Tomáš Moravčík xmorav41

ISS Projekt 2019 / 20

15.12.2019

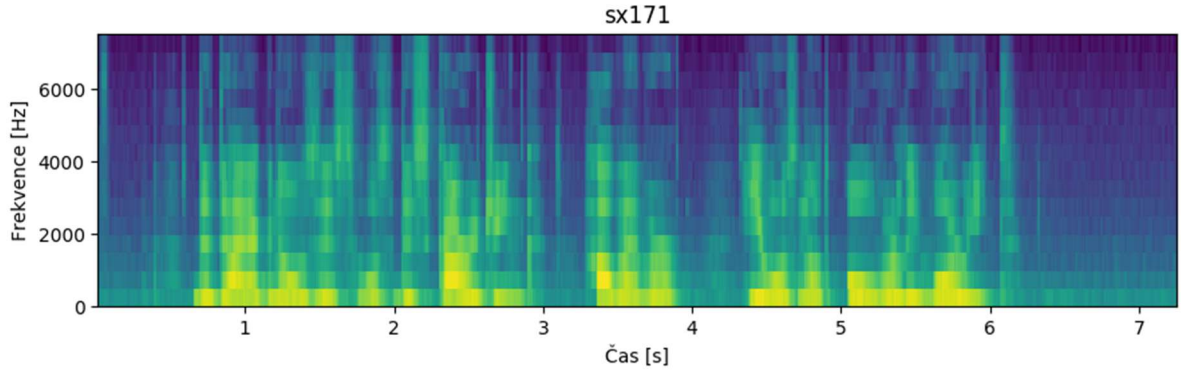
Úkol 1.

Názov	Dĺžka [vzorky]	Dĺžka[s]	Použitie
sa1.wav	82938	5.18	B
sa2.wav	75429	4.71	B
si711.wav	91472	5.71	B
si1341.wav	56314	3.51	B
si1841.wav	116730	7.29	B
sx81.wav	70992	4.43	B
sx171.wav	116389	7.27	B
sx261.wav	73381	4.58	B
sx351.wav	113317	7.08	B
sx441.wav	77818	4.86	B

Úkol 2.

Názov	Dĺžka [vzorky]	Dĺžka[s]
q1.wav	9419	0.58
q2.wav	9198	0.57

Úkol 3.



Úkol 4.

Do matice *‚features‘* ukladám zredukované vektory z matice *‚sgr‘*, ktoré získam nasledovne: Prechádzam prvkami poľa *‚sgr‘* a ich hodnoty sčítavam. Keď dôjdem na 16. prvok (podmienka *(j + 1)%16* pretože sa indexuje od 0), urobím priemer týchto hodnôt a výsledok uložím ako prvok do matice *‚features‘* na príslušný index. Počet vektorov je daný počtom rámcov nahrávky.

```
f, t, sgr = spectrogram(s, fs, nperseg=400, noverlap=240, nfft=511, scaling='density')  #  400 240 511

features = np.empty((len(sgr[1]), 16))  # np array do ktorej budem vkladat aritmeticke priemery

sgr = 10 * np.log10(sgr + 1e-20)
i = 0
result = 0
while i < len(sgr[1]):
    k = 0
    j = 0
    while j < 256:
        result = result + sgr[j][i]
        if (j + 1) % 16 == 0:
            result = result / 16
            features[i][k] = result
            k += 1
            result = 0
        j += 1
    i += 1
```

Úkol 5.

Do poľa budem vkladať *Pearsonove koeficienty podobnosti*, ktoré získam na základe výpočtu podobnosti segmentu vety a *features* slova. Segmenty sa posúvajú po 1 vektore až dokým nenarazia na *len(veta) – len(slovo)*.

```
counter = 0                                 # počítadlo pre výpočet koeficientu
n_of_rep = len(features_s) - len(features_w)    # počet opakovaní, rovná sa dĺžke vety mínus dĺžke slova aby slovo "nepretieklo" za vetu
list1 = np.empty((len(sgr1[1]), 16))        # segment z vety
list2 = []                                  # array s koeficientmi podobnosti
counter2 = 0                                # počítadlo pre naberanie vektorov segmentu vety

# prve query
while counter < n_of_rep:
    while counter2 < len(features_w):                       # segment o dĺžke slova
        list1[counter2] = features_s[counter + counter2]    # vloženie vektoru do arraye
        counter2 += 1                                       # navýšenie počítadla

    counter2 = 0                                            # vynulovanie počítadla
    kf = pearsonr(list1, features_w)                        # výpočet pearsonovho korelačného koeficientu
    list2.append(kf[0])                                     # koeficient sa vloží do arraye
    counter += 1                                            # navýšenie počítadla
```

Úkol 6.

