build_songcorpus

October 29, 2022

0.1 Materials

At the start, we have audio and annotated textgrids of **regilaul** songs, annotated for ictus/off-ictus and phrase text, then force-aligned using Praat's built in eSpeak forced aligner for Estonian to word and then segment. Then, we use the estnltk vabamorf package to syllabify the words so that we can annotate the textgrid further with syllable quantity (Estonian has 3) and whether or not it is accented at the word level. We end up with a dataframe containing the data from three of the(Interval) tiers of the textgrid, acquiring duration data for words, individual segments, and (eventually) syllables.

[ ]:
```
import pandas as pd
import parselmouth
from estnltk.vabamorf.morf import syllabify_word
import tgt
import string
import unicodedata

#test method on a single TextGrid:
gridDir2 = "/Users/sarah/Git/regilaul_project/songs/txtgrids/09.TextGrid"



def syllShape(syll,quant):
    syll1 = unicodedata.normalize('NFC',syll)
    #remove punctuation for this method to avoid false CCs
    syll = syll1.strip(string.punctuation)
    prenorms = ['e','o','õ','ö', 'i', 'a', 'u', 'ä', 'ü']
    vowels = []
    for v in prenorms:
        vowels.append(unicodedata.normalize('NFC', v))

    shortC = ['b','g','d', 'j','l','r', 's', 'n','m', 'h']
    #syllables = ['or','jal','ood','mull', 'muks', 'tan']
    onset = ""
    nucleus = ""
    coda = ""
    codapos = (len(syll) -1 )
    shape = ""
```

```
    #all onsets are either null or singleton
    if syll[0] not in vowels:
        onset = "C"
    else:
        onset = "V"
    if quant == 1:
        if len(syll) <= 1 :
            nucleus = ""
            coda = ""
        else:
            nucleus = "V"

    if quant == 2:
        if syll[codapos] not in vowels:
            if syll[codapos] in shortC:
                if syll[codapos-1] in vowels:
                    nucleus = "V"
                    coda = "C"
                elif syll[codapos-1] in shortC:
                    nucleus = "V"
                    coda = "CC"
            else:
                nucleus = "V"
                coda = "CC"
        else:
            nucleus = "V"
            coda = ""



    if quant == 3:
        if syll[codapos] not in vowels:
            if syll[codapos] in shortC:
                if syll[codapos-1] in vowels:
                    nucleus = "V"
                    coda = "VC"
                elif syll[codapos-1] in shortC:
                    nucleus = "V"
                    coda = "CC"
                else:
                    nucleus = "V"
                    coda = "CCC"
            else:
                nucleus = "V"
                coda = "CC"
        else:
            nucleus = "V"
```

```
            coda = "V"

    shape = onset + nucleus + coda


    return shape


def get_duration_labels(textgrid, wordTier,s1,s2,ictusTier):
    #tmp = codecs.open(textgrid,'r','utf-8')
    tmp = tgt.io.read_textgrid(textgrid)
    words = tmp.get_tier_by_name(wordTier)
    firstSyll = tmp.get_tier_by_name(s1)
    secSyll = tmp.get_tier_by_name(s2)
    ictus = tmp.get_tier_by_name(ictusTier)
    segments = []
    wordlist = words.intervals

    for interval in wordlist:
        onset = interval.start_time
        offset = interval.end_time
       # wordms = offset-onset
        word = interval.text
        syllablist = syllabify_word(word,as_dict=True)
        i = 0

        while i < len(syllablist):
            if i >= 2: break
            tmpsy = syllablist[i]
            ortho = tmpsy.get('syllable')
            ortho = ortho.strip(string.punctuation)
            q = tmpsy.get('quantity')
            a = tmpsy.get('accent')
            shape = syllShape(ortho,q)

            if i == 0:
                tmpinterval = firstSyll.
 ↪get_annotations_between_timepoints(onset,offset)
            elif i == 1:
                tmpinterval = secSyll.
 ↪get_annotations_between_timepoints(onset,offset)

            #skip syllables with no annotations in the analysis tiers
            if len(tmpinterval)==0:
                i+= 1
                break
            for vowel in tmpinterval:
```

```
                segment = vowel.text
                vOnset = vowel.start_time
                vOffset = vowel.end_time
                dur = vOffset-vOnset
                vMidpoint = vOffset - (dur/2)
                tmpick = ictus.get_annotations_by_time(vMidpoint)
                if len(tmpick) > 0 :
                    ick = tmpick[0].text
                else: ick = "off"
                row = (word,ortho,shape,i,segment,q,a,ick,dur,vMidpoint)
                segments.append(row)

            i+= 1

    nu_df = pd.
↪DataFrame(segments,columns=["word","syll","shape","index","segment","quantity","stressed","
    return nu_df
```

