Technische Universität Berlin

Grundlagen der automatischen Spracherkennung

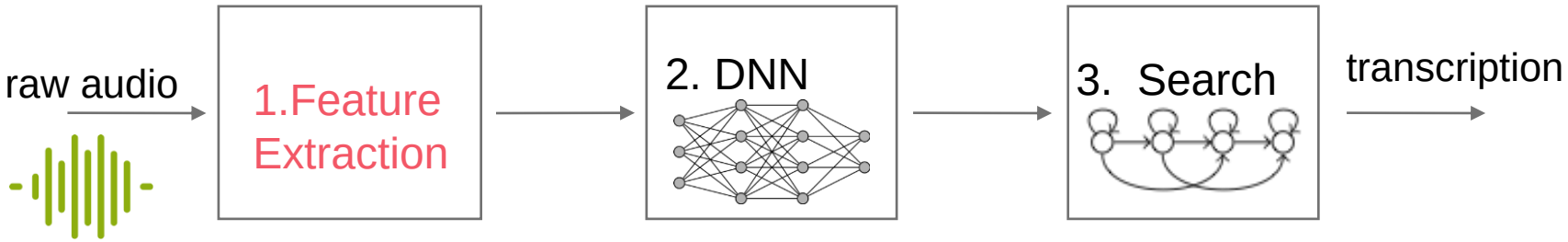
Aufgabe 2 – Spektralanalyse und mel-skalierte Dreiecksfilterbank

15.11.2023

Wentao Yu

Technische Universität Berlin

Hybride Spracherkennung



Technische Universität Berlin

ISIS-Kursdatei

Im ISIS-Kurs ist das Übungsblatt Aufgabe2.pdf mit den genauen Aufgabenstellungen.

Außerdem ist dort die Datei **uebung2.py**.

Technische Universität Berlin

Spektralanalyse

1. compute_absolute_spectrum(frames)

2. compute_features(audio_file, window_size=25e-3, hop_size=10e-3, feature_type='STFT', n_filters=24, fbank_fmin=0, fbank_fmax=8000, num_ceps=13):

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

```
def compute_absolute_spectrum(frames):
    ...
```

Diese Funktion berechnet das Betragsspektrum der gegebenen Frames.

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

```
def compute_features(audio_file, window_size=25e-3, hop_size=10e-3,
                     feature_type='STFT', n_filters=24, fbank_fmin=0,
                     fbank_fmax=8000, num_ceps=13):
    ...
```

Wenn feature_type=STFT ist, liefert diese Funktion die Short-Time Fourier Transform (STFT) für normalisierte Audio-Frames als Features.

Nicht verwendete Eingabeparameter können vorerst ignoriert werden.

Technische Universität Berlin

Erstellen der Hauptfunktion: uebung2.py

1. Verwenden Sie die compute_features() Funktion, um die STFT für die Audiodatei 'TEST-MAN-AH-3O33951A.wav' zu berechnen.

2. Stellen Sie das Spektrum in Dezibel (dB) dar, wobei Sie die Formel 20 * log10(.) verwenden.

Ein Beispiel wie ein Ergebnis aussehen könnte, und mehr Hinweise zur Implementierung, finden Sie im Übungsblatt Aufgabe2.pdf im ISIS-Kurs.

Technische Universität Berlin

Mel-Skalierte Dreiecksfilterbank

1. hz_to_mel(x)
2. mel_to_hz(x)
3. get_mel_filters(sampling_rate, window_size_sec, n_filters, f_min=0, f_max=8000)
4. apply_mel_filters(abs_spectrum, filterbank)
5. computer_features()

Technische Universität Berlin

Benötigte Funktionen in tools.py

```
def hz_to_mel(x):

...
```

Diese Funktion berechnet den Wert der Mel-Skala für den entsprechenden Frequenzwert x.

$$\text{Mel}\,(x) = 2595\,\log_{10}(1 + x/700)$$

Technische Universität Berlin

Benötigte Funktionen in tools.py

```
def mel_to_hz(x):
    ...
```

Die Umkehrfunktion von `hz_to_mel(x)` wandelt einen Frequenzwert x von der Mel-Skala in die lineare Frequenzskala in Hz um.

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

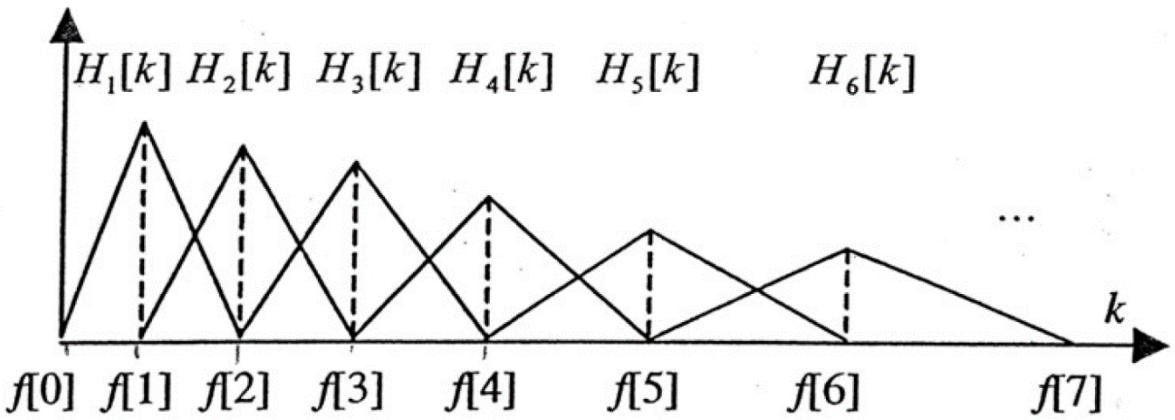
```
def get_mel_filters(sampling_rate, window_size_sec, n_filters,
                    f_min=0, f_max=8000):
    ...
```

Die Rückgabe (die Dreiecksfilterbank) sollte in Form einer Matrix erfolgen.

