Jiří Žák, xzakji02

1.

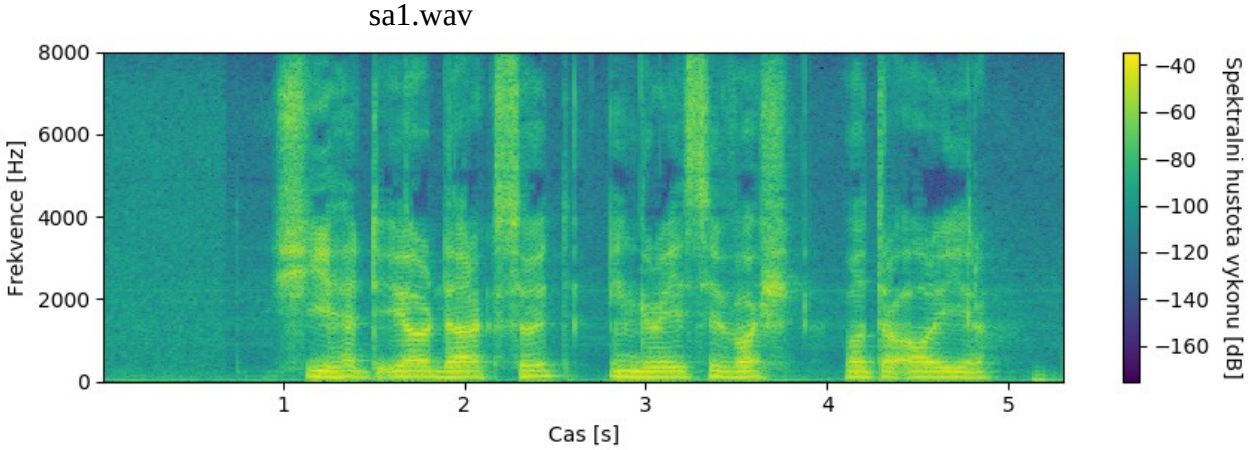
Název	Délka vět ve vzorcích	Délka vět v sekundách
sa1.wav	85055	5.32
sa2.wav	60906	3.81
si1067.wav	59420	3.71
si1697.wav	73909	4.62
si2327.wav	60906	3.81
sx167.wav	64250	4.02
sx24.wav	51247	3.20
sx257.wav	48274	3.02
sx437.wav	70566	4.41
sx77.wav	59420	3.71

Můžete to využít pro b).

2.

Název	Slova	Délka vět ve vzorcích	Délka vět v sekundách
q1.wav	employee	12195	0.76
q2.wav	reorganization	20253	1.27

3.




```
s, fs = sf.read('sa1.wav')
s = s[:250000]
t = np.arange(s.size) / fs
f, ti, sgr = spectrogram(s, fs, nperseg=400, noverlap=240, nfft=511)
sgr_log = 10 * np.log10(sgr+1e-20)
```

Jiří Žák, xzakji02

4.
Funkce pro výpočet parametrů

```
def ctyrka (sgr, sgr_log):
    new_sgr = np.zeros([sgr.shape[0] // 16, sgr.shape[1]])
    sestnact = vypocet = pom = 0
    for i in range(len(sgr_log[0])):
        for j in range(len(sgr_log)):
            vypocet += sgr_log[j][i]
            sestnact += 1
            if sestnact == 16:
                new_sgr[pom][i] = vypocet
                sestnact = vypocet = 0
                pom += 1
        pom = 0
    return new_sgr
```

Nejdříve si uděláme matici podle výstupu ze spektogramu s tím, že vydělíme počet řádků 16. Tuto matici si vynulujeme. Matice ze spektotogramu je parametrem této funkce. Druhý parametr je zlogaritmovaná matice spektogramu. Do vytvořené matice si budeme ukládat vypočtené parametry. Poté sčítáme hodnoty v jednom sloupci po 16 řádcích. Vypočtené hodnoty postupně ukládáme do naší matice. Po naplnění matici ji vrátíme. Jsou zde ještě 3 pomocné proměnné, které ošetřují, aby vše fungovalo jak má.

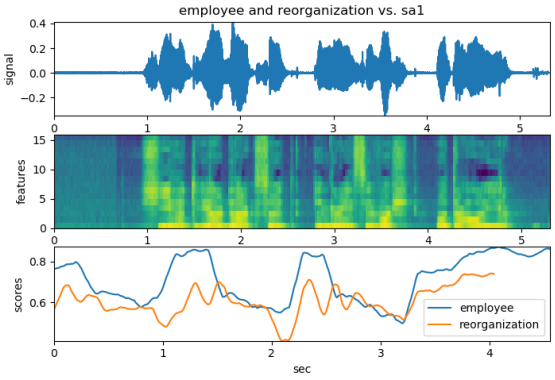
5.
Funkce pro výpočet skóre klíčového slova pro jednu větu