Obrázky signálov priložené na konci

Úkol 7.

Z grafov je vidieť, že podobnosť miestami ‚vyskočí' tam kde nemá, spravidla na začiatku alebo na konci viet. Avšak ignorujúc túto vadu, tak počas priebehu vyslovovania vety skutočne našlo dané slovo. Prah som empiricky definoval ako koeficient *kf < 0.93* pre *query1* a *kf < 0.92* pre *query2* na základe pozorovaní, aby som sa dostal nad hranicu vady a pod hranicu hitu.

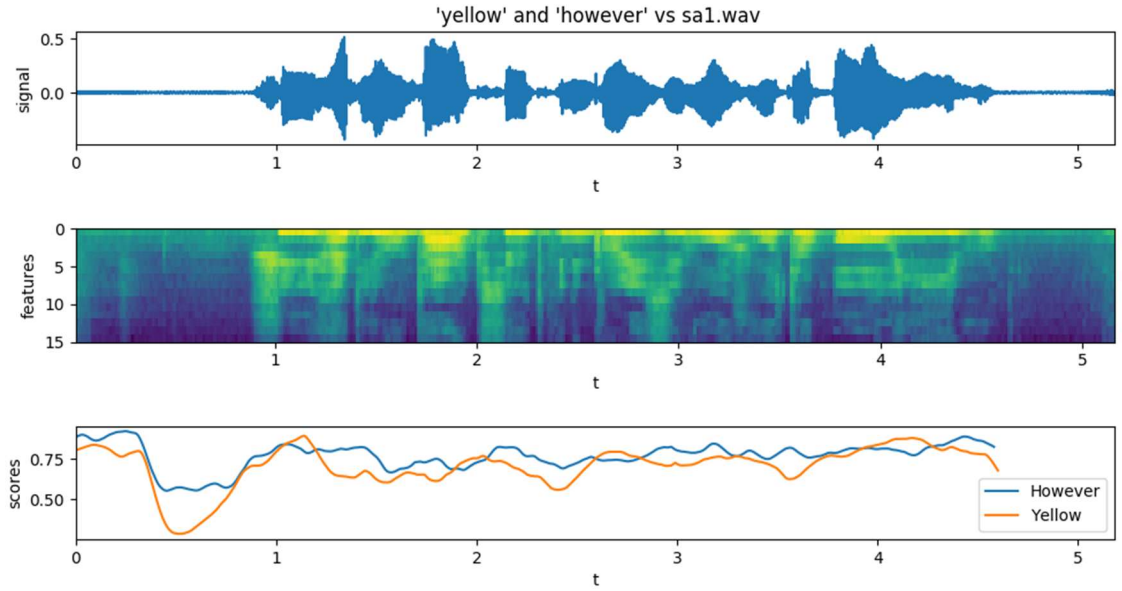
Úkol 8.

