Препознавање песама у програмском језику Python

Електротехнички факултет, Универзитет у Београду
Предмет: Основи телекомуникација
Студенти: Милена Вујчић 2020/0325, Марко Стојановић 2020/0080

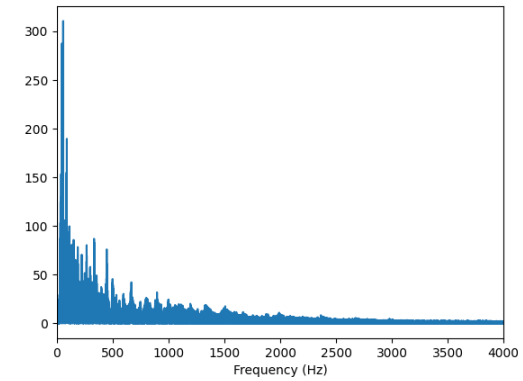
У овом задатку је коришћен Python3 са библиотекама NumPy, SciPy, OS, Wave и Matplotlib.

На почетку смо учитали песме и одредили њихове отиске – специфичне карактеристике.

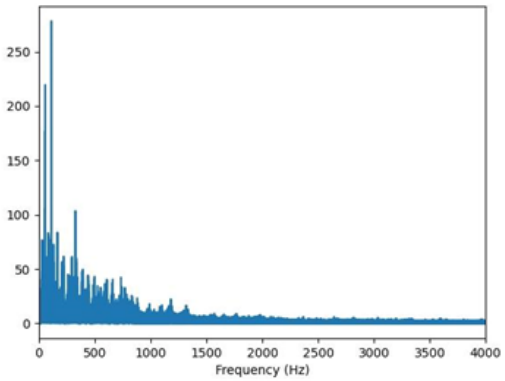
Учитавање песме смо одрадили помоћу функција из библиотеке Wave.

```
wav_obj = wave.open(file, 'rb')
Fs = wav_obj.getframerate() # odabirci po sekundi
n_samples = wav_obj.getnframes()

signal_wave = wav_obj.readframes(n_samples)
song = np.frombuffer(signal_wave, dtype=np.int16)
```

Отисак је оно по чему ћемо разликовати сваку песму. Циљ јесте пронаћи фреквенције које се појављују. Фуријеовом трансформацијом можемо лако да добијемо жељене фреквенције. Иначе, има доста шума, који отежава проналазак важнијих максимума и других карактеристика, али Фирујеова трансформација помаже да се превазиђу ти проблеми. У домену фреквенције након Фуријеове трансформације максимуми су врло изражени, чак и уз велико присуство шума. Након ограничавања фреквенцијског спектра, постаје јасно које се фреквенције истичу.

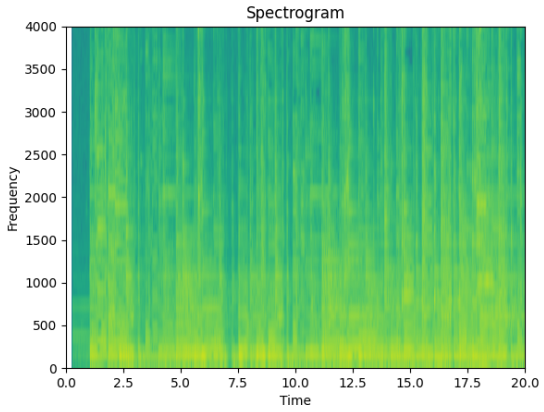


Фуријеова трансформација сигнала песме “Heart - Crazy on You”

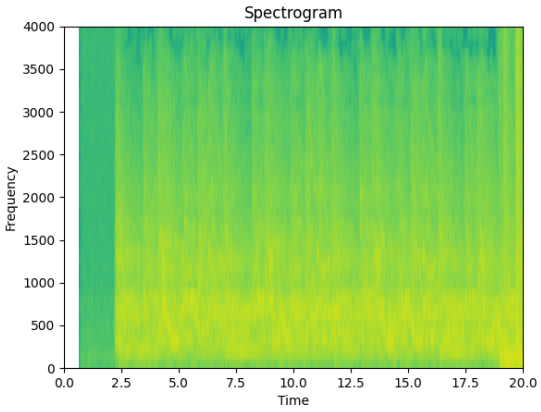


Фуријеова трансформација сигнала песме “ЕКВ - Земља”

Како се песма мења, мења се и њен спектар, па почетак и рефрен песме имају потпуно другачији спектар. То значи да се спектар исечка песме неће поклапати са спектром целокупне песме. Зато радимо Фуријеову транформацију над краћим интервалима. На тај начин можемо да проверимо да ли се исечак песме поклапа са било којим делом песме.



