Analiza i Przetwarzanie Dźwięku Projekt 3 - Rozpoznawanie mowy

Karolina Dunal

29 maja 2025

Spis treści

1 Opis Zadania

Celem projektu było rozpoznawanie mowy. Projekt zakładał wybór jednej z trzech metod parametryzacji mowy oraz jednej z trzech głównych funkcji rozpoznawczych:
Metody parametryzacji mowy:

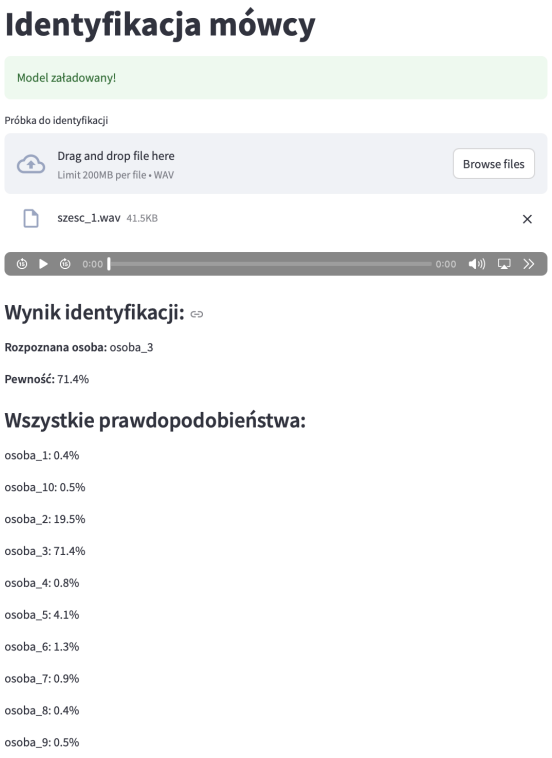
- **MFCC** – współczynniki mel-cepstralne (wybrana metoda),
- **DTW (Dynamic Time Warping)** – dopasowanie dynamiczne w czasie,
- **Formanty**.

Typ zadania:

1. **Weryfikacja osoby** – odpowiedź: „tak/nie”, czy to dana osoba,
2. **Identyfikacja osoby** – dopasowanie próbki głosu do osoby z bazy (wybrane podejście),
3. **Rozpoznawanie słów** – np. identyfikacja liczb.

2 Opis Aplikacji

Aplikacja umożliwia przeprowadzenie identyfikacji osoby na podstawie próbki głosu. Zostały w niej zaimplementowane funkcje pozwalające na obliczanie współczynników MFCC dla zadanej próbki dźwiękowej. Uzyskane cechy stanowią dane wejściowe dla klasyfikatora, który można wytrenować na podstawie zbioru próbek głosowych różnych mówców (wymagane są odpowiednie etykiety dla próbek). Identyfikacja odbywa się wyłącznie wśród osób dostępnych w bazie danych, na których wcześniej model został wytrenowany – aplikacja nie rozpoznaje głosów spoza znanego zestawu. Interfejs graficzny aplikacji został zaimplementowany wyłącznie w celu wygodniejszego korzystania z algorymtu niż w środowisku notebookowym. Ma on charakter pomocniczy i nie oferuje rozbudowanej funkcjonalności — umożliwia jedynie wczytanie próbki dźwiękowej oraz prezentację wyników predykcji wraz z odpowiadającymi jej prawdopodobieństwami.



Rysunek 1: Zrzut ekranu przedstawiający pomocniczy interfejs graficzny aplikacji.

3 Metody Użyte w Aplikacji

Zaimplementowany został pełny pipeline przetwarzania mowy służący do wyznaczania współczynników MFCC. Proces został zrealizowany bez użycia wyspecjalizowanych bibliotek do przetwarzania mowy, a jego etapy przedstawiono poniżej. Do implementacji wykorzystano biblioteki takie jak `librosa` do wczytywania i przetwarzania sygnału audio, `numpy` do operacji numerycznych, `xgboost` i `sklearn` do procesu treningu i walidacji klasyfikatora oraz `streamlit` do stworzenia prostego interfejsu.

