数据标注

标注工具

praat 和转化工具

- praat 官网：http://www.fon.hum.uva.nl/praat/download_linux.html
- 转化工具地址：https://github.com/ecooper7/label-convert/blob/master/textgrid_to_lab.py

安装教程

- sudo apt-get install praat
- sudo apt-get install fonts-sil-charis
- sudo apt-get install fonts-sil-doulos
- **key key**：for lab in labels/mono/*.lab; do sed -i "s/ $//g" ${lab}; done

prrat 标注mono label

praat是一个强大的音频编辑软件，可以是实现和编辑以下文件：（最关键是可视化音频）
SoundEditor
LongSoundEditor
TextGridEditor
ManipulationEditor
SpectrumEditor
PitchEditor
PointEditor
PitchTierEditor
IntensityTierEditor
DurationTierEditor
SpectrogramEditor
ArtwordEditor
OTGrammarEditor

(any type: Inspect)

praat 安装：

```
sudo apt install praat
sudo apt-get install fonts-sil-charis
sudo apt-get install fonts-sil-doulos
```

praat 教程

普通话歌声合成初步方案
一、数据准备（30天）：
1. 多个说话人的清唱歌声文件，WAV格式，44.8kHz采样，16位量化。文件数至少100个。
2. 每首歌对应的MIDI文件。
3. 每首歌的歌词文件，txt格式。
要求：覆盖所有音节、声韵母均匀分布、覆盖所有调且均匀分布。
对录音文件进行手工切分和标注

二、上下文特征：
采用四层上下文相关标注格式:
1. 声韵母层：当前声母或韵母，前，前前，后，后后声韵母
2. 音节层（音调层）：当前音节的音高，当前音调的信息；前一音节；后一音节；
3. 短语层：当前短语中音调的数目；长度（2/3调）；长度（毫秒）
4. 歌曲层：调值，时间戳和节奏；音调的数目；长度（2/3调）；长度（毫秒）。
编写程序，根据midi和歌词自动产生上下文相关标注。
编写程序，生成决策树问题集。
三、声学模型训练：
以声韵母作为合成的基元，采用HMM建模。基频F0和颤音使用多分布的HMMs建模。
模型及决策树以hts_engine格式保存。

