Rozpoznawanie człowieka metodami biometrii
Projekt 2. — Rozpoznawanie na podstawie głosu
Raport

Bartłomiej Dach

12 kwietnia 2019

Poniższy dokument stanowi sprawozdanie z implementacji aplikacji dokonującej rozpoznawania człowieka na podstawie zarejestrowanych próbek głosu. W dokumencie opisano zastosowaną metodę opartą na współczynnikach mel-cepstralnych oraz zawarto wyniki działania dla próbek zarejestrowanych przez studentów uczęszczających na zajęcia.

1 Wstęp

Ludzki głos jest cechą biometryczną z pogranicza cech biologicznych i behawioralnych. Podczas gdy barwa głosu jest determinowana przez wrodzone czynniki anatomiczne, ton głosu i akcent stosowany podczas wymawiania określonych fraz stanowi cechę nabytą podczas nauki mowy.

Rozpoznawanie człowieka na podstawie mowy to prężnie rozwijające się zagadnienie. Ze względu na rosnącą popularność urządzeń interpretujących frazy wymawiane przez użytkowników takich, jak Google Home czy Amazon Echo, identyfikacja na podstawie głosu stanowi ważny kierunek rozwoju.

W ramach projektu zaimplementowano prostą metodę klasyfikacji próbek na podstawie danego wcześniej zbioru treningowego, używającą współczynników mel-cepstralnych. Szczegółowy opis metody znajduje się w sekcji 3.

2 Opis aplikacji

Opracowana aplikacja została zaimplementowana w języku skryptowym Python w wersji 3.5.2. Głównym uzasadnieniem wyboru tego języka był czas implementacji — istniejące biblioteki *open source* pozwalają na szybkie opracowanie algorytmu i usprawniają proces przetwarzania danych.

Aplikacja ma postać zbioru skryptów konsolowych — zdecydowano się na rezygnację z interfejsu graficznego ze względu na jego niewielką przydatność w stosunku do czasu potrzebnego na jego opracowanie. Sposób wywołania skryptów opisany jest w podsekcji 2.2.

2.1 Zastosowane biblioteki

W implementacji zastosowano kilka bibliotek *open source*, aby ominąć konieczność implementacji operacji niezbędnych do zaimplementowania algorytmu rozpoznawania, takich, jak m.in. transformata Fouriera. Pełna lista zastosowanych bibliotek, łącznie z nazwami licencji, znajduje się w tabeli 1.

Nr	Nazwa	Opis	Licencja	
1	`matplotlib` 3.0.3	Tworzenie wykresów i wizualizacji	PSF	[2]
2	`numpy` 1.16.2	Wielowymiarowe tablicowe struktury danych	BSD	[5]
3	`pandas` 0.24.2	Struktury do manipulacji i analizy danych	BSD	[4]
4	`seaborn` 0.9.0	Rozszerzone wizualizacje danych	BSD	[6]
5	`scipy` 1.2.1	Algorytmy pomocnicze (transformata Fouriera, manipulacja dźwiękiem)	BSD	[3]

Tablica 1: Lista bibliotek użytych w projekcie

2.2 Instrukcja obsługi

W celu uruchomienia skryptów do rozpoznawania konieczne jest zainstalowanie interpretera Python oraz bibliotek zawartych w tabeli 1. Aby zainstalować wymagane biblioteki, należy wywołać polecenie

```
$ pip3 install -r requirements.txt
```

gdzie plik `requirements.txt` to plik dołączony do źródeł aplikacji.

Dla uproszczenia założono uspójnione nazewnictwo i format rozpoznawanych próbek. Próbki powinny być plikami `.wav` o częstotliwości próbkowania 22050 Hz, o nazwach formatu `speaker_XX_PHRASE_Y.wav`, gdzie

- `XX` to numer identyfikacyjny mówcy,
- `PHRASE` to nazwa identyfikująca frazę, która została zarejestrowana (dla rozważanego zbioru zarejestrowane zostały cztery frazy: `biometria`, `chrzaszcz`, `poniedzialek` i `wycieczka`),
- `Y` to numer próbki (w rozważanym zbiorze każda fraza była rejestrowana trzy razy).

