General tips and guidelines

Working with the command line

In all tutorials, code to be typed into a command prompt is indicated in this way:

```
 > type code in Courier New into the command line
```

Don't actually type in the > - this is just shown to indicate that you'll be typing at a command prompt.

To go to a folder, use `cd` to get to the desired directory, e.g.

```
> cd directory/subdirectory/targetDirectory
```

If you see an argument in brackets, e.g.

```
> python predict.py [input file] [output file]
```

This means that the item should be replaced with your own argument, e.g.

```
> python predict.py test.wav test.textgrid
```

Get started

To download all tutorial tools and dependencies, follow the instructions found here:
https://github.com/MLSpeech/DeepPhoneticToolsTutorial

Operating systems

The tools presented in these tutorials are currently available for Mac and Linux. If you have a PC, not to worry – you'll just need to download a virtual machine to be able to run Linux on your computer. To get started, you'll need to install Virtual Box (https://www.virtualbox.org/), which will give you the capability to set up a virtual machine on your computer. Note that you'll need at least 14 GB free, and the machine will work best with at least 1 GB of RAM dedicated to it.

You can start a machine from scratch. Alternately, we recommend the use of the Berkeley Phonetics Machine, a system that comes pre-built with several tools of use to speech scientists and linguists:
http://linguistics.berkeley.edu/plab/guestwiki/index.php?title=Berkeley_Phonetics_Machine
Even if you choose not to use the BPM, there are helpful setup tips for virtual machines on their site.

Structure of these tutorials

In each tutorial, we will start by looking at a manually-annotated file, using a Praat plugin (currently only available for Mac) of each tool to process that file, and then practice running the relevant tool at the command line (an alternate way to process files). The Praat plugin and command line tools to run these single-file processes are all available through the repositories linked at the start of each tutorial.

After visually inspecting and comparing the manual and automatic measurement, we'll move on to process an entire directory of files, and look at some visualizations of the aggregated data in R. The scripts we use for batch processing and analysis are just examples of how you can process and post-process your data – we'll make them available on our tutorial websites, but you can also use your own scripts to accomplish similar analyses.

Tutorial 1: Vowel durations and formants

Tools (find installation instructions here)

- AutoVowelDuration: https://github.com/MLSpeech/AutoVowelDuration
- DeepFormants: https://github.com/MLSpeech/DeepFormants

Introduction

The tools introduced in this tutorial are designed to automatically measure properties of vowels. They are optimized for measuring the stressed vowel of a single word, ideally in a consonant-vowel-consonant structure. AutoVowelDuration creates a textgrid with an interval tier that gives a best estimate of the onset and offset of the vowel. DeepFormants creates a CSV file with an estimate for F1-F4, in two possible modes: (a) a single estimate for a defined time window, or (b) tracking every 10 ms across a sound file. (Note that the estimates for F4 are currently less reliable than F1-F3).

In this tutorial, we will use AutoVowelDuration to measure the duration of vowels, and use the onset and offset it finds as input to DeepFormants, where we will find estimates of the vowel's formants.

Stage A: Measure a single file's vowel duration using the Praat plugin. (Note: Mac only)

Open up Praat to look at our example file (**example.wav**) and a textgrid (**example_vowelManual.TextGrid**) with a manual annotation of the vowel. The example files are found in the **example/** directory.

To process the data using the Praat plugin, select the wav file and click the "AutoVowelDuration" button. This will make a textgrid with the vowel onset and offset marked.

To compare the manual and automatic measurements, select both and click "Merge". Select the newly-merged textgrid and the wav file, and click "View and edit" to compare.

Stage B: Process a single file at the command line.

Now we'll take those same vowel measurements from the command line. We will also use that measurement as input to measure the formants of that vowel. Open a terminal and `cd` to the directory containing the tutorial files and tools.

Step 1: Measure vowel duration

The AutoVowelDuration tool will create a textgrid with a tier called VOWEL, with an interval defined by the start and end of the vowel.