Parselmouth

0. Parselmouth
1. Parselmouth是praat的python接口。
2. 论坛讨论地址：https://groups.google.com/g/parselmouth

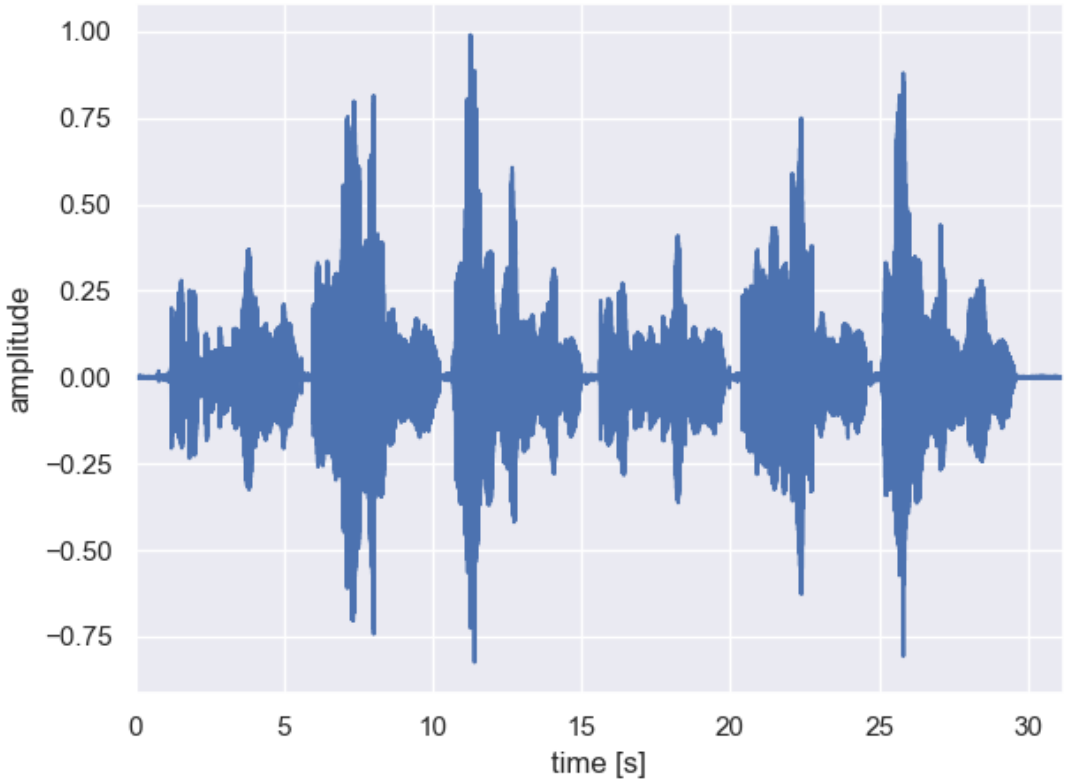
install

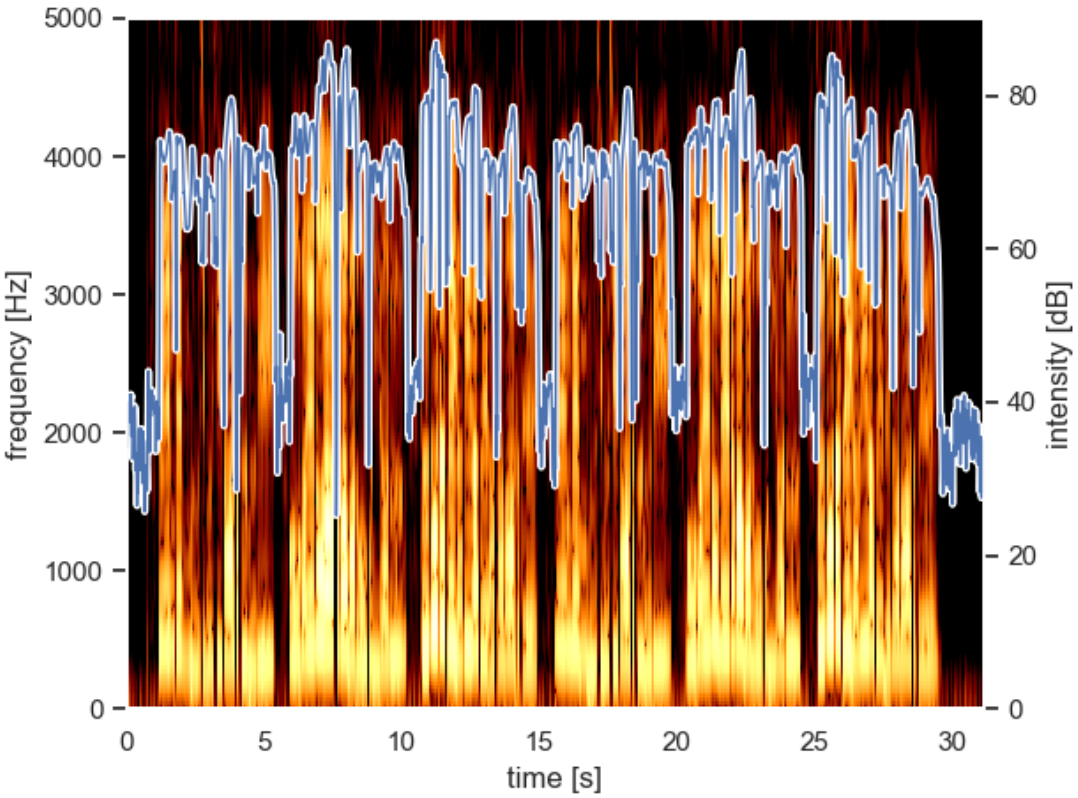
```
$ pip install praat-parselmouth
```

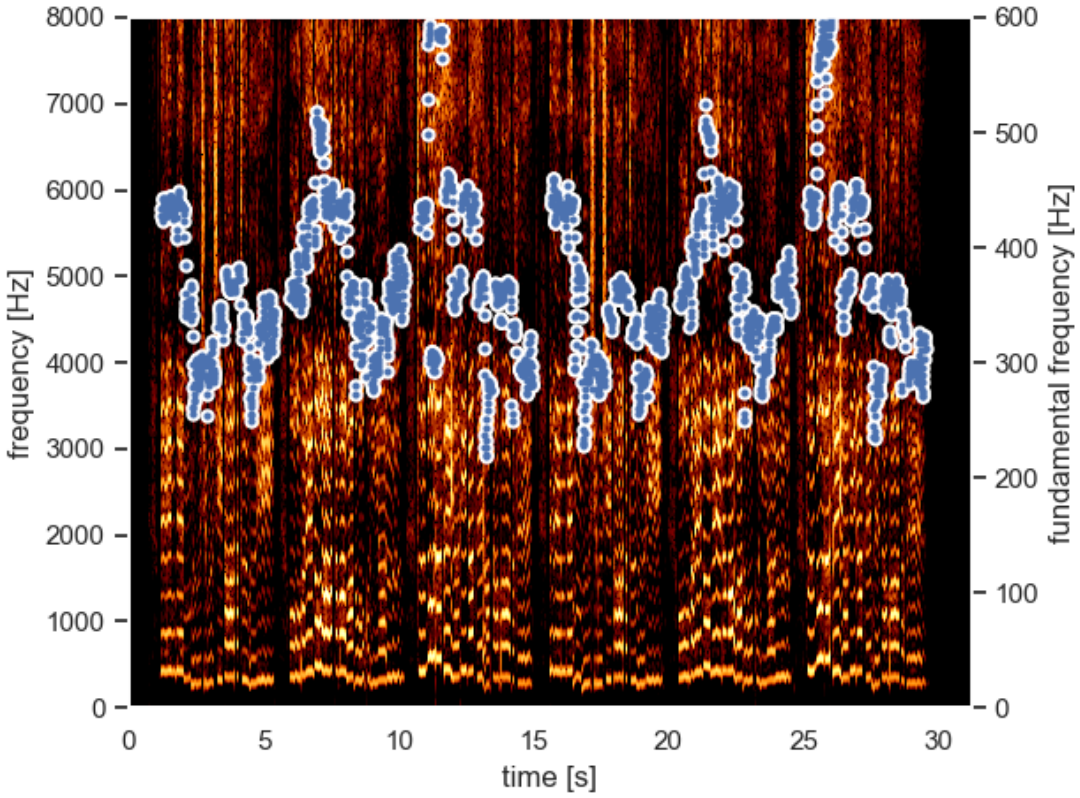
画图

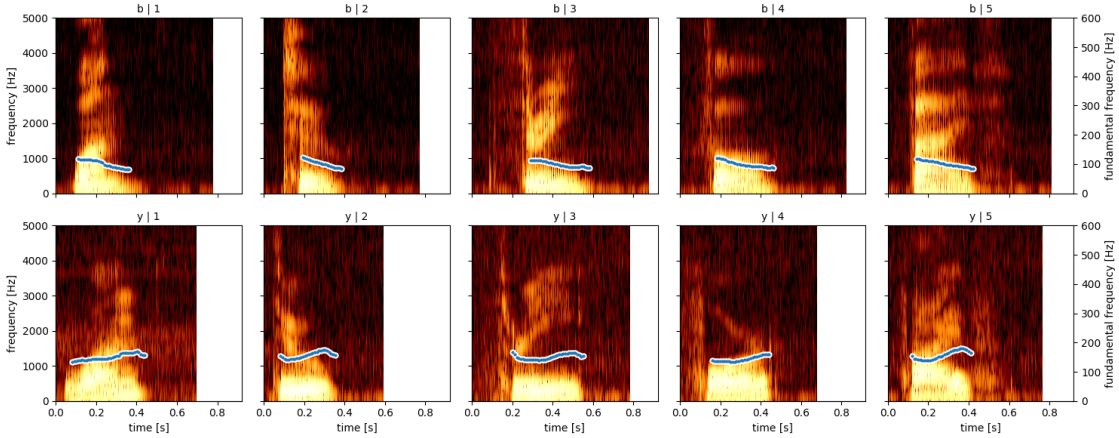
```
import parselmouth

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd


def draw_spectrogram(spectrogram, dynamic_range=70):
    X, Y = spectrogram.x_grid(), spectrogram.y_grid()
    sg_db = 10 * np.log10(spectrogram.values)
    plt.pcolormesh(X, Y, sg_db, vmin=sg_db.max() - dynamic_range, cmap='afmhot')
    plt.ylim([spectrogram.ymin, spectrogram.ymax])
    plt.xlabel("time [s]")
    plt.ylabel("frequency [Hz]")

def draw_intensity(intensity):
    plt.plot(intensity.xs(), intensity.values.T, linewidth=3, color='w')
    plt.plot(intensity.xs(), intensity.values.T, linewidth=1)
    plt.grid(False)
    plt.ylim(0)