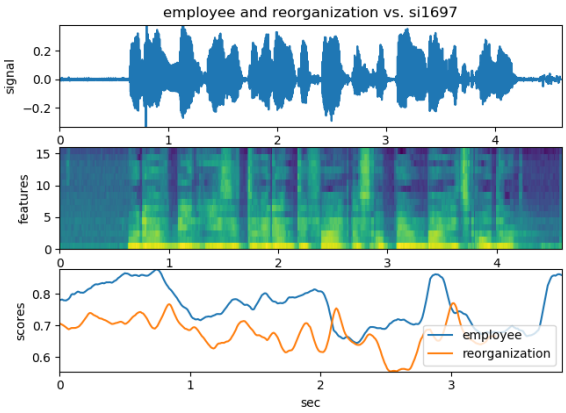
```
def petka(new_sgr, new_sgr2):
    F = np.array(new_sgr)
    F = F.transpose()
    Q = np.array(new_sgr2)
    Q = Q.transpose()
    pp = 0
    score = 0.0
    rating_array = np.zeros([len(F)-len(Q)])

    while pp < len(F)-len(Q) :
        for k in range(len(Q)):
            rcoef, pval = pearsonr(Q[k], F[k + pp])
            score += rcoef
        rating_array[pp] = score/len(Q)
        pp += 1
        score = 0
    return rating_array
```

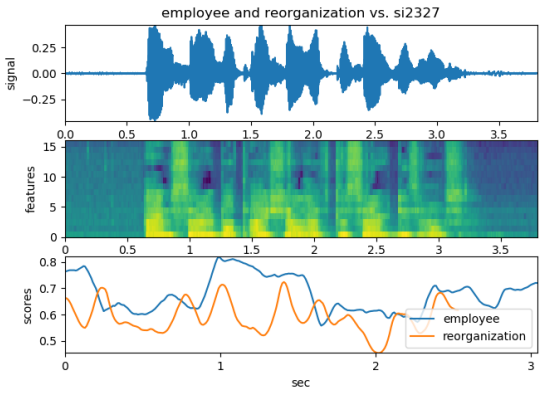
Vstupem této funkce, jsou dvě pole s parametry. První je z věty, kterou prohledáváme a druhé z klíčového slova. Obě pole nejdříve musíme transponovat, aby šli porovnávat ve funkci pearsonr. Poté pole věty postupně procházíme a zkoušíme porovnáváme s polem klíčového slova. Výsledky sčítáme do pomocné proměnné score a ukládáme do pomocného pole, které na konci vracíme. Než však score uložíme do výsledného pole musíme ho ještě podělit kvůli normalizaci. Máme zde ještě jednu pomocnou proměnnou pro krokování.

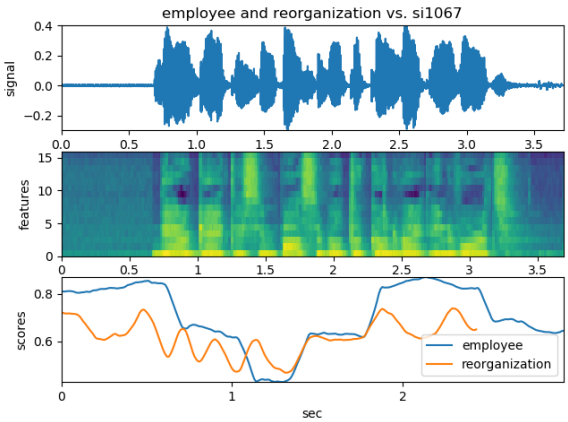
Jiří Žák, xzakji02

6.

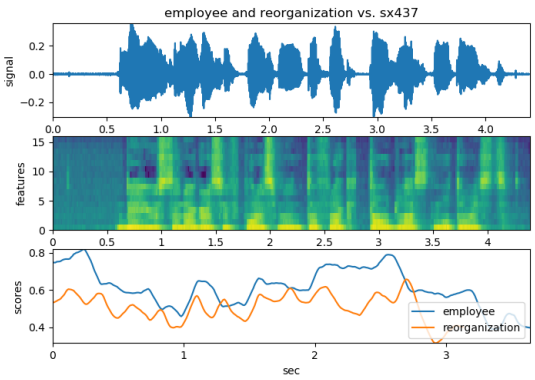


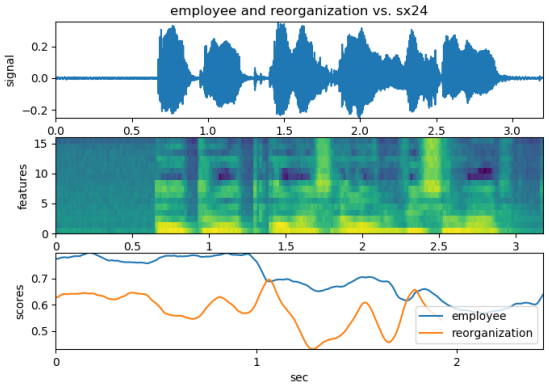






Jiří Žák, xzakji02





7.
Hledané querry by se mohlo vyskytovat, když score přesáhne hranici, kterou bych určil asi něco nad 0.85 . Podle těch grafů co jsem vygeneroval se na tuto hranici dostáváme opravdu zřídka, takže by se tam to querry mohlo vyskytovat. Je i možné, že je tam čistě jen podobná nějaká část slova, takže se může jednat pouze o chybnou podobnost.

8.

9. Závěr
Řekl bych, že detektor by mohl fungovat. Podle výsledků si myslím, že by dokázal dané querry najít. Na druhou stranu je i dost pravděpodobné, že by dělal chyby. Proto aby byli výsledky přesnější, tak by se určitě ještě hodilo kód zoptimalizovat a popřípadě udělat i lepší nahrávky. Vím, že moje výslovnost nebude moc dobrá.