- The basic call:
  ```
  > python AutoVowelDuration.py [input wav file] [output textgrid file] [output csv file]
  ```

- The call to measure the sample file (type this into the terminal):
  ```
  > python AutoVowelDuration.py example/example.wav example/example_vowelAuto.TextGrid example/example_vowelAuto.csv
  ```

Step 2: Estimate formants

The DeepFormants tool will use the interval defined in the textgrid created in step 1 to estimate F1-F4 and save the estimated values to a .csv file.

- The basic call:
```
> python DeepFormants.py [input wav file] [input textgrid] [output file]
--tier_name [tier name]
```

- The call to measure the sample file (type this into the terminal):
```
> python DeepFormants.py example/example.wav
example/example_vowelAuto.TextGrid example/example_formantsAuto.csv
--tier_name VOWEL
```

Stage C: Process a directory of files.

To efficiently process several files, you can also use these scripts to process a whole directory at once. In the following example, we will measure the duration of every vowel in a directory using AutoVowelDuration, and then extract those measurments from their textgrids into text files. We will then use those measurements as input to define the window in which we will estimate F1-F4 using DeepFormants.

We've provided a set of sample wav files in the **sampleFiles/** directory. All example words are monosyllabic, with the vowels /a/, /i/, or /u/, each spoken by a male and a female speaker.

Step 1: Measure durations

- The basic call:
```
> python AutoVowelDuration.py [input wav directory] [output textgrid
directory] [output csv file]
```

This will create a textgrid for each file. It will also create a single csv file which extracts the onset, offset, and duration from each of those textgrids.

- To process the data in **sampleFiles/**, type:
```
> python AutoVowelDuration.py sampleFiles/waveforms
sampleFiles/vowel_durations sampleFiles/vowel_durations.csv
```

Step 2: Measure formants

Note that DeepFormants' estimation mode requires a defined time window, so you'll need to measure vowel durations (step 1) before estimating formants (step 2).

- The basic call:
```
> python DeepFormants.py [input wav directory] [input textgrid directory]
[output csv file]
```

This looks up the textgrids created in step 1 to find a window to estimate formants for each file. It creates a single csv file with an estimate for F1, F2, F3, and F4 in that window.

- To process the data in **sampleFiles/**, type:
  ```
  > python DeepFormants.py sampleFiles/waveforms sampleFiles/vowel_durations
  sampleFiles/formants.csv
  ```

Stage D: look at vowel properties in R.

The sample R code **analyzeVowels.R** gives you a jumping-off point to investigate the vowel durations and formants measured by these tools. It reads in the .csv files created in step 3 and uses these to produce some sample visualizations and simple data comparisons.

In this script, we:

- Read in the vowel duration and formant data
- Process the data files to extract information from the file names
- Merge the two datasets together
- Compare durations across vowels and across speakers
- Plot vowels (F1/F2) by speaker

Tutorial 2: Voice onset time and word durations

Tools (find installation instructions here)

- AutoVOT: https://github.com/MLSpeech/AutoVOT
  (note: AutoVOT needs to be compiled; follow instructions in readme file)
- DeepWDM: https://github.com/MLSpeech/DeepWDM

Introduction

The tools included in this tutorial will allow you to measure the voice onset time of a stop consonant at the beginning of a word, without prior annotation or transcription of the word.

The VOT tool requires a pre-defined window to search an audio file. To narrow the search space, we will use DeepWDM, a tool that automatically measures the duration of a word (assuming a single word in an audio file) and creates a textgrid with an interval defined by the start and end points of the word.

In this tutorial, we will use DeepWDM to measure the duration of words, and use the onset and offset as input to AutoVOT, where we will get a measurement of the VOT of the initial stop in the word.

Stage A: Process a single file using the Praat plugins. (Note: Mac only)

Word duration

- Open up Praat to look at our example file (**example.wav**) and a textgrid (**example_wordManual.TextGrid**) with a manual annotation of the word. The example files are found in the **example/** directory.
- To process the data using the Praat plugin, select the wav file of interest, and press the "DeepWDM" tool. This will create a textgrid with the word onset and offset marked.
- To compare the manual and automatic measurements, select both textgrids and click "Merge". Select the newly-merged textgrid and the wav file, and click "View and edit" to compare them.

