

Metode de recunoaștere a vorbitorului

Cristina Torodoc

UNIVERSITATEA TEHNICĂ DIN CLUJ-NAPOCA

FACULTATEA DE ELECTRONICĂ, TELECOMUNICAȚII ȘI TEHNOLOGIA INFORMAȚIEI

Acest proiect își propune evidențierea unor aspecte teoretice ce țin de o caracteristică foarte importantă a sistemelor om-mașină, și anume, recunoașterea vorbitorului. De asemenea, încearcă o implementare practică minimalistă a unui astfel de sistem și oferind o imagine de ansamblu asupra principiilor de la care se poate porni pentru dezvoltări mai avansate în domeniu.

1. INTRODUCERE

Recunoașterea vorbitorului este un proces care se ocupă cu recunoașterea în mod automat a persoanelor care vorbesc pe baza informațiilor individuale care pot fi extrase din semnalul vocal. Această tehnică face posibilă o multitudine de aplicații ce verifică identitatea și oferă accesul la anumite servicii, printre care se numără apelarea vocală, servicii bancare prin telefon, accesarea unor anumite baze de date, poșta vocală, control de securitate pentru informații confidențiale, etc. [1]

Acest proces este în strânsă legătură cu sistemele de recunoaștere automată a vorbirii (eng. automatic speech recognition). Ele se pot clasifica în 2 grupe: dependente de vorbitor și independente de vorbitor. Cele dependente sunt antrenate cu un singur vorbitor și recunoașterea este făcută numai pentru acesta.

Există câteva probleme în recunoașterea vorbirii care nu au fost încă descoperite. Cu toate acestea, s-au identificat în ultimele decade anumite aspecte dintre care o parte rămân nerezolvate. Principalele probleme în sistemele de recunoaștere automată a vorbirii sunt:

Determinarea granițelor dintre cuvinte;

- Variația accentelor;
- Vocabularul foarte extins;
- Variațiile existente în acustica încăperii;
- Variațiile temporale (diferiți vorbitori vorbesc la viteze diferite).

Recunoașterea vorbitorului poate fi de două feluri: identificare și verificare.

Identificarea vorbitorului determină dacă subiectul face parte dintr-un grup cunoscut de vorbitori. Avem de-a face, în acest caz, cu o comparație de tip unu la mai mulți (eng. one-to-many).

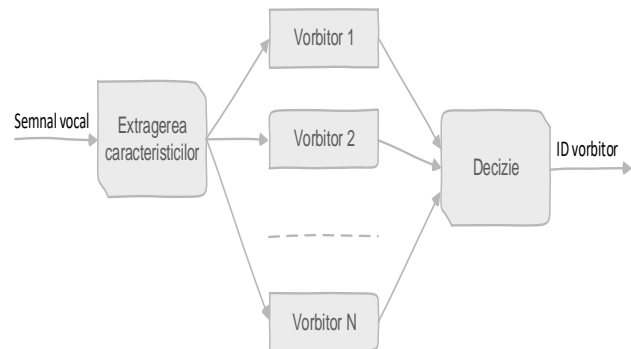


Figura 1: Sistem ce realizează identificarea vorbitorului

Verificarea vorbitorului determină dacă apelantul este cine susține că este. Aceasta este o comparație de tip unu la unu. [2]

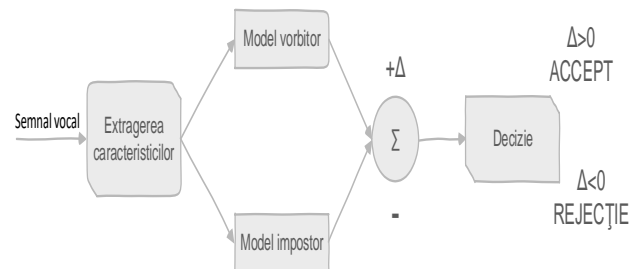


Figura 2: Sistem de verificare a vorbitorului

2. EXTRAGEREA CARACTERISTICILOR

Principalul scop al acestui stadiu este simplificarea recunoașterii prin sumarizarea unei mari cantități de informații vocale fără a-i pierde proprietățile acustice. Reprezentarea schematică a nivelului de extragere a caracteristicilor este prezentată în figura 3.

