Aligners and AutoVOT: A VERY Detailed Tutorial

Zhiyan Gao

George Mason University

zgao@gmu.edu

Abstract

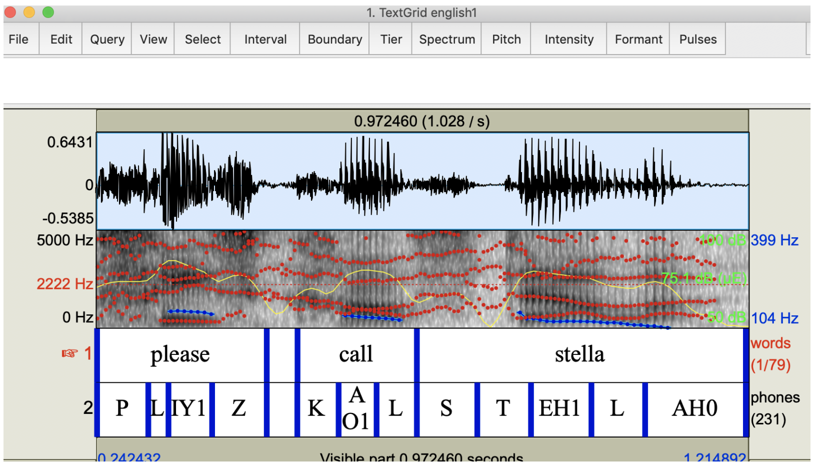
In this tutorial, I will show you (1) how to use forced-aligners, (2) how to bulk-process textgrid files using the ***rPraat*** package in **R**. I will introduce two types of aligners: the general purpose aligners (e.g., the Montreal Forced Aligner) and an aligner for aligning Voice Onset Time (i.e, AutoVOT). I am going to show you how to do grapheme-to-phoneme alignment on corpus data. Based on the autovot.praat script shipped with the AutoVOT plugin, I created a Praat script called ***autovot_bulk.praat*** to help measure VOTs of multiple audio files. I assume you have some basic knowledge of **R** and **Praat**. This handout was prepared for students who have just started doing speech analysis.

Introduction

Speech annotation is the first and the most critical step toward meaningful linguistic analyses. The Figure 1 demonstrates this process. We got a speech sample "please call stella". We listen to it and then we label out the word boundaries and the boundaries for each phonetic segment using **Praat** (Boersma & Weenink, 2018).

We could manually annotate speech samples if the sample size is relatively small. It might be too labor-intensive when there are thousands of files to label. This tutorial will introduce techniques of automated speech annotation.

Figure 1. Speech Annotation



Aligners

Software programs for speech annotation/segmentation is often termed as "forced aligners". "Forced-aligners" are software that could generate time-stamped files showing temporal boundaries of phonetic segments and/or lexical items (i.e., words). For example, the Praat textgrid displayed in Figure 1 was generated computationally using a "forced aligner". The results are reasonably accurate.

There are quit a few "forced aligners". You might have heard of the **The Penn Phonetics Lab Forced Aligner** (Yuan & Liberman, 2008), which is widely used in the field. The Penn Forced Aligner has a new update called **FAVE-align** (Rosenfelder, Fruehwald, Evanini, & Yuan, 2011). However, the installation of both the old Penn forced aligner and the FAVE-align are less than straight-forward, especially for PC users.

I will introduce two aligners that are easy to use (relatively speaking, of course). The first one is called the **the Montreal Forced Aligner** (McAuliffe, Socolof, Mihuc, Wagner, & Sonderegger, 2017), which requires some copying and pasting of codes. The second one is a website called the **The Munich Automatic Segmentation System** (MAUS) (Kisler, Reichel, & Schiel, 2017), which provides a user interface, and it is extremely easy to use.

Montreal Forced Aligner

The most current forced aligner is the **Montreal Forced Aligner** (MFA) (McAuliffe et al., 2017), which is compatible with both Mac and PC systems. Details about the installation of the MFA is here. Once the MFA zip file is downloaded on your computer, unzip it. You will see a folder called **montreal-forced-aligner**. Inside the **montreal-forced-aligner** folder, there are 3 folders, namely **bin**, **lib**, and **pretrained_models**.