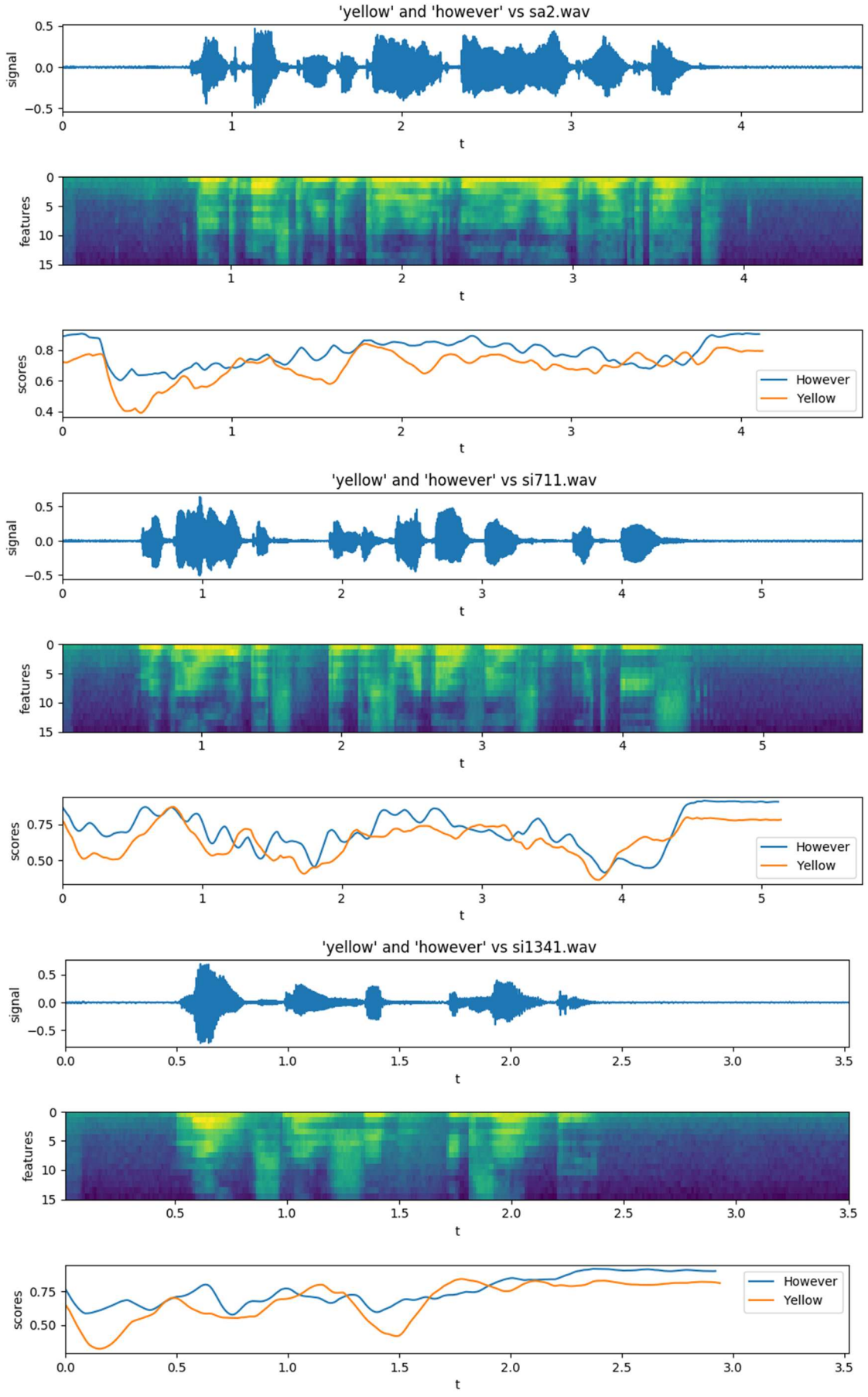
Query	Koeficient	Vektor (poradie)	Veta
1	0.9328	331	sx171
1	0.9403	332	sx171
1	0.9413	333	sx171
1	0.9370	334	sx171
1	0.9272	335	sx171
2	0.9224	130	sx261
2	0.9238	131	sx261
2	0.9222	132	sx261