2.2.1 Skrypt `recognize.py`

Pierwszy ze skryptów wykonywalnych, o nazwie `recognize.py`, dokonuje podstawowego rozpoznawania podanej próbki na podstawie zbioru próbek treningowych. Składnia wykonania skryptu to:

```
python3 recognize.py [-h] [-t THRESHOLD]
                     TRAIN_SAMPLE [TRAIN_SAMPLE ...]
                     SAMPLE
```

gdzie:

- opcja `-h` wyświetla pomoc dot. wywołania skryptu,
- opcja `-t` (lub `-threshold`) pozwala na wyspecyfikowanie progu używanego w klasyfikacji. Domyślnie przyjmowany jest próg $t = 1000$. Więcej informacji dot. progu znajduje się w podsekcji 3.2.
- Parametry `TRAIN_SAMPLE` to ścieżki do próbek używanych do treningu klasyfikatora. Pliki z próbkami powinny mieć nazwę zawierającą podciąg `speaker_XX`, gdzie `XX` to numer identyfikujący mówcę. Wymagane jest podanie co najmniej jednej próbki.
- Parametr `SAMPLE` to ścieżka do próbki, która powinna być zaklasyfikowana. Możliwe jest podanie tylko jednej próbki.

Prawidłowe wywołanie skryptu powoduje wypisanie na wyjście linii poleceń wyniku klasyfikacji. W zależności od wyniku klasyfikatora próbka może być zakwalifikowana jako należąca do jednego z mówców, lub jako nieznana próbka.

2.2.2 Skrypt `crossvalidator.py`

Drugi ze skryptów o nazwie `crossvalidator.py` dokonuje kroswalidacji algorytmu klasyfikacji na podstawie podanego zbioru próbek. Składnia wykonania skryptu to:

```
python3 crossvalidator.py [-h] SAMPLE_DIR N PHRASE
```

gdzie:

- opcja `-h` wyświetla pomoc dot. wywołania skryptu,
- parametr `SAMPLE_DIR` to katalog z próbkami, na podstawie których wykonywana jest kroswalidacja,
- parametr `N` to liczba próbek dla każdego mówcy i frazy zawartych w folderze,

- parametr `PHRASE` oznacza frazę, dla której powinna być wykonana kroswalidacja.

Schemat kroswalidacji jest następujący:

1. Zbiór próbek dzielony jest na zbiory: testowy i treningowy na N sposobów. Dla $i = 1, \ldots, N$ i-ty podział ma i-te próbki w zbiorze treningowym i wszystkie pozostałe w zbiorze testowym. W związku z tym każdy mówca ma $N - 1$ próbek w zbiorze treningowym i jedną w zbiorze testowym.

2. Dla każdego z podziałów następuje klasyfikacja zbioru testowego (czyli jednej próbki dla każdego mówcy). Wynik klasyfikacji porównywany jest z docelowym wynikiem, zliczane są: błędy pierwszej i drugiej kategorii oraz pozytywne klasyfikacje.

3. Skrypt bada również wpływ progu na jakość klasyfikacji, rozważając wartości progu z przedziału $[500, 1500]$ z krokiem 100.

Skrypt tworzy w katalogu roboczym zbiór wyjściowych plików `.csv`, na podstawie których można oszacować jakość klasyfikacji i optymalny dobór progu:

- Pliki `confusion_matrix_PHRASE_t=THRESHOLD.csv` zawierają macierze pomyłek dla wybranej frazy i wartości progu równej `THRESHOLD`.

- Plik `acceptance_rates_PHRASE.csv` zawiera liczby: prawidłowych klasyfikacji, fałszywych pozytywów i fałszywych negatywów dla danej frazy i rozważanych wartości progu.

Wykresy z sekcji 4 zostały wygenerowane na podstawie tych plików.

3 Opis metody

U podstawy zastosowanej metody rozpoznawania głosu leży opracowana przez Bridle'a i Browna [1] metoda obliczania współczynników melowo-cepstralnych (ang. MFCC — *mel-frequency cepstral coefficients*). Pozwala ona na przekształcenie sygnału dźwiękowego w wektory liczbowe, które stanowią cechy, których można użyć w połączeniu z klasyfikatorami do wyznaczenia osoby, do której należy zarejestrowana próbka. Dokładniejszy opis zaimplementowanego algorytmu przetwarzania znajduje się poniżej.

