faseAlign Documentation

Release 1.1.14

Eric Wilbanks

Oct 14, 2022

Contents:

1 Features **3**

2 Citation **5**

3 Contribute **7**

4 Contact **9**

4.1 Installation 9
4.2 Preparing your Data 15
4.3 Aligning 18
4.4 Model Development 20
4.5 Changelog 21

faseAlign is a Python-based suite for doing automatic forced alignment of text to audio for Spanish data using the HTK Speech Recognition Toolkit. The included acoustic models and dictionary are appropriate for many varieties of Latin American Spanish (see development for more details).

CHAPTER 1

Features

- force-aligned .TextGrid output files
- support for .txt or .TextGrid input transcriptions
- custom dictionaries
- support for stereo audio
- automatic phonemicization of unknown words
- automatic syllabification and tonicity

CHAPTER 2

Citation

Wilbanks, E. (2022). faseAlign (Version 1.1.14) [Computer software]. Retrieved Oct 14, 2022 from https://github.com/EricWilbanks/faseAlign.

CHAPTER 3

Contribute

- Issue Tracker: https://github.com/EricWilbanks/faseAlign/issues
- Source Code: https://github.com/EricWilbanks/faseAlign

CHAPTER 4

Contact

If you're experiencing issues or have questions about the usage or installation of faseAlign, please feel free to contact me at wilbanks.ericw@gmail.com

4.1 Installation

4.1.1 Build from Source

Note: The easiest installation uses conda environments for *Linux Installation* or *macOS Installation*.

Another option is to use a pre-compiled virtual machine image: *Virtual Machine Installation* which has much of the required software already included. This option however takes up a large (~14GB) amount of disk space. This is the only supported installation options for Windows users.

Linux Installation

Downloading HTK

The HTK Toolkit is required to perform the backend acoustic modeling and alignment. Because of license requirements, HTK cannot be distributed with other software, but it is free to download for individual users.

First, register on the HTK website

Then, download the HTK source code. faseAlign was developed using the stable release 3.4.1 of HTK.

Compiling HTK

Once the zipped source code has been downloaded. Navigate to the downloaded file and execute the following command to unpack it:

```
tar -xvzf HTK-3.4.1.tar.gz
```

Now move into the newly created *htk* directory:

```
cd htk
```

Finally, execute the following lines of code to compile and install HTK:

```
export CPPFLAGS=-UPHNALG
./configure --disable-hlmtools --disable-hslab --without-x
make all
sudo make install
```

If your installation was successful, the following command should print out the version information for the HTK toolkit:

```
HVite -V
```

Making sure you have miniconda

We'll be using conda to build an environment for faseAlign. Follow the instructions on the miniconda linux installation page and ensure that you're following the instructions for *Miniconda - Python 3.9*.

Build the Conda env

We'll be using conda to manage our python packages and prevent other version issues. For more details on conda usage, please consult the conda documentation.

To build the conda environment, we'll need to download the environment.yml file, either manually or with the following command:

```
wget https://raw.githubusercontent.com/EricWilbanks/faseAlign/master/environment.yml
```

Next, we'll build the environment with conda.

```
conda env create -f environment.yml
```

You should receive a success message along the lines of "Done. To activate this environment..."

Correctly Configure UTF-8

At this point you need to ensure that accented (UTF-8) characters are correctly interpreted. To do so, enter the following to the terminal:

```
echo export LC_ALL=en_US.UTF-8 >> ~/.bashrc
echo export LC_ALL=en_US.UTF-8 >> ~/.profile
echo export LANG=en_US.UTF-8 >> ~/.bashrc
echo export LANG=en_US.UTF-8 >> ~/.profile
echo export LANGUAGE=en_US.UTF-8 >> ~/.bashrc
echo export LANGUAGE=en_US.UTF-8 >> ~/.profile

source ~/.bashrc
```

Activating the fase environment

In order to use use faseAlign on the command line, we'll now have to activate the conda environment we've just built. You have to activate this environment each time you restart the session, using the following code:

```
conda activate fase
```

