Documentation of data processing of *Vowel duration and consonant voicing: An articulatory study* (English)

Stefano Coretta

2019-05-20

1 Convert and resample stereo files

```
<<<script header>>>

stereo$ = "../data/raw/stereo"
mono$ = "../data/raw/mono"
Create Strings as file list: "file_list", "'stereo$'/*.wav"
files = Get number of strings

createDirectory: mono$

for file from 1 to files
  selectObject: "Strings file_list"
  file$ = Get string: file
  participant$ = file$ - "-stereo.wav"
  mono_file$ = "'mono$'/'participant$'.wav"

  if fileReadable(mono_file$)
    appendInfoLine: "Skipping 'participant$'.wav..."
  else

    stereo = Read from file: "'stereo$'/'file$'"
    file_name$ = selected$("Sound")

    # Audio is in channel 1
    ch_1 = Extract one channel: 1

    # Doumsample
    ch_1_22050 = Resample: 22050, 50

    Save as WAV file: mono_file$

    removeObject: stereo, ch_1, ch_1_22050

  endif

endfor
```

2 Detect burst onset in C1 and C2 and create a merged TextGrid

This script detects the burst onset of C1 (/t/) and C2 and creates a new TextGrid with sentences, words, phones, and releases. The algorythm for the detection of the burst onset is based on @avanthapadmanabha2014. Note that the alrgorythm can find bursts, rather than releases. In the case of a non-audible release, the algorythm will not return a time point (which is the intended outcome).

```
<<<script header>>>

<<<initialise>>>

<<<detect>>>
```

```
writeInfoLine: "Initialise...'newline$'"
mono_dir$ = "../data/raw/mono"

wav_list = Create Strings as file list: "wav_list", "'mono_dir$'/*.wav"
files = Get number of strings
appendInfoLine: "> Found 'files' sound files."

for file from 1 to files
  selectObject: wav_list
  file_name$ = Get string: file
  speaker$ = file_name$ - ".wav"
  appendInfoLine: "'newline$'Processing speaker 'speaker$'."
  appendInfo: ""

  if fileReadable("'mono_dir$'/'speaker$'-annotation.TextGrid")
    appendInfoLine: "Found annotation file for 'speaker$'. Skipping to the next speaker."
    appendInfo: ""
    goto next
  endif

  sound = Read from file: "'mono_dir$'/'file_name$'"
  # The new tg:
  textgrid = To TextGrid: "sentence, word, phones, release", "release"
    # Tiers indexes of textgrid
    sentence_tier = 1
    word_tier = 2
    phones_tier = 3
    release_tier = 4

  palign = Read from file: "'mono_dir$'/'speaker$'-palign-corrected.TextGrid"
    # Tiers indexes of paling
    activity_tier = 4
    tokens_tier = 2
    phon_tier = 1

  # The tg with the sentences:
  sentences = Read from file: "'mono_dir$'/'speaker$'.TextGrid"

  <<<sentence loop>>>

  removeObject: sound, textgrid, palign, sentences

  label next
endfor
```