3.1 Wyznaczanie współczynników mel-cepstralnych

Współczynniki mel-cepstralne obliczane są zarówno dla próbek treningowych, jak i rozpoznawanej próbki. Pozwalają one na znaczną redukcję ilości danych, które trzeba podać do końcowego klasyfikatora. Poszczególne kroki obliczania opisane są w poniższych akapitach.

Normalizacja. Przed rozpoczęciem przetwarzania, wejściowy sygnał dźwiękowy jest normalizowany do zakresu $[0, 1]$, aby uprościć obliczenia w dalszych fazach.

Preemfaza. Po normalizacji stosowana jest preemfaza, będąca metodą filtracji formującej. Eliminuje ona składowe o niskich częstotliwościach i wzmacnia te o wysokich częstotliwościach. Z matematycznego punktu widzenia operację tą opisuje wzór

$$s'_i = s_i - a \cdot s_{i-1}$$

gdzie s_i to i-ta próbka z sygnału wejściowego, zaś a to regulowany parametr filtra. W zaimplementowanym wariancie wybrano wartość $a = 0.97$.

Ramkowanie sygnału. Następnie sygnał dzielony jest na krótkie ramki, dla których wyznaczane będą współczynniki melowo-cepstralne. Ramki mogą na siebie nachodzić. W zaimplementowanym rozwiązaniu zdecydowano się na układ ramek, w którym początki ramek są od siebie odległe o 15 milisekund, co przy założeniu częstotliwości próbkowania 22050 Hz przekłada się na

$$\frac{15}{1000} \cdot 22050\,[\mathrm{s} \cdot \mathrm{Hz}] = 330$$

próbek odstępu między kolejnymi ramkami. Każda ramka składa się z 2048 próbek.

Zastosowanie funkcji okna. Sygnał podzielony na ramki następnie zostaje poddany okienkowaniu. Każda z N próbek w ramce jest modulowana za pomocą wybranej funkcji okna $w(n)$ według wzoru:

$$s'_i = s_i \cdot f(i)$$

gdzie $i = 0, \ldots, N-1$ to numer próbki w ramce. Przy wyznaczaniu MFCC najczęściej stosowane jest okno Hamminga, określone wzorem

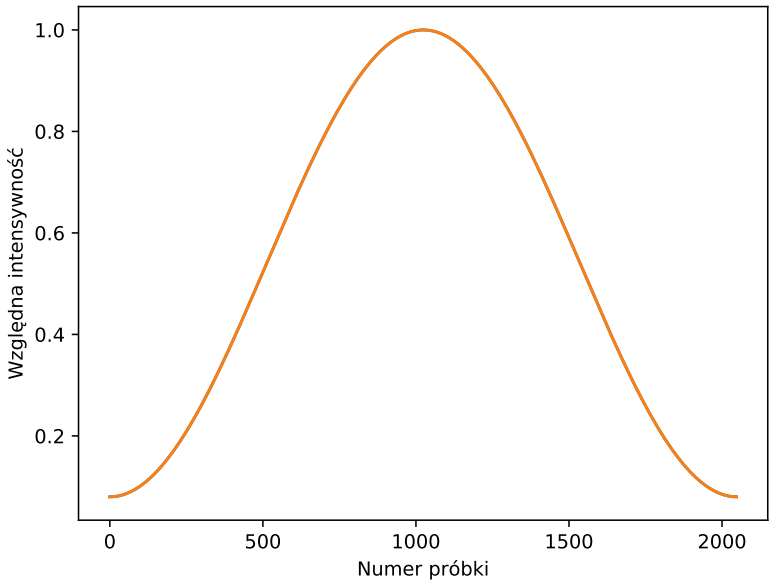
$$w(n) = a - (1-a)\cos\left(2\pi\frac{n-1}{N-1}\right)$$

gdzie $a = \frac{25}{46} \approx 0.54$. Wykres funkcji okna Hamminga dla $N = 2048$ znajduje się na rysunku 1.

Szybka transformata Fouriera. Okienkowany sygnał jest następnie przekształcany z dziedziny czasowej w dziedzinę częstotliwościową przy pomocy szybkiej dyskretnej transformaty Fouriera (ang. FFT, *fast Fourier transform*), określonej wzorem:

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \exp\left(-\frac{2\pi i}{N} kn\right)$$

gdzie $k = 0, \ldots, N-1$. W zastosowanym schemacie przetwarzania przyjęto, że każde z okien transformowanych przy pomocy FFT składa się z $N = 2048$ próbek. W wyniku przekształceń uzyskiwane jest zatem 2048 liczb zespolonych.



