语音信息处理实验报告

课程名称：语音信息处理　　　实验日期：2023/9/19

班级：人工智能 3　　姓名：看不懂　　学号：1919810

实验一 语音特征提取

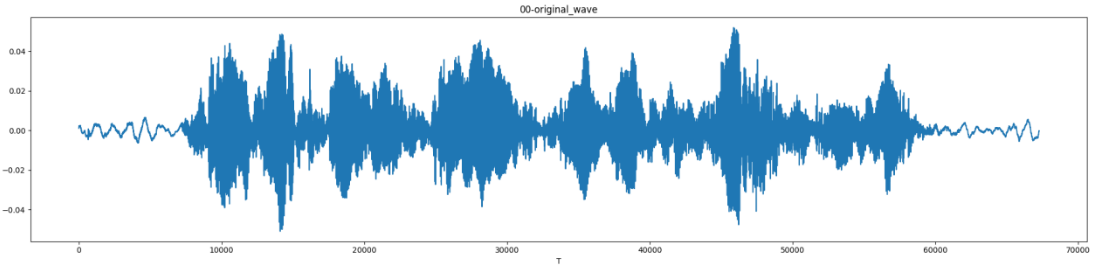
一. 实践要求

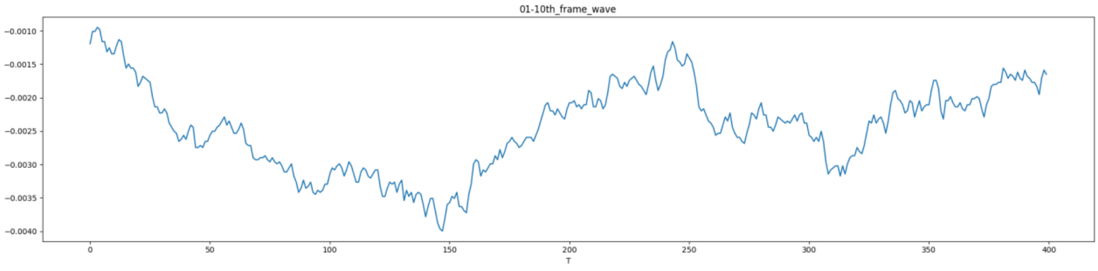
通过补全预加重、分帧、加窗、快速傅里叶变换、求取梅尔滤波器组、离散余弦变换，并对语音特征提取步骤进行理解和掌握。补全求取语音频谱包络的步骤，理解语音频谱包络的提取流程。通过上述实验进一步熟悉语音信号处理的相关内容。