Kod został podzielony na trzy moduły: `audio_analysis.py` – zawierający funkcje przetwarzania sygnału i ekstrakcji cech MFCC oraz `model.py` – implementujący trening, ewaluację i obsługę modelu klasyfikacyjnego oraz `app.py` – odpowiedzialny za interfejs użytkownika i logikę aplikacji.

Poniżej opisano najważniejsze z zaimplementowanych metod oraz ich działanie.

3.1 Plik `audio_analysis.py`

3.1.1 Metoda `pre_emphasis`

Funkcja implementuje filtracje wstępną, której celem jest wzmocnienie wyższych częstotliwości w sygnale mowy. Wykorzystywany jest filtr I rzędu:

$$y[n] = x[n] - \alpha \cdot x[n-1]$$

gdzie $x[n]$ to sygnał wejściowy, a α to współczynnik pre-emfazy (zdomyślnie $\alpha = 0{,}97$).

3.1.2 Metoda `framing`

Funkjca dzieli sygnał na krótkie fragmenty. Przyjęto domyślny rozmiar ramki (frame size) 25 ms oraz przesunięcie (stride) 10 ms.

3.1.3 Metoda `hamming_window`

Dla każdej ramki funkcja stosuje okno Hamminga w celu ograniczenia efektów obcięcia ramki.

3.1.4 Metoda `compute_fft`

Funkcja na każdej ramce wykonuje szybką transformację Fouriera (FFT) w celu przejścia do dziedziny częstotliwości. Następnie wyznaczana jest moc widma na podstawie poniższego wzoru:

$$P[k] = \frac{1}{N} \left|\mathrm{FFT}\{x[n]\}\right|^2$$

gdzie $x[n]$ to sygnał wejściowy, N to liczba punktów w dyskretnej transformacji Fouriera, k to indeks odpowiadający konkretnej składowej częstotliwości w widmie FFT.

3.1.5 Metoda `mel_filterbank`

Funkcja przekształca widmo mocy sygnału na skalę Mel, wykorzystując zestaw trójkątnych filtrów. Punkty graniczne filtrów są równomiernie rozmieszczone w skali Mel, a następnie konwertowane do skali Hz. Każdy filtr $H_m(k)$ ma postać trójkąta i nakładany jest na widmo mocy $P[k]$. Na wyjsciu każdego z filtrów otrzymywana jest energia pasma zgodnie z wzorem:

$$E_m = \sum_{k=0}^{N/2} P[k] \cdot H_m(k)$$

Energia ta następnie jest logarytmowana.

3.1.6 Metoda `dct_filterbanks`

Funkcja `dct_filterbanks` przeprowadza dyskretną transformację kosinusową (DCT) na logarytmowanych wartościach energii z filtrów Mel. DCT przekształca dane do przestrzeni współczynników cepstralnych, które reprezentują cechy sygnału mowy.
Dla każdej ramki n oraz współczynnika cepstralnego k obliczana jest wartość:

$$c_k(n) = \alpha_k \sum_{m=0}^{M-1} \log(S(n,m)) \cdot \cos\left[\frac{\pi k}{M}\left(m - \frac{1}{2}\right)\right]$$

gdzie $S(n,m)$ to energia filtra Mel o indeksie m w ramce n, M to liczba filtrów Mel, a współczynnik normalizacyjny $\alpha_k = \sqrt{\frac{2}{M}}$.

3.1.7 Metoda `extract_mfcc`

Funkcja ta łączy cały pipeline przetwarzania i opisane powyżej metody w spójną całość przeprowadzając cały proces wyznaczania współczynników MFCC.

3.2 Plik `model.py`

3.2.1 Metoda `load_dataset`

Metoda ta jest odpowiedzialna za wczytanie zbioru danych audio z podanego folderu. Dla każdego pliku wyodrębnia współczynniki MFCC, a także przypisuje etykietę mówcy na podstawie nazwy podfolderu. Zwraca listę cech oraz etykiet.

3.2.2 Metoda `train_model`

Metoda przeprowadza trening modelu XGBoost na podstawie wyekstrahowanych cech MFCC. Po wytrenowaniu model oraz enkoder etykiet są zapisywane do odpowiednich plików.