Rysunek 1: Wykres funkcji okna Hamminga dla $N = 2048$ próbek.

Obliczanie mocy sygnału. W celu powrotu do dziedziny liczb rzeczywistych, zespolone współczynniki dyskretnej transformaty Fouriera są przekształcane poprzez wzięcie kwadratu ich modułu:

$$X'_k = |X_k|^2$$

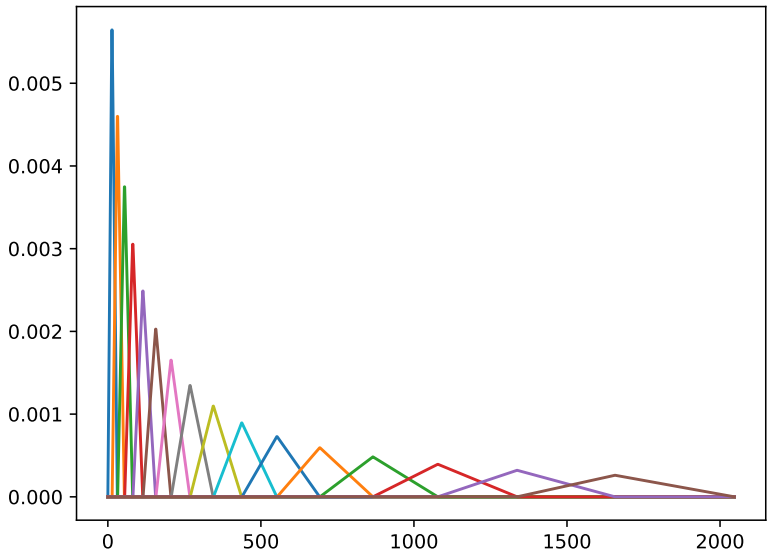
Zastosowanie filtrów trójkątnych. Po obliczeniu mocy sygnału każde okno ma postać wektora rzeczywistego o długości 2048. W celu dalszej analizy tworzone jest 16 filtrów trójkątnych, rozłożonych równomiernie w przedziale $[0, 2048]$ w skali melowej. Skala melowa jest skalą częstotliwości, w przybliżeniu proporcjonalną do logarytmu częstotliwości w hercach. Konwersja między skalą melową a hercową odbywa się poprzez zastosowanie wzorów:

$$x \text{ [Hz]} \to 2595 + \log_{10}\left(1 + \frac{x}{700}\right) \text{ [mel]}$$
$$x \text{ [mel]} \to 700 \cdot \left(10^{x-2595} - 1\right) \text{ [Hz]}$$

Filtry trójkątne są ponadto normalizowane pod względem pola trójkąta w skali hercowej:

$$f^j_{\text{norm}}(n) = \frac{f^j(n)}{\sum_{i=1}^{N} f^j(i)}$$

gdzie j oznacza numer filtra, $j = 0, \ldots, 15$. Ostateczną postać krzywych filtrów obrazuje rysunek 2.



Rysunek 2: Zbiór $k = 16$ filtrów trójkątnych rozłożonych równomiernie w paśmie melowym stosowanych do filtracji.

Filtry są stosowane dla każdego okna poprzez wymnożenie próbek z okna z wartością filtra:

$$p_j = \sum_{k=0}^{N-1} X'_k \cdot f^j_{\text{norm}}(k)$$

gdzie j oznacza numer filtra, $j = 0, \ldots, 15$, zaś k — numer współczynnika transformaty, $k = 0, \ldots, 2047$.

Logarytmowanie mocy po filtracji. W wyniku zastosowania filtrów trójkątnych dla każdego okna uzyskiwany jest wektor długości 16, reprezentujący jedno z przefiltrowanych pasm częstotliwościowych. W celu zamodelowania nieliniowej odpowiedzi ludzkiego ucha na bodźce, wektory te są logarytmowane element po elemencie, np. według wzoru

$$p'_j = 10 \log_{10} p_j$$

Dyskretna transformata kosinusowa. Ostatecznie, dane z pasm dla każdego okna są transformowane przy pomocy dyskretnej transformaty kosinusowej (ang. DCT, *discrete cosine transform*). Końcowe wartości współczynników MFCC dla każdego okna liczone są

wg wzoru

$$c_i = \sqrt{\frac{2}{M}} \sum_{j=0}^{M-1} p'_j \cos\left(\frac{\pi i}{M}\left(j + \frac{1}{2}\right)\right)$$

gdzie i to numer współczynnika transformaty kosinusowej (w projekcie wybrano $i = 0, \ldots, 39$), $M = 16$ to liczba zastosowanych filtrów trójkątnych.