The pretrained_models folder contains the model for English. The MFA offers pre-trained models for several other languages. We are going to use the English model for this tutorial. If you need to use other models, just download the model from the MFA website.

Data Preparation

We need 4 things to implement the MFA.

1. the audio file we want to segment;
2. the orthographic transcription for the audio file;
3. the model that matches the language of the audio file;
4. a phonetic dictionary for the language.

Let's use an audio file from the Speech Accent Archive (Weinberger, 2019). The audio file I have is called *arabic6.wav* which is a non-native English speech sample produced by an Arabic speaker. This speaker read the "Stella" passage as listed below

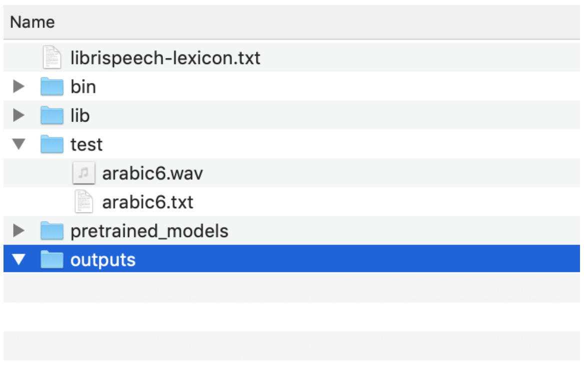
Please call Stella. Ask her to bring these things with her from the store. Six spoons of fresh snow peas, five thick slabs of blue cheese, and maybe a snack for her brother Bob. We also need a small plastic snake, and a big toy frog for the kids. She can scoop these things into three red bags, and we will go meet her Wednesday at the train station.

I created a folder called **test** to store the audio file and the orthographic transcription of the audio file (i.e., the "Stella" passage). We need to name the audio file and its corresponding transcription file using the same name (See Figure 2). ! VERY IMPORTANT

For this audio file, the language is English. We therefore need a phonetic dictionary for English. We could use the "LibriSpeech lexicon" dictionary provided by the MFA. Download the "LibriSpeech lexicon" here, and put it in the MFA folder.

Let's also create an ***outputs*** folder. Now the folder is empty. We are going to store textgrid files in this folder.

Figure 2. the MFA folder



Running the MFA

Now let's run the MFA with the pre-trained English model. Open Terminal (Command Prompt on PCs) and navigate to your MFA folder. I put the folder on my Desktop. So here are my codes ("cd" stands for "change directory").

```
cd ~/Desktop/montreal-forced-aligner
```

Hit Return (enter).

Now we are in the MFA folder, let's tell MFA the things it wants to know.

1. location of the audio file and its transcript
2. location of the phonetic dictionary
3. which language model to use
4. where to store the output

We begin by calling the ***mfa_align*** function and then specify the parameters (i.e., the things we want to tell MFA). Here are the codes.

```
bin/mfa_align test/ librispeech-lexicon.txt english outputs/
```

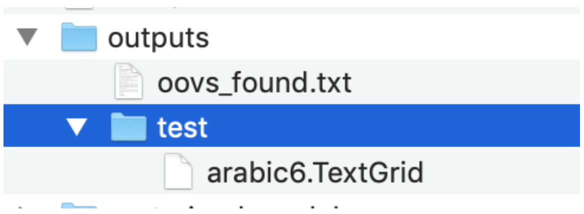
These codes tell MFA that both the audio file and the transcript file are in the **test** folder. There's no need to type out the complete directory of the **test** folder, because it is in the MFA folder, and we are in the MFA folder already (Recall we did something like "cd /Desktop/montreal-forced-aligner"). If your files are not in the MFA folder, then you need to type out the full directory. Same thing applies to the phonetic dictionary and the output folder.

After we specify the location of the audios and the transcripts. We tell MFA the location of the phonetic dictionary, the language of the audio (e.g., english), and where to store the outputs. The MFA starts to run and starts to give us the status of the alignment. Here's what I got when processing the *arabic6.wav* file.

```
Setting up corpus information...
Number of speakers in corpus: 1, average number of utterances per speaker: 1.0
Creating dictionary information...
Setting up training data...
Calculating MFCCs...
Calculating CMVN...
Number of speakers in corpus: 1, average number of utterances per speaker: 1.0
Done with setup.
100%|| 2/2 [00:01<00:00,  1.20it/s]
Done! Everything took 6.955101013183594 seconds
```

I got only one speaker and one audio file. The automated alignment took 6.955 seconds to complete. In my **outputs** folder, I see a new folder called **test**, in which there is a textgrid file called *arabic6.TextGrid* (See Figure 3).