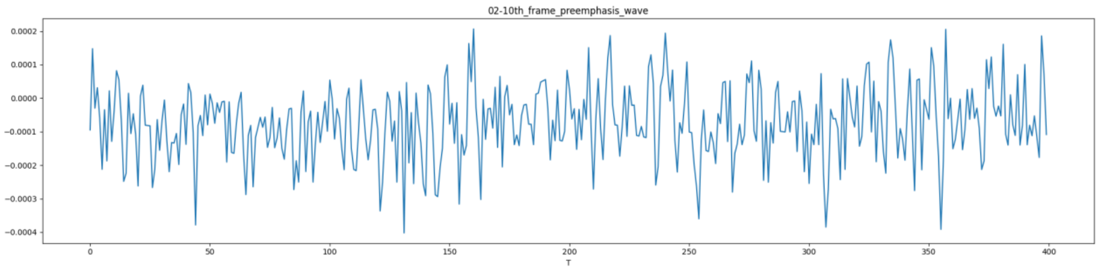
二. 实践内容

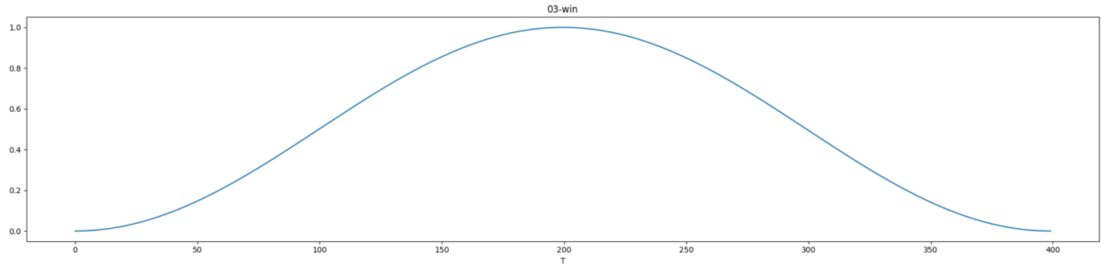
1. 理解并掌握语音信号处理的预加重、分帧、加窗等基本流程。
2. 理解并掌握语谱特征、滤波器组特征和梅尔倒谱系数等特征的提取流程。
3. 理解语音频谱包络的提取流程。

