Universitatea ''POLITEHNICA'' din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Recunoaşterea genului vorbitorului pe baza vocii şi detecţia termenilor vorbiţi în discursuri multi-lingve (Speech-based gender detection and multilingual spoken term detection)

PROIECT DE DIPLOMĂ

prezentat ca cerință parțială pentru obținerea titlului de
Inginer în domeniul *Electronică și Telecomunicații*
programul de studii de licență *Rețele și Software de Telecomunicații*

Conducător științific

Ş.l. Dr.Ing. Horia CUCU

Absolvent ,

Ancuța ARDELEANU

ANUL 2014

Anexa 1

Universitatea "Politehnica" din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei
Departamentul * Telecomunicaţii

Aprobat Director de Departament *:

Prof. Dr. Ing .Silviu Ciochina

TEMA PROIECTULUI DE DIPLOMĂ
a studentei Ardeleanu Gh. Ancuta,grupa 442D

1. Titlul temei: Recunoaşterea genului vorbitorului pe baza vocii şi detecţia termenilor vorbiţi în discursuri multi-lingve(Speech-based gender detection and multilingual spoken term detection)

2. Contribuţia practică, originală a studentului va consta în (*în afara* părţii de documentare):

Identificarea termenilor vorbiţi:

- generarea şi clusterizarea coeficienţilor mffc ai query-urilor şi contentului;
- calculul costului de similitudine între ferestrele fiecărui query şi content pe baza distanţei euclidiene;
- căutarea căii optime pe baza algoritmului DTW(Dynamic time warping);

Detecţia genului vorbitorului:

- crearea unui model acustic cu înregistrări de voci masculine şi feminine;
- evaluarea modelului cu alţi vorbitori;
- realizarea unei aplicaţii client-server;

3. Proiectul se bazează pe cunoştinţe dobîndite în principal la următoarele 3-4 discipline: Tehnici de programare în Internet, Microcontrolere,Prelucrarea digitală semnalelor

4. Realizarea practică/ proiectul rămân în proprietatea: Laboratorul de Cercetare SpeeD, student Ancuţa Ardeleanu

5. Proprietatea intelectuală asupra proiectului aparţine: Laboratorul de Cercetare SpeeD, student Ancuţa Ardeleanu

6. Locul de desfăşurare a activităţii: UPB

7. Data eliberării temei: iunie 2013

CONDUCĂTOR LUCRARE: *Lect.Horia Cucu*

STUDENT: *Ancuţa Ardeleanu*

Declaraţie de onestitate academică

Prin prezenta declar că lucrarea cu titlul "*Algoritmi adaptivi cu convergenţă variabilă*", prezentată în cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei a Universităţii "Politehnica" din Bucureşti ca cerinţă parţială pentru obţinerea titlului de *Inginer/ Master* în domeniul *Electronică şi Telecomunicaţii* programul de studii *Tehnologii şi Sisteme de Telecomunicaţii (ETC – TST)* este scrisă de mine şi nu a mai fost prezentată niciodată la o facultate sau instituţie de învăţământ superior din ţară sau străinătate.

Declar că toate sursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referinţe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar şi în traducere proprie din altă limbă, sunt scrise între ghilimele şi fac referinţă la sursă. Reformularea în cuvinte proprii a textelor scrise de către alţi autori face referinţă la sursă. Înţeleg că plagiatul constituie infracţiune şi se sancţionează conform legilor în vigoare.

Declar că toate rezultatele simulărilor, experimentelor şi măsurătorilor pe care le prezint ca fiind făcute de mine, precum şi metodele prin care au fost obţinute, sunt reale şi provin din respectivele simulări, experimente şi măsurători. Înţeleg că falsificarea datelor şi rezultatelor constituie fraudă şi se sancţionează conform regulamentelor în vigoare.

Bucureşti, 1 iulie 2014 Absolvent Ancuţa ARDELEANU

_____

CUPRINS

Lista Figurilor

Lista cu Tabele

Lista acronime

ASR	Automatic Speech Recognition
ATWV	Actual Term Weighted Value
AVCSR	Audio-Video Continuous Speech Recognition
BIC	Bayesian Information Criterion
ChER	Character Error Rate
CLR	Cross Likelihood Ratio
CMU	Carnegie Mellon University
DER	Diarization Error Rate
DET	Detection Error Tradeoff
DCT	Discrete Cosinus Transform
DFT	Discrete Fourier Transform
DTW	Dynamic Time Warping
DTWSS	Dynamic Time Warping String Search
EDT	Event Dispatch Thread
EM	Expectation Maximization
FAP	False Alarme Probability
FFT	Transformata Fourier Discreta
FST	Finite-state Transducer
FSG	Finite State Grammar
GMM	Gaussian Mixture Model
HMM	Hidden Markov Model
HP	Hewlett Packard
HTK	Hidden Markov Toolkit
JSGF	Java Speech Grammar Format
LIUM	The Laboratoire d'Informatique de l'Université du Maine
LMG	Literal movement grammars
MAP	Maximum a posteriori
MERL	Laboratoarele Mitsubishi Electric Research Labs
MIT	Massachusetts Institute of Technology
MFCC	Mel-frequency cepstrum coefficients
ML	Maximum Likelihood
MP-	Miss Probability

MTWV	Maximum Term Weighted Value
NIST	National Institute of Standards and Technology
OOV	Out Of Vocabulary
PhER	Phone Error Rate
PLP	Perceptual Linear Prediction
PPL	Perplexity
RAV	Recunoaşterea Automată a Vorbirii
RTTM	Real-Time Trade Matching
SER	Sentence Error Rate
SNR	Signal to Noise Ratio
ST	Sausage Technique
STD	Spoken Term Detection
TCP	Transmission Control Protocol
TWV	Term Weighted Value
UBM	Universal Background Model
UCSC	University Of California, Santa Cruz
UML	Unified Modeling Language
UTF	Universal Character Set Transformation Format
WER	Word Error Rate
XML	Extensible Markup Language

CAPITOLUL 1

INTRODUCERE

1.1 Motivarea lucrării

Lucrarea oferă o privire de ansamblu asupra tehnologiilor în domeniul recunoaşterii vorbirii, accentul punându-se pe recunoaşterea independentă de vorbitor. Progresele recente în tehnologie au crescut asteptările utilizatorilor în legatură cu diversitatea aplicaţiilor de vorbire. Una dintre aceste schimbări principale o reprezintă nevoia de a susţine ca date de intrare înregistrările în mai multe limbi. Astăzi există o diversitate de informaţii text disponbile pe Internet în cadrul carora se pot efectua căutări, dar cele în format audio sau video reprezintă o sursă mai comodă de informare. Procesarea informaţiilor a devenit o activitate economică principală în lume, informaţiile rostite reprezentând sursa majoră a acestor informaţii.

Lucrarea este motivată de toţi aceşti factori şi a fost creată pentru a ilustra mai mult forme de recunoaştere bazate pe calităţile vocii, ce pot servi la diverse aplicaţii utile în lumea reală. Procesul de recunoaştere automată a vorbirii (ASR) se ocupă de maparea unui semnal acustic cu o secvenţă de cuvinte.

Scopul detecţiei de termeni vorbiţi este acela de a găsi un anumit termen într-o bază de înregistrări audio şi reprezintă o oportunitate de a găsi informaţii în arhivele de vorbire înregistrate.

Vocea umană nu furnizează doar semantica cuvintelor rostite, ci reprezintă un semnal cu o infinită informaţie, caracteristice dependente de vorbitor precum gen, emoţii, etnie, dialect,vârstă, identitate. În cadrul apelurilor telefonice de zi cu zi, ne bazăm pe aceste caracteristici ale vocii interlocutorului şi încercăm să ne adaptăm propriul stil de vorbire în concordanţă cu al său.

Recunoaşterea genului cu ajutorul vocii se bazează pe forma tractului vocal, accent, ritm, stilul intonaţiei şi este utilă în simplificarea sarcinilor în domeniul vorbirii şi îmbunătăţirea serviciurilor destinate clienţilor.

Cea de-a treia parte abordată în lucrare, detectarea cifrelor rostite, prezintă importanţa în diverse aplicaţii precum: apelarea telefonică vocală, sisteme automate bancare, intrarea automată a datelor, intrarea PIN-ului. Este o sarcină de recunoaştere cu un vocabular foarte redus şi fiind pentru limba română, limba ţintă a acestui subcapitol, avem puţine resurse text, audio sau video disponibile. Experimentele făcute în cadrul acestei abordări, variind numărul de senone şi de densităţi gaussiene ajută la determinarea modelului acustic optim pentru a obţine o rată de detecţie greşită cât mai mică.

1.2 Obiectivele lucrării

Având în vedere observaţiile făcute în motivarea lucrării, obiectivul principal al acestei lucrări este de a arăta diversele forme pe care le poate lua un sistem de recunoaştere a vorbirii. Sistemul de recunoaştere a cifrelor, o parte a lucrării, trebuie să fie capabil să recunoască cifrele rostite de un vorbitor şi să le afişeze în forma textuală, iar sistemul de recunoaştere a genului. Pentru a realiza aceste lucruri, au fost folosite următoarele obiective specifice:

a) Crearea modelelor (acustic, fonetic şi lingvistic) pentru sistem de recunoaştere automată a vorbirii în limba română;

b) Realizarea unor experimente folosind diverse modele de gen şi configuraţii ale procesului de decodare pentru a evalua cele mai bune rezultate;

c)Efectuarea unor experimente pentru a găsi un termen rostit într-o bază de clipuri audio și optimizarea procesului de căutare.

d) Dezvoltarea unor aplicaţii care să realizeze aceste detecţii de cifre, respectiv de gen şi returnează rezultatele.

Lucrarea este organizată în 5 capitolele, precum urmează:

Capitolul 1 prezintă introducerea şi conţine motivarea care stă la baza realizării sale. Începe cu prezentarea tehnologiilor din domeniul vorbirii, importanţa lor în diversitatea actuală a informaţiilor audio şi domeniile lor de aplicabilitate şi continuă cu obiectivele lucrării.

Capitolul 2 dezvoltă aspectele teoretice ale sistemelor de recunoaştere automată a vorbirii, a detecţiei de gen şi a detecţiei de termeni vorbiţi. Sunt descrise, de asemenea, arhitecturile sistemelor, algoritmii de căutare, uneletele folosite şi metodele de evaluare.

Capitolul 3 subliniază experimenţele făcute cu scopul de găsi modelul acustic optim pentru recunoaşterea de cifre şi cele realizate în cadrul competiţiei MediaEval 2013. Astfel, se începe cu construcţia unui sistem de recunoaştere a vorbirii continue dependent de vorbitor, se continuă cu un sistem independent de vorbitor, pentru ambele variindu-se parametrii (numărul de senone şi numărul de densitaţi gaussiene). Capitolul se concentrează apoi, pe algoritmii de căutare şi metodele utilizate în sistemul de căutare a termenilor vorbiţi, precum şi pe rezultatele obţinute prin varierea parametrilor α şi β.

În capitolul 4 sunt descrise cele două aplicaţii demonstrative împreună cu principiile lor de funcţionare. Aplicaţia de detecţie de cifre leagă modelul acustic optim obţinut în urma experimenţelor cu modelul

lingvistic şi dictionarul. Acest penultim capitol este încheiat cu o scurtă descriere a dezvoltărilor ulterioare ce pot fi efectuate asupra aplicaţiei de detecţie a genului.

Capitolul 5 este dedicat concluziilor şi contribuţiilor personale în realizarea lucrării.

CAPITOLUL 2

TEHNOLOGIA LIMBAJULUI VORBIT

2.1 Recunoașterea automată a vorbirii [ASR]

Recunoașterea automată a vorbirii reprezintă procesul de transcriere a vorbirii în text, adică transformarea semnalului acustic într-o secvență de cuvinte. Până de curând limbajul vorbit era o modalitate de interacțiune între subiecții umani, însă, o dată cu dezvoltatea microelectronicii, comunicarea prin voce a trecut la stadiul de conectare subiect uman-mașină de calcul, având în momentul de față o gamă largă de aplicabilitate. Cel mai important domeniu ar fi cel al interfețelor hands-free și eyes-free. Este util de asemenea în aplicațiile de dictare automată, în sistemele de dialog pentru call centere, sistemele de traducere speech-to-speech, sau când dorim transcrierea unui monolog al unui vorbitor anume (cazul transcrierii discuțiilor din timpul proceselor de judecată, editarea unor documente de către jurnaliști sau scriitori). Arhitectura generală a unui sistem de recunoaștere automată a vorbirii este prezentată în figura 2.1.

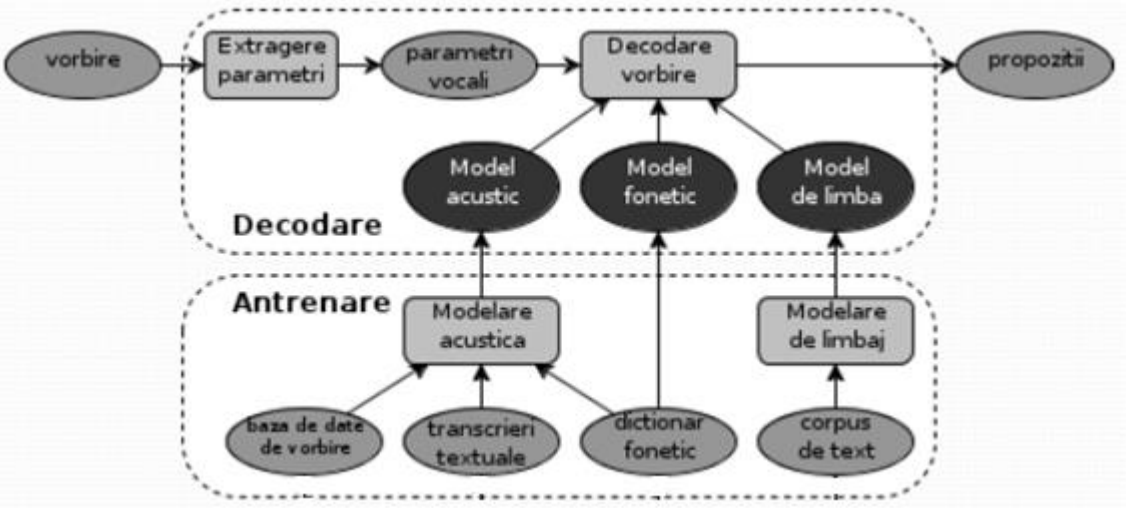


Figura 2.1 Arhitectura generală a unui sistem de recunoaștere automată a vorbirii

Modelul unui asemenea sistem statistic poate fi simplicat astfel, figura 2.2 :

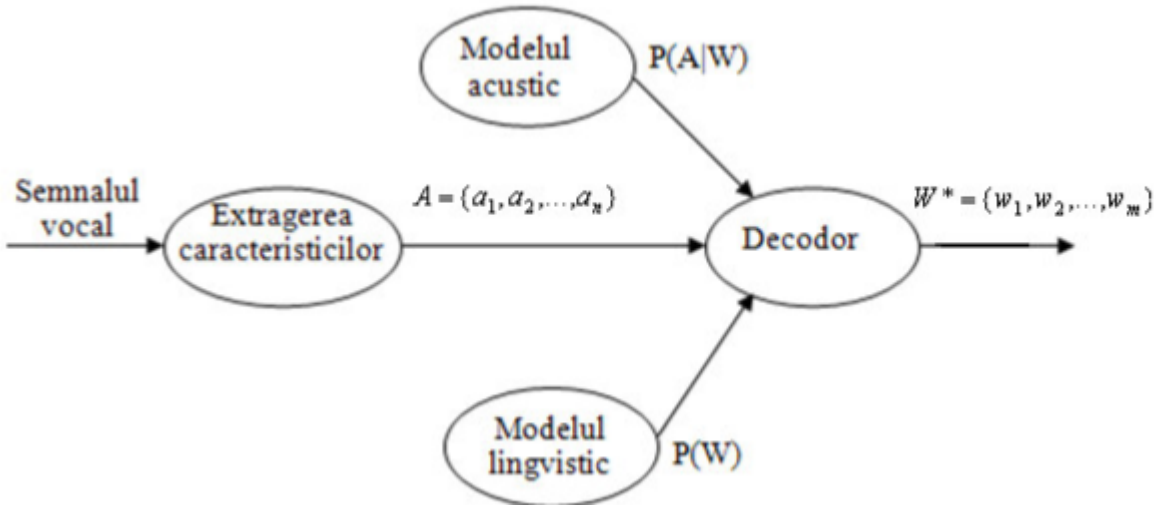


Figura 2.2 Arhitectura unui sistem de recunoaștere a vorbirii statistic

Având după prelucrarea și extragerea parametrilor semnalului vocal un set de vectori acustici $A=\{a_1,a_2,\ldots..a_n\}$, căutăm o secvență ce se potrivește cel mai bine $W^*=\{w_1,w_2,\ldots..w_m\}$:

$$W^* = \arg\max_W p(W|A) \qquad (2.1)$$

Și folosind regula lui Bayes, din relația 2.1. rezultă:

$$W^* = \arg\max_W \frac{p(A|W)p(W)}{p(A)} = \arg\max_W (p(A|W)p(W)) \qquad (2.2)$$

unde s-a ținut cont de faptul că probabilitatea mesajului vorbit p(A), este independentă de secvența de cuvinte W și poate fi ignorată, p(A|W) fiind probabilatea mesajului vorbit și P(W) probabilitatea de apariție a secvenței de cuvinte W. Problema găsirii unei secvențe de cuvinte pe baza mesajului vorbit a fost astfel împărțită în două sub-probleme mai simple:
-estimarea probabilității apriorice p(W), utilizând un model de limbă;
-estimarea probabilității p(A|W) cu ajutorul unui model acustic.

2.1.1. Resursele fonetice, acustice și lingvistice necesare

Precum se observă din schema arhitecturii sistemului de recunoaștere a vorbirii, pentru decodare avem nevoie de un model acustic, unul fonetic și unul de limbă, pe care trebuie să le pregătim în prealabil.

- *Modelul acustic*

Este construit pe baza setului de înregistrări audio și a transcrierilor textuale ale acestora.

- *Modelul fonetic*

Conține un model de pronunție pentru fiecare cuvânt, chiar dacă există mai multe pronunții, acestea sunt comasate.

- *Modelul de limbă*

Se bazează pe un corpus de text de dimensiuni mari.

1) Dicţionarul fonetic conține transcrierea cuvintelor din forma lor literară în forma fonetică (precum se pronunță o succesiune de foneme). Acesta face legătura între modelul acustic și cel de limbă și trebuie să conțină cuvintele ce sunt posibile într-o anumită sarcină de recunoaștere a vorbirii. Fiecare intrare în dicţionar conține atât cuvântul cât și forma lui fonetică și se poate realiza manual, automat sau semi-automat. În limba română, de foarte multe ori unei litere îi corespunde același fonem, însă există și excepții, când o literă se pronunță diferit în funcție de contextul în care apare.

Tabel 2.1 Clase de ambiguitate în fonetizarea cuvintelor limbii române

Litera	Fonemul	Exemple
e	e	c**e**r, r**e**st, r**e**c**e**
	e1	d**e**al, t**e**-am, c**e**ainic, gh**e**orghe
	i3e	**e**u, **e**l, **e**ram, **e**rai
o	o	gol,potrivit
	o1	voal,soare

2) Baza de date de vorbire constituie într-o mulțime de înregistrări salvate în format wav și se folosește pentru antrenarea modelului acustic, cuprinzând 3 părți:
-un set de clipuri audio;
-un set ce corespunde transcrierilor textelor rostite în clipuri audio, precum și delimitarea fonemelor;
-în formații suplimentare despre domeniul vorbirii, identitatea vorbitorului;

Ea este un element important de care depinde atât rata de eroare cât și viteza de recunoaștere. Pentru fiecare bază de date se păstrează informații legate de tipul de înregistrare, numărul de cuvinte, dimensiunea bazei de date, calitatea etc. Indiferent de algortimul de antrenare, cu cât baza de date este mai mare cu atât rezultatele obținute vor fi mai bune.

Înregistrarea acestor clipuri presupune parcurgerea unor pași precum: alegerea locului de înregistrare (laborator, studio), alegerea microfonului, stației de înregistrare, precum și alegerea textelor ce trebuiesc înregistrate. Acest ultim pas ține cont de mai multe criterii:
-frazele să fie corecte din punct de vedere lexical și gramatical;
-frazele să fie scurte (maxim 15-20 de cuvinte);
-frazele să fie conectate semantic pentru a-i fi mai ușor vorbitorului să înțeleagă sensul și să obțină o pronunție fluentă;

În cazul în care vorbitorul cunoaște aprioric textul ce va fi înregistrat, erorile de pronunție, bâlbele sau lipsa de fluență sunt mult mai rare.

3) Corpusul de text este folosit la antrenarea modelului lingvistic, iar achiziția sa este realizată în mod automat, prin folosirea unui sisteme de achiziție a textelor de pe Internet (pe principiul Web-as-Corpus).

Deoarece sistemele de recunoaștere a vorbirii transcriu vorbirea și scot la ieșire text simplu (fără simboluri speciale, litere mari etc.) textele trebuiesc normalizate și restaurate astfel:
-transformarea literelor mari în litere mici;
-transcrierea numerelor formate din cifre în numere din litere;
-înlăturarea caracterelor speciale și a semnelor de punctuație;
-scoaterea etichetelor HTML în cazul în care textele sunt de pe internet;

2.1.2 Principiile de bază ale modelării acustice [J01]

Sistemele de recunoaștere a vorbirii continue nu estimează probabilitatea mesajului, ci se bazează pe unități de vorbire mai mici (foneme) și calculează probabilitățile acestora. Astfel modelele pentru cuvinte stau la baza modelelor pentru secvențele de cuvinte și se bazează pe modelele pentru unitățile sub-lexicale. Ele sunt utilizate în final pentru a estima probabilitatea ca mesajul rostit de vorbitor să fie format dintr-o succesiune sau alta de cuvinte.

2.1.2.1 Parametrii acustici

Modelul acustic este implementat utilizând modelele Markov ascunse (HMM). Un astfel de HMM este un automat probabilistic cu stări finite, ce constă într-un set de stări conectate prin tranziții, dar înşiruirea stărilor este ascunsă. Aceste foneme legate formează modelele cuvintelor și apoi, secvențe de cuvinte ce sunt utilizate în estimarea lui p(A|W).

Definite HMM:
Un HMM [R01] este complet definit prin:
-spaţiul de stări: $Q = \{1, 2, 3, \ldots N\}$;

-numărul de evenimente: $E = \{e_1, e_2, e_3, \ldots, e_M\}$;
-probabilitatea ca stare inițială să fie i: $\pi_j = P[q_1 = j]$;
-A={ a_{ij} } matricea cu probabilitățile de tranziție, unde $a_{ij} = P[q_t = j \mid q_{t-1} = i]$ este probabilitatea de a trece din starea i în starea j;
-B={ $b_j(k)$} matricea de ieșire, unde $b_j(k)$ este probabilitatea să se emită simbolul o_k la starea i;
- $O = \{o_1, o_2, o_3, \ldots, o_M\}$- mulțimea observațiilor de ieșire, în cazul în care această mulțime este discretă;

Un model întreg se poate nota cu : $\lambda = (A,B, \pi)$;
Cum parametrii de voce iau valori într-un spațiu continuu trebuie ca și vectorul de ieșire al HMM-ului să aibă valori continue. Din acest motiv se folosesc mixturile gaussiene pentru modelarea ieșirii stărilor HMM-ului.

Sistemul de recunoaștere a vorbirii bazat pe HMM este format din HMM-urile unităților ‘subcuvintelor’ numite foneme. Numărul stărilor HMM distincte într-un sistem depinde de mărimea setului de unități ale subcuvintelor [H02]. În figura 2.3 este prezentata spectograma unei propoziții.

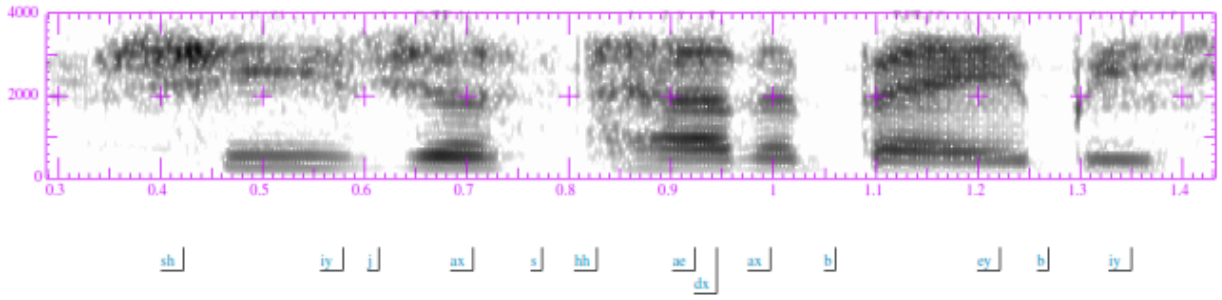


Figura 2.3 Spectograma propoziției ‘Ea a avut doar un copil’

Deoarece semnalul de vorbire este unul nestaționar, analiza spectrală nu poate fi făcută pe întreg semnalul, ci pe cadre scurte (20-30ms), cvasi-staționare. Astfel semnalul în domeniul timp este împărțit în ferestre, iar pentru fiecare fereastră sunt extrași parametrii de tip spectral sau cesptral.

Cei mai folosiți coeficienți în recunoașterea vorbirii sunt cei MEL(*Mel Frequency Cepstral Coefficients)* cepstrali MFCC, deoarece sunt în număr mai mic față de cei spectrali (13 în loc de 64 sau 128 în celălalt caz), nu sunt corelați între ei.

Coeficienții MFCC sunt bazați pe anvelopa spectrală a logaritmului semnalului, ce folosește o scară neliniară de frecvență, deoarece simulează mai bine modul neliniar de percepție al omului. Semnalului x(n) i se aplică o transformată Fourier discrete (DFT) cu formula:

$$X_a[k] = \sum_{n=0}^{N-1} x[n] e^{-j2\pi nk/N} \quad ,0 \le n < N$$

Se folosește apoi un banc de M filtre (m=1,2,…M) triunghiulare, ce sunt plasate pe axa frecvențelor astfel încât frecvența centrală a filtrelor să urmeze scara mel. Banda lor crește o dată cu indexul m, conform figurii 2.4 [B01] :

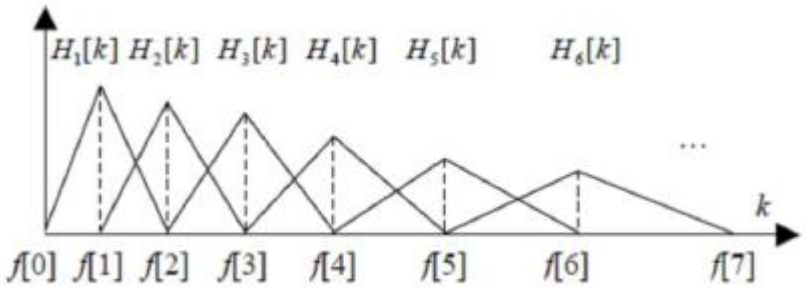


Figura 2.4 Bancul de filtre pe frecvențe în scara MEL

După cum se poate observa din figură, filtrarea cu scala Mel subliniază frecvențele joase. Relaţia între frecvenţele MEL şi cele liniare este, figura 2.5 :

$$\text{Frecvenţa\_mel} = 2595 * \log\left(1 + \frac{\text{Frecvenţa\_liniară}}{700}\right)$$

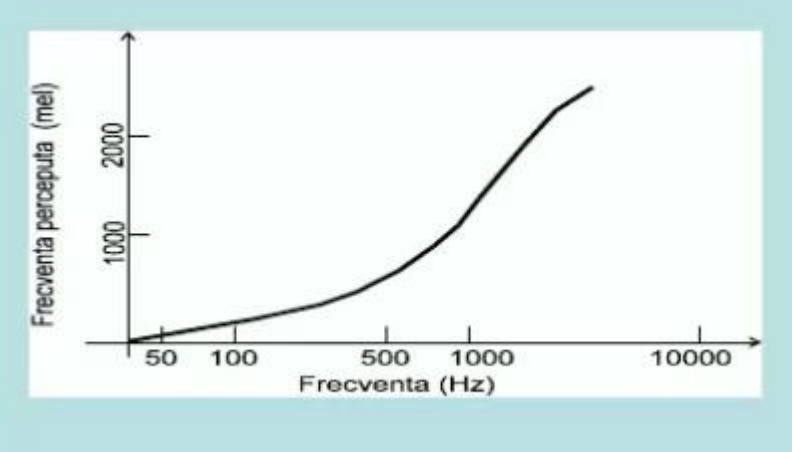


Figura 2.5 Scara Mel în percepția auditivă

Apoi, se calculează energia logaritmică la ieșirea fiecărui filtru:

$$S[m] = \ln\left(\sum_{n=0}^{N-1} |Xa[k]|^2 * H_m[k]\right) \quad , 0 \le m \le M$$

Această reprezentare este nivelată şi ortogonalizată prin aplicarea transformatei cosinus discrete (*DCT – Discrete Cosinus Transform*) cu ajutorul formulei :

$$c[n]=\sum_{m=0}^{M-1} S[m] * \cos\left(\pi n\left(m+\frac{1}{2}\right)M\right) \ , \ 0\leq n\leq M$$

şi astfel rezultă 7 coeficienţi cepstrali pentru fiecare vector mel.M poate varia de la 24 la 40, dar pentru recunoaşterea de vorbire se folosesc 13.Aceşti vectori, în mod normal, se calculează la fiecare 10ms cu ferestre Hamming de 25ms în interiorul semnalului. Coeficienţii MFCC sunt creaţi după modelul din figura 2.6 :

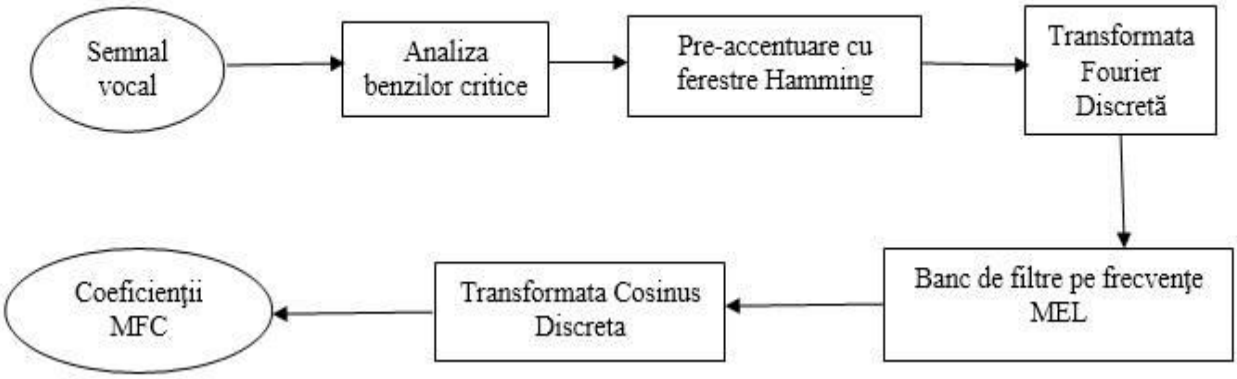


Figura 2.6. Schema bloc pentru analiza coeficienţilor MFC

2.1.2.2 Platforma HMM/GMM

Aşa cum am văzut mai sus, un HMM este un automat cu un număr finit de stări, format dintr-un set de stări conectate prin arce, ce reprezintă tranziţiile. Prima şi ultima stare a unui HMM este non-emisivă.

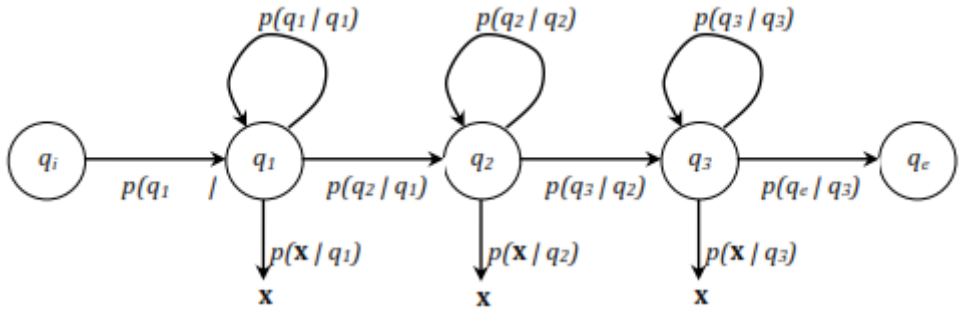


Figura 2.7. Reprezentarea unui HMM ca un automat probabilist cu număr finit de stări

Probabilităţile observaţiilor de ieşire a unei stări q_i sunt modelate de o funcţie densitate de probabilitate, sub forma unei sume de K funcţii gaussiene.

Dacă X este o variabilă aleatoare continuă, atunci densitatea de probabilitate a lui X este o funcţie f(x) pentru care:

$$P(a\leq X\leq b)=\int_a^b f(x)dx,$$

deci probabilitatea ca X să aibă valori între a şi b reprezintă aria densităţii între a şi b.Orice funcţie arbitrară a densităţii de probabilitate poate fi construită însumând N Gaussiene ponderate (mixture de Gaussiene):

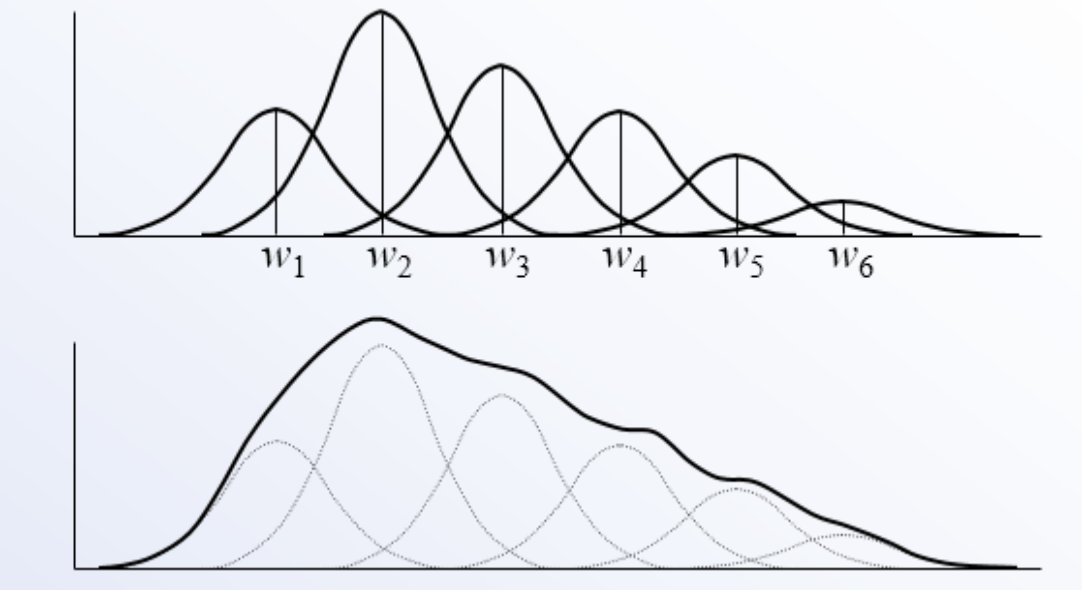


Figura 2.8. Însumarea a șase gaussiene ponderate

Astfel se formează un GMM (din engleză Gaussian Mixture Model).
În cazul în care ieşirea unei stări este o mixtură gaussiană (GMM), funcţia densitate de probabilitate se calculează cu formula:

$$b_i(x)=p(x|\lambda_j) = \sum_{i=1}^{K} w_{ik}\mathcal{N}(x;\mu ik, \Sigma_i),$$

în care x este un vector D-dimensional de date, cu valori continue (vectorul trăsăturilor), w_{ik} sunt mixturile ponderate, iar $\mathcal{N}(x_t;\mu_{ik}, \Sigma_i)$, i = 1, . . . , K, sunt componentele densităţilor Gaussiene, cu vectorul mediilor μ_i şi matricea de varianta Σi:

$$\mathcal{N}(x, \mu ik, \Sigma_i)=\frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma i|^{\frac{1}{2}}}\exp\left(-\frac{1}{2}(x-\mu ik)^T \Sigma i^{-1}(x-\mu ik)\right)$$

Când realizăm modelarea vorbirii cu ajutorul HMM-urilor putem face două presupuneri [B01], [C02] :

Independenţa observaţiilor: probabilitatea ca o stare să genereze un anumit vector acustic este independenţa de vectorii generaţi în stările anterioare.

Proces Markov: tranziţia între stările HMM este un proces Markov de ordinul întâi în care probabilitatea tranziţiei următoare depinde numai de starea actuală.

Cele trei probleme ale platformei HMM/GMM sunt: evaluarea, decodare şi estimarea, pentru care există 3 algoritmi:

- Antrenarea (estimarea diverşilor parametrii ai unui sistem HMM/GMM) foloseşte unul din algoritmii Forward-Backword sau Baum-Welch (o particularizare a lui expectation-maximization-EM);
- Decodarea (identificarea celei mai probabile secvenţe de stări ce ar putea genera secvenţa de observaţii) se rezolvă cu algoritmul Viterbi;
- Evaluarea (aflarea probabilităţii ca o secvenţă de observaţii să fie emisă de un HMM) se realizează cu algortimul Forward.

Cheia algoritmului Forward se rezumă la calculul probabilităţii care urmează $\alpha_t(q_j)=p(x_1,\ldots, x_t,q_t = q_j|\lambda)$, adică probabilitatea de observare a secvenţei $x_1,\ldots, x_t$, fiind în starea q_j la momentul t. Cu ajutorul presupunerilor Markov putem folosi formula:

$$\alpha_t(q_i)=\sum_{i=1}^{N} \alpha_{t-1}(\mathrm{qi})\, \alpha_{ij} b_j(x_t)$$

Algoritmul Viterbi funcţionează în maniera asemănătoare cu cel anterior, doar că sumarea nu se mai realizează la fiecare pas, ci se calculează valoarea maximă.

2.1.3 Modelarea fonetică

O unitate de vorbire potrivită trebuie sa aibă cel puţin trei caracteristici, potrivit autorilor [H01] :

- Unitatea să fie precisă: să aibă realizarea acustică care apare în diferite contexte;
- Unitatea să fie antrenabilă: să dispunem de suficiente date astfel încât să estimăm parametrii asociaţi ei;
- Unitatea să fie generalizabilă, astfel încât să putem construi orice cuvânt dintr-un set de unităţi de bază.

Sistemele de recunoaştere a vorbirii nu folosesc cuvinte, deoarece acestea nu sunt nici antrenabile, nici generalizabile, ci se bazează pe foneme, ca unităţi de vorbire de bază, ce sunt independente de vocabularul cerinţei de recunoaştere.Modelul fonetic nu este potrivit deoarece, se presupune, că un fonem este pronunţat identic, indiferent de context, iar oamenii nu pot pronunţa un cuvânt ca un set de foneme independente.

Dacă am utiliza foneme dependente de context şi am avea la dispoziţie suficiente informaţii pentru antrenare şi implicit pentru estimarea corecta a parametrilor modelelor corespunzătoare, am avea o precizie de recunoaştere mult mai bună.

Modelul de trifoneme este un model fonetic ce ţine cont şi de fonemele din dreapta şi stânga sa. Dacă găsim două foneme identice în contexte diferite, ele sunt considerate trifoneme diferite.Folosirea de trifoneme ar conduce la o precizie mai puternică, deoarece conţin şi efectele co-articulare, însă datorită numărului mare de modele diferite, am avea un număr ridicat de parametrii estimaţi, ce trebuie de asemenea antrenaţi.

Scara mel pentru percepţia auditivă bazată pe HMM este formată din HMM-urile unităţilor ‘subcuvintelor’ numite foneme.Numărul stărilor HMM distincte într-un sistem depinde de mărimea setului de unităţi al subcuvintelor.
Modelul fonetic conţine un dicţionar de pronunţii, mapând toate cuvintele din vocabular cu o secvenţă de foneme.Construcţia acestuia se poate face manual sau automat, însă ultimul mod nu oferă o precizie la fel de bună ca primul.

2.1.4. Principiile de bază ale modelării limbajului natural

2.1.4.1. Modele de limbă statistice

Cea mai mare diferenţă între un sistem de recunoaştere a cuvintelor izolate şi unul de vorbire continuă este modelul de limbă.Scopul sau în cadrul decodării, aşa cum este arătat în figura 2.1., este de a estima probabilitatea ca o secvenţă de cuvinte $W=w_1,w_2,.....w_n$ să fie o secvenţă validă pentru o sarcină de recunoaştere.Această probabilitate este complementare cei de la modelul acustic.
Probabilitatea se poate calcula cu formula :

$$p(W)=p(w_1,w_2,.....w_n)=p(w_1)p(w_2|w_1).....p(wn|\ w_1,w_2,.....w_{n-1})$$

şi se poate observa că depinde de istoria cuvântului, adică de secvenţa de cuvinte anterioare lui, ce nu poate include un număr infinit de cuvinte, ci doar m cuvinte.Astfel s-a ajuns la modelul de limbă de tip n-gram [R01], ce este un model mult mai evoluat.Acesta prezice următorul cuvânt pe baza celor n-1 cuvinte anterioare.

Se folosesc modele de tip trigram (n=3), unde este nevoie de o istorie de două cuvinte pentru a-l putea prezice pe al treilea sau bigram (n=2).

În recunoaşterea de vorbire, sunetele de intrare se pot confunda foarte uşor, deoarece multe cuvinte sună foarte asemănător.

2.1.4.2. Construcţia modelelor de tip n-gram

În cazul unui model de limbă bigram, probabilităţile $p(w_i|wj)$ pentru fiecare pereche de cuvinte (w_i,w_j) se calculează cu formula probabilităţii maxime (Maximum Likelihood-ML) :

$$p(w_i|w_j)=\frac{count(wi,wj)}{\sum_w count(wi,wj)}$$

în care se numără de câte ori cuvântul w_i este urmat de w_j faţă de alte cuvinte;
Pentru un trigram formula se modifică astfel:

$$p(w_k|w_i,w_j)=\frac{count(wi,wj,wk)}{\sum_w count(wi,wj,wk)}$$

Avem nevoie de o cantitate mare de date de antrenare pentru a obţine un model n-gram precis.Există însă şi probleme precum existenţa unor n-grame ce nu apar în textul de antrenare însă apar în textele de evaluare, indiferente de mărimea corpusului, sau existenţa unor n-grame ce apar de puţine ori (mai puţin de zece ori).Probleme se rezolvă cu metodele de back-off, care folosesc multiple modele de limbă pentru a crea un model de limbă interpolat, care să poată utiliza toate părţile ce îl compun.

2.1.5. Gramatici cu stări finite FSG (Finite State Grammar)

O gramatica cu stări finite FSG este un graf în care fiecare nod reprezintă un cuvânt al vocabularului sarcinii de recunoaştere, iar arcele dintre noduri sunt tranziţii între cuvinte.Astfel sunt specificate toate secvenţele de cuvinte permise de gramatica sarcinii respective.Dacă atribuim acele costuri, specificăm şi probabilitatea ca un cuvânt să fie precedat de altul.În cazul sarcinilor de mici dimensiuni (recunoaştere de cifre, apelarea unui număr de telefon), acest model poate fi folosit cu succes.

2.1.6.Evaluarea sistemelor

Evaluarea unui sistem de recunoaştere a vorbirii necesită existenţa unei baze de date de înregistrări, numită baza de date de evaluare şi se întemeiază pe transcrierile clipurilor audio din bază. Acestea descriu cât de bun este un sistem de recunoaştere a vorbirii pentru a transcrie vorbirea dintr-un alt domeniu. Criteriile de bază pentru evaluarea unui model de limbă sunt rata de cuvinte în afara vocabularului(Out of vocabulary OOV) şi perplexitatea(PPL), ce măsoară capacitatea de predicţie a unui model de limbă.Cu cât e mai mică perplexitatea, cu atât e mai bună capacitatea de predicţie a acelui model [H01].

Perplexitatea unui cuvânt din afara vocabularului este infinită şşi acest lucru face evaluarea performanţei unui model de limbă dificilă.În acest caz, procentul cuvintelor OOV trebuie specificată când evaluăm un model de limbă.

Evaluarea completă a sistemului de recunoaştere a vorbirii este realizată prin compararea transcrierilor clipurilor audio de evaluare cu transcrierile generate în urma procesului de decodare,conform cu figura 2.9 :

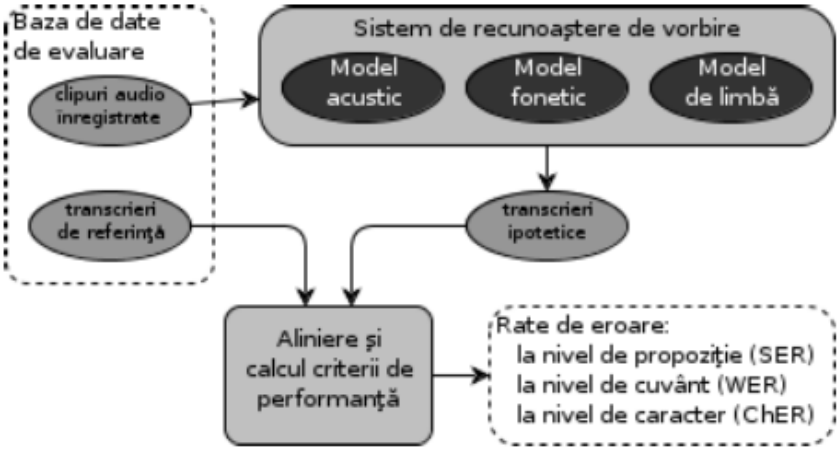


Figura 2.9 Evaluarea unui sistem de recunoastere

Rata de eroare la nivel de propoziţie (SER-Sentence Error Rate) este un criteriu de performanţă utilizat în evaluare şi este raportul între numărul de propoziţii eronate (ce conţin cel puţin o greşeală) şi numărul total de propoziţii din transcrierea de referinţă:

$$SER[\%]=100*\frac{\#PROPOZIŢII\ ERONATE}{\#PROPOZIŢII\ ÎN\ TRANSCRIEREA\ DE\ REFERINŢĂ}$$

Aceasta rată se foloseşte doar în situaţiile în care orice propoziţie ce conţine o greşeală este inutilă (sistem de recunoaştere a unui IBAN, CNP).

Metrica standard de evaluare a sistemelor de recunoaştere a vorbirii este rata de eroare a cuvintelor (WER-Word Error Rate).Această rată se referă la cât de mult cuvântul returnat de recunoascător (numit şi sir ipotetic) diferă de transcrierea de referinţă.Primul pas în aflarea ratei de eroare consta în determinarea distanţei minine între cuvântul corect şi cel recunoscut. Rezultatul acestui calcul va da numărul minim de cuvinte substituite, inserate sau şterse şi astfel, formula ratei de eroare va fi:

$$WER[\%]=100*\frac{\#INSERŢII+\#SUBSTITUŢII+\#ŞTERGERI}{\#NUMĂRUL\ TOTAL\ DE\ CUVINTE\ DIN\ TRANSCRIEREA\ CORECTĂ}$$

În cazul în care, procentul de caractere eronate din cuvântul ipotetic este redus, eroarea de substituţie se poate redresa mult mai uşor prin utilizarea ratei de eroare la nivel de caracter ChER (Character Error Rate).Este foarte asemănătoare cu rata de eroare la nivel de cuvânt, diferenţa dintre ele limitându-se la nivelul unităţii folosite de baza folosite în comparaţie: caracterul şi nu cuvântul:

$$ChER[\%]=100*\frac{\#ERORI\ DE\ INSERŢIE\ DE\ CARACTER+\#ERORI\ DE\ SUBSTITUŢIE\ DE\ CARACTER+\#ERORI\ DE\ ŞTERGERE\ DE\ CARACTER}{\#CARACTERE\ IN\ TRANSCRIEREA\ CORECTĂ}$$

2.1.7 Unelte folosite în recunoașterea vorbirii [Sphinx]

Pentru lucrarea de faţă am folosit setul de intrumente, pus la dispoziţie în mod gratuit pentru recunoaşterea vorbirii de CMU Sphinx.

Pe langă versiunile oferite de Sphinx, există numeroase sisteme utilizate în recunoaşterea vorbirii precum : HTK [Y01] , ISIP[D01], AVCSR[L01]. Pentru a facilita toate inovaţiile în domeniul recunoaşterii vorbirii, a fost creat un Sphinx-4: o platformă la dispoziţia utilizatorului ce încorporează metodologiile de ultimă generaţie şi acoperă şi ariile în curs de dezvoltare. Include componente separate pentru fiecare sarcină specifică, iar modulele sale pot fi înlocuite în timpul rulării.

Sphinx-4 este un system de recunoaştere a vorbii scris în limbajul de programare Java. A fost creat în urma unei colaborări dintre grupurile Sphinx de la Universitatea Carnegie Mellon, Laboratoarele Sun Microsystems, Laboratoarele Mitsubishi Electric Research Labs (MERL) şi Hewlett Packard (HP), cu unele contribuţii de la Universitatea din California la Santa Cruz (UCSC) şi Institutul de Tehnologie Massachusetts (Massachusetts Institute of Technology (MIT)).

Principalele trei module din arhitectura Sphinx-4 sunt:

- FrontEnd
- Decodor
- Linguist

Dispunând în momentul de faţă de cei mai mulţi parametrii configurabili, de acest aspect ocupându-se ConfigurationManager. Sphinx-4 foloseşte pentru a putea oferi statistici în legatură cu procesul de decodare precum rata de eroare a cuvintelor, viteza de rulare, memoria utilizată diverse ustensile (din engleză Tools) şi de asemenea Utilitati (din engleză Utilities) ce suportă procesarea la nivel de aplicaţie a rezultatelor decodării. Se pot obţine grile de rezultate, scoruri de confidențialitate sau intelegerea limbajului natural.

Scopul modulului FrontEnd este acela de a transforma un semnal de intrare (audio) într-o secvenţă de ieşire de trasături. Conţine mai multe lanţuri paralele de procesare astfel încat se pot calcula simultan tipuri diferite de parametrii, precum MFCC şi PLP din acelaşi semnal de intrare sau din semnale diferite. Cu ajutorul arhitecturii FrontEnd ce conţine blocuri de procesare de date (din engleză DataProcessor) şi de date (din engleza Data), Sphinx-4 oferă suport pentru urmatoarele tehnici: citirea de la diferite formate de intrare, citirea de la dispozitivele sistemului audio pentru funcţionarea în mod live, pre-accentuare, folosirea de ferestre (ex. ferestre