You should now see a *(fase)* to the left of your command line, indicating that the environment is active. To deactivate the environment, use the following:

```
conda deactivate
```

For more details on conda environment usage, please consult the conda documentation.

macOS Installation

Xcode Compiler

First we have to make sure that Xcode (and included GCC compiler) are installed. Open the terminal application and call the following command:

```
xcode-select -p
```

If this command is not successful, install Xcode through the terminal:

```
xcode-select --install
```

And select "Install"

Homebrew Installation

Next, we need a package manager. Install Homebrew through the terminal:

```
ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/
↪install)"
```

Downloading HTK

The HTK Toolkit is required to perform the backend acoustic modeling and alignment. Because of license requirements, HTK cannot be distributed with other software, but it is free to download for individual users.

First, register on the HTK website

Then, download the HTK source code. faseAlign was developed using the stable release 3.4.1 of HTK.

Compiling HTK

Once the zipped source code has been downloaded. Navigate to the downloaded file and execute the following command to unpack it:

```
tar -xvzf HTK-3.4.1.tar.gz
```

Now move into the newly created *htk* directory:

```
cd htk
```

Finally, execute the following lines of code to compile and install HTK:

```
export CPPFLAGS=-UPHNALG
./configure --disable-hlmtools --disable-hslab --without-x
make all
sudo make install
```

If your installation was successful, the following command should print out the version information for the HTK toolkit:

```
HVite -V
```

Making sure you have miniconda

We'll be using conda to build an environment for faseAlign. Follow the instructions on the miniconda macOS installation page and ensure that you're following the instructions for *Miniconda - Python 3.9*.

Build the Conda env

We'll be using conda to manage our python packages and prevent other version issues. For more details on conda usage, please consult the conda documentation.

To build the conda environment, we'll need to download the environment.yml file, either manually or with the following command:

```
wget https://raw.githubusercontent.com/EricWilbanks/faseAlign/master/environment.yml
```

Next, we'll build the environment with conda.

```
conda env create -f environment.yml
```

You should receive a success message along the lines of "Done. To activate this environment..."

Correctly Configure UTF-8

At this point you need to ensure that accented (UTF-8) characters are correctly interpreted. To do so, enter the following to the terminal:

```
echo export LC_ALL=en_US.UTF-8 >> ~/.bashrc
echo export LC_ALL=en_US.UTF-8 >> ~/.profile
echo export LANG=en_US.UTF-8 >> ~/.bashrc
echo export LANG=en_US.UTF-8 >> ~/.profile
echo export LANGUAGE=en_US.UTF-8 >> ~/.bashrc
echo export LANGUAGE=en_US.UTF-8 >> ~/.profile

source ~/.bashrc
```

Activating the fase environment

In order to use use faseAlign on the command line, we'll now have to activate the conda environment we've just built. You have to activate this environment each time you restart the session, using the following code:

```
conda activate fase
```

You should now see a *(fase)* to the left of your command line, indicating that the environment is active. To deactivate the environment, use the following:

```
conda deactivate
```

For more details on conda environment usage, please consult the conda documentation.

4.1.2 Virtual Machine Installation

Download BPM Image

Another way to run faseAlign on your own computer is to use a virtual machine image with the required software already included. The image we'll be using is based on the Berkeley Phonetics Machine (BPM), in turn derived from the Berkeley Common Environment (BCE).

Download the faseAlign OVA image.

Set Up Virtual Machine

First, you'll need to install VirtualBox in able to be able to run virtual machines on your computer.

Run VirtualBox and select *File, Import Appliance...* and select the .ova file you downloaded earlier. Then adjust the amount of RAM the virtual machine is allotted (defaults are typically fine).

You can now start the virtual machine by clicking the green "Show" arrow.

Warning: If you're on a Windows computer and run into errors at this step, virtualization might be disabled on your computer.

Follow the BCE Guide to Enabling Virtualization in order to enable virtualization. This involves editing BIOS settings, so if you don't feel comfortable making such changes consult with someone who does.

Note: The files on your virtual machine and the files on your host machine are completely separated and do not interact.

