Документација – пројекат из дигиталне електронике

Autotune

Аутотјун је аудио процесор креиран од стране компаније Антерс Аудио Технологија. Служи за корекцију у гласу и инструменталном извођењу. Уведен је за корекцију само гласа, али је после коришћен за корекцију проклизавањем френквентних таласа.

Студијски трик захваљујући коме глас може да дође до жељене ноте. Аутотјун је као фотошоп за људски глас. Не омогућава баш сваком да пева као професионалац, али може да трансформише лошу изведбу у, малтене, технички савршену.

Иза овог стоји Енди Хилдебранд, који никада није био везан за ноте, већ је радио у нафтној индустрији. Користећи математички метод зван аутокорелација, Хилдебранд је слао таласе на тло и снимао њихово одбијање, и на тај начин је добијао прецизну информацију о потенцијалним нафтним бушотинама. Захваљујући тој техници, нафтне компаније могу да уштеде приличну своту новца.

Неколико година касније, познати музички продуценти овај уређај су унапредили и прилагодили га потребама музичке индустрије.

Као област која је у повоју, за истраживање музичких информација и унапређење музичке индустрије данас су веома корисне области као што су **музикологија**, **дигитална обрада сигнала**, **машинско учење**, проналажење информација и разне **библиотеке појединих програмских језика**. Негде почетком 2000.-их година, када је музичка индустрија почела врло брзо да се развија због настанка ИТјунс-а, Спотифаја и других платформи, алати и скрипте које су се користиле у музичкој индустрији су углавном биле писане у Матлаб-у или C++.

Последњих неколико година за потребе музичке индустрије се све више користи програмски језик Пајтон, али његове функционалности су биле успорене због непостојања неке стабилне, квалитетне библиотеке која пружа основне конфигурације на којима су изграђене многе апликације у музичкој индустрији.

Зато је и створена библиотека која се зове librosa, Пајтон библиотека које се користи за аудио и музичку обраду. Функције у овој библиотеци имају ту тенденцију да све релевантне параметре изложе позиваоцу.

Овде се аудио сигнал репрезентује као једнодимензионални numpy низ, означен као **y** у целој librosa библиотеци.

```
>>> np.array([1, 2, 3])
array([1, 2, 3])
```

Сигнал у прати брзина узорковања (означава се у библиотеци, а и у оквиру овог пројекта са sr), која означава фреквенцију (у Hz) на којој се вредности и узоркују. Трајање сигнала у се може израчунати дељењем броја узорака према стопи узорковања.

У овој библиотеци се аудио датотеке учитавају преко librosa.load() функције и ова функција омогућава да се смањи монофонски звук (то је звук који потиче из једне позиције).

librosa је пакет који има више подмодула.

1. librosa.display - служи за визеулизацију и приказује поставке помоћу display својства

display(могућности):

- waveplot(y[, sr, max_points, x_axis, …]) - исцртава амплитудну енвелопу таласног облика

waveplot у потпуности треба да изгледа овако: waveplot(y, sr=22050, max_points=50000.0, x_axis='time', offset=0.0, max_sr=1000, ax=None). Овде је важно следеће: ако је у монофонични звук(сигнал), тј долази од једног извора, крива ће бити нацртана између [-abs(y), abs(y)].

Ако је у стереофонски звук(звук допире од више извора), крива ће бити нацртана између [-abs(y[1]), abs(y[0])].

Waveplot(назнаке) - серије звука(монофоничног или стереофонског), sr је мера узорковања сигнала Y, max_points је максимални број временских тачака које треба да се исцртају

x_axis дефинише чим меримо време - у секундама(c), милисекундама(мс)

ax служи за цртање уместо подразумеваног plt.gca().

- specshow(data, x_coords=None, y_coords=None, x_axis=None, y_axis=None, sr=22050, hop_length=512, fmin=None, fmax=None, tuning=0.0, bins_per_octave=12, ax=None). Овде је важно следеће: задатак параметра data је да прикаже матрицу.
  hop_length - дефинише временску скалу x-осе
  bins_per_octave - служи за фреквенцијско скалирање помоћу трансформације константно Q(constant-Q transform)

2. chroma - спада у спектралне могуцности либроса библиотеке

2. chroma - спада у спектралне могућности librosa библиотеке

- chroma_stft([y, sr, S, norm, n_fft, …]) - израчунава хромограм на основу таласног облика или спектограма

3. librosa.util.normalize служи да нормализује низ дуж одабране осе

- нпр. norm(S, axis=axis) == 1 нормализује сваки члан дводимензионалног низа
- да смо ставили norm(S, axis=axis) == 2, вршила би се нормализација сваког другог члана низа, врши се скалирање

4. librosa.load - учитава аудио фајл као временску секвенцу с помичним зарезом

5. librosa.effects.pitch_shift - pomera visinu talasa na osnovu broja koraka (steps)

6. **librosa.feature** је модул који имплементира разне спектралне репрезентације и већина тих репрезентација се заснива на краткорочној Фуријеовој трансформацији.(краткорочна Фуријеова трансформација се израчунава тако што се дужи временски сигнали поделе на краће сегменте једнаке дужине и онда се израчуна Фуријеова трансформација за сваки тај сегмент.).