1. sampling_rate: die Abtastfrequenz in Hz
2. window_size_sec: die Fensterbreite in Sekunden
3. n_filters: die Anzahl der Dreiecksfilter
4. f_min und f_max: minimale bzw. maximale Frequenz in Hz

Benötigte Funktionen in feature_extraction.py

```
def get_mel_filters(sampling_rate, window_size_sec, n_filters,
                    f_min=0, f_max=8000):
    ...
```



n_filters=6 Filtern von $H_1[k]$ bis $H_6[k]$
n_filters + 2 = 8 Frequenzstützstellen von $f_{Hz}[0]$ bis $f_{Hz}[7]$ (in Hz)
entsprechende 8 DFT-Indizes $f[0]$ bis $f[7]$

Technische Universität Berlin

Benötigte Funktion in feature_extraction.py

```
def get_mel_filters(sampling_rate, window_size_sec, n_filters,
                    f_min=0, f_max=8000):
    ...
```

M + 2 = 8 Frequenzstützstellen von $f_{Hz}[0]$ bis $f_{Hz}[7]$ sollen:

1. im Mel-Frequenzbereich angenähert äquidistant sein.

Benötigte Funktion in feature_extraction.py

M + 2 = 8 Frequenzstützstellen von $f_{Hz}[0]$ bis$f_{Hz}[7]$ sollen:

2. genau mit den Frequenzstützstellen im Fourierbereich übereinstimmen.

Fouriertrafo (Frequenzstützstellen)		DFT Indizes
f_{Hz} = 0 Hz		f = 0
f_{Hz} = Fs/2 Hz		f = N/2
f_{Hz} = 0 …Fs/2 Hz		f = 0, 1, …, N/2

Tabelle 1: Auf den Folien der Vorlesung 3-4, Seite 26

i-te Frequenzstützstelle $f_{Hz}[\mathrm{i}] = (f[i] \cdot \frac{Fs}{2} \cdot \frac{2}{N})$ Hz

3. mit $f[0] = 0$ und $f[\mathrm{M}+1] = \frac{N}{2}$, wobei N der FFT-Größe entspricht.

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

```
def get_mel_filters(sampling_rate, window_size_sec, n_filters,
                    f_min=0, f_max=8000):
```

Die Gewichte der Dreiecksfilter sind bestimmt durch:

```
Für alle m = 1 bis M und k = 0 bis N/2:
```

$$H_m[k] = \begin{cases} 0 & \text{falls } k < f[m-1] \\ \dfrac{2(k - f[m-1])}{(f[m+1] - f[m-1])(f[m] - f[m-1])} & \text{falls } f[m-1] \leq k < f[m] \\ \dfrac{2(f[m+1] - k)}{(f[m+1] - f[m-1])(f[m+1] - f[m])} & \text{falls } f[m] \leq k \leq f[m+1] \\ 0 & \text{falls } k > f[m+1] \end{cases}$$

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

```
def apply_mel_filters(abs_spectrum, filterbank):
    ...
```

Wenden Sie die berechnete Dreiecksfilterbank $H_m[k]$ auf ein Betragsspektrum $S_{\mathrm{LIN}}[k,\tau]$ an

$$S_{\mathrm{MEL}}[m,\tau] = \sum_{k=0}^{K} H_m[k] S_{\mathrm{LIN}}[k,\tau]$$

Technische Universität Berlin

Benötigte Funktionen in feature_extraction.py

```
def compute_features(audio_file, window_size=25e-3, hop_size=10e-3,
                     feature_type='STFT', n_filters=24, fbank_fmin=0,
                     fbank_fmax=8000, num_ceps=13):
    ...
```

- Erweitern Sie die compute_features() Funktion für den Fall, dass FBANK als feature_type angegeben wird.
- n_filters, fbank_fmin und fbank_fmax können Sie für die FBANK feature type benutzen
- Nicht verwendete Eingabeparameter (num_ceps) können weiter ignoriert werden.

Erweitern der Hauptfunktion: uebung2.py

1. Plotten Sie die berechnete Dreiecksfilterbank.

2. Plotten Sie das Mel-Spektrum der Audiodatei 'TEST-MAN-AH-3O33951A.wav‘.

Ein Beispiel wie ein Ergebnis aussehen könnte, und mehr Hinweise zur Implementierung, finden Sie im Übungsblatt Aufgabe2.pdf im ISIS-Kurs.

Technische Universität Berlin

Melden Sie sich gerne bei Fragen