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

Bartłomiej Dach

11 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	[1]
2	`numpy` 1.16.2	Wielowymiarowe tablicowe struktury danych	BSD	[4]
3	`pandas` 0.24.2	Struktury do manipulacji i analizy danych	BSD	[3]
4	`seaborn` 0.9.0	Rozszerzone wizualizacje danych	BSD	[5]
5	`scipy` 1.2.1	Algorytmy pomocnicze (transformata Fouriera, manipulacja dźwiękiem)	BSD	[2]

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 pró"ki, 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

3.1 Wyznaczanie współczynników mel-cepstralnych

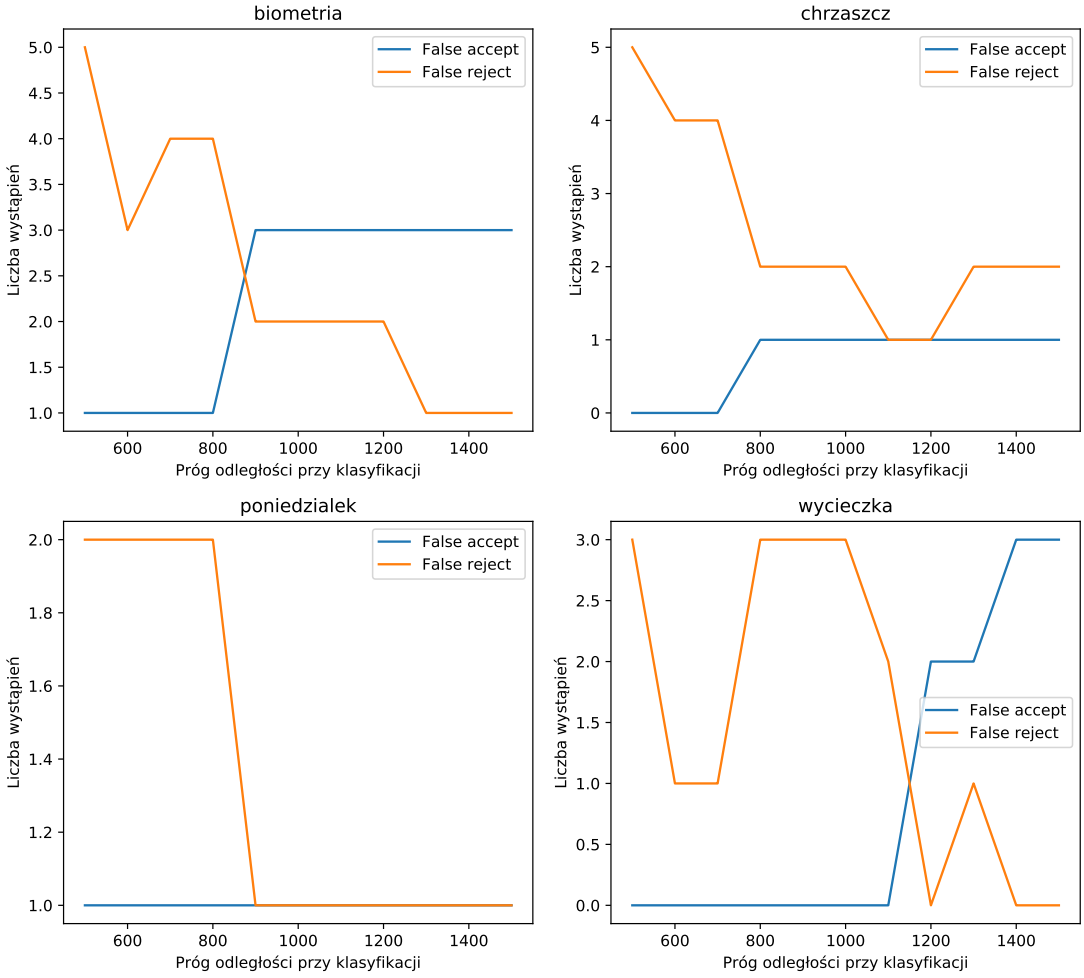
3.2 Klasyfikacja nowych próbek

4 Wyniki eksperymentalne

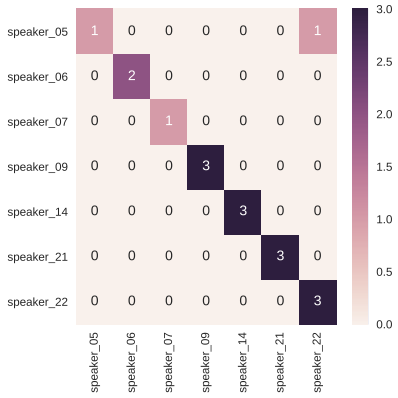
5 Podsumowanie

Literatura

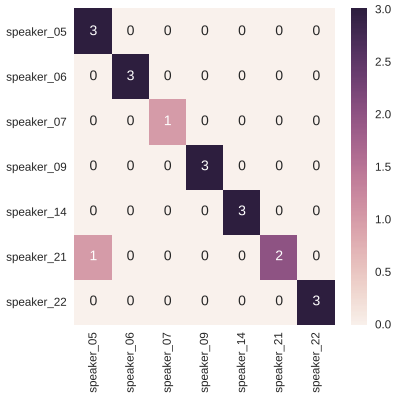
[1] Hunter, J.D., „Matplotlib: A 2D graphics environment", *Computing In Science & Engineering*, tom 9, nr 3, s. 90–95, 2007.



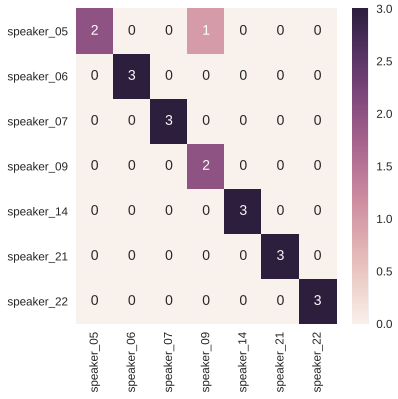
Rysunek 1: Liczba fałszywych pozytywów (ang. *false accept*) i fałszywych odrzuceń (ang. *false reject*) próbek głosów z testowanego zbioru w zależności od przyjętego progu odległości między próbkami podczas klasyfikacji. Oddzielono wyniki dla każdej z czterech rejestrowanych fraz.



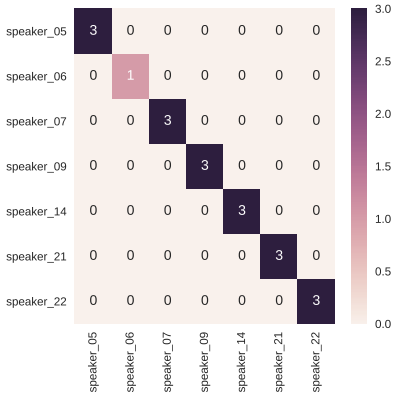
(a) Macierz pomyłek dla słowa *biometria* przy progu $t = 800$.



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

[2] 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]

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

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

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