If you want to transfer files between the two you need to set up a *Shared Folder*.

- Create a folder on your main computer wherever you like
- Open up the VirtualBox application and select the virtual machine you want to add the folder to and go to *Settings*
- Go to the *Shared Folders* tab and click the *Adds new Shared Folder* button.
- Select the Folder you want to share and make sure you select *Auto-Mount*
- The next time you start up the virtual machine you should find the shared folder at /media/sf_myfoldername/ or ~/Desktop/Shared/sf_myfoldername

---

Install HTK

The HTK Toolkit is required to perform the backend acoustic modeling and alignment. Because of license requirements, HTK cannot be distributed with other software, but it is free to download for individual users.

First, register on the HTK website.

Next, open the terminal application and install/download HTK with the following code:

```
sudo bpm-update htk
```

You will be prompted for the username and password from your HTK registration.

Build the Conda env

We'll be using conda to manage our python packages and prevent other version issues. For more details on conda usage, please consult the conda documentation.

To build the conda environment, we'll need to download the environment.yml file, either manually or with the following command:

```
wget https://raw.githubusercontent.com/EricWilbanks/faseAlign/master/environment.yml
```

Next, we'll build the environment with conda.

```
conda env create -f environment.yml
```

You should receive a success message along the lines of "Done. To activate this environment..."

Correctly Configure UTF-8

At this point you need to ensure that accented (UTF-8) characters are correctly interpreted. To do so, enter the following to the terminal:

```
echo export LC_ALL=en_US.UTF-8 >> ~/.bashrc
echo export LC_ALL=en_US.UTF-8 >> ~/.profile
echo export LANG=en_US.UTF-8 >> ~/.bashrc
echo export LANG=en_US.UTF-8 >> ~/.profile
echo export LANGUAGE=en_US.UTF-8 >> ~/.bashrc
echo export LANGUAGE=en_US.UTF-8 >> ~/.profile

source ~/.bashrc
```

Activating the fase environment

In order to use use faseAlign on the command line, we'll now have to activate the conda environment we've just built. You have to activate this environment each time you restart the session, using the following code:

```
conda activate fase
```

You should now see a *(fase)* to the left of your command line, indicating that the environment is active. To deactivate the environment, use the following:

```
conda deactivate
```

For more details on conda environment usage, please consult the conda documentation.

Windows Installation

Not supported. Please use the *Virtual Machine Installation*.

4.2 Preparing your Data

4.2.1 Transcription Files

The input to the force aligner is a wav file and the orthographic transcription associated with it. Transcriptions can be in either .TextGrid or .txt formats.

.TextGrid transcription

It is **strongly suggested** to use transcriptions in Praat TextGrid format as in the above example. This approach has many benefits over the .txt transcription approach:

1. *Each speaker on a separate tier*: more accurate alignments, especially in overlap or noisy conditions