Figure 3. the Outputs Folder

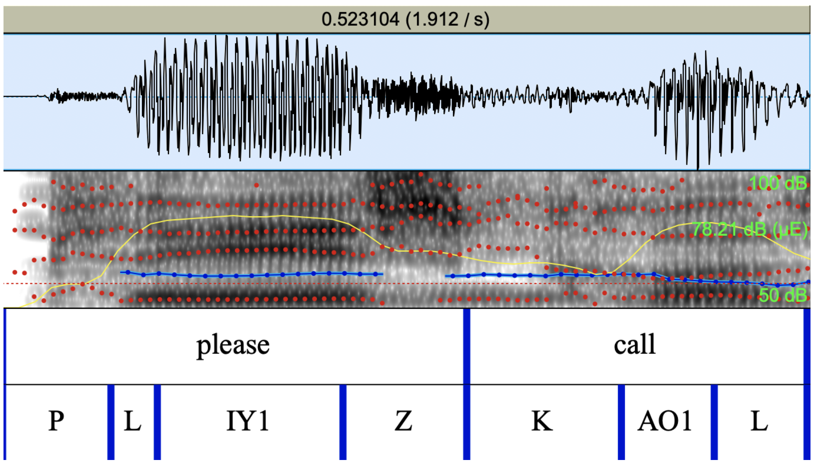


outputs
oovs_found.txt
test
arabic6.TextGrid

The arabic6.TextGrid is what the MFA generated. Let's take a look at this file in **Praat**. MFA generated two tiers. The first tier shows the temporal boundaries for words.

The second tier shows temporal boundaries of phonetic segments. The phonetic segments are represented using the ARPABET symbols. The results are reasonably good (See Figure 4).

Figure 4. MFA Alignment Result



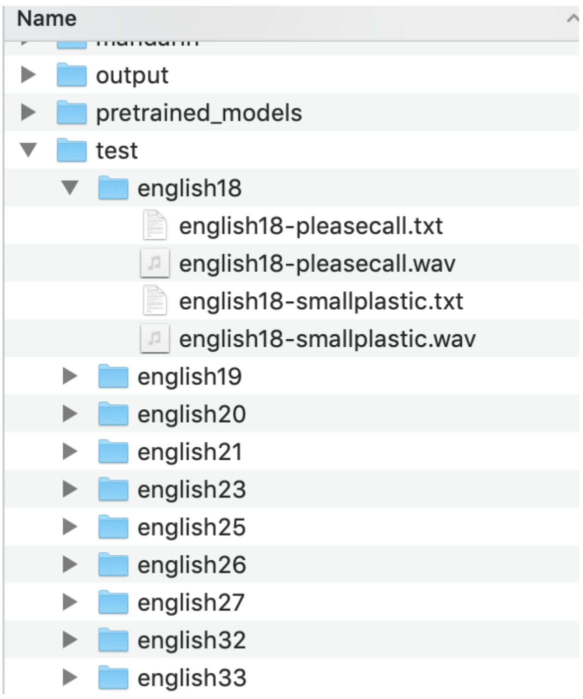
Multiple Speakers with Multiple Recordings

We only dealt with one speaker with one audio recording. For multiple speakers, the codes are the same. However, the folders should be organized in a way that the MFA understands that there are more than one speakers. As shown in Figure 5, my **test** folder now includes multiple sub-folders, each of which is the name of a specific speaker. For speaker *english18*, I got 2 audio files. I created two .txt files to tell MFA what *english18* was saying. For example, in *english18-pleasecall.txt* file, I simply wrote *please call*, which tells MFA that the *english18-pleasecall.wav* file is the English phrase *please call.*

Once you run the following codes. MFA starts to go through every sub-folders in **test** and then create textgrids for each .wav file in the **outputs** folder.

```
bin/mfa_align test/ librispeech-lexicon.txt english outputs/
```

Figure 5. Folder Organization for Multiple Speakers



Using MFA for Other Languages

Suppose we want to annotate Mandarin audio files. What should we do? We still need 4 things. The audio files, the transcripts, the dictionary, and the pre-trained mandarin model.