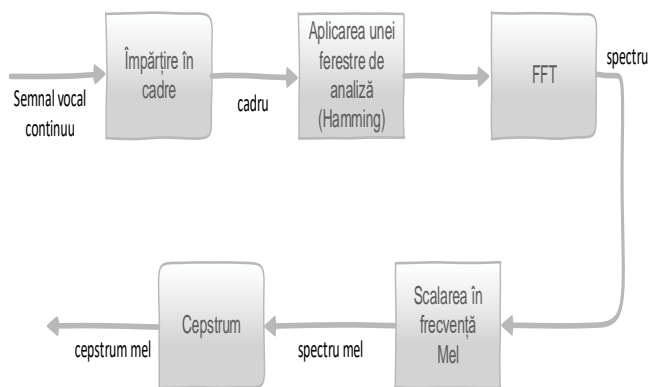


Figura 3: Extragerea caracteristicilor-pași

2.1. Împărțirea în cadre

S-a arătat că semnalul caracteristicile semnalului vocal rămân staționare într-o perioadă dintr-un interval de timp suficient de mic (cvasi-staționaritate). De aceea, semnalul vocal este procesat în intervale de timp mici, fiind împărțit în bucăți cuprinse între 30 și 100 de milisecunde. Fiecare cadru se suprapune cadrului anterior cu o dimensiune predefinită. Scopul suprapunerii este de a netezi trecerea de la cadru la cadru.

2.1.1. Aplicarea unei ferestre de analiză (eng. windowing)

Următorul pas este împărțirea tuturor cadrelor în ferestre de analiză, cu scopul de a elimina discontinuitățile de la marginile cadrelor. Dacă funcția care realizează acest proces se definește ca $w(n)$, $0 < n < N - 1$ unde N este numărul de eșantioane din fiecare cadru, semnalul rezultat va

fi $y(n) = x(n)w(n)$. Cel mai des se utilizează fereastra Hamming:

$$w(n) = 0.4 - 0.46 \cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1 \quad (1)$$

2.2. Transformata Fourier Rapidă

Urmează aplicarea Transformatei Fourier Rapide pentru fiecare cadru. Această transformată este o metodă rapidă a Transformatei Fourier Discrete și schimbă domeniul de analiză de la timp la frecvență.

2.3. Scalarea în frecvență Mel (eng. Mel Frequency Warping)

Auzul uman nu are o caracteristică liniară ci se aproximează cu una logaritmă. Aceasta conduce la ideea că urechea se comportă ca un filtru. Filtrele distanțate liniar la frecvențe joase și logaritmice la frecvențe înalte au fost utilizate pentru a surprinde caracteristicile fonetice importante ale vorbirii. Acest lucru este exprimat în scala mel-frecvență. Relația dintre frecvența în Hz și frecvența în scala Mel este dată de:

$$m = 1125 \ln\left(1 + \frac{f}{700}\right) \quad (2)$$

$$f = 700 \left(e^{\frac{m}{1125}} - 1\right) \quad (3)$$

2.4. Cepstrumul

Modelul sursă-filtru (MSF) de modelare a semnalului vocal pornește de la o simplificare extremă a tipurilor de semnale vocale și face separarea lor strict pe baza sonorității (sonor-nesonor). Sonoritatea se referă la prezența sau absența periodicității din forma de undă, periodicitatea fiind un rezultat al utilizării corzilor vocale în fonație. Astfel că, transpus în noțiuni lingvistice se referă la discriminarea vocale - consoane. Totodată consideră totalitatea organelor fonatoare (cu excepția plămânilor și a corzilor vocale) ca fiind un simplu filtru ce modulează sursa de aer, fie ea periodică, în cazul vocalelor, sau aperiodică, în cazul consoanelor.

Ca urmare, MSF va modela sursa pentru semnalele sonore ca fiind un tren de impulsuri distanțate cu perioada fundamentală (T_0), iar pentru semnalele nesonore, sursa va fi un zgomot alb gaussian. În funcție de semnalul vocal dorit la ieșirea modelului, filtrul dat de tractul vocal va fi implementat folosind coeficienții estimați pentru acesta.

Pentru a putea utiliza MSF este nevoie ca informațiile legate atât de sursă, cât și de filtru să poată fi extrase din semnalul vocal. Însă, conform teoriei semnalelor, ieșirea unui sistem va fi dată de convoluția dintre intrare $s(n)$ și funcția de transfer a sistemului $h(n)$:

$$y[n] = x[n] \otimes h[n] \quad (4)$$

În cazul semnalului vocal, sursa este dată de fluxul de aer expirat și modulată (sau nu) de vibrația corzilor vocale, iar filtrul sau sistemul este reprezentat de tractul vocal.