W rezultacie strumień przetwarzania zwraca macierz rozmiaru $40 \times W$, gdzie W to liczba okien, na które podzielona została próbka głosu, zależna od jej długości w sekundach.

3.2 Klasyfikacja próbek weryfikowanych

Niech dany będzie zbiór N próbek z głosu razem z etykietami: $\{(s_1, c_1), \ldots, (s_N, c_N)\}$, gdzie s_i to współczynniki cepstralne wyznaczone z i-tej próbki, zaś c_i będzie etykietą identyfikującą mówcę zarejestrowanego w danej próbce. Klasyfikacja nieznanej próbki rozpoczyna się obliczeniem jej współczynników cepstralnych, które oznaczymy przez s'.

Oznaczmy przez s_i^j współczynniki cepstralne j-tego okna i-tej próbki treningowej ($j = 0, \ldots, W_i - 1$) i przez s'^j współczynniki j-tego okna identyfikowanej próbki ($j = 0, \ldots, W' - 1$). Dodatkowo zdefiniujmy miarę odległości współczynników cepstralnych dwóch okien jako odległość euklidesową:

$$d(s'_j, s_i^k) = \|s'_j - s_i^k\|_2$$

Wówczas możemy zdefiniować odległość próbki identyfikowanej od jednej z próbek treningowych wzorem

$$d(s_i, s') = \frac{1}{W'} \sum_{j=0}^{W'-1} \min_{k=0,\ldots,W_k-1} d(s'_j, s_i^k)$$

Interpretacja tej odległości jest następująca:

- dla każdego okna próbki identyfikowanej obliczamy odległość współczynników okna od współczynników wszystkich okien próbki treningowej,
- wybieramy najmniejsze takie odległości dla każdego okna próbki identyfikowanej i sumujemy je.

W ten sposób otrzymujemy jedną liczbę, określającą podobieństwo próbki weryfikowanej i próbki treningowej. Zatem można obliczyć taką odległość dla wszystkich próbek treningowych i zastosować dowolne proste klasyfikatory, takie jak np. k-NN (który zastosowano w zaimplementowanym programie).

W celu uniknięcia fałszywej identyfikacji osób niebędących w bazie treningowej jako jedna z istniejących, przed klasyfikacją przy pomocy k-NN wprowadzono progowanie odległości między próbkami. Po obliczeniu odległości między próbką klasyfikowaną a wszystkimi treningowymi, te próbki treningowe, które mają większą odległość niż zadany próg t od klasyfikowanej, są odrzucane. Klasyfikator k-NN operuje tylko na pozostałych punktach.

4 Wyniki eksperymentalne

Zaimplementowany potok przetwarzania i klasyfikacji został przetestowany na zebranym wewnętrznie przez studentów uczęszczających na zajęcia z przedmiotu zbiorze próbek głosu. W momencie przeprowadzenia testów w zbiorze znajdowały się 84 pliki dźwiękowe, składające się z trzech nagrań czterech fraz wypowiadanych przez 7 mówców. Zarejestrowane frazy to:

- słowo *biometria*,
- słowo *chrząszcz*,
- słowo *poniedziałek*,
- zdanie *Jutro pojadę na wycieczkę albo zostanę w domu.*

Testy klasyfikatora polegały na kroswalidacji przy użyciu skryptu `crossvalidate.py`, którego działanie opisano w podsekcji 2.2.2. Na podstawie plików wygenerowanych przez ten skrypt opracowano wykresy FAR i FRR (rysunek 3) i macierze pomyłek dla najlepszych wartości progu (rysunek 4).

