Documentation of data processing of *Vowel duration, voicing duration, and vowel height: Acoustic and articulatory data from Italian*

Stefano Coretta

2019-05-21

1 Pre-process data for force-alignment

```
<<<script header>>>

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

<<<file loop>>>
```

The files in `raw/` are read and processed for alignment.

```
for file from 1 to files
    select Strings file_list
    file$ = Get string: file
    sound = Read from file: "'raw$'/'file$'"
    file_name$ = selected$("Sound")

    # The EGG output signal is inverted
    Multiply: -1

    # Filter audio
    audio = Extract one channel: 1

    audio_f = Filter (pass Hann band): 40, 10000, 100

    Save as WAV file: "'derived$'/audio/'file_name$'.wav"

    removeObject: audio, audio_f

    # Filter EGG
    selectObject: sound

    egg = Extract one channel: 2

    egg_f = Filter (pass Hann band): 40, 10000, 100

    Save as WAV file: "'derived$'/egg/'file_name$'_egg.wav"

    removeObject: egg, egg_f
endfor
```

For each file, read the file, extract the left channel (audio), filter within range 40-10000 Hz, save the file in `derived/`.

2 Stop release detection

This script detects the release of C1 and C2. The algorythm is based on @avanthapadmanabha2014.

```
<<<script header>>>

<<<file loop>>>

<<<findRelease>>>

appendInfoLine: "newline$Done!"
```

```
stereo$ = "../data/raw/stereo"
audio$ = "../data/raw/audio"

Create Strings as file list: "tg_list", "'stereo$'/*-palign-corrected.TextGrid"
tg_number = Get number of strings

writeInfoLine: "'tg_number' files found.'newline$'Starting now.'newline$'"

for file from 1 to tg_number

  selectObject: "Strings tg_list"
  file$ = Get string: file
  Read from file: "'stereo$'/'file$'"
  palign = selected("TextGrid")
  speaker$ = file$ - "-palign-corrected.TextGrid"

  appendInfoLine: "Processing 'speaker$'..."

  <<<find release>>>

  removeObject: palign, sound, textgrid

endfor
```

The following procedure defines the algorithm.

```
###################
# Define findRelease procedure
###################

procedure findRelease: .start_time, .end_time, .label$
  selectObject: sound

  .sound_consonant = Extract part: .start_time, .end_time,
    ..."rectangular", 1, "yes"

  <<<hilbert>>>

  <<<plosion index>>>

  removeObject: .sound_consonant, sound_band, spectrum, spectrum_hilbert, sound_hilbert, matrix, poi

endproc
```

To calculate the plosion index, it is first necessary to create the hilbert transform of the sound.

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

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

spectrum_hilbert = Copy: "hilbert"
Formula: "if row=1 then Spectrum_original[2,col] else -Spectrum_original[1,col] fi"
sound_hilbert = To Sound
.samples = Get number of samples
Formula: "abs(self)"
matrix = Down to Matrix
.period = Get column distance
```

We can now calculate the plosion index.

```
.m1_time = 0.006
.m2_time = 0.016

for .sample from 1 to .samples
  .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

matrix_sound = To Sound
Shift times by: .start_time
pointprocess = To PointProcess (extrema): 1, "yes", "no", "Sinc70"
.half_consonant = .start_time + ((.end_time - .start_time) / 3) * 2
Remove points between: .start_time, .half_consonant
.release = Get time from index: 1

selectObject: textgrid
if .release <> undefined
  Insert point: 1, .release, .label$
endif
```

We start by identifying the inverval that corresponds to C2.

```
speech_intervals = Get number of intervals: 3
sound = Read from file: "'audio$'/'speaker$'.wav"
textgrid = To TextGrid: "releases","releases"

for speech_interval to speech_intervals

  selectObject: palign
```

```
  speech_label$ = Get label of interval: 3, speech_interval

  if speech_label$ == "speech"
    speech_start = Get start time of interval: 3, speech_interval
    frame_interval = Get interval at time: 2, speech_start
    word_1$ = Get label of interval: 2, frame_interval

    if word_1$ == "ha"
      frame_end = Get end time of interval: 2, frame_interval + 1
      c1_interval = Get interval at time: 1, frame_end
      c2_interval = c1_interval + 2
    else
      frame_end = Get end time of interval: 2, frame_interval
      c1_interval = Get interval at time: 1, frame_end
      c2_interval = c1_interval + 2
    endif


    c1_start = Get start time of interval: 1, c1_interval
    c1_end = Get end time of interval: 1, c1_interval
    c2_start = Get start time of interval: 1, c2_interval
    c2_end = Get end time of interval: 1, c2_interval