The following is the loop that iterates through each sentence, fills in the new TextGrid, and runs the burst onset detection algorythm. The algorythm calculates a 'plosion index' to find the onset of the burst.

```
selectObject: palign
speech_intervals = Get number of intervals: activity_tier

for speech_interval to speech_intervals
    selectObject: palign
    speech_label$ = Get label of interval: activity_tier, speech_interval
    if speech_label$ == "speech"

        # Sentence
        speech_start = Get start time of interval: activity_tier, speech_interval
        speech_end = Get end time of interval: activity_tier, speech_interval
        selectObject: textgrid
        Insert boundary: sentence_tier, speech_start
        Insert boundary: sentence_tier, speech_end
        selectObject: sentences
        this_sentence = Get interval at time: 1, speech_start
        this_sentence$ = Get label of interval: 1, this_sentence
        if this_sentence$ == "#"
          this_sentence$ = Get label of interval: 1, this_sentence + 1
        endif
        selectObject: textgrid
        this_sentence_new = Get interval at time: sentence_tier, speech_start
        Set interval text: sentence_tier, this_sentence_new, this_sentence$

        # Word
        selectObject: palign
        word_1 = Get interval at time: tokens_tier, speech_start
        word_interval = word_1 + 2
        word$ = Get label of interval: tokens_tier, word_interval
        word_start = Get start time of interval: tokens_tier, word_interval
        word_end = Get end time of interval: tokens_tier, word_interval
        selectObject: textgrid
        Insert boundary: word_tier, word_start
        Insert boundary: word_tier, word_end
        word_new = Get interval at time: word_tier, word_start
        Set interval text: word_tier, word_new, word$

        # Phones
        selectObject: palign
        p_1 = Get interval at time: phon_tier, word_start
        p_1$ = Get label of interval: phon_tier, p_1
        p_1_start = Get start time of interval: phon_tier, p_1
        if p_1_start != word_start
          appendInfoLine: "'tab$'Found a misaligned phone at interval 'p_1'!"
        endif
        p_2_start = Get end time of interval: phon_tier, p_1
        p_2$ = Get label of interval: phon_tier, p_1 + 1
        p_3_start = Get start time of interval: phon_tier, p_1 + 2
        p_3_end = Get end time of interval: phon_tier, p_1 + 2
        p_3$ = Get label of interval: phon_tier, p_1 + 2
        selectObject: textgrid
        Insert boundary: phones_tier, p_1_start
        Insert boundary: phones_tier, p_2_start
        Insert boundary: phones_tier, p_3_start
        Insert boundary: phones_tier, p_3_end
        p_1_new = Get interval at time: phones_tier, p_1_start
```

```
        Set interval text: phones_tier, p_1_new, p_1$
        Set interval text: phones_tier, p_1_new + 1, p_2$
        Set interval text: phones_tier, p_1_new + 2, p_3$

        @detect: p_1_start, p_2_start
        @detect: p_3_start, p_3_end

    endif
endfor

selectObject: textgrid
Save as text file: "'mono_dir$'/'speaker$'-annotation.TextGrid"
```

```
procedure detect: start_time, end_time
  selectObject: sound
  sound_consonant = Extract part: start_time, end_time,
      ..."rectangular", 1, "yes"

  <<<filter>>>

  <<<plosion index>>>

  selectObject: textgrid
  if burst_onset <> undefined
      Insert point: release_tier, burst_onset, "release"
  endif

  removeObject: sound_consonant, sound_filt, spectrum, spectrum_hilbert, sound_hilbert, matrix,
    ...plosion_sound, plosion_pp
endproc
```

Before calculating the plosion index, we need to filter the audio and perform a Hilbert transform.

```
Filter (pass Hann band): 400, 0, 100
sound_filt = selected("Sound")

spectrum = To Spectrum: "no"
Rename: "original"

spectrum_hilbert = Copy: "hilbert"
# Hibbert transform
Formula: "if row=1 then Spectrum_original[2,col] else -Spectrum_original[1,col] fi"
sound_hilbert = To Sound
# We need the num of samples in "plosion index"
samples = Get number of samples
Formula: "abs(self)"
matrix = Down to Matrix
period = Get column distance
```

We can now calculate the plosion index.

```
# Defaults in @avanthapadmanabha2014
m1_time = 0.006
m2_time = 0.016

for sample from 1 to samples ; the number of samples of sound_hilbert
    current = sample * period
    selectObject: sound_hilbert
    mean_before = Get mean: 1, current - m1_time - m2_time, current - m1_time
```

```
    mean_after = Get mean: 1, current + m1_time, current + m1_time + m2_time
    window_average = (mean_before + mean_after) / 2
    current_value = Get value at time: 1, current, "Sinc70"
    plosion = current_value / window_average

    if plosion == undefined
        plosion = 0
    elif plosion < 3
        plosion = 0
    endif

    selectObject: matrix
    Set value: 1, sample, plosion
endfor

To Sound
plosion_sound = Shift times by: start_time
plosion_pp = To PointProcess (extrema): 1, "yes", "no", "Sinc70"

# To reduce detection error when there is noise in the first part of the consonant
Remove points between: start_time, start_time + 0.015
# The time of the burst onset
burst_onset = Get time from index: 1
```

3 Correct burst onset

This script facilitates the manual correction of burst onset detection by zooming to the relevant intervals in a loop.

```
<<<script header>>>

<<<open textgrid>>>

<<<consonants loop>>>
```

We first ask the user to chose which participant will be processed, and the corresponding sound and annotation TextGrid are loaded.

```
form Select participant
  word speaker en01
endform

mono_dir$ = "../data/raw/mono"

if fileReadable("'mono_dir$'/'speaker$'-annotation-corrected.TextGrid")
  pauseScript: "Corrected annotation found. Continue anyway?"
endif

sound = Read from file: "'mono_dir$'/'speaker$'.wav"
annotation = Read from file: "'mono_dir$'/'speaker$'-annotation.TextGrid"

selectObject: sound, annotation
```