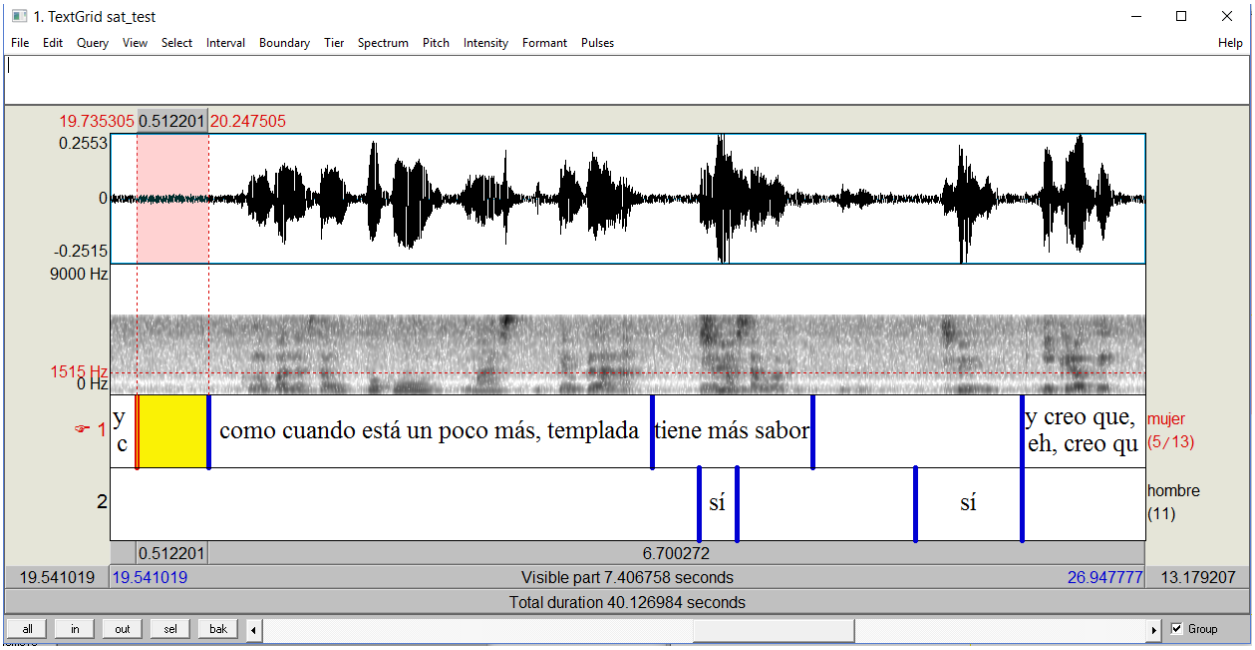

Fig. 1: *Example TextGrid Transcription*

2. *Exclusion of non-transcribed sections*: the aligner will only align where there is transcribed speech, this is helpful for ignoring speech not of interest to the analysis.
3. *Aligning in small intervals*: each interval is aligned separately, increasing accuracy.
4. *Definition of speaker labels through tier names*

.txt transcriptions

```
{S1} pues yo como aquí, comida y o o bebes té y es más ma volcánico {LG} no puedo␣
↪comer o o tomar el té durante quince minutos porque, está muy caliente
{S2} sí
{S1} y cuando la comida está muy caliente, en mi opinión, no tiene tanto sabor
{S2} sí
{S1} como cuando está un poco más, templada, tiene más sabor
{S2} sí
{S1} tiene más sabor
{S2} sí
{S1} y creo que, eh, creo que es porque en España, una cosa que he notado es que,␣
↪normalmente las familias esperan hasta que todo el mundo esté sentado a la mesa,
{S2} sí
{S1} para comer
{S2} mhm
```

faseAlign also supports transcriptions in .txt format as in the above example. This approach has the benefit of not needing any determination of turn boundaries, but suffers from slightly poorer alignments. This is due to the fact that the entire wav file is aligned to the entire transcription. In this method, speakers cannot overlap in the output alignment (since the file is processed in one linear chunk).

Speakers are still separated into different tiers following alignment, through the use of speaker labels (e.g., {Julia}, {Speaker1}, {Marcos}) which mark utterances belonging to that speaker.

Note: Make sure to use the same speaker tags throughout the transcription. *{Julia}* and *{JULIA}* would be treated as two separate speakers.

4.2.2 Transcription Best Practices

Regardless of transcription method chosen, the process of transcribing is similar; you'll want to pay attention to these key points:

1. **Be as Accurate as Possible!** - This includes transcribing false starts, repetitions, omissions, etc as accurately as possible. The aligner can't find a word if it's not in the transcription!
2. **Transcribe in Breath Groups** - While this is most helpful for the TextGrid transcriptions, I also find it helpful for the txt ones as well.
3. **Spell out all accents and tildes** - otherwise words might be not be found in the dictionary
4. **Spell out all numbers and dates** - ("El veinte de marzo" instead of "el 20 de Mar.")

Additionally, the following can be used to note various non-speech items:

- {BR} - breath
- {CG} - cough
- {SIL} - silence longer than ~ 2 seconds
- {LG} - laughter
- {NS} - random noise

4.2.3 How to Add Missing Words

You'll often run into words in your transcription that are not included in the dictionary; these may include place names, speech errors, novel words, etc. as decribed in *Missing Words During Alignment*.