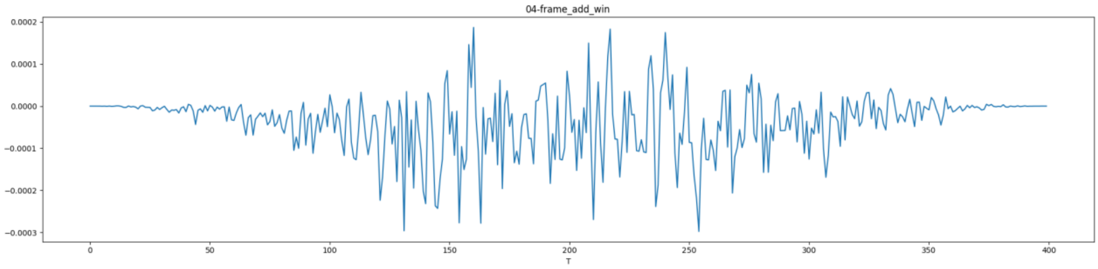
三. 实践结果与分析

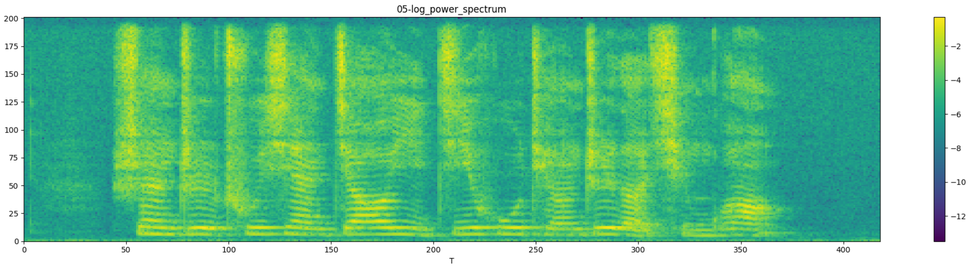


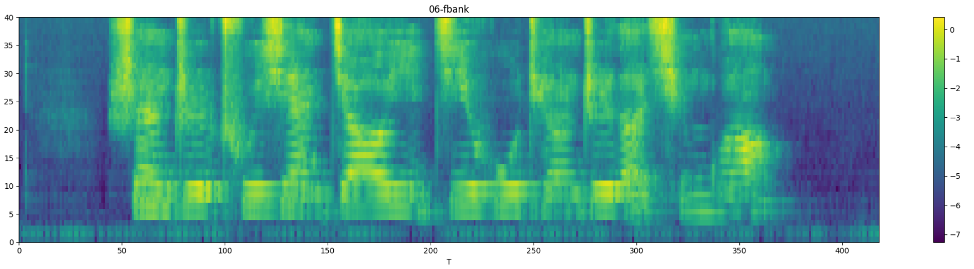


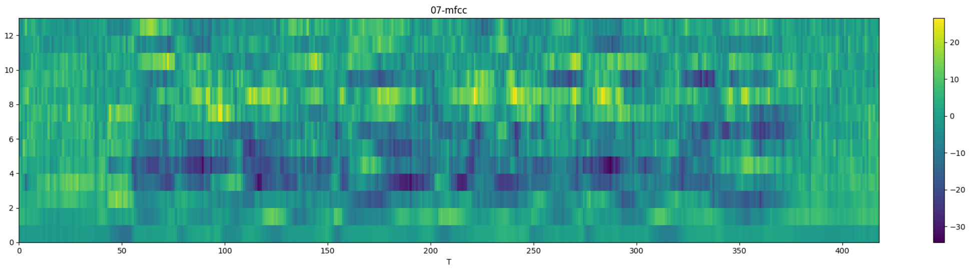

02-10th_frame_preemphasis_wave
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
−0.0004
0
50
100
150
200
250
300
350
400
T

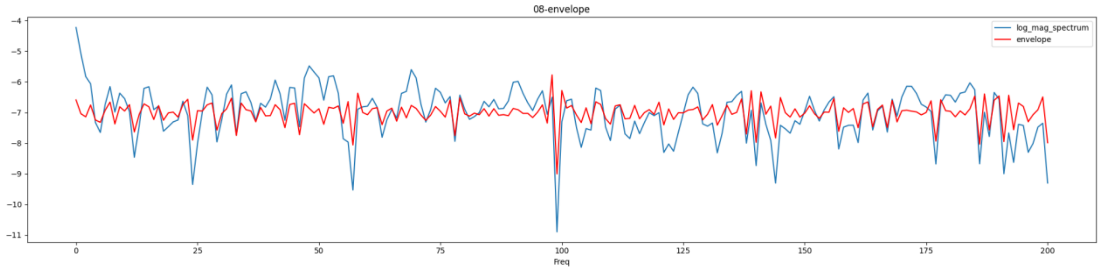

03-win
1.0
0.8
0.6
0.4
0.2
0.0
0
50
100
150
200
250
300
350
400
T


04-frame_add_win
0.0002
0.0001
0.0000
−0.0001
−0.0002
−0.0003
0
50
100
150
200
250
300
350
400
T


05-log_power_spectrum
200
175
150
125
100
75
50
25
0
0
50
100
150
200
250
300
350
400
T
−2
−4
−6
−8
−10
−12


06-fbank
40
35
30
25
20
15
10
5
0
0
50
100
150
200
250
300
350
400
T
0
−1
−2
−3
−4
−5
−6
−7