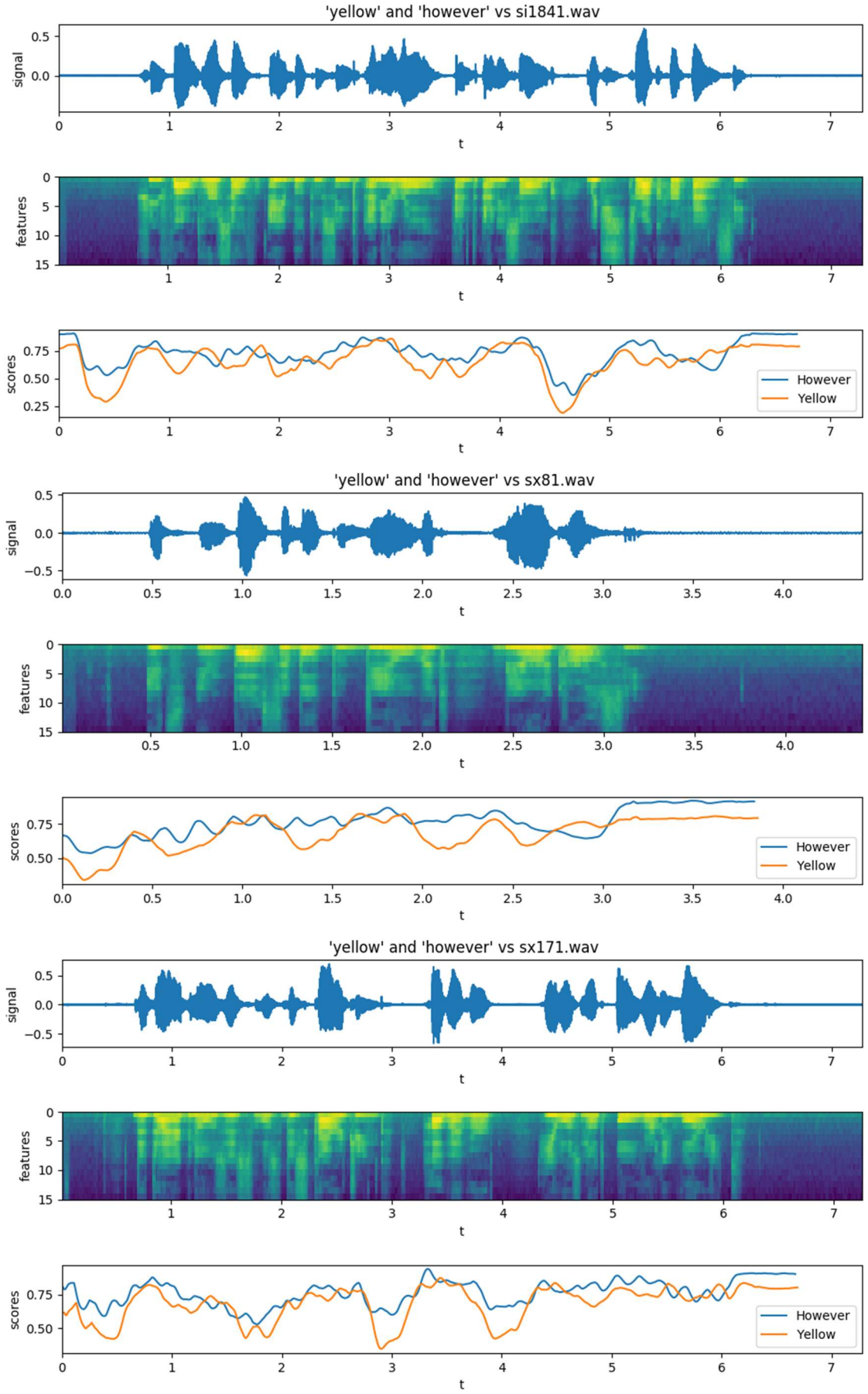
Záver:

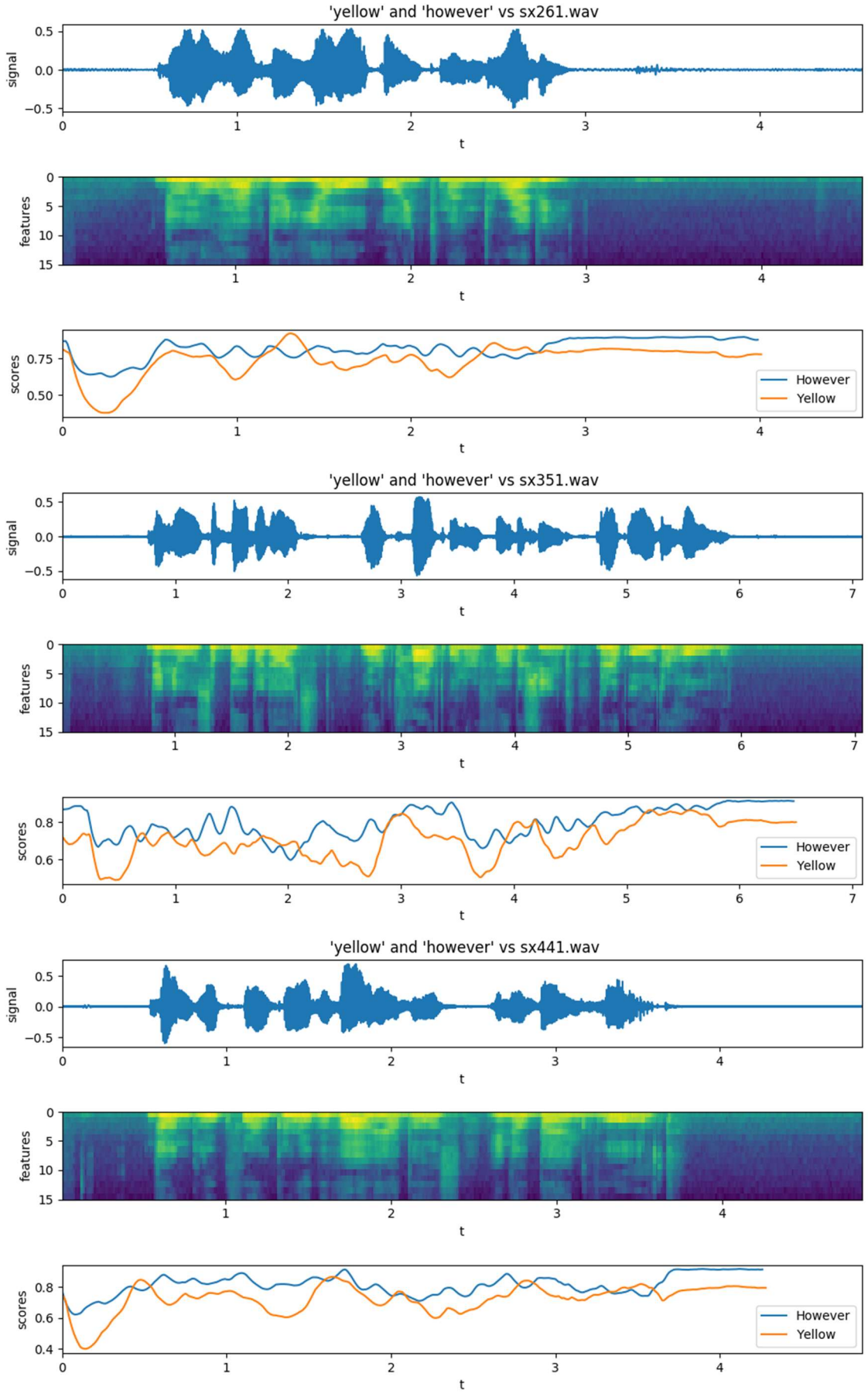
S mojím detektorom som bol priaznivo prekvapený, očakával som oveľa skreslenejšie a nepresnejšie výsledky. Detektor je určite použiteľný a dokáže nájsť slová, avšak je to vysoká šanca falošných nálezov a funguje korektne len za špeciálnych podmienok ako napríklad zbavenie slova ticha. Najlepšie výsledky vytvorí počas hovorenia vety a najviac problémov robí pre detektor začiatok a koniec nahrávky, kde sa nachádza najviac ticha, čo môže vyhodnotiť ako falošný nález. Zlepšiť presnosť by mohlo napríklad predspracovanie nahrávok viet a až následné vyhľadávanie slov, ako napríklad zbavenie viet ticha.

Obrázky




'yellow' and 'however' vs sa2.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs si711.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs si1341.wav
signal
features
scores
However
Yellow
t


'yellow' and 'however' vs si1841.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs sx81.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs sx171.wav
signal
features
scores
However
Yellow
t


'yellow' and 'however' vs sx261.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs sx351.wav
signal
features
scores
However
Yellow
t
'yellow' and 'however' vs sx441.wav
signal
features
scores
However
Yellow
t