Сем могућности librosa библиотеке, овде ћу објаснити и друге делове кода везаног за Аутотјун. Наиме, многе функционалности које сам овде користио су део numpy библиотеке, коју користимо тако што импортујемо ту библиотеку помоћу

```
import librosa
import Notes
import numpy as np
```

, а коришћењем библиотеке за исцртавање графика matplotlib у могућности смо да прикажемо графичке резултате појединих операција у коду. Нпр. погледајмо следећу линију кода:

```
yD = librosa.stft(y,n_fft=self.INPUT_SR)
arr = np.argmax(yD,axis=0)
#np.argmax враћа индекс максималне вредности у низу/дуж осе
```

np.argmax враћа индексе максималних вредности дуж осе. Следећих пар линија кода су пример како функционише np.argmax.

```
>>> b = np.arange(6) #овде дефинишемо низ од 6 чланова, напоменимо да у пајтону индекси низа иду од 0
>>> b[1] = 5 #стављамо да члан са индексом 1, а то је други члан низа, има вредност 5
>>> b #овде само штампамо низ
#  резултат:       array([0, 5, 2, 3, 4, 5])
>>> np.argmax(b)  #тражимо да нам се испише индекс у низу оне вредности која је највећа
#излаз ће бити 1, јер то је највећа вредност на коју смо наишли(исписује се само 1, не и 5, јер се штампа само први
#индекс највеће вредности на коју се наиђе при проласку кроз цео низ
```

np.mean - израчунава аритметичку средину вредности у наведеном низу

```
fq = np.mean(arr)
#израчунава аритметичку средину вредности у наведеном низу
```

np.empty прави нови низ одређеног типа и облика(димензије), али се не врши додела вредности.

У овом коду у више наврата можемо видети кључну реч **self.** Кључну реч self. користимо за приступ атрибутима и методама дефинисане класе(као this у Јави или C++). Ова кључна реч у ствари представља инстанце те саме класе.

Користимо **self.** јер Пајтон не користи @ да би референцирао на инстанцу атрибута. У Пајтону су методе креиране тако да се инстанца којој припада метода аутоматски прослеђује, али се не прима аутоматски. Први параметар методе је инстанца на коју се позива метода.

Matplotlib, тј. функционалности ове библиотеке користимо на следећи начин:

plt.show() - приказује исцртану фигуру

Ову команду треба да ставимо углавном само једном у току неке пајтон скрипте (ако се стави више пута, а пошто зависи од графичког backend-а рачунара, може да прави проблеме.

plt.subplot(321) (овде је као пример узета жељена линија кода, где је параметар 321. Последњи од ова три броја се мења зависно од тога који је редни број цртежа који треба да се исцрта) - subplot користимо кад хоћемо на једном графику да исцртамо више функција(321 значи да мрежа на графику буде 3x2, а 1 значи да је то први цртеж на заједничком графику)

Сада ћемо се дотаћи још пар детаља.

Погледајмо следећу линију кода:

```
                                        #istoimenoj funkciji u Javi
self.OUTPUT_WAVE = np.empty(shape=self.INPUT_WAVE.shape)
```

Када ставимо својство **shape**, тиме изражавамо да желимо да добијемо тренутну димензију матрице(дводимензионалног или вишедимензионалног низа).

Даље, да би све ово било функционално, врло је важно и који тип датотеке овај програм може да учита:

```
def load(self):
    if ".wav" in self.FILE_PATH:
        import wave
```

Кад импортујемо **wave,** тиме се омогућује рад са **WAV** датотекама.

WAV датотеке не подржавају компресију/декомпресију, али подржавају моно/стерео звуке(један извор/више извора звука).

Важно је дотаћи се и исцртавања у овом програму, тј. објаснити шта ми то исцртавамо.

```
def plotWave(self,y,sr):
    print('Iscrtavanje talasnog oblika',end='') #ово end само специфицира шта се исписује на крају
    disp.waveplot(y,sr)
    print('Završeno.')
    return

def plotSpec(self,y,sr):
    print('Iscrtavanje spektograma snage',end="")
    #Спектрограм је визуелни приказ спектра фреквенција у звук или други сигнал који варира временом или неком другом
    променљивом.
    yD = librosa.stft(y,n_fft=sr) #враћа матрицу комплексних вредности
    disp.specshow(librosa.amplitude_to_db(yD),y_axis='log',x_axis='time')
    print('Završeno.')
    return

def plotChroma(self,y,sr):
    print('Iscrtavanje hromatografa',end='')

    cD = librosa.feature.chroma_stft(y,n_fft=sr)
```

Спектрограм је визуелни приказ спектра фреквенција у звук или други сигнал који варира временом или неком другом променљивом.

Формат је графикон са две геометријске димензије; хоризонтална оса представља време, а вертикална оса је фреквенција; трећа димензија указује на амплитуду одређене фреквенције у одређеном тренутку који представља интензитет и боју сваке тачке на слици.

Аудио ће обично бити представљен са логаритамском осом амплитуде (вероватно у децибелима дБ) - otuda u kodu **librosa.amplitude_to_db** - prebacuje amplitudni spektogram u spektogram koji se predstavlja u decibelima(dB).

Хроматограф је хроматографски инструмент који подразумева различиту поделу узорка између две фазе, при чему је једна фаза мобилна, а друга стационарна(у једној прикупљамо узорке, у другој их испитујемо).