3.2.3 Metoda `evaluate_model`

Metoda ładuje wytrenowany model i enkoder etykiet, a następnie przeprowadza ewaluację na zbiorze testowym (analogicznie jak zbiór treningowy wczytywany za pomocą `load_dataset`). Oblicza i zwraca podstawowe miary jakości klasyfikacji.

3.2.4 Metoda `main`

Metoda umożliwia interaktywny wybór, pozwalający użytkownikowi zdecydować, czy trenować model, przeprowadzić ewaluację, czy wykonać obie czynności razem i uruchamia cały proces zgodnie z wybraną opcją.

4 Zbiór treningowy i testowy

Zbiór treningowy składał się z nagrań 10 różnych osób. Dla każdej osoby wybrano ten sam zestaw próbek 12 różnych słów, które posłużyły do nauki modelu. Zabieg ten miał na celu umożliwienie modelowi skuteczniejszego nauczenia się rozróżniania cech charakterystycznych dla poszczególnych mówców, a także eliminację wpływu specyfiki samych słów na końcowe współczynniki.

Zbiór testowy również obejmował nagrania tych samych 10 osób. Dla każdej osoby wybrano po 10 próbek, oczywiście innych niż w zbiorze treningowym. Co istotne, w przeciwieństwie do zbioru treningowego, zestawy wypowiadanych słów różniły się pomiędzy osobami.

5 Wyniki

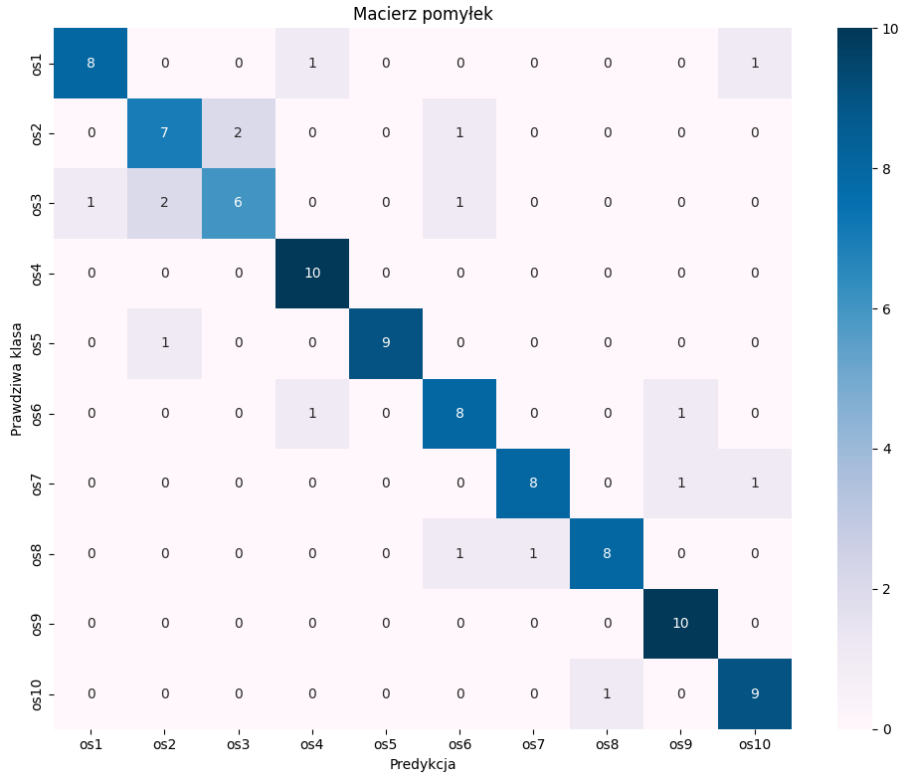
System identyfikacji mówców został poddany ewaluacji na zbiorze testowym składającym się ze 100 próbek głosu (tak jak opisano w sekcji wyżej). Uzyskano skuteczność klasyfikacji na poziomie **83%**. Szczegółowe wyniki klasyfikacji dla każdej klasy (osoby) przedstawiono w poniższej tabeli.