The MFA has a great number of pre-trained models. Click here to find the model you want. The dictionary could be generated via codes described here.

Instead of using the language-specific models, we could simply using the English model for other languages. To achieve this, we just need to add some "words" in the librispeech-lexicon.txt file.

Let's say my audio file is a Mandarin sentence "ni chi le ma" containing 4 mandarin words. All I need to do is pretend that these words are English words. I add these words to the *librispeech-lexicon.txt* file by adding the following line at the beginning:

ni NIH

chi CHAXR

le LAX

ma MAA

The first column lists the words and the second column uses English phonemes to indicate how these words are pronounced using the ARPABET symbols. Obviously, English phonemes cannot correctly describe Mandarin pronunciation. However, our goal is to segment the audio files. It should be fine to use this type of “hacking” techniques as long as it gets the job done.

The Bavarian Archive for Speech Signals

A much easier way of doing speech annotation/segmentation is through the web interface of the Bavarian Archive for Speech Signals (BAS) [1]. It is so easy that I believe no tutorial is actually needed for the basic functions.

Just go to the BAS website, click the **WebMAUSBasic** tap, drag your .wav files and their corresponding .txt files into the box in the middle. Scroll down to select the language you need, check the little box to accept terms of usage, and hit **Run Web Service**. Your textgrid files will be created on the BAS server and ready for download.

AutoVOT Praat Plugin

The MFA and the BAS are general purpose aligners, which could help locate temporal boundaries of phonetic segments. The AutoVOT software program (Keshet, Sonderegger, & Knowles, 2014) was designed to do just one thing: locate Voice Onset Time (VOT) of plosive consonants.

Regrettably, AutoVOT does not seem to work on PCs. We need a Mac to get it to work. AutoVOT provides a command line tool and a praat plugin. This tutorial discusses only the praat plugin. Consult the AutoVOT GitHub repository for more information on the command line tool.

Installation

For this tutorial, please download the **Bulk_AutoVOT** plugin from the GitHub repository I created for this tutorial. The difference between the **Bulk_AutoVOT** plu-

[1]The segmentation function the BAS employs is called the Munich Automatic Segmentation System (MAUS)

gin and the officially released AutoVOT plugin is that **Bulk_AutoVOT** contains a *autovot_bulk.praat* file that allows bulk-processing of audio files.

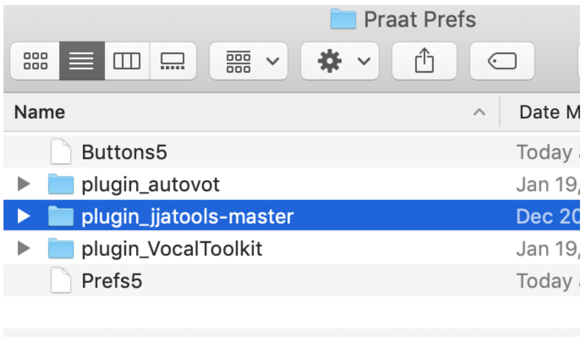
Once the zip file is downloaded. Unzip it. You will see a folder called **plugin_autovot**.

Move the **plugin_autovot** folder to the **Praat Prefs** folder located in your Library/Preferences folder. We are going to do this via Terminal. Please close and re-open your Terminal. Type in the following codes:

```
open Library/Preferences/Praat\ Prefs/
```

Now the **Praat Prefs** folder is open. Drag the **plugin_autovot** folder into it.

Figure 6. the Praat Prefs Folder



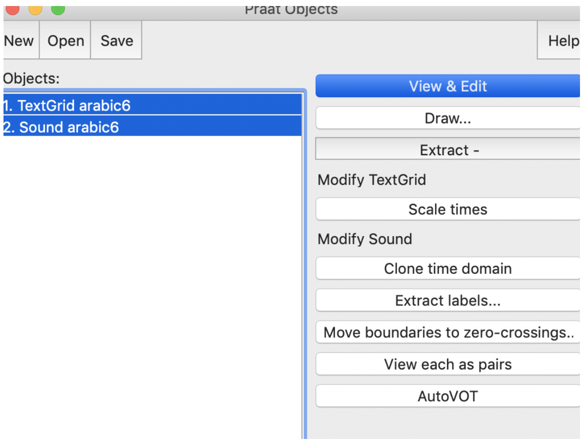
Buttons5 Today
plugin_autovot Jan 19,
plugin_jjatools-master Dec 20
plugin_VocalToolkit Jan 19,
Prefs5 Today

A side-note, the “Praat Prefs” folder is the location for all your praat plugins. Let’s take a look my “Praat Prefs” folder. I got three praat plugins. The codes for the AutoVOT plugin are in the **plugin_autovot** folder.