    plt.ylabel("intensity [dB]")
```

```
def draw_pitch(pitch):
    # Extract selected pitch contour, and
    # replace unvoiced samples by NaN to not plot
    pitch_values = pitch.selected_array['frequency']
    pitch_values[pitch_values==0] = np.nan
    plt.plot(pitch.xs(), pitch_values, 'o', markersize=5, color='w')
    plt.plot(pitch.xs(), pitch_values, 'o', markersize=2)
    plt.grid(False)
    plt.ylim(0, pitch.ceiling)
    plt.ylabel("fundamental frequency [Hz]")

def facet_util(data, **kwargs):
    digit, speaker_id = data[['digit', 'speaker_id']].iloc[0]
    sound = parselmouth.Sound("{}_{}.wav".format(digit, speaker_id))
    draw_spectrogram(sound.to_spectrogram())
    plt.twinx()
    draw_pitch(sound.to_pitch())
    # If not the rightmost column, then clear the right side axis
    if digit != 5:
        plt.ylabel("")
        plt.yticks([])

results = pd.read_csv("digit_list.csv")

grid = sns.FacetGrid(results, row='speaker_id', col='digit')
grid.map_dataframe(facet_util)
grid.set_titles(col_template="{col_name}", row_template="{row_name}")
grid.set_axis_labels("time [s]", "frequency [Hz]")
grid.set(facecolor='white', xlim=(0, None))
plt.show()
```


1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
amplitude
0
5
10
15
20
25
30
time [s]


5000
4000
3000
2000
1000
0
frequency [Hz]
80
60
40
20
0
intensity [dB]
0
5
10
15
20
25
30
time [s]





音高处理

```
import parselmouth
from parselmouth.praat import call

sound = parselmouth.Sound("other/4_b.wav")
```

```
manipulation = call(sound, "To Manipulation", 0.01, 75, 600)

print(type(manipulation))

pitch_tier = call(manipulation, "Extract pitch tier")

call(pitch_tier, "Multiply frequencies", sound.xmin, sound.xmax, 2)

call([pitch_tier, manipulation], "Replace pitch tier")
sound_octave_up = call(manipulation, "Get resynthesis (overlap-add)")

print(type(sound_octave_up))

sound_octave_up.save("4_b_octave_up.wav", "WAV")
```