    @findRelease: c1_start, c1_end, "release_c1"

    @findRelease: c2_start, c2_end, "release_c2"
  endif

endfor

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

3 Voice onset/offset detection

This script finds the onsent and offset of the voicing interval that includes V1.

```
<<<script header>>>

<<<egg loop>>>

appendInfoLine: "Done!"

<<<smoothing>>>
```

Each EGG file is smoothed with a weighted moving average and a VUV textgrid is created.

```
stereo$ = "../data/raw/stereo"
egg$ = "../data/raw/egg"

Create Strings as file list: "tg_list", "'stereo$'/*-palign-corrected.TextGrid"
tg_number = Get number of strings

writeInfoLine: "Found 'tg_number' files.'newline$'Starting now!'newline$'"

for file from 1 to tg_number

  selectObject: "Strings tg_list"
```

```
  file$ = Get string: file

  speaker$ = file$ - "-palign-corrected.TextGrid"
  appendInfoLine: "Processing 'speaker$'..."

  Read from file: "'stereo$'/'file$'"
  palign = selected("TextGrid")
  Read from file: "'egg$'/'speaker$'_egg.wav"
  egg = selected("Sound")

  <<<vuv>>>

endfor
```

The following chunk contains the functions for extracting the VUV intervals. The EGG signal is smoothed using a weighted average filter, with width = 11. A PointProcess object is created which indicates the individual glottal periods. From this object, a TextGrid with voiced/unvoiced (VUV) intervals is obtained. Since the delay created by the smoothing is corrected by shifting the times in the EGG, the times of the TextGrid are extended by the same lag at the beginning of the TextGrid (so that the EGG file and the TextGrid start at 0). The TextGrid is written in `./data/raw/egg/`.

```
appendInfoLine: "'tab$'Smoothing."

@smoothing: 11

egg_smoothed = selected("Sound")

Create Sound from formula: "silence", 1, 0, time_lag, 44100, "0"
silence = selected("Sound")
selectObject: egg_smoothed, silence
Save as WAV file: "'egg$'/'speaker$'_egg_smoothed.wav"

appendInfoLine: "'tab$'Getting VUV"

selectObject: egg_smoothed

noprogress To PointProcess (periodic, cc): 75, 600

To TextGrid (vuv): 0.02, 0

# Extend time: time_lag, "Start"

Write to text file: "'egg$'/'speaker$'-vuv.TextGrid"
```

The following chunk defines the smoothing procedure. The EGG signal is smoothed by using the weighted average filter formula. This kind of smoothing filter creates a small delay in the signal, which is corrected by shifting the times of the signal.

```
procedure smoothing : .width
    .weight = .width / 2 + 0.5

    .formula$ = "( "

    for .w to .weight - 1
        .formula$ = .formula$ + string$(.w) + " * (self [col - " + string$(.w) + "] +
            ...self [col - " + string$(.w) + "]) + "
    endfor

    .formula$ = .formula$ + string$(.weight) + " * (self [col]) ) / " +
```

```
        ...string$(.weight ^ 2)

    Formula: .formula$

    .sampling_period = Get sampling period
    time_lag = (.width - 1) / 2 * .sampling_period
    Shift times by: time_lag
endproc
```

4 Merge TextGrids

This script merge the TextGrids with the IPUs, force-alignment, VUVs, and releases into one TextGrid, for each speaker. The TextGrids with IPUs and releases are merges as they are, while for the TextGrids with force-alignment and VUVs only the relevant intervals/points are copied in the merged textgrid

```
<<<script header>>>

<<<textgrid loop>>>

appendInfoLine: "'newline$'Done!"
```