Open the **plugin_autovot** folder, double-click the setup.praat file. Or use Praat to open it. Run it in Praat. This script added a button called *AutoVOT* to your Praat object window.

Let’s take a look at the basic functions of AutoVOT. Close your praat, then re-open it. Open a sound file and a textgrid file in your Praat. Choose these two files by click them one by one while holding the shift key. You will see a clickable button called **AutoVOT** on the button right. This means that the AutoVOT plugin has been installed (See Figure 7).

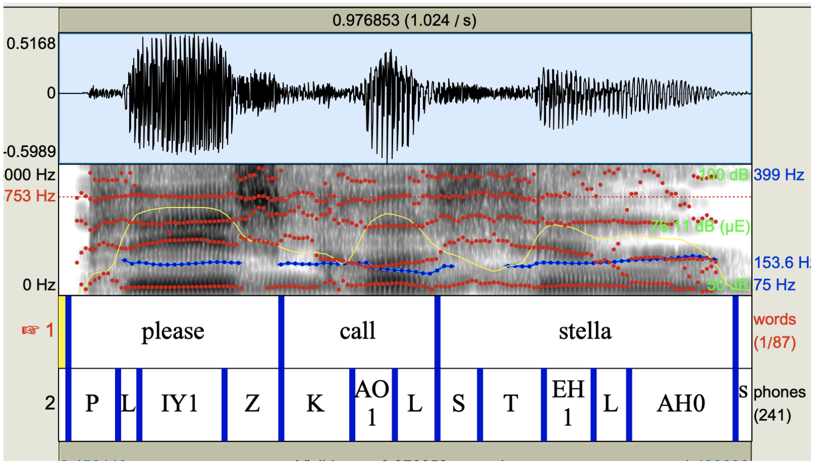
Figure 7. the AutoVOT button



Usage: One File at a Time

Click the AutoVOT button, a windown will pop up asking us which tier we want to analyze and which segments should be analyzed. Before we do that, let's first take a look at my sound file and the textgrid file.

Figure 8. the Original TextGrid

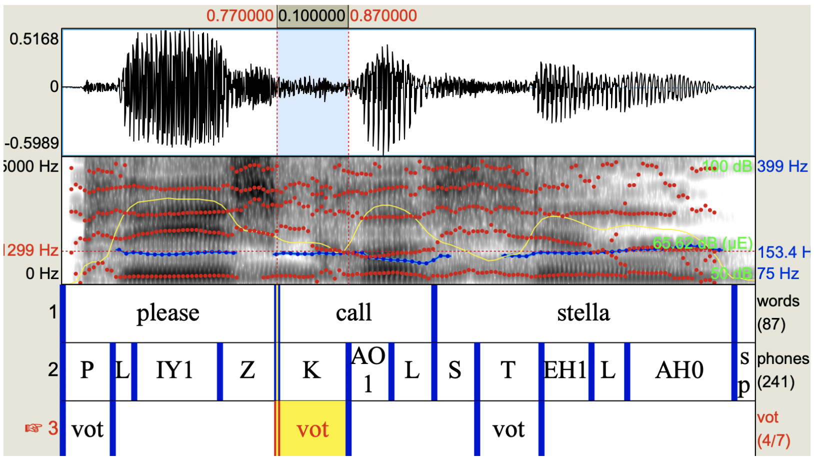


The phones tier includes plosives p, t and k, which were labeled using P, T and K. So the tier I want AutoVOT to focus on is the phones tier. I still need to specify the segments for AutoVOT to analyze, and this is the annoying part.

AutoVOT can analyze all the intervals in the phones tier. We just need to set the *interval_mark* option to *. However, what we really need is to have AutoVOT analyze P, T and K, and ignore other intervals. Unfortunately, this is impossible with the AutoVOT button.