Pentru a putea separa sursa de filtru, este nevoie de o operație matematică homomorfică ce va transforma operația de convoluție (neliniară) într-o operație liniară, cum ar fi adunarea. O astfel de transformare sau filtrare este și cepstrumul.

Cepstrumul este definit ca fiind transformata Fourier inversă a logaritmului modulului transformatei Fourier directe:

$$c[n] = \mathcal{F}^{-1}\{\log |\mathcal{F}(x[n])|\} \quad (5)$$

Domeniul de definiție al cepstrumului nu va fi însă timpul, deoarece din transformata Fourier s-a păstrat doar spectrul de amplitudini, informația de fază fiind eliminată prin aplicarea modulului. Acest nou domeniu este denumit quefrequency pentru a face diferențierea dintre domeniul de frecvență și acesta.

Urmărind operațiile matematice aplicate asupra semnalului de intrare, se poate observa că transformata Fourier va transforma convoluția în înmulțire, iar logaritmul va transforma

operația de înmulțire în adunare. Ca urmare, cepstrumul va fi alcătuit din suma reprezentării cepstrale a sursei și reprezentarea cepstrală a tractului vocal.

Rezultatul acestor două procesări este cunoscut sub numele de coeficienți Mel-cepstrali (eng. Mel Frequency Cepstral Coefficients - MFCC). Aceștia reprezintă energia medie din benzile de frecvență date un banc de filtre de lungime N, egal distanțate pe scala Mel.

Primul filtru va începe la primul punct, îți va atinge vârful într-al doilea și apoi se va întoarce în zero în cel de-a treilea punct. Al doilea filtru va începe cu al doilea punct, va atinge maximum în punctul 3 și va trece în zero în punctul 4 ș.a.m.d. Formula pentru acest calcul este dată de ecuația următoare:

$$\begin{cases} 0, & k < f(m-1) \\ \frac{k - f(m-1)}{f(m) - f(m-1)}, & f(m-1) \leq k \leq f(m) \\ \frac{f(m+1) - k}{f(m+1) - f(m)}, & f(m) \leq k \leq f(m+1) \\ 0, & k > f(m+1) \end{cases} \quad (6)$$

Unde m este nr. de filtre pe care îl dorim, iar funcția f este lista de m+2 frecvențe pe scara Mel. [3]

2.5. LPC (coeficienții de predicție liniară)

Coeficienții LPC reprezintă o altă metodă interesantă de extragere a caracteristicilor. Este o metodă autoregresivă, eșantionul curent fiind prezis pe baza a p eșantioane anterioare. Această metodă se bazează pe modelul sursă-filtru.

Fiecare eșantion are următoarea formă:

$$x(n) = -\sum_{k=1}^p a^k x(n-k) + u(n) \quad (7)$$

,fiecare eșantion la momentul n depinzând de p eșantioane anterioare, la care se adaugă zgomot gaussian u(n).

O metodă de estimare a coeficienților LPC se realizează prin utilizarea ecuațiilor Yule-Walker, definite în ecuația următoare:

$$\sum_{k=1}^p a^k R(l-k) = -R(l) \quad (8)$$

Se folosește funcția de autocorelație, dată astfel:

$$R(l) = \sum_{n=1}^N x(n) x(n-l) \quad (9)$$

Pentru implementarea în Python se poate folosi o metodă numită Box-Jenkins care scalează corelația la fiecare întârziere cu câte un eșantion obținându-se autocorelația 1 când întârzierea este zero.

Soluțiile ecuațiilor Yule-Walker sunt date de coeficienții, sub forma:

$$\alpha = -R^{-1}r \quad (10)$$

3.POTRIVIREA CARACTERISTICILOR (eng. feature matching)

Cei mai populari algoritmi de potrivire a caracteristicilor sunt algoritmul bazat pe Modele Markov ascunse, cel de deformare dinamică a timpului (Dynamic Time Warping) și algoritmul de Cuantizare Vectorială.

3.1. Cuantizarea vectorială

Cuantizarea Vectorială [4] este un proces de mapare a vectorilor dintr-un spațiu vectorial extins într-un număr finit de regiuni din acel spațiu. Fiecare regiune se numește cluster și poate fi reprezentată prin centrul ei numit „cuvânt de cod” (eng. codeword). O colecție de astfel de cuvinte de cod poartă denumirea de codebook (carte de coduri).