If you want to override the automatic phonemicization (often appropriate for words not following native Spanish orthography to phonemes), add each of the missing words to a txt file with their corresponding phones, as shown below:

```
MATAMOROS m a t a m o r o s
TORREÓN t o R e o n
CAMPECHE k a m p e CH e
ZÚÑIGA s u NY i g a
GUASAVE g u a s a b e
ALLENDE a y e n d e
```

The format of the dictionary is the word (in captials) following by the corresponding phones, each separated by a space.

Phones correspond to ipa with the following exceptions:

- Palatal fricative: y
- Palatal affricatve: CH
- Palatal nasal: NY
- Alveolar tap: r
- Alveolar trill: R
- Approximants and stops are both lower-case: (b,d,g)

Note: Extra words in your custom dictionary won't affect alignment! I suggest keeping one dict.local and adding new words as you encounter them.

4.3 Aligning

4.3.1 Command Line

Note: All faseAlign calls are carried out via the command line or terminal. It may be useful for you to review basic terminal commands used by Linux and OSX when interacting with files. I'd suggest finding a short introduction such as this Brief Tutorial.

4.3.2 Basic Example

In most cases, the default options for faseAlign are sufficient and can be called with the following command:

```
faseAlign -w input.wav -t transcript.TextGrid
```

The transcript file passed to the *-t* flag can be of type *.TextGrid* or *.txt*

Note: There are no positional arguments so you can order the flags however you want.

4.3.3 Additional Options

On top of the basic aligning call, there is additional functionality available. Examples for each of these additional features are shown below.

Change Output Location

The default location of the output TextGrid is your current working directory. This can be overwritten by passing an new directory (absolute or relative path) to the *-o* flag:

```
faseAlign -w input.wav -t transcript.TextGrid -o /home/user/Desktop/project/
```

Specify Output Name

The default name for output files can be changed with the *-n* flag:

```
faseAlign -w input.wav -t transcript.TextGrid -n new_file_name
```

Missing Words During Alignment

If you encounter words in your transcript which are not included in the default dictionary, you can choose to generate automatic phonemicizations for these missing words following Spanish orthography to phoneme rules. These automatic phonemicizations are usually correct for native Spanish vocabulary, but are often incorrect for loanwords or some place names.

To use these automatic phonemicizations, you can use the *-p* flag:

```
faseAlign -w input.wav -t transcript.TextGrid -p
```

If there are words in your transcript that you'd like to give a custom phonemicization, you should add them to a custom local dictionary (see missing). Then pass the path to the new file to the *-m* flag:

```
faseAlign -w input.wav -t transcript.TextGrid -m dict.local
```

Speaker Tags for .txt Transcriptions

Recall that .txt transcriptions have speaker tags in between braces (e.g., {Julia}, {S10}). To use these speaker tags, you have to use the *-g* flag:

```
faseAlign -w input.wav -t transcript.txt -g Julia Marco S4
```

This command will correctly match the speaker tags *{Julia}*, *{Marco}*, and *{S4}*.

Single Tier Alignment for .TextGrid transcriptions

By default, all tiers of .TextGrid transcripts will be aligned. If you'd prefer to only align one tier of a .TextGrid, you can use the *–tier*/*-i* flag:

```
faseAlign -w input.wav -t transcript.TextGrid -i tier-name
```

This command will only align a tier with the name 'tier-name'.

Stereo Options

If you have stereo audio with speakers on separate channels, alignment can be improved by separating out speakers into their respective channels.

First, determine which speaker is in channel 1 (left) and channel 2 (right). Now, pass those speaker labels to the *-l* and *-r* flags as well as using the *-s* flag to indicate stereo.

```
faseAlign -w input.wav -t transcript.TextGrid -s -l S1 -r S2
```

Automatic Syllabification

To generate automatic syllable boundaries, as well as stressed/unstressed status, use the *-y* flag in your call:

```
faseAlign -w input.wav -t transcript.TextGrid -y
```

Note that this does not correctly account for exceptional hiatus class words. For example, dueto will be syllabified as 2, rather than 3 syllables.