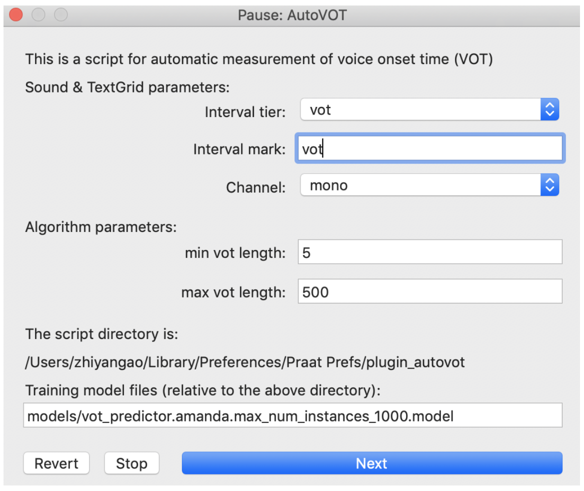
A workaround is to create another tier and change P, T, and K to the same name. Something like what is shown in Figure 9:

Figure 9. the Re-labeled TextGrid



I created a third tier called *vot*, and I added 3 intervals that match intervals of the 3 plosives. I named these 3 intervals *vot*. Now I select both the sound file and this newly created textgrid (click them one by one while holding the shift key). The AutoVOT button appears. Click on the button. A window pops up (See Figure 10)

Figure 10. Pop-up Window


Pause: AutoVOT

This is a script for automatic measurement of voice onset time (VOT)

Sound & TextGrid parameters:

Interval tier: vot

Interval mark: vot

Channel: mono

Algorithm parameters:

min vot length: 5

max vot length: 500

The script directory is:

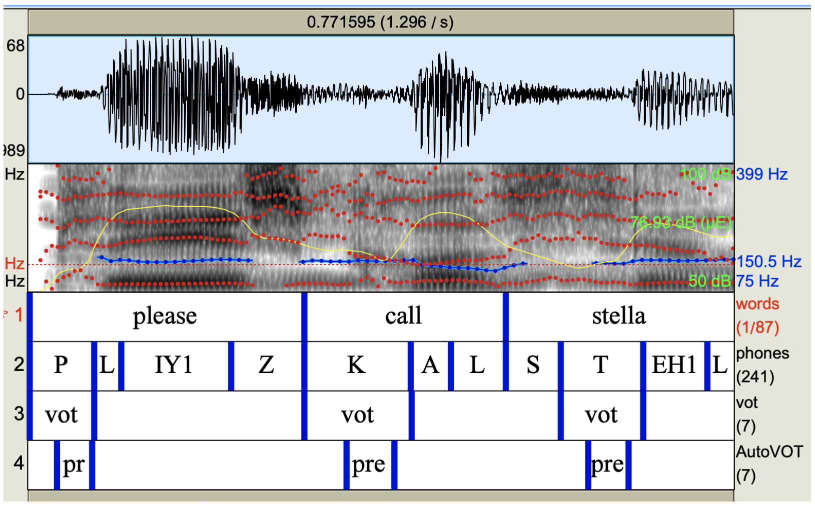
/Users/zhiyangao/Library/Preferences/Praat Prefs/plugin_autovot

Training model files (relative to the above directory):

models/vot_predictor.amanda.max_num_instances_1000.model

I set the interval tier to the vot tier and set the interval mark to vot. Click Next. A new textgrid file is then created to contain a tier called AutoVOT, which shows the VOT for each plosive. The VOTs are labeled as *Pred* (for "prediction") (See Figure 11).

Figure 11. TextGrid with VOT Labeled



Usage: Bulk-Processing

The AutoVOT button only allows to process one file at a time. It is not efficient when we have hundreds or thousands of files to work on. I have created a praat script for processing multiple audios.

Go to your **plugin_autovot** folder by typing in the following codes in Terminal:

```
open Library/Preferences/Praat\ Prefs/plugin_autovot/
```

In this folder, there is a praat script called **autovot_bulk.praat** that can be used for processing multiple audios. Before we can use the autovot_bulk.praat, we still need to do same modifications of the textgrids. That is, we need to create an additional tier and then label all the plosives as *vot*. There are many ways of doing this. I am going to do it in R with the rPraat package (Bořil & Skarnitzl, 2016).