O colecție de cuvinte de cod formează un codebook unic specific fiecărui vorbitor. Pentru identificarea unui vorbitor, se calculează distanța (sau distorsiunea) dintre caracteristicile vorbitorului din toate codebook-urile antrenate. Codebook-ul asociat care are distorsiunea minimă față de caracteristicile vorbitorului este cel corect. [5]

Algoritmul, cunoscut ca și LBG [Linde, Buzo și Gray], este utilizat pentru a grupa un set de L vectori de antrenare într-un set de M vectori asociați codebook-urilor. Implementarea este una recursivă, urmând următoarea procedură:

- **Pasul 1:** Se proiectează un codebook cu un vector care va reprezenta centroidul întregului set de vectori de antrenare;
- **Pasul 2:** Se dublează dimensiunea codebook-ului prin împărțirea fiecărui codebook curent y_n după regula:
 - $y_n^+ = y_n(1 + \varepsilon); \quad (11)$
 - $y_n^- = y_n(1 - \varepsilon). \quad (12)$

,unde n variază de la 1 la dimensiunea curentă a codebook-ului, și ε este un parametru de împărțire;

- **Pasul 3:** Căutarea vecinului cel mai apropiat: pentru fiecare vector de antrenare, se găsește cuvântul de cod în codebook-ul care este cel mai apropiat și se asignează acel vector celei corespunzătoare;
- **Pasul 4:** Actualizarea centroidului: se actualizează cuvântul de cod din fiecare celulă utilizând centroidul vectorilor de antrenare asigurați acelei celule;
- **Pasul 5:** Prima iterație: repeat pașii 3 și 4 până când distorsiunea iterației curente scade sub o fracțiune din distorsiunea asociată iterației anterioare.
- **Pasul 6:** Iterația 2: repeat pașii 2, 3 și 4 până când se obține un codebook de dimensiune M. [6]

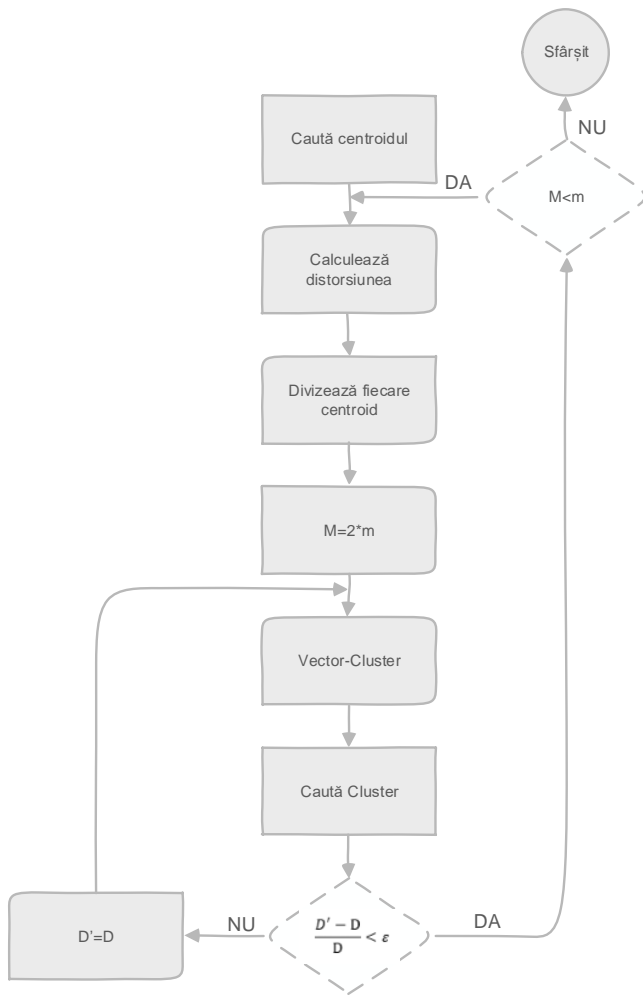


Figura 4: Schema bloc a algoritmului LBG

3.2. Metoda K-means

Tehnica celor k-medii, cunoscută și sub denumirea K-means este unul dintre cei mai simpli algoritmi de învățare nesupervizată care rezolvă bine problemele de clustering. Reprezintă o metodă de clasterizare a vectorilor de antrenare pentru a obține vectori de caracteristici.

Procedul urmează o cale simplă și ușoară pentru a clasifica setul datelor de intrare într-un număr de K grupuri. Varianta clasică a acestui algoritm presupune ca numărul K de clusteri să fie cunoscut dinainte.

Obiectivul este acela de a minimiza varianza totală intra-cluster:

$$V = \sum_{i=1}^k \sum_{j \in S_i} |x_j - \mu_i|^2 \quad (13)$$

,unde există k clusteri S_i , $i=1,2,\dots,k$ și μ_i este centroidul sau punctul mediu al tuturor punctelor, $x_j \in S_i$.

Ideea de bază este aceea de a defini K centre de greutate, numite centroizi, câte unul pentru fiecare cluster. Aceste centre de greutate trebuie fixate rațional, deoarece locații diferite pot conduce la rezultate diferite. Alegerea cea mai bună este să le fixăm cât mai depărtate unele de altele. Următorul pas este luarea pe rând a fiecărui element din setul de date de intrare și asocierea acestuia celui mai apropiat centroid. Prima etapă a grupării se termină atunci când nu mai există elemente negrupate. În acest moment e necesar să fie calculate noii K centroizi pentru clusterii determinați în pasul anterior. Procesul continuă până în momentul în care pozițiile noilor centroizi nu se mai modifică semnificativ.

3.3. Distanța euclidiană

În faza de recunoaștere a vorbitorului, vocea unui vorbitor necunoscut este reprezentată de o secvență de vectori de caracteristici $\{x_1, x_2, \dots, x_i\}$; i apoi comparată cu codebook-ul din baza de date. Pentru a identifica vorbitorul necunoscut se poate măsura distanța distorsiunii dintre 2 seturi de vectori prin minimizarea distanței Euclidiene. Aceasta reprezintă distanța obișnuită între 2 puncte, având următoarea relație:

$$\begin{aligned} & \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \dots + (p_n - q_n)^2} \\ &= \sqrt{\sum_{i=1}^n (p_i - q_i)^2} \end{aligned} \quad (14)$$

Vorbitorul cu cea mai mică distorsiune este ales ca fiind persoana necunoscută. [7]

3.4. Metoda SVM (Support Vector Machine)

SVM este un algoritm linear de Machine Learning utilizat în mod frecvent pentru rezolvarea problemelor de clasificare. Ideea de bază constă în găsirea unui hiperplan capabil să împartă optim datele din setul de antrenare. Deși formulele matematice care stau la baza lui sunt destul de complexe, acest algoritm are capacități în lucrul cu vectori de dimensiune foarte mare și reușește performanțe deosebite în cazul învățării datelor neliniar separabile.

Clasificarea este o tehnică de învățare supervizată care utilizează o mulțime de secvențe etichetate pentru antrenare și încearcă să găsească regulile după care sunt împărțite cel mai bine mulțimile de antrenare, în timp ce clustering-ul este o metodă nesupervizată de grupare a exemplurilor în clase, obiectele din aceeași clasă obținându-se similare iar cele din clase diferite dimpotrivă.

Algoritmul de bază este unul binar, extinderea pe mai multe clase putându-se face sub forma o clasă versus celelalte clase,

clasificare pe perechi de clase, codificarea ieșirii pe baza corecției erorii și funcțiile obiectiv pe mai multe clase.

4. ANTRENAREA CARACTERISTICILOR

Modelele asociate vorbitorilor și cele legate de clasificări nu sunt dependente numai de caracteristici cât și de task-urile care urmează a fi implementate. Mulți factori, cum ar fi tipul semnalului vocal, ușurința antrenării, cerințele de stocare și cele computaționale pot influența drastic rezultatele care urmează a fi obținute.

În faza de antrenare, folosind un algoritm de grupare (eng. clustering), se generează câte un codebook bazat pe cuantizarea vectorială caracteristic fiecărui vorbitor în parte (pentru unul din programele ce urmează a fi testate).

6. IMPLEMENTARE

În această fază, am testat și preluat secvențe de cod din 2 programe open source disponibile în internet.

6.1. [Speaker Recognition-master](#)

Primul program realizează identificarea vorbitorului astfel încât să se preteze la secvențe audio scurte conținând câte un cuvânt de test și folosește MFCC și, respectiv, LPC pentru extragerea de caracteristici urmate de un algoritm bazat pe cuantizare vectorială în faza de antrenare.

Programul este structurat având următoarele module:

- test
- train
- LBG
- LPC
- mel_coefficient

Primul pas în implementarea cu MFCC este procesarea semnalului vocal asociat fișierelor de intrare prin transformarea acestuia într-o secvență de vectori acustici dați prin coeficienții MFCC folosind, în ordine, pașii descriși mai sus. Astfel, citim fișierele de intrare. Împărțim semnalul vocal (un vector) în cadre cu suprapunere. Rezultatul este o matrice în care fiecare coloană este un cadru de N eșantioane din semnalul de vorbire original. Aplicăm pașii de „ferestruire” și „FFT” pentru a transforma semnalul în domeniul frecvență. Acest proces este utilizat în multe aplicații diferite și este denumit în literatură ca Windowed Fourier Transform (WFT) sau Short-Time Fourier Transform (STFT). Rezultatul este adesea numit spectru sau periodogramă. Ultimul pas în procesare este convertirea spectrului de putere în coeficienți cepstrali pe scara mel.

Rezultatul ultimei secțiuni este că transformăm semnalele de vorbire în vectori din spațiul acustic. Apoi vom aplica tehnica de recunoaștere a modelelor bazată pe VQ pentru a construi modele de referință a vorbitorilor din acei vectori în faza de antrenare și apoi putem identifica orice secvență de vectori acustici provenită de la vorbitori necunoscuți.

În continuare, construim o funcție care să calculeze într-un proces iterativ distanțele euclidiene perechi între cuvinte de cod și vectori de antrenare.

Analog, pe cealaltă ramură se obțin coeficienții LPC, se obțin codebook-uri aferente și se aplică algoritmul LBG și restul prelucrărilor vectorilor respectivi.

6.2. [Speaker Recognition-main](#)

Cel de-al doilea program utilizează funcții din librării deja implementate în python pentru a ilustra utilizarea SVM în identificarea vorbitorilor.

Programul este structurat având următoarele module:

- preprocess.py → extrage caracteristicile pentru fiecare clip din setul de date și pune rezultatele într-un fișier CSV (principala caracteristică este reprezentată tot de către coeficienții MFCC descriși mai sus; pentru obținerea lor se utilizează librosa).
- train.py → se utilizează mai multe modele pentru prezicerea claselor (le-am lăsat doar pe cele bazate pe vectori suport, pe restul le-am comentat). Se afișează acuratețea fiecărui algoritm pe baza prezicerilor efectuate.
- test.py → Se testează alte secvențe audio pentru a se vedea dacă se potrivesc vorbitorii.

7. SIMULARE ȘI EVALUARE

7.1. [Speaker-Recognition-master](#)

Pentru testarea caracteristicilor implementate am folosit niște secvențe audio de test open-source disponibile internet. Acestea sunt numerotate după ID-ul asociat fiecărui vorbitor și au fost înregistrate folosind formatul Microsoft WAV. [1]

- primul set de date:

Am folosit setul de date furnizat de autorul programului.

Fiecare fișier de sunet conține o singură comandă vocală reprezentând cuvântul ‘zero’ înregistrat în limba engleză de către vorbitori de gen feminin.

În continuare, se urmărește antrenarea unui model asociat fiecărui vorbitor de la s1 la s8 folosind fișierul de sunet corespunzător. După această fază, sistemul va avea cunoștințele necesare despre caracteristicile vocale specific fiecărui vorbitor cunoscut.

ACURATEȚE

MFCC

33.33333333333333 %

LPC	50.0 %
------------	--------

- al doilea set de date:


Fiecare fișier de sunet conține o singură comandă vocală reprezentând cuvântul ‘one’ înregistrat în limba engleză de către 6 vorbitori de gen masculin.

ACURATEȚE	
MFCC	66.66666666666666 %
LPC	83.33333333333334 %

Rezultate din consolă:


```
In [58]: runtime('C:/Users/CRIS/Desktop/pr_PSV/Spokeer-Recognition-master/Spokeer-Recognition-master')
Reloaded modules: LBG, mel_coefficients, LPC, train
Now speaker 1 features are being trained
Now speaker 2 features are being trained
Now speaker 3 features are being trained
Now speaker 4 features are being trained
Now speaker 5 features are being trained
Now speaker 6 features are being trained
Training complete
Now speaker 1 features are being tested
Speaker 1 in test matches with speaker 1 in train for training with MFCC
Speaker 1 in test matches with speaker 1 in train for training with LPC
Now speaker 2 features are being tested
Speaker 2 in test matches with speaker 2 in train for training with MFCC
Speaker 2 in test matches with speaker 2 in train for training with LPC
Now speaker 3 features are being tested
Speaker 3 in test matches with speaker 3 in train for training with MFCC
Speaker 3 in test matches with speaker 3 in train for training with LPC
Now speaker 4 features are being tested
Speaker 4 in test matches with speaker 3 in train for training with MFCC
Speaker 4 in test matches with speaker 4 in train for training with LPC
Now speaker 5 features are being tested
Speaker 5 in test matches with speaker 5 in train for training with MFCC
Speaker 5 in test matches with speaker 5 in train for training with LPC
Now speaker 6 features are being tested
Speaker 6 in test matches with speaker 5 in train for training with MFCC
Speaker 6 in test matches with speaker 5 in train for training with LPC
Accuracy of result for training with MFCC is 66.66666666666666 %
Accuracy of result for training with LPC is 83.33333333333334 %
```

Dacă se modifică numărul de filtre pentru coeficienții MFCC, și ordinul LPC, se observă că se pot obține performanțe mai bune pentru metoda cu MFCC și, respectiv, LPC:

-  nfilbank = 15, orderLPC = 12

ACURATEȚE	
MFCC	66.66666666666666 %
LPC	66.66666666666666 %

```
In [59]: runtime('C:/Users/CRIS/Desktop/pr_PSV/Spokeer-Recognition-master/Spokeer-Recognition-master')
Reloaded modules: LBG, mel_coefficients, LPC, train
Now speaker 1 features are being trained
Now speaker 2 features are being trained
Now speaker 3 features are being trained
Now speaker 4 features are being trained
Now speaker 5 features are being trained
Now speaker 6 features are being trained
Training complete
Now speaker 1 features are being tested
Speaker 1 in test matches with speaker 1 in train for training with MFCC
Speaker 1 in test matches with speaker 1 in train for training with LPC
Now speaker 2 features are being tested
Speaker 2 in test matches with speaker 2 in train for training with MFCC
Speaker 2 in test matches with speaker 1 in train for training with LPC
Now speaker 3 features are being tested
Speaker 3 in test matches with speaker 3 in train for training with MFCC
Speaker 3 in test matches with speaker 3 in train for training with LPC
Now speaker 4 features are being tested
Speaker 4 in test matches with speaker 3 in train for training with MFCC
Speaker 4 in test matches with speaker 4 in train for training with LPC
Now speaker 5 features are being tested
Speaker 5 in test matches with speaker 5 in train for training with MFCC
Speaker 5 in test matches with speaker 5 in train for training with LPC
Now speaker 6 features are being tested
Speaker 6 in test matches with speaker 5 in train for training with MFCC
Speaker 6 in test matches with speaker 5 in train for training with LPC
Accuracy of result for training with MFCC is 66.66666666666666 %
Accuracy of result for training with LPC is 66.66666666666666 %
```

-  nfilbank = 17, orderLPC = 17

ACURATEȚE	
MFCC	66.66666666666666 %
LPC	100 %

ACURATEȚE	
MFCC	66.66666666666666 %
LPC	83.33333333333334 %

Se observă că dacă creștem prea mult ordinul LPC, se obțin performanțe mai proaste.

- al treilea set de date:

Păstrăm cele 6 fișiere pentru antrenare, și folosim pentru test fișiere cu vocile aceluiași vorbitori, dar conținând înregistrat cuvântul ‘five’.

ACURATEȚE	
MFCC	50.0 %
LPC	50.0 %

Pentru primul și al doilea set de date am folosit drept sursă fișiere provenite din [Free Spoken Digit Dataset](#) [8].

- al 4-lea set de date:

Pentru al 4-lea set de date am folosit drept sursă fișiere provenite din [URDU-DATASET](#), extrăgând informațiile de la 6 vorbitori neutri, dintre care sunt 5 vorbitori de gen

masculin și unul feminin. Rezultatele obținute sunt mai slabe din punct de vedere calitativ, astfel:

ACURATEȚE	
MFCC	16.666666666666664 %
LPC	33.33333333333333 %

- al 5-lea set de date:

Am folosit propoziții de la 11 vorbitori diferiți, dar cuvintele din secvențele de test sunt diferite de cele din secvențele de antrenare, astfel încât acuratețea este foarte slabă:

ACURATEȚE	
MFCC	9.090909090909092 %
LPC	18.181818181818183 %

Pentru acest set de date, și dacă se modifică numărul de filtre pentru coeficienții MFCC, și ordinul LPC, se observă că nu se pot obține performanțe mai bune, astfel încât aplicația nu se pretează pentru volume mari de date.

7.2. Speaker-Recognition-main

Folosind date din al 5-lea set de date de mai sus la antrenarea celui de al 2-lea program, se estimează o acuratețe de peste 90% a algoritmilor SVM, astfel:

ACURATEȚE	
SVM	94.5945945945946 %
SVM OVR ("ONE VERSUS REST")	94.5945945945946 %
SVM OVO ("ONE VERSUS ONE")	91.8918918918919 %

```
In [49]: runfile('C:/Users/CRIS/Desktop/Speaker_Recognition-main/train.py', wdir='C:/Users/CRIS/Desktop/Speaker_Recognition-main/Speaker_Recognition-main')
(84, 26) (84,) (37,)
SVM Classifier
accuracy: 94.5945945945946 %
SVM OVR
accuracy: 94.5945945945946 %
SVM OVO
accuracy: 91.8918918918919 %
```

Pentru a putea obține aceste rezultate, s-au folosit câte 10 fișiere de sunet diferite conținând mai multe propoziții pentru fiecare dintre cei 11 vorbitori pentru care s-a realizat antrenarea, prin urmare am oferit mult mai multe informații sistemului. Pe același principiu se pot testa și algoritmii care folosesc arbori pentru potrivirea de caracteristici (nu am

abordat această problemă în prezentul document și am comentat secvențele de cod corespunzătoare din programul menționat.

În faza de test se poate observa o acuratețe de 100% a algoritmului SVM, pentru datele de intrare menționate:

```
IPython console
Console 1/A
Speaker_Recognition-main/Speaker_Recognition-main')
Predicted Class for file 1_test.wav is: [1]
Predicted Class for file 0_test.wav is: [0]
Predicted Class for file 2_test.wav is: [2]
Predicted Class for file 3_test.wav is: [3]
Predicted Class for file 4_test.wav is: [4]
Predicted Class for file 5_test.wav is: [5]
Predicted Class for file 6_test.wav is: [6]
Predicted Class for file 7_test.wav is: [7]
Predicted Class for file 8_test.wav is: [8]
Predicted Class for file 9_test.wav is: [9]
Predicted Class for file 10_test.wav is: [10]
```

8. CONCLUZII

În stadiul actual există algoritmi avansați în computarea problemelor biometrice, incluziv a celor ce țin de autentificarea și identificarea vorbitorilor. Pentru a putea trece la un astfel de nivel însă este util a avea cunoștințele de bază pe care am încercat să le ating în soluția prezentată în conținutul acestui proiect și de a dobândi o abordare de jos în sus în acest mod.

REFERINȚE

- [1] „DSP Mini-Project: An Automatic Speaker Recognition System,”
http://www.ifp.uiuc.edu/~minhdo/teaching/speaker_recognition.
- [2] A.-a.-M. MD, M. Firoz, I. Aminul și T. Z. Syed , „Automatic Speaker Recognition system using Mel Frequency Cepstral Coefficients (MFCC) and Vector Quantization (VQ) approach,” în *International Conference on Electrical, Computer and Telecommunication Engineering*, Bangladesh, 2012.
- [3] . F. M. Haytham, „Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between,” 2016.
- [4] M. Darshan și G. Pravin, „Speaker Recognition Using MFCC and Vector Quantization Model,” Institute of Technology, Nirma University, Ahmedabad, 2011.
- [5] H. Perez-Meana, *Advances in Audio and Speech Signal Processing: Technologies and Applications*, Mexic: Idea Group Publishing, 2007.
- [6] R. B. Mtech și P. B. Mtech, „Real Time Speaker Recognition System using MFCC and Vector Quantization Technique,” *International Journal of Computer Applications (0975 – 8887)*, vol. 117, nr. 1, 2015.
- [7] P. Kashyap și R. Prasad, „Speech Recognition and Verification Using MFCC & VQ,” *International Journal of Emerging Science and Engineering (IJESE)*, vol. 1, 2013.
- [8] Z. Jackson, C. Souza, J. Flaks, Y. Pan, H. Nicolas și A. Thite, „Jakobovski/free-spoken-digit-dataset: v1.0.8”.
- [9] <https://scikit-learn.org/stable/modules/svm.html>

MENȚIUNI

Diagramele au fost realizate folosind utilitarul Microsoft Visio Professional 2013.