Спектограм сигнала песме “Heart - Crazy on You”



Спектограм сигнала песме “ЕКВ - Земља”

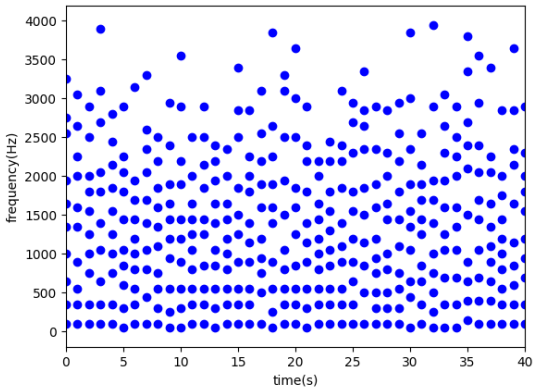
```
window_length_seconds = 1
window_length_samples = int(window_length_seconds * Fs)
window_length_samples += window_length_samples % 2

frequencies, times, stft = signal.stft(
```

```
        song, Fs, nperseg=window_length_samples,
        nfft=window_length_samples, return_onesided=True
    )
```

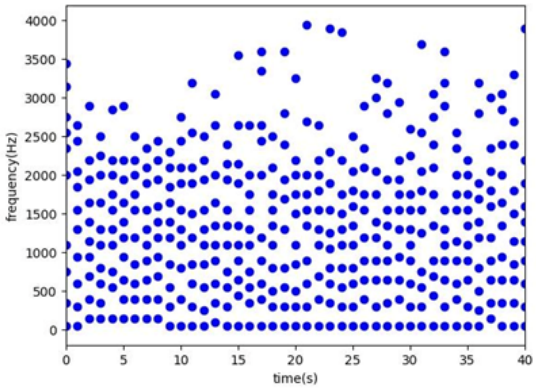
Затим смо издвојили најбитније максимуме помоћу библиотечких функција. Ова функција узима у обзир само најистакнутије врхове. Дефинишемо минимално растојање и налазимо 10 најбитнијих врхова. Ове истакнуте фреквенције карактеришу сигнал који анализирамо.

```
num_peaks = 10
fmax = 4000 # u pesmama uglavnom varira do ove frekvencije
freq_slice = np.where((frequencies >= 0) & (frequencies <= fmax))
frequencies = frequencies[freq_slice]
stft = stft[freq_slice,:][0]
```

Отисак посматране песме представља матрица чије ћелије представљају зависност фреквенције у времену. Исти такав отисак генеришемо и за посматрани исечак помоћу функције findFeatures(file, trim=False) .



Отисак песме “Heart - Crazy on You"



Отисак песме “ЕКВ - Земља”

```
constellation_map = []
for time_idx, window in enumerate(stft.T):
    if(trim and time_idx >= 20): break # da ne racuna sve za odsecke
    spectrum = abs(window) # posto je sprektum kompleksan
    peaks, props = signal.find_peaks(spectrum, prominence=0, distance=200)

    n_peaks = min(num_peaks, len(peaks))
```

```
    if(n_peaks != 10): continue
    largest_peaks = np.argpartition(props["prominences"], -n_peaks)[-n_peaks:]
    for peak in peaks[largest_peaks]:
    frequency = frequencies[peak]
    constellation_map.append([time_idx, frequency])

features = []
for i in range (0,len(constellation_map),num_peaks):
    collumn = []
    for j in range(num_peaks):
    f = constellation_map[i+j][1]
    # zaokruzivanje vrednosti frekvencija
    mod25 = f % 25
    mod50 = f % 50
    if(mod50 >= 25): collumn.append(int(f+25-mod25))
    else: collumn.append(int(f-mod25))
features.append(sorted(collumn))
```

Да бисмо одлучили постоји ли песма са којом се поклапа исечак, морамо да проверимо да ли је матрица исечка приближно једнака некој подматрици матрице песме. Овај део је реализован функцијом Matching(mat, submat). Поред провере да ли је матрица подматрица посматране матрице, додали смо и граничну вредност која ће проверавати колико су различите вредности матрица, тако да ће се увећавати бројач који нам говори колико има приближно поклапања.

```
def Matching(mat, submat): # prolazak da li je isecak priblizno jednak nekom delu pesme
    m_len = len(mat)
    sm_len = len(submat)
    cntmax = 0
    for i in range (m_len-sm_len):
        cnt = 0
        for si in range (sm_len):
            for j in range (10):
                if (abs(mat[i+si][j]- submat[si][j]) <=  50):
                    cnt = cnt + 1
        if (cnt > cntmax): cntmax = cnt

    return cntmax
```

Затим у главном програму пролазимо кроз све песме, и тражимо да ли се посматрани исечак поклапа са иједном песмом. Прво налазимо песму са киком има највише преклапања. Експерименталним путем смо дошли до уобичајене вредности која се јавља ако песма постоји у бази. Уколико је вредност коју је

вратила функција Matching(mat, submat) већа од граничне вредности и са том песмом има највише преклапања, онда се сматра да је исечак део те песме.

```
max_matches = 0
final_song = ""
for song in songs:
    matches = Matching(song['features'],recording_features)
    if (matches > max_matches):
        max_matches = matches
        final_song = song['path']

if ( max_matches > 100) : print("\nDobijena pesma: " + final_song)
else: print("No matches")
```

Уколико постоји песма, а ипак није препозната, треба смањити критеријум за max_matches.