Change TextGrids with rPraat

Here are the codes you can use in R (R Core Team, 2019). I first created a function called label_vot(), which takes 4 arguments:

1. textgrid_directory : folder of your original textgrids.
2. textgrids_output_folder : folder for the textgrids that are going to be created.
3. labels_to_change : which original labels should be changed? For me, they are P, T and K.
4. new_label : to what label should the original labels be changed? For me, I want to change all Ps, Ts and Ks to vot.

Beware that my original textgrids were generated using the MFA. Therefore, there are 2 tiers in my original textgrids, called *words* and *phones*. The label_vot function first duplicates the *phones* tier and then change the original labels to new ones.

```
### install rPraat if you don't have it on your computer
install.packages("rPraat")

###create function label_vot
label_vot<-function(textgrid_directory,
                    textgrids_output_folder,
                    labels_to_change,
                    new_label){
  library(rPraat)
  textgrids<-list.files(path = textgrid_directory, pattern = ".TextGrid")
  for (t in textgrids){
    tg<-tg.read(paste(textgrid_directory,t,sep=""))
    ### insert a 3rd tier, called vot
    tg2<-tg.insertNewIntervalTier(tg,3,"vot")
    ### duplicate the phones tier
```

```r
    tg2$vot<-tg2$phones
    tg2$vot$name<-"vot"
    for (i in c(1:length(tg2$vot$label))){
      if (tg2$vot$label[i]%in%labels_to_change){
      ## change labels
        tg2$vot$label[i] <- new_label
      }
    }
    tg.write(tg2,paste(textgrids_output_folder,t,sep=""))
  }
}
```

Run the R codes above first to create the label_vot() function in your R environment. Then set the 4 arguments:

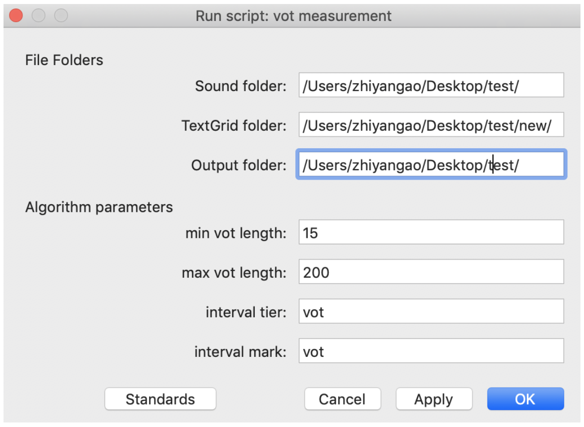
```r
## set parameters
### location of my original textgrids
textgrid_directory<-"/Users/zhiyangao/Desktop/test/"
### I want to change labels "P","T","K","p","t","k"
labels_to_change<-c("P","T","K","p","t","k")
### I want to change the labels to "vot"
new_label<-"vot"
### where to save the new textgrids
textgrids_output_folder<-"/Users/zhiyangao/Desktop/test/new/"

## After setting-up, run the label_vot() function
label_vot(textgrid_directory,
          textgrids_output_folder,
          labels_to_change,
          new_label)
```

You could download the R codes above from here .

Run autovot_bulk.praat

Open **autovot_bulk.praat** in your **plugin_autovot/** folder. Run it with Praat. A window will pop up, asking you to specify where the sound files and the textgrids are, where you want to store the new textgrids, and which tier should be analyzed, etc.

Figure 12. Pop-up Window


Run script: vot measurement

File Folders

Sound folder: /Users/zhiyangao/Desktop/test/

TextGrid folder: /Users/zhiyangao/Desktop/test/new/

Output folder: /Users/zhiyangao/Desktop/test/

Algorithm parameters

min vot length: 15

max vot length: 200

interval tier: vot

interval mark: vot

Remember that I just used R to create new textgrids? We are going to use these textgrids with autovot_bulk.praat. The **TextGrids folder** is where I stored the R-generated textgrids, which is the value of the textgrids_output_folder (i.e., /Users/zhiyangao/Desktop/test/new/).