Klasa	**Precision**	**Recall**	**F1-score**	**Support**
osoba_1	0.8889	0.8000	0.8421	10
osoba_2	0.7000	0.7000	0.7000	10
osoba_3	0.7500	0.6000	0.6667	10
osoba_4	0.8333	1.0000	0.9091	10
osoba_5	1.0000	0.9000	0.9474	10
osoba_6	0.7273	0.8000	0.7619	10
osoba_7	0.8889	0.8000	0.8421	10
osoba_8	0.8889	0.8000	0.8421	10
osoba_9	0.8333	1.0000	0.9091	10
osoba_10	0.8182	0.9000	0.8571	10
Accuracy			0.8300	100
Macro avg	0.8329	0.8300	0.8278	100
Weighted avg	0.8329	0.8300	0.8278	100

Tabela 1: Metryki klasyfikacji dla każdej klasy.

Macierz pomyłek

Macierz pomyłek przedstawia liczbę błędnych i poprawnych klasyfikacji dla każdej klasy. Wiersze odpowiadają klasom rzeczywistym, kolumny – klasom przewidzianym przez model.



Rysunek 2: Macierz pomyłek dla zbioru testowego.

Prawdziwy mówca	O1	O2	O3	O4	O5	O6	O7	O8	O9	O10
O2	0.2	**93.2**	2.1	0.3	0.3	2.0	0.3	0.4	1.1	0.2
O4	1.4	1.8	0.7	**62.3**	1.4	0.6	1.8	26.4	2.1	1.4
O5	0.9	21.3	12.2	1.3	**52.6**	1.1	2.0	5.9	1.7	0.9
O6	9.6	4.8	2.7	6.0	3.0	**36.9**	4.0	9.0	22.0	1.9
O9	0.3	0.4	0.4	0.5	0.7	0.8	0.8	9.8	**86.0**	0.4

Tabela 2: Porównanie pewności modelu (w %) przy rozpoznawaniu słowa *masło* pochodzącego od 5 różnych mówców względem wszystkich osób z bazy. W pierwszej kolumnie podane jest od którego mówcy pochodzi dana próbka. Dla każdej próbki pogrubione zostało największe prawdopodobieństwo.

Uzyskane wyniki prezentują się całkiem dobrze – model okazuje się być stosunkowo skuteczny, szczególnie biorąc pod uwagę fakt, że proces wyznaczania współczynników MFCC został zaimplementowany ręcznie na podstawie wzorów, a nie z wykorzystaniem gotowych, zoptymalizowanych bibliotek. Można przypuszczać, że przy jeszcze większym zbiorze treningowym model miałby szansę nauczyć się rozpoznawać mówców jeszcze trochę skuteczniej. Warto także pamiętać, że jakość predykcji może być zależna od jakości samego nagrania – obecność szumów, zakłóceń czy zmiennej głośności mogą wpływać na końcowy wynik klasyfikacji. Poza tym żadna osoba nie wypowiada danego słowa w sposób identyczny za każdym razem, co dodatkowo komplikuje zadanie rozpoznawania mówcy. Jak pokazuje przykład z tabeli 2, mimo że finalnie model prawidłowo klasyfikuje nagrania dla wszystkich pięciu mówców, poziom pewności predykcji jest zróżnicowany – dla niektórych osób wynosi zaledwie nieco ponad 35%, a dla innych przekracza 90%. Ogółem jednak, osiągnięte wyniki można uznać za zadowalające, a skuteczność predykcji za obiecującą.

6 Podsumowanie

Projekt skutecznie realizuje postawione założenia, umożliwiając efektywną identyfikację mówców na podstawie dostarczonego zbioru nagrań. Mimo że rozwiązanie nie jest pozbawione ograniczeń i nie osiąga pełnej perfekcji, jego działanie można uznać za w pełni satysfakcjonujące – szczególnie biorąc pod uwagę jego relatywnie prostą strukturę oraz zastosowany algorytm klasyfikacji. System wykazuje stabilność działania i rozsądny poziom trafności.

7 Źródła

- dr inż. J. Rafałko, *Wykłady i materiały z Analizy i Przetwarzania dźwięku*, Politechnika Warszawska, 2025
- `https://medium.com/@derutycsl/intuitive-understanding-of-mfccs-836d36a1f779`
- OpenAI, ChatGPT (GPT-4), `https://openai.com/chatgpt`