The script searches for all the `.txt` files in `data/raw/stereo/` and then merges the TextGrids.

```
stereo$ = "../data/raw/stereo"
audio$ = "../data/raw/audio"
egg$ = "../data/raw/egg"

txt_list = Create Strings as file list: "txt_list", "'stereo$'/*.txt"
n_files = Get number of strings

writeInfoLine: "'n_files' files found. Start processing.'newline$'"

for file from 1 to n_files
  selectObject: txt_list
  file$ = Get string: file
  speaker$ = file$ - ".txt"
  appendInfoLine: "Processing 'speaker$'"

  ipu = Read from file: "'audio$'/'speaker$'.TextGrid"

  palign = Read from file: "'stereo$'/'speaker$'-palign-corrected.TextGrid"
  vuv = Read from file: "'egg$'/'speaker$'-vuv-corrected.TextGrid"

  <<<palign vuv loop>>>

  releases = Read from file: "'audio$'/'speaker$'-rel-corrected.TextGrid"

  selectObject: ipu, palign_2, vuv_2, releases
  merged = Merge
  Save as text file: "'stereo$'/'speaker$'-align.TextGrid"

  removeObject: ipu, palign, palign_2, vuv, vuv_2, releases, merged

endfor
```

```
selectObject: palign
n_intervals = Get number of intervals: 3
end_time = Get end time
```

```
palign_2 = Create TextGrid: 0, end_time, "sentence word segments", ""
vuv_2 = Create TextGrid: 0, end_time, "vuv", ""

for sentence from 1 to n_intervals

  selectObject: palign
  speech$ = Get label of interval: 3, sentence

  if speech$ == "speech"

    speech_start = Get start time of interval: 3, sentence
    speech_end = Get end time of interval: 3, sentence
    first_word = Get interval at time: 2, speech_start
    first_word$ = Get label of interval: 2, first_word

    if first_word$ == "#"
      first_word = first_word + 1
      appendInfoLine: "'tab$'Misaligned sentence at 'speech_start's"
    endif

    if first_word$ == "ha"
      frame_end = Get end time of interval: 2, first_word + 1
    else
      frame_end = Get end time of interval: 2, first_word
    endif

    word_start = frame_end
    word = Get interval at time: 2, word_start
    word$ = Get label of interval: 2, word
    word_end = Get end time of interval: 2, word

    c1 = Get interval at time: 1, word_start
    c1$ = Get label of interval: 1, c1

    if c1$ == "e" or c1$ == "o"
      c1 = c1 + 1
      appendInfoLine: "'tab$'Misaligned word at 'speech_start's"
    endif

    c1_end = Get end time of interval: 1, c1
    v1_end = Get end time of interval: 1, c1 + 1
    v1$ = Get label of interval: 1, c1 + 1
    c2_end = Get end time of interval: 1, c1 + 2
    c2$ = Get label of interval: 1, c1 + 2
    v2$ = Get label of interval: 1, c1 + 3

    selectObject: palign_2
    Insert boundary: 1, speech_start
    Insert boundary: 1, speech_end
    sentence_2 = Get interval at time: 1, speech_start
    Set interval text: 1, sentence_2, "sentence"
    Set interval text: 1, sentence_2 - 1, "#"

    Insert boundary: 2, word_start
    Insert boundary: 2, word_end
    word_2 = Get interval at time: 2, word_start
    Set interval text: 2, word_2, word$
```

```
    Insert boundary: 3, word_start
    Insert boundary: 3, c1_end
    c1_2 = Get interval at time: 3, word_start
    Set interval text: 3, c1_2, c1$

    Insert boundary: 3, v1_end
    v1_2 = Get interval at time: 3, c1_end
    Set interval text: 3, v1_2, v1$

    Insert boundary: 3, c2_end
    c2_2 = Get interval at time: 3, v1_end
    Set interval text: 3, c2_2, c2$

    Insert boundary: 3, word_end
    v2_2 = Get interval at time: 3, c2_end
    Set interval text: 3, v2_2, v2$

    <<<vuv loop>>>

  elsif speech$ == ""

    speech_start = Get start time of interval: 3, sentence
    speech_end = Get end time of interval: 3, sentence

    selectObject: palign_2
    Insert boundary: 1, speech_start
    Insert boundary: 1, speech_end
    sentence_2 = Get interval at time: 1, speech_start
    Set interval text: 1, sentence_2, ""
    Set interval text: 1, sentence_2 - 1, "#"

  endif

endfor
```

```
selectObject: vuv

v1_mid = c1_end + ((v1_end - c1_end) / 2)

vuv_i = Get interval at time: 1, v1_mid
vuv_label$ = Get label of interval: 1, vuv_i

if vuv_label$ == "V"
  voice_start = Get start time of interval: 1, vuv_i
  voice_end = Get end time of interval: 1, vuv_i

  selectObject: vuv_2
  Insert boundary: 1, voice_start
  Insert boundary: 1, voice_end
  voice = Get interval at time: 1, voice_start
  Set interval text: 1, voice, "voicing"
endif
```