[ ]:
```
#test duration label method:

onetwo_df = get_duration_labels(gridDir2,"word","s1","s2","ictus")
onetwo_df
```

[ ]:

	word	syll	shape	index	segment	quantity	stressed	ictus	\
0	Lõpe,	lõ	CV	0		2	1	ictus	
1	lõpe,	lõ	CV	0		2	1	ictus	
2	lõpe,	pe	CV	1	e	2	0	x	
3	linakene,	li	CV	0	i	1	1	ictus	
4	linakene,	na	CV	1		2	0	x	
5	kui	kui	CVV	0	ui	3	1	ictus	
6	sa	sa	CVV	0	a	3	1	x	
7	lõpe,	lõ	CV	0		2	1	ictus	
8	siia	sii	CV	0	i	2	1	ictus	
9	siia	a	V	1	ja	1	0	x	
10	jätan,	jä	CV	0	æ	2	1	ictus	
11	jätan,	tan	CVC	1	a	2	0	x	
12	siia	sii	CV	0	i	2	1	ictus	
13	siia	a	V	1	a	1	0	x	
14	jätan	jä	CV	0	æ	2	1	ictus	
15	jätan	tan	CVC	1	a	2	0	x	
16	tsirgu	tsir	CVC	0	i	2	1	ictus	
17	tsirgu	gu	CV	1	u	1	0	x	
18	süia,	sü	CV	0	yi	2	1	ictus	
19	süia,	a	VV	1	a	2	0	x	
20	pääsukesel	pää	CV	0	æ	2	1	ictus	

21	pääsukesel	su	CV	1	u	2	0	x
22	pääle	pää	CV	0	æ	2	1	ictus
23	pääle	le	CV	1	e	1	0	x
24	tulla.	tul	CVC	0	u	2	1	ictus
25	tulla.	la	CV	1	a	2	0	x
26	Jätän	jä	CV	0	æ	2	1	ictus
27	Jätän	tän	CVC	1	æ	2	0	x
28	päivä	päi	CV	0	æi	2	1	x
29	päivä	vä	CV	1	æ	1	0	ictus
30	palada,	pa	CV	0	a	1	1	x
31	palada,	la	CV	1		1	0	ictus
32	jätän	jä	CV	0	æ	2	1	ictus
33	jätän	tän	CVC	1	æ	2	0	x
34	vihma	vih	CVC	0	i	2	1	x
35	vihma	ma	CV	1	a	1	0	ictus
36	valada,	va	CV	0	a	1	1	x
37	valada,	la	CV	1		1	0	ictus
38	ligi	li	CV	0	i	1	1	ictus
39	ligi	gi	CV	1		1	0	x
40	maada	maa	CV	0	a	2	1	ictus
41	maada	da	CV	1	a	1	0	x
42	ligunema.	li	CV	0	i	1	1	ictus
43	ligunema.	gu	CV	1		1	0	x
44	Perenaine,	pe	CV	0	e	1	0	ictus
45	Perenaine,	re	CV	1		1	0	x
46	linnukene,	lin	CVC	0	i	2	1	ictus
47	linnukene,	nu	CV	1	u	2	0	x
48	tule	tu	CV	0	u	1	1	ictus
49	tule	le	CV	1		1	0	x
50	talu	ta	CV	0	a	1	1	ictus
51	talu	lu	CV	1		1	0	x
52	tuastagi,	tuas	CVVC	0	u	3	1	ictus
53	tuastagi,	ta	CV	1	a	1	1	x
54	orjal	or	VVC	0	o	2	1	ictus
55	orjal	jal	CVC	1	a	2	0	x
56	oodakuta!	ood	VVVC	0	o	3	1	ictus
57	oodakuta!	a	VV	1	a	2	1	x

	duration	midpoint
0	0.221401	0.398469
1	0.208552	1.227423
2	0.293176	1.714267
3	0.222958	2.045270
4	0.312941	2.379412
5	0.382281	5.463635
6	0.165045	5.846221
7	0.216827	6.230085

