eesti_regilaul

April 16, 2022

I don't use folksongs, folksongs use me >Velljo Tormis, *regilaul* revitalist composer

1 Stress in Estonian

2 Kalevala Meter

3 Research Questions

4 Previous Work

5 Hypotheses

6 Methods

This section is outlined as follows: 1. the *regilaul* corpus is described. 1. singers/informants background info 2. physical recording info (collection/collectors) 2. Then, the annotation process is described. 1. beat detective: 1.Logic method 2. Nyquist and Audacity 3. (manual check: definition of a beat) 2. midi to textgrid (annotate silences script) 3. txt file of lyrics to phrases with praatio (check: definition of a phrase, verses instead of choruses decision?) 4. forced aligner: word level (check: definition of word) 5. syllable level (coded for stress and quanitity!) with varbomorf manual check: definition of a syllable (use ictus/off-ictus, duh!) 6. segmental level (run from syllable level) (manual check: definition of each segment and the segment's environmental criteria to be included in analysis) 3. The measurement process is described 1. temporal aspects: 1. vowel duration 2. consonant duration 3. syllable duration 2. spectral aspects: 1. vowel space 2. spectral tilt

1. the *regilaul* corpus is described.
    1. singers/informants background info
    2. physical recording info (collection/collectors)
2. Then, the annotation process is described.
    1. beat detective: 1.Logic method
        2. Nyquist and Audacity
        3. (manual check: definition of a beat)
3. midi to textgrid (annotate silences script)
    3. txt file of lyrics to phrases with praatio (check: definition of a phrase, verses instead of choruses decision?)

[ ]:
```
import os
from os.path import join
from praatio import textgrid
from praatio import praat_scripts
from praatio.utilities.constants import (
    Interval)

praat_exe = "/Applications/Praat.app/Contents/MacOS/Praat"
root = "/Users/sarah/Git/eesti_regilaul_corpus/audio"
input_wavs = join(root,"clicks")
input_tg = join(root, "ictus_tier")
output_tg = join(root,"lyric_tier")

SILENCE_LABEL = "x"
SOUND_LABEL = "ictus"

for fn in os.listdir(root):
        if ".wav" not in fn:
            continue
        fn.beats_to_textgird()
        #then, add the phrase tier
        if ".TextGrid" not in fn:
            continue
        tg = textgrid.openTextgrid(fn,True)
        ictus_dict = tg.tierDict
        myTier = textgrid.IntervalTier("phrase",entryList= ictus_dict[1])
        lyric_lines = open(join(root,fn,".txt")).readlines()
        for interval, line in myTier, lyric_lines:
            #TODO first edge case thing
            interval.text = lyric_lines
        tg.addTier(myTier,[2],"warning")
        tg.save(fn,'long_textgrid')

        #tg.addTier(input_tg,output_tg,"phrase")
#file format: NAME + "beat"
def beats_to_textgird(input_wav):

        name = os.path.splitext(fn)[0]
        #remove "beat" so it matches song file
        tgFn = name.strip("beat") + ".TextGrid"
        praat_scripts.annotateSilences(
         praat_exe, join(input_wavs, fn), join(input_tg, tgFn),100,0,-25,0.1,0.
 ↪1,SILENCE_LABEL,SOUND_LABEL
        )

#def addTier(input_tg, output_tg,tier_label):
        # tg = textgrid.openTextgrid(join(input_tg, fn),True)
```

```
        # myTier = textgrid.IntervalTier(tier_label, entryList= [('0', tg.
↪maxTimestamp,
        # 'test'),],minT=0,maxT=tg.maxTimestamp)a
        # tg.addTier(myTier)
        # tg.save(join(output_tg,fn),'long_textgrid', True)
#finds the txt file that matches the
def add_lines_intervals(input_tg,input_txt,tier_num):
        tg = textgrid.openTextgrid(input_tg,True)
        myTier = tg.tierDict[tier_num]
        lyrics = open(input_txt,"r")
        lyricList = lyrics.readlines()
        dur = (tg.maxTimestamp/len(lyricList))
        start= 0.0
        length = start + dur
        line = 0
        while length <= tg.maxTimestamp:
            tmpInterval = Interval(start,length,lyricList[line])
            myTier.insertEntry(tmpInterval,'replace','warning')
            start = start+dur
            length = length +dur
            line = line + 1
        tg.save(join(input, fn),'long_textgrid',True)




#autoSegmentSpeech(praatEXE, inputWavPath, input_tg)
#markIctus(praatEXE,inputWavPath,input_tg)
#addPhraseTier(input_tg, finalTGPath)
#addIntervalsLyrics(finalTGPath)
#getLyrics(input_tg)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/var/folders/s2/hj078vmj60j2qr4wwsp2bgvm0000gp/T/ipykernel_68630/354519275.py i␣
↪<module>
     18         if ".wav" not in fn:
     19             continue
---> 20         fn.beats_to_textgird()
     21         #then, add the phrase tier
     22         if ".TextGrid" not in fn:

AttributeError: 'str' object has no attribute 'beats_to_textgird'
```

4. forced aligner: word level (check: definition of word)
5. syllable level (coded for stress and quanitity!) with varbomorf manual check: definition of a syllable (use ictus/off-ictus, duh!)

[ ]:
```
from praatio import textgrid
import string
import pandas as pd
def add_syll_tier(grids,out):
    for fn in os.listdir(grids):
        sylly_name = (fn.strip(".TextGrid") + "_sillies.csv")
        if ".TextGrid" not in fn:
            continue
        tg = textgrid.openTextgrid(join(grids,fn), True)
        wordTier = tg.tierDict["word"]
        tmpTier = wordTier.new("varbo")

        syllies = pd.read_csv(join(grids,sylly_name)).to_dict()
        #print(syllies)
        punctuation = string.punctuation
        for entry in tmpTier.entryList:
            if not entry.label:
                continue
            tmpLabel = str(entry.label)
            for p in punctuation:
                cleanlabel = tmpLabel.replace(p,'')
            if syllies.get(cleanlabel):
                newLabel = syllies[cleanlabel]
                tmpInterval = Interval(entry.start,entry.end,newLabel)
                tmpTier.insertEntry(tmpInterval,'replace','silence')
        tg.addTier(tmpTier,3)
        tg.save(join(out,fn),'long_textgrid',True,reportingMode='silence')




grids_in = "/Users/sarah/Git/eesti_regilaul_corpus/grids/syll/"
#dict_in = "/Users/sarah/Git/eesti_regilaul_corpus/grids/syll/009_sillies.csv"
grid_out = "/Users/sarah/Git/eesti_regilaul_corpus/grids/syll/out/"
add_syll_tier(grids_in,grid_out)
```

7 segmental level (run from syllable level) (manual check: definition of each segment and the segment's environmental criteria to be included in analysis)

7.1 The measurement process is described:

1. temporal aspects:
    1. vowel duration
    2. consonant duration
    3. syllable duration
2. spectral aspects:
    1. vowel space
    2. spectral tilt

8 Results

8.1 Ictus and Off-Ictus (song prominence)

8.1.1 temporal

8.1.2 spectral

8.2 Stressed and unstressed syllables (word-level prominence)

8.2.1 temporal

8.2.2 spectral

8.3 Conflicts: stressed syllables in off-ictus position

8.3.1 temporal

8.3.2 spectral

8.4 Conflicts: unstressed syllables in off-ictus position

8.4.1 temporal

8.4.2 spectral

9 Discussion

10 Conclusion