5 Extract measurements

```
<<<script header>>>
```

```
<<<align loop>>>

appendInfoLine: "'newline$'Done!"
```

```
stereo$ = "../data/raw/stereo"
result_file$ = "../datasets/measurements.csv"
header$ = "speaker,ipu,stimulus,sentence_ons,sentence_off,word_ons,word_off,v1_ons,c2_ons,v2_ons,voi

writeFileLine: result_file$, header$

align_list = Create Strings as file list: "align_list", "'stereo$'/*-align.TextGrid"
n_files = Get number of strings

writeInfoLine: "'n_files' files found. Start processing.'newline$'"

for textgrid from 1 to n_files

  selectObject: align_list
  file$ = Get string: textgrid
  speaker$ = file$ - "-align.TextGrid"
  appendInfoLine: "Processing 'speaker$'"

  align = Read from file: "'stereo$'/'file$'"

  n_sentences = Get number of intervals: 3

  for interval from 1 to n_sentences - 1

    selectObject: align
    interval$ = Get label of interval: 3, interval

    if interval$ == "sentence"

      sentence_start = Get start time of interval: 3, interval
      sentence_end = Get end time of interval: 3, interval
      sentence_mid = sentence_start + ((sentence_end - sentence_start) / 2)

      ipu_i = Get interval at time: 1, sentence_mid
      ipu_i$ = Get label of interval: 1, ipu_i
      sentence = Get interval at time: 2, sentence_mid
      sentence$ = Get label of interval: 2, sentence
      sentence$ = replace$(sentence$, """", "'", 2)

      pre_word = Get interval at time: 5, sentence_start
      c1 = pre_word + 1
      c1_start = Get start time of interval: 5, c1
      v1_start = Get start time of interval: 5, c1 + 1
      v1_end = Get end time of interval: 5, c1 + 1
      c2_end = Get end time of interval: 5, c1 + 2
      v2_end = Get end time of interval: 5, c1 + 3

      v1_mid = v1_start + ((v1_end - v1_start) / 2)
      voicing = Get interval at time: 6, v1_mid
      voicing$ = Get label of interval: 6, voicing

      if voicing$ == "voicing"
        voice_start = Get start time of interval: 6, voicing
        voice_end = Get end time of interval: 6, voicing
```

```
        if voice_start < c1_start or voice_end > c2_end
          voice_start = undefined
          voice_end = undefined
        endif

      else
        voice_start = undefined
        voice_end = undefined
      endif

      c1_rel_i = Get nearest index from time: 7, c1_start
      c1_rel = Get time of point: 7, c1_rel_i

      if c1_rel < c1_start or c1_rel > v1_start
        c1_rel = undefined
      endif

      c2_rel_i = Get nearest index from time: 7, c2_end
      c2_rel = Get time of point: 7, c2_rel_i

      if c2_rel < v1_end or c2_rel > c2_end
        c2_rel = undefined
      endif

      results$ = "'speaker$','ipu_i$','sentence$','sentence_start','sentence_end','c1_start','v2_end

      appendFileLine: result_file$, results$

    elsif interval$ == ""

      sentence_start = Get start time of interval: 3, interval
      sentence_end = Get end time of interval: 3, interval
      sentence_mid = sentence_start + ((sentence_end - sentence_start) / 2)

      ipu_i = Get interval at time: 1, sentence_mid
      ipu_i$ = Get label of interval: 1, ipu_i
      sentence = Get interval at time: 2, sentence_mid
      sentence$ = Get label of interval: 2, sentence
      sentence$ = replace$(sentence$, """", "'", 2)

      results$ = "'speaker$','ipu_i$','sentence$',--undefined--,--undefined--,--undefined--,--undefi

      appendFileLine: result_file$, results$

    endif

  endfor

  removeObject: align

endfor
```

6 Script header

```
#######################################
# This is a script from the project 'Vowel duration and consonant voicing: An
```

```
# articulatory study', Stefano Coretta
#######################################
# MIT License
#
# Copyright (c) 2016-2018 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
#
#######################################
```