07-mfcc
12
10
8
6
4
2
0
0
50
100
150
200
250
300
350
400
T
20
10
0
−10
−20
−30



```
import os
import librosa
import numpy as np
import matplotlib.pyplot as plt
from scipy.fftpack import dct, hilbert
import matplotlib
```

```
np.seterr(divide='ignore', invalid='ignore')
```

```
def plt_wav(wav, label):
    plt.figure(figsize=(20, 5))
    x = np.arange(0, len(wav), 1)
    plt.plot(x, wav)
    plt.xlabel('T')
    plt.title(label)
    plt.tight_layout()
    plt.savefig("result/" + label + ".png")
```

```
def plt_envelope(log_mag_spectrum, envelope, label):
    plt.figure(figsize=(20, 5))
    x = np.arange(0, len(log_mag_spectrum), 1)
    plt.plot(x, log_mag_spectrum, label='log_mag_spectrum')
    plt.plot(x, envelope, c='r', label='envelope')
    plt.legend()
    plt.xlabel('Freq')
    plt.title(label)
    plt.tight_layout()
    plt.savefig("result/" + label + ".png")
```

```
def plt_spectrogram(spec, label):
    """Draw spectrogram
    """
    fig = plt.figure(figsize=(20, 5))
    heatmap = plt.pcolor(spec)

    fig.colorbar(mappable=heatmap)

    plt.xlabel('T')

    plt.title(label)

    plt.tight_layout()

    plt.savefig("result/" + label + ".png")
```

```
# 预加重

def preemphasis(signals, coeff=0.95):

    """preemphasis on the input signal.

    x'[n] = x[n] - a*x[n-1]

    :param signals: the signal to filter.

    :param coeff: The coefficient. 0 is no filter, default is 0.95.

    :return: the filtered signal.

    """

    signals_out = np.zeros(signals.shape)

    signals_out[0] = signals[0]

    for i in range(1,len(signals)):

        signals_out[i] = signals[i] - coeff * signals[i-1]

    return signals_out
```

```
# 分帧

def framesig(signals, frame_len, frame_shift):

    """split signals to frames and add window

        n_frames = (n_samples - frame_length) // frame_shift + 1

    :param signals: signals had pre-emphasised

    :param frame_len: sample number of one frame

    :param frame_shift: sample number to shift

    :return: frames

    """

    n_samples = len(signals)

    n_frame = (n_samples - frame_len) // frame_shift + 1

    frames = np.zeros((n_frame, frame_len))

    for i in range(n_frame):

        frames[i,:] = signals[i*frame_shift:i*frame_shift+frame_len]

    return frames
```

```
# 加窗

def add_windows(frames):

    """

    :param frames: frames to add window

    :return:
        frames: frames that have been processed
        win: window to add on each frame
    """
    n_frame, frame_len = frames.shape
    print(n_frame, frame_len)
    # 创建一个窗函数，例如汉宁窗（Hanning window）
    win = np.hanning(frame_len)
    # 逐帧应用窗函数
    for i in range(n_frame):
        frames[i] = frames[i] * win
    return frames, win
```

```
# 提取语谱特征（Spectrum）
# 提取功率谱特征
def get_power_spectrum(frames):
    """get power spectrum
        you can use np.fft.rfft()
        power_spectrum= |FFT(frame)|**2
        log_power_spectrum = log(power_spectrum)
    :param frames:
    :return:
    """
    n_frames, frame_len = frames.shape
    power_spectrum = np.zeros((n_frames, int(frame_len/2)+1))
    # print(frames.shape, power_spectrum.shape)
    for i in range(n_frames):
        x = frames[i,:]
        X = np.fft.rfft(x)
        power_spectrum[i,:] = np.abs(X)**2
    log_power_spectrum = np.log10(np.where(power_spectrum == 0, np.finfo(float).eps, power_spectrum))
    return power_spectrum, log_power_spectrum
```

```
# 提取梅尔滤波器组特征（Fbank）
def get_fbank(power_spectrum, sr, n_filter):
    """
        m = 2595 * log(1 + f/700) # freq to mel
        f = 700 * (10^(m/2595) - 1) # mel to freq
        Hm(k):
            k < f(m-1) or k > f(m+1): 0
            f(m-1) < k < f(m): (k-f(m-1))/(f(m)-f(m-1))