Now we can loop through the consonants, which are on tier 3.

```
cons_tier = 3

selectObject: annotation
consonants_num = Get number of intervals: cons_tier
```

```
for i from 1 to consonants_num - 3
  selectObject: annotation
  consonant$ = Get label of interval: cons_tier, i
  if consonant$ <> ""
    c_start = Get start time of interval: cons_tier, i
    c_end = Get end time of interval: cons_tier, i

    selectObject: sound, annotation
    View & Edit
    editor: annotation
      Select: c_start, c_end
      Zoom to selection
      pauseScript: "Annotate then continue"
      Close
    endeditor

    selectObject: annotation
    c_start = Get start time of interval: cons_tier, i + 2
    c_end = Get end time of interval: cons_tier, i + 2

    selectObject: sound, annotation
    View & Edit
    editor: annotation
      Select: c_start, c_end
      Zoom to selection
      pauseScript: "Annotate then continue"
      Close
    endeditor

    selectObject: annotation
    Save as text file: "'mono_dir$'/'speaker$'-annotation-corrected.TextGrid"

    i += 3
  endif
endfor
```

4 Get measurements

This is the script that extracts the duration measurements from the corrected annotation TextGrids.

```
<<<script header>>>

<<<initialise results>>>

<<<speakers loop>>>
```

```
mono_dir$ = "../data/raw/mono"
results_dir$ = "../datasets"
results_file$ = "'results_dir$'/english-durations.csv"
results_header$ = "speaker,sentence,sentence_ons,sentence_off,c1_rel,c2_rel,v1_ons,v1_off"
writeFileLine: results_file$, results_header$

file_list = Create Strings as file list: "file_list", "'mono_dir$'/*-annotation-corrected.TextGrid"
textgrid_num = Get number of strings
```

```
sentence_tier = 1
phones_tier = 3
release_tier = 4
```

```
for speaker from 1 to textgrid_num
  selectObject: file_list
  annotation$ = Get string: speaker

  annotation = Read from file: "'mono_dir$'/'annotation$'"
  speaker$ = annotation$ - "-annotation-corrected.TextGrid"
  phones_num = Get number of intervals: phones_tier

  <<<annotation loop>>>

  removeObject: annotation

endfor
```

```
for phone from 1 to phones_num - 3
  label$ = Get label of interval: phones_tier, phone

  if label$ != ""
    c1_start = Get start time of interval: phones_tier, phone
    c1_end = Get end time of interval: phones_tier, phone
    sentence = Get interval at time: sentence_tier, c1_start
    sentence$ = Get label of interval: sentence_tier, sentence
    sentence_start = Get start time of interval: sentence_tier, sentence
    sentence_end = Get end time of interval: sentence_tier, sentence

    c1_rel_i = Get nearest index from time: release_tier, c1_start
    c1_rel = Get time of point: release_tier, c1_rel_i
    if c1_rel < c1_start or c1_rel > c1_end
      c1_rel = undefined
    endif

    v1 = phone + 1

    v1_start = Get start time of interval: phones_tier, v1
    v1_end = Get end time of interval: phones_tier, v1

    c2 = phone + 2

    c2_start = Get start time of interval: phones_tier, c2
    c2_end = Get end time of interval: phones_tier, c2

    c2_rel_i = Get nearest index from time: release_tier, c2_start
    c2_rel = Get time of point: release_tier, c2_rel_i
    if c2_rel < c2_start or c2_rel > c2_end
      c2_rel = undefined
    endif

    results_line$ = "'speaker$','sentence$','sentence_start','sentence_end',
      ...'c1_rel','c2_rel','v1_start','v1_end'"
    appendFileLine: results_file$, results_line$

    phone += 3
  endif

endfor
```

5 Script header

```
#######################################
# This is a script from the project 'Vowel duration and consonant voicing: An
# articulatory study', subproject 'English'.
# Author: Stefano Coretta.
#######################################
# MIT License
#
# Copyright (c) 2016-2019 Stefano Coretta
#
# Permission is hereby granted, free of charge, to any person obtaining a copy
# of this software and associated documentation files (the "Software"), to deal
# in the Software without restriction, including without limitation the rights
# to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
# copies of the Software, and to permit persons to whom the Software is
# furnished to do so, subject to the following conditions:
#
# The above copyright notice and this permission notice shall be included in all
# copies or substantial portions of the Software.
#
# THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
# IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
# FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
# AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
# LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
# OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
# SOFTWARE.
#######################################
#
# !!! WARNING !!!
#
# This script is generated automatically, DO NOT EDIT
#
#######################################
```