Del Libro Parlato “Adriano Sernagiotto” (Feltre, Italy) partially sponsored the development of the v1.7 series

### Supporting

Would you like supporting the development of **aeneas**?

I accept sponsorships to

- fix bugs,
- add new features,
- improve the quality and the performance of the code,
- port the code to other languages/platforms, and
- improve the documentation.

Feel free to get in touch.

### Contributing

If you think you found a bug or you have a feature request, please use the GitHub issue tracker to submit it.

If you want to ask a question about using **aeneas**, your best option consists in sending an email to the mailing list.

Finally, code contributions are welcome! Please refer to the Code Contribution Guide for details about the branch policies and the code style to follow.

---

## Acknowledgments

Many thanks to **Nicola Montecchio**, who suggested using MFCCs and DTW, and co-developed the first experimental code for aligning audio and text.

**Paolo Bertasi**, who developed the APIs and Web application for ReadBeyond Sync, helped shaping the structure of this package for its asynchronous usage.

**Chris Hubbard** prepared the files for packaging aeneas as a Debian/Ubuntu `.deb`.

**Daniel Bair** prepared the `brew` formula for installing **aeneas** and its dependencies on Mac OS X.

**Daniel Bair**, **Chris Hubbard**, and **Richard Margetts** packaged the installers for Mac OS X and Windows.

**Firat Ozdemir** contributed the `finetuneas` HTML/JS code for fine tuning sync maps in the browser.

**Willem van der Walt** contributed the code snippet to output a sync map in TextGrid format.

**Chris Vaughn** contributed the MacOS TTS wrapper.

All the mighty GitHub contributors, and the members of the Google Group.