Wyniki uzyskane w eksperymentach są bardzo dobre — przy korzystnym dobraniu progu dla danej frazy identyfikacja oprócz pojedynczych pomyłek kończy się zwróceniem właściwego wyniku. Dla zdania o wycieczce udało się zupełnie wyeliminować pomyłki, dla jednego z mówców zdarzały się jednak fałszywe odrzucenia. Optymalne progi dla wszystkich fraz oscylowały w okolicy wartości $t = 1000$.

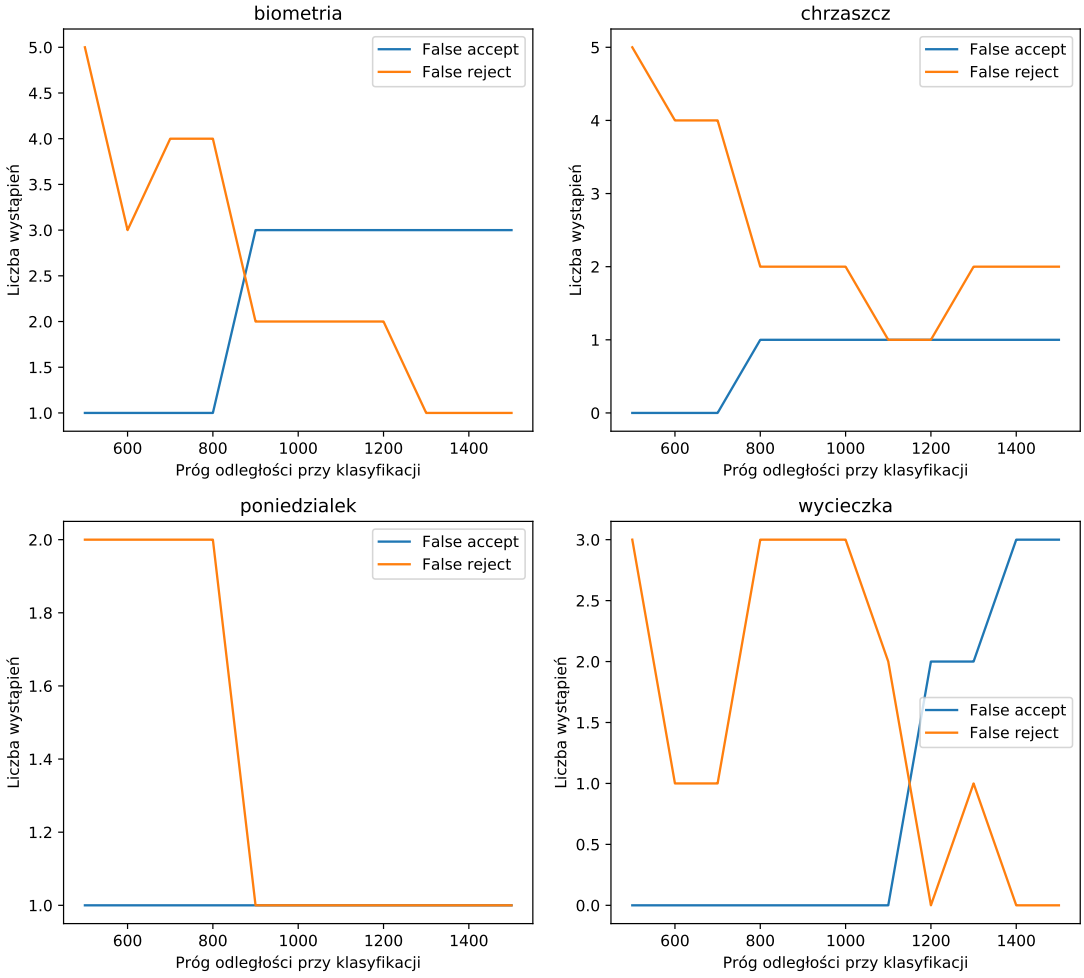
Biorąc pod uwagę prostotę zaimplementowanego rozwiązania są to bardzo satysfakcjonujące wyniki. Z wyników można wnioskować, że dla dobrego działania algorytmu korzystne jest gromadzenie jak najdłuższych próbek (najlepsze wyniki odnotowano dla najdłuższej wypowiadanej frazy). Można również domniemywać, że zwiększenie liczby próbek treningowych dla każdej frazy jeszcze bardziej polepszyłoby jakość klasyfikacji — w testach zbiór treningowy zawierał po 2 próbki każdej frazy, przy zastosowaniu klasyfikatora 3-NN. Zwiększenie liczby próbek najprawdopodobniej zwiększyłoby stabilność procesu identyfikacji.