VOT

- Open the example file (**example.wav**) and the manual textgrid (**example_votManual.TextGrid**) from the **example/** directory.
- The AutoVOT tool needs a defined window around the start of the word to make an estimate. A good window would start 180 ms before the onset of the word, to 100 ms after the word onset.
- To process the data using the Praat plugin, you'll first need to create that manual window. Create a new textgrid with a tier called "window", add an interval to search in. Then select the wav file and that textgrid and press the "AutoVOT" tool. Make sure to specify the tier as "window" in the dialog box that pops up. This will create a textgrid with the VOT measured.
- To compare the manual and automatic VOT measurements, select both and click "Merge". Select the newly-merged textgrid and the wav file, and click "View and edit".

Stage B: Process a single file at the command line.

Now we'll take those same measurements from the command line. Open a terminal and `cd` to the directory containing the tutorial files and tools.

Step 1: Measure word duration

First, we'll use DeepWDM to define the boundaries of the word. This will create a textgrid with an interval bounded by the start and end of the word, on a tier called WORD.

- The basic call:
  ```
  > python DeepWDM.py [input wav] [output textgrid] [output csv file]
  ```

- The call to measure the sample file (type this into the terminal):
  ```
  > python DeepWDM.py example/example.wav example/example_wordAuto.textgrid
  example/example_wordAuto.csv
  ```

Step 2: Generate search window

Now we'll use a script to generate a search window for the VOT tool to search in. Here, we specify a window that starts 180 ms before the word onset and 100 ms after onset. This will create a second tier called WINDOW on the textgrid created in step 1.

- The basic call:
  ```
  > python generateSearchWindows.py [textgrid] --tier [tiername] --before [time
before word onset] --after [time after word onset]
  ```

- The call to measure the sample file:
  ```
  > python generateSearchWindows.py example/example_wordAuto.textgrid –-tier
  WORD --before 0.18 –-after 0.1
  ```

Step 3: VOT

Finally, we'll use the search window generated in step 2 to search for the VOT interval. This will create a third tier on that same textgrid with the estimated VOT interval marked.

- The basic call:
  ```
  > python AutoVOT.py [input wav file] [textgrid] [csv]
  ```

- The call to measure the sample file (type this into the terminal):
  ```
  > python AutoVOT.py example/example.wav example/example_wordAuto.TextGrid
  example/example_votAuto.csv
  ```

Stage C: Process a directory of files.

We can use these same tools to measure VOT from every wav file in a directory. In the following example, we will measure the duration of every word in a directory using DeepWDM, use those measurements to define a search window, and then measure VOT within those windows using AutoVOT. We've provided a set of sample wav files in the **sampleFiles/** directory. All example words are monosyllabic and start with a stop consonant, and each is spoken by a male and a female speaker.

Step 1: Measure word durations

- The basic call:
  ```
  > python DeepWDM.py [input wav directory] [output textgrid directory] [output csv file]
  ```

This will create a textgrid for each file. It will also create a single csv file which extracts the onset, offset, and duration from each of those textgrids.

- To process the data in **sampleFiles/**, type:
  ```
  > python DeepWDM.py sampleFiles/waveforms sampleFiles/word_durations sampleFiles/word_durations.csv
  ```

Step 2: Generate search windows

- The basic call:
  ```
  > python GenerateSearchWindows.py [textgrid directory]
  ```

This looks up the textgrids created in step 1, and creates a search window centered around the word onset, to serve as input to the VOT tool.

- To process the data in **sampleFiles/**, type:
  ```
  > python GenerateSearchWindows.py sampleFiles/word_durations
  ```

Step 3: Measure VOT

- The basic call:
  ```
  > python AutoVOT.py [wav directory] [textgrid directory]
  ```

This looks up the textgrids created in step 1, and creates a search window centered around the word onset, to serve as input to the VOT tool.

- To process the data in **sampleFiles/**, type:
  ```
  > python AutoVOT.py sampleFiles/waveforms sampleFiles/word_durations sampleFiles/vot.csv
  ```

Stage D: look at VOT in R.

The sample R code **analyzeVOT.R** is an example analysis of the measurements extracted in this tutorial. In this script, we:

- Load in word durations measured from DeepWDM
- Load in VOT measurements from AutoVOT
- Plot VOT as a function of voicing
- Look at VOT as a percentage of overall word duration