Remember that we created a 3rd interval tier called *vot*? This is the tier we want AutoVOT to focus on. In the *vot* tier, I created intervals all named *vot* to match intervals of the plosive segments labeled in the 2nd tier. So the interval mark is also *vot*.

Hit OK. The program starts running. If you have lots of sound files to process, then it might take a while. For psychological assurance, you could open your Output folder to see textgrids being generated one after another.

After all the files are processed, a Praat Info window will pop up, telling you how many audio files have been processed. The new TextGrid files are in the Output folder, all with a suffix "_new" in their names.

Conclusion

This tutorial introduced the following:

1. How to use the MFA and the BAS
2. How to manipulate TextGrids using rPraat
3. How to automatically measure VOT using AutoVOT

Only the most basic functions of MFA, BAS, rPraat, and AutoVOT were introduced. You might want to check out the documentations of these tools for more advanced functionalities. Other than the MFA, BAS, and the UPenn forced aligners, you might also want to try out the following alignment tools:

1. Train and Align (Brognaux, Roekhaut, Drugman, & Beaufort, 2012): provides a web interface. Unlike BAS which relies on pre-trained models. The Train and Align tool allows you to train your own model right on their server.
2. EasyAlign (Goldman, 2011): a praat plugin, provide sentence-level and phone-level alignment.
3. ALISA (Stan et al., 2016): a standalone software program. works for any languages which employ alphabetic scripts.
4. the Korean Phonetic Aligner (Yoon & Kang, 2013): the name is self-explanatory.

After we have all our files labeled. The next step is to check the labeling manually. After all the errors are corrected, we need to extract acoustic information from the sound files based on the textgrid files. For example, now we have all the VOTs labeled, how are we suppose to get the duration of the VOTs? Suppose we want to know the formant information of all the vowels, how should we do that? We could do it one file, one segment at a time if we only have a handful of files. What if we have hundreds or thousands of files to measure?

The next tutorial will introduce ways of extracting acoustic information from a large number of files using Praat scripts.

References

Boersma, P., & Weenink, D. (2018). Praat: Doing phonetics by computer [computer program]. version 6.0. 37. *RetrievedFebruary*, *3*, 2018.

Bořil, T., & Skarnitzl, R. (2016). Tools rpraat and mpraat. In *International conference on text, speech, and dialogue* (pp. 367–374).

Brognaux, S., Roekhaut, S., Drugman, T., & Beaufort, R. (2012). Train&align: A new online tool for automatic phonetic alignment. In *2012 ieee spoken language technology workshop (slt)* (pp. 416–421).

Goldman, J.-P. (2011). Easyalign: an automatic phonetic alignment tool under praat. In *Interspeech'11, 12th annual conference of the international speech communication association.*

Keshet, J., Sonderegger, M., & Knowles, T. (2014). Autovot: A tool for automatic measurement of voice onset time using discriminative structured prediction [computer program]. *Version 0.93, retrieved November 2018*.

Kisler, T., Reichel, U., & Schiel, F. (2017). Multilingual processing of speech via web services. *Computer Speech & Language*, *45*, 326–347.

McAuliffe, M., Socolof, M., Mihuc, S., Wagner, M., & Sonderegger, M. (2017). Montreal forced aligner: Trainable text-speech alignment using kaldi. In *Interspeech* (pp. 498–502).

R Core Team. (2019). R: A language and environment for statistical computing [Computer software manual]. Vienna, Austria. Retrieved from https://www.R-project.org/

Rosenfelder, I., Fruehwald, J., Evanini, K., & Yuan, J. (2011). *Fave (forced alignment and vowel extraction) program suite.* Retrieved from http://fave.ling.upenn.edu

Stan, A., Mamiya, Y., Yamagishi, J., Bell, P., Watts, O., Clark, R. A., & King, S. (2016). Alisa: An automatic lightly supervised speech segmentation and alignment tool. *Computer Speech & Language*, *35*, 116–133.

Weinberger, S. (2019). *Speech accent archive. george mason university.* Retrieved from http://accent.gmu.edu

Yoon, T., & Kang, Y. (2013). *The korean phonetic aligner program suite.* Retrieved from http://korean.utsc.utoronto.ca/kpa/

Yuan, J., & Liberman, M. (2008). Speaker identification on the scotus corpus. *Journal of the Acoustical Society of America*, *123*(5), 3878.