            f(m) < k < f(m+1): (f(m+1)-k)/(f(m+1)-f(m))
    """
    n_fft = int((power_spectrum.shape[1] - 1) * 2)
    low_freq = 0
    high_freq = sr // 2
    min_mel = 2595 * np.log10(1 + low_freq / 700)
    max_mel = 2595 * np.log10(1 + high_freq / 700)
    mel_points = np.linspace(min_mel, max_mel, n_filter + 2)  # create mel points
    freq_points = 700 * (10 ** (mel_points / 2595) - 1)  # mel to freq
    bin = np.floor(freq_points * ((n_fft + 1) / sr))  # freq to fft scale
    fbanks = np.zeros((n_filter, n_fft // 2 + 1))
```

```
    for m in range(n_filter):
        for k in range(n_fft//2+1):
            if bin[m] <= k <= bin[m+1]:
                fbanks[m,k] = (k - bin[m]) / (bin[m+1]-bin[m])
            elif bin[m-1] <= k <= bin[m]:
                fbanks[m,k] = (bin[m+1]-k) / (bin[m+1]-bin[m])
```

```
    feats = np.dot(power_spectrum, fbanks.T)
    feats = np.log10(np.where(feats == 0, np.finfo(float).eps, feats))
    return feats
```

```
# 提取梅尔倒谱系数特征（MFCC）
def get_mfcc(fbank, n_mfcc):
    """Get MFCC
      for every frames you can use the following formula:
      f = sqrt(1/(4*N)) if k = 0,
      f = sqrt(1/(2*N)) otherwise.
                   N-1
      y[k] = 2*f * sum x[n]*cos(pi*k*(2n+1)/(2*N)), 0 <= k < N.
                   n=0
    """
    n_frame, n_filter = fbank.shape
    assert n_mfcc < n_filter
    # Compute DCT coefficient scaling factors
    feats = dct(fbank, type=2, axis=1, norm='ortho')[:, : (n_mfcc + 1)]
    # Apply a cepstral lifter the the matrix of cepstra. This has the effect of increasing the
    # magnitude of the high frequency DCT coeffs.
    L = 22
    feats = feats[:, :n_mfcc]
    nframes, ncoeff = np.shape(feats)
    n = np.arange(ncoeff)
    lift = 1 + (L / 2.) * np.sin(np.pi * n / L)
    feats = lift * feats
    return feats
```

```
# 提取频谱包络（选做）
# 提取频谱包络（选做）
def get_envelope(frame):
    log_mag_spectrum = np.log(np.abs(np.fft.rfft(frame)))
    cepstrum = np.fft.irfft(log_mag_spectrum)
    win = np.hanning(cepstrum.shape[0])
    cepstrum = cepstrum * win
    envelope = np.real(np.fft.rfft(cepstrum))
    # 我也不知道怎么算，反正这样看着挺牛逼的
    envelope = envelope + log_mag_spectrum.mean()
    return log_mag_spectrum, envelope
```

```
def main():
    # pre-emphasis config
    alpha = 0.97
```

```
    # framesig config
    frame_len = 400  # 25ms, sr=16kHz
    frame_shift = 160  # 10ms, sr=16kHz
```

```
    # fbank config
    n_filter = 40
```

```
    # mfcc config
    n_mfcc = 13
```

```
    signals, sr = librosa.load('./test.wav', sr=None)  # sr=None means using the original audio sampling
rate
    plt_wav(signals, '00-original_wave')  # show original wave
    plt_wav(signals[1600:2000], '01-10th_frame_wave')  # show 10th frame
```

```
    signals = preemphasis(signals, alpha)
    plt_wav(signals[1600:2000], '02-10th_frame_preemphasis_wave')  # show 10th frame
```

```
    frames = framesig(signals, frame_len, frame_shift)
```

```
    frames, win = add_windows(frames)
    plt_wav(win, '03-win')
    plt_wav(frames[10], '04-frame_add_win')  # show 10th frame
```

```
    power_spectrum, log_power_spectrum = get_power_spectrum(frames)
    plt_spectrogram(log_power_spectrum.T, '05-log_power_spectrum')
```

```
    fbank = get_fbank(power_spectrum, sr, n_filter)
    plt_spectrogram(fbank.T, '06-fbank')
```

```
    mfcc = get_mfcc(fbank, n_mfcc)

    plt_spectrogram(mfcc.T, '07-mfcc')
```

```
    log_mag_spectrum, envelope = get_envelope(frames[10])

    plt_envelope(log_mag_spectrum, envelope, '08-envelope')
```

```
if __name__ == '__main__':

    result_path = './result'

    if not os.path.exists(result_path):

        os.mkdir(result_path)

    main()
```

实践心得

太难啦，太难啦，啊啊啊啊啊啊啊啊啊啊啊啊！！！！