8	0.287818	7.058594
9	0.373816	7.389411
10	0.219520	7.823673
11	0.290910	8.275510
12	0.372053	10.502911
13	0.196502	10.880522
14	0.170946	11.222064
15	0.213219	11.614226
16	0.268399	12.018500
17	0.254819	12.451586
18	0.460370	12.953363
19	0.261608	13.374466
20	0.406963	15.458911
21	0.200522	15.860154
22	0.382602	17.079941
23	0.263279	17.460234
24	0.293701	17.852665
25	0.313637	18.341674
26	0.182297	20.353781
27	0.195067	20.710641
28	0.386169	21.524724
29	0.206285	21.882764
30	0.289298	22.239541
31	0.238170	22.580172
32	0.174812	25.295391
33	0.172461	25.646129
34	0.285254	26.356856
35	0.246190	26.784018
36	0.247466	27.142424
37	0.253880	27.472883
38	0.343919	30.391618
39	0.279709	30.796216
40	0.256067	31.145193
41	0.262658	31.591090
42	0.293766	31.967375
43	0.293904	32.386313
44	0.358602	35.551608
45	0.287859	35.910022
46	0.241380	37.009935
47	0.235123	37.401839
48	0.318001	40.261886
49	0.230516	40.622335
50	0.314768	40.999864
51	0.231065	41.351082
52	0.309984	41.729664
53	0.249898	42.009606
54	0.247051	45.672860

```
55  0.217923  46.147530
56  0.317374  46.493654
57  0.220725  46.881315
```

[ ]:
```
#test syllable shape method:
syll = "biip"
syllShape(syll,3)
```

[ ]:
```
'CVCC'
```

0.2 Adding Spectral data

now that we have the duration data from the textgrid, we can query specific timepoints for information about the acoustic signal. The following function uses the midpoint (which we snagged while we were making the dataframe above) and get the first three formants(Hz) for each segment.

[ ]:
```
import parselmouth

test = "/Users/sarah/Git/regilaul_project/songs/wavs_aligned/65.wav"

def get_formants(syl_dur_df, wave):
    song = parselmouth.Sound(wave)
    formant = song.to_formant_burg()
    f1 = []
    f2 = []
    for float in syl_dur_df.midpoint:
        time = float
        first = formant.get_value_at_time(1,time)
        f1.append(first)
        second = formant.get_value_at_time(2, time)
        f2.append(second)
    syl_dur_df["f1"] = f1
    syl_dur_df["f2"] = f2
    return syl_dur_df
nu_df = get_formants(onetwo_df,test)
nu_df.head()
```

[ ]:
```
        word syll shape  index segment  quantity  stressed  ictus  duration  \
0      Lõpe,   lõ    CV      0                2         1  ictus  0.221401
1      lõpe,   lõ    CV      0                2         1  ictus  0.208552
2      lõpe,   pe    CV      1       e        2         0      x  0.293176
3  linakene,   li    CV      0       i        1         1  ictus  0.222958
4  linakene,   na    CV      1                2         0      x  0.312941

   midpoint          f1           f2
0  0.398469  537.955860  2636.111467
1  1.227423  376.720589  1061.124425
2  1.714267  511.874473  1462.904048
```

```
3  2.045270  477.258475  1491.004187
4  2.379412  793.983102  1348.612283
```

[ ]:
```
nu_df['shape'].unique()
```

[ ]:
```
array(['CV', 'CVV', 'V', 'CVC', 'VV', 'CVVC', 'VVC', 'VVVC'], dtype=object)
```

[ ]:
```
from os.path import join
#runs a for loop over a directory using the above-specified functions

test = "/Users/sarah/Git/regilaul_project/songs/txtgrids"
songs = "/Users/sarah/Git/regilaul_project/songs/wavs_aligned"
datum = "/Users/sarah/Git/regilaul_project/songs/datum/"
for fn in os.listdir(test):
    if '.TextGrid' not in fn:
        continue
    n = fn.strip('.TextGrid')
    wave = join(songs, n + '.wav')
    data = join(datum, n)
    data_file = open(data +".csv",'w')
    #make a dataframe with the interval tiers of the textgrid
    tmp = pd.DataFrame(get_duration_labels(join(test,fn),␣
 ↪"word","s1","s2","ictus"))
    #add the formant data to the dataframe
    nu_df = get_formants(tmp,wave)
    #print(nu_df.head())
    nu_df.to_csv(data_file)
    data_file.close()
```

1 Now we put it into a big pile!

Here we concatenate all the data we have so far into one large pandas dataframe. At this point, we can keep annotating songs for the corpus, and as textgrids are finished we can run the scripts above to add them into the larger dataset. We're also gonna take the opportunity to add some metadata to the dataframes: fileid(song) and performer initials as potential grouping factors.

[ ]:
```
import os
import pandas as pd
import statsmodels.formula.api as smf
folder = "/Users/sarah/Git/regilaul_project/songs/datum"
meta =  pd.read_csv("/Users/sarah/Git/regilaul_project/songs/song_metadata.csv")


songs_dfs = []
for fn in os.listdir(folder):
    if '.csv' not in fn: continue
    whole_name = os.path.join(folder,fn)
```

```
    song_df = pd.read_csv(whole_name)
    fileid1 = fn.strip('.csv')
    fileid = int(fileid1)
    row = meta.index[meta['track'] == fileid].tolist()
    if len(row) !=0 :
        performer = meta.performer[row[0]]
    else: performer = "couldn't get match"
    for index in song_df:
        song_df['fileid'] = fileid
        song_df['performer'] = performer


    songs_dfs.append(song_df)

big_frame = pd.concat(songs_dfs, ignore_index=True)
#move ictus-off replacement to here!
big_frame.ictus = big_frame.ictus.replace("x","off")

# big_frame.describe()




# big_frame
#clean_frame = pd.
 ↪DataFrame(big_frame[['quantity','stress','segment','seg_duration','ictus','euc','fileid','p
#clean_frame.head()

big_frame
```

[ ]:

	Unnamed: 0	word	syll	shape	index	segment	quantity	stressed	\
0	0	sain	sain	CVVC	0	ai	3	1	
1	1	mal´lika,	mal´	CVCC	0	a(i)	2	1	
2	2	mal´lika,	li	CV	1	i	2	0	
3	3	mal´likast	mal´	CVCC	0	a(i)	2	1	
4	4	mal´likast	li	CV	1	i	2	0	
..	…	…	…	…	…	…	…	…	
753	73	südant	dant	CVCC	1		3	1	
754	74	sülle	sül	CVC	0	yl	2	1	
755	75	sülle	le	CV	1	e	1	0	
756	76	rabadaie.	ra	CV	0	a	1	0	
757	77	rabadaie.	ba	CV	1		1	0	

	ictus	duration	midpoint	f1	f2	fileid	performer
0	ictus	0.217500	4.263002	714.071536	1129.944778	41	LK
1	off	0.179055	4.614738	946.021821	1448.348662	41	LK

```
2    ictus  0.130760   4.936603   574.096119  1864.237985      41        LK
3    ictus  0.193353   5.891856  1075.429489  1570.329209      41        LK
4      off  0.122714   6.186455   589.298998  1688.844658      41        LK
..     …       …         …           …          …     …       …
753    off  0.172132  94.811813   654.258343  1112.070873      65        LK
754  ictus  0.336953  95.331866   640.080936  1706.830954      65        LK
755    off  0.308579  95.744141   721.464844  1631.467177      65        LK
756  ictus  0.333783  96.176488   935.850426  1478.448942      65        LK
757    off  0.319876  96.618055   824.139440  1571.052387      65        LK

[758 rows x 15 columns]
```

[ ]:
```
#big_frame['shape'].unique()
```

[ ]:
```
big_frame['index'] = big_frame['index'].astype(object)
big_frame['quantity'] = big_frame['quantity'].astype(object)
big_frame['stressed'] = big_frame['stressed'].astype(object)
big_frame['shape'] = big_frame['shape'].astype(object)

big_frame['fileid'] = big_frame['fileid'].astype(object)

corpus_data = open('regilaul_vowels.csv','w')
big_frame.to_csv(corpus_data)
corpus_data.close()
big_frame.dtypes
```

[ ]:
```
Unnamed: 0      int64
word           object
syll           object
shape          object
index          object
segment        object
quantity       object
stressed       object
ictus          object
duration      float64
midpoint      float64
f1            float64
f2            float64
fileid         object
performer      object
dtype: object
```