5 Podsumowanie

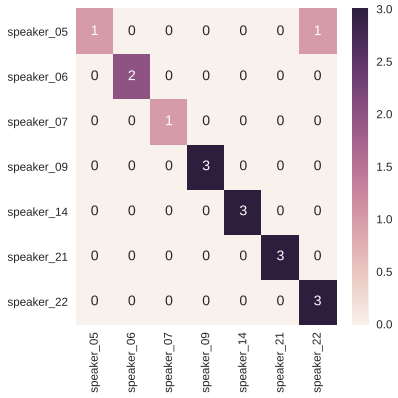
Literatura

[1] Bridle, J.S., Brown, M.D., „An Experimental Automatic Word-Recognition System", *JSRU Report*, nr 1003, Joint Speech Research Unit, 1974.

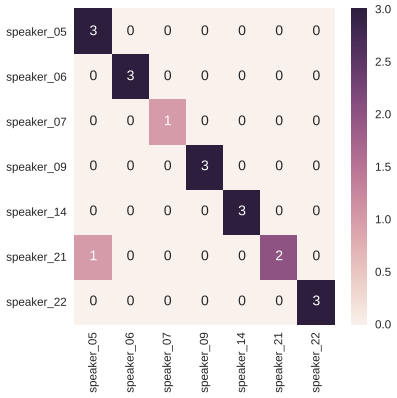
[2] „Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, tom 9, nr 3, s. 90–95, 2007.



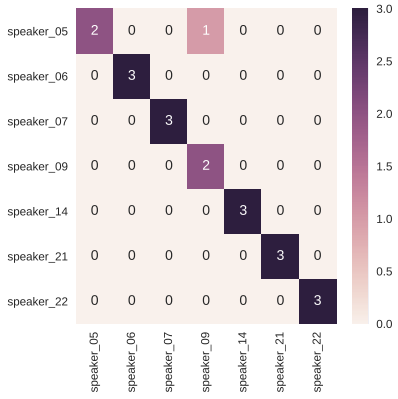
Rysunek 3: Liczba fałszywych pozytywów (ang. *false accept*) i fałszywych odrzuceń (ang. *false reject*) próbek głosów z testowanego zbioru w zależności od przyjętego progu odległości między próbkami podczas klasyfikacji. Oddzielono wyniki dla każdej z czterech rejestrowanych fraz.



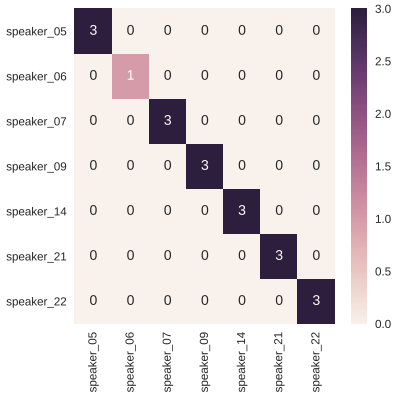
(a) Macierz pomyłek dla słowa *biometria* przy progu $t = 800$.



(b) Macierz pomyłek dla słowa *chrząszcz* przy progu $t = 800$.



(c) Macierz pomyłek dla słowa *poniedziałek* przy progu $t = 900$.



(d) Macierz pomyłek dla zdania *Jutro pojadę na wycieczkę, albo zostanę w domu* przy progu $t = 1100$.

Rysunek 4: Macierze pomyłek dla wartości progu minimalizujących liczbę fałszywych pozytywów i negatywów dla poszczególnych zarejestrowanych fraz.

[3] Jones, E., Oliphant T.E., Peterson P. i inni, „SciPy: Open source scientific tools for Python". [Online]
Dostępne: `https://www.scipy.org/`. [Dostęp 7 kwietnia 2019]

[4] McKinney, W., „Data Structures for Statistical Computing in Python", *Proceedings of the 9th Python in Science Conference*, s. 51–56, 2010.

[5] Oliphant, T.E., *A Guide to NumPy*, Trelgol Publishing, Stany Zjednoczone, 2006.

[6] Waskom, M. i inni, „`seaborn`: statistical data visualization". [Online]
Dostępne: `https://seaborn.pydata.org/`. [Dostęp 7 kwietnia 2019]