4.4 Model Development

4.4.1 Monophones

	labial	dental	alveolar	palatal	velar
plosives - voiceless	p	t			k
plosives - voiced	b	d			g
fricatives - voiceless	f		s		x
fricative - voiced				j (y)	
affricate				tʃ (CH)	
nasals	m		n	ɲ (NY)	
lateral			l		
rhotic - tap			ɾ (r)		
rhotic - trill			r (R)		

Vowels: /a,e,i,o,u/ correspond to their ipa symbols
Non-Speech: Laughing (lg), Coughing (cg), Breath (br), Noise (ns), short pause (sp), silence (sil)

Fig. 2: *Monophone symbols*

For the most part, phones correspond to their IPA symbols. Exceptions are noted in the table with symbols in parentheses. This phonemic inventory corresponds to many varieties of Latin American Spanish, but notably does not include theta. Additionally, palatal laterals are not included.

4.4.2 Notes on Development

The acoustic models used in this tool are triphone-HMMs trained using the HTK on data from 20 spontaneous interviews from the Corpus del Español de Raleigh-Durham (CERD) created and maintained by Drs. Jim Michnowicz and Rebecca Ronquest. A detailed description of the development of this tool can be found in the slides from our NWAV 2015 presentation.

4.5 Changelog

1. **Version 1.1.14**: October 13, 2022
    1. Fixed issue where words in *dict_sort* whose pronunciations differ from the automatic phonemicizations would cause syllabification to fail with an IndexError.
    2. Fixed issue where syllabification could fail when custom dictionaries weren't also enabled.
    3. Fixed issue where digraphs (NY and CH) were not correctly treated when syllabifying words from custom dictionaries.
2. **Version 1.1.13**: June 12, 2021
    1. Fixed bug where custom dictionary entries (-m) did not correctly interact with automatic syllabification (-y)
3. **Version 1.1.12**: April 25, 2021
    1. New installation instructions for conda environment method
4. **Version 1.1.11**: March 30, 2021
    1. Fixed bug where -n flag was removing characters from beginning and end of argument
    2. Fixed bug where output TextGrids tiers may have been added in a random order; intended behavior is words and then phones.
    3. Fixed bug where rounding errors in HTK output could cause output textgrids to have tiny empty intervals
5. **Version 1.1.10**: October 20, 2020
    1. Changes to directory structure for more consistent distribution
    2. Bug-fixes with dictionary access and phonemicization flag
    3. Change to https git protocol (over git+git) for installation
6. **Version 1.1.9** : October 10, 2018
    1. Updates to package directory structure and creation of utils.py.
    2. Import custom classes and functions in a Python interpreter by importing from faseAlign.utils as in the following example: *from faseAlign.utils import spanish_word*
7. **Version 1.1.8** : September 26, 2018
    1. Added -i/–tier tag to allow for aligning of individual tiers of .TextGrid transcripts
8. **Version 1.1.7** : August 20, 2018
    1. Bug fixes to .txt input transcripts and processing of speaker labels
9. **Version 1.1.6** : August 19, 2018
    1. Addition of version printout (-v flag)
    2. Change of version numbering scheme to be consistent with semantic versioning (note the jump from 0.1.5 to 1.1.6)
10. **Version 0.1.5** : March 28, 2018
    1. Sampling rate bug fixes
    2. Change in default behavior: phonemicization is no longer carried out by default, but must be chosen using the *p* flag
11. **Version 0.1.4** : October 18, 2017

    1. Dictionary encoding bug fixes

12. **Version 0.1.3** : August 22, 2017
    1. Addition of custom output name functionality (-n flag)
13. **Version 0.1.2** : July 24, 2017
    1. Addition of custom speaker tag functionality (-g flag)
14. **Version 0.1.1** : July 19, 2017
    1. Addition of syllabification functionality (-y flag)
    2. Addition of automatic phonemicization of unknown words (and prevention with -p flag)
15. **Version 0.1.0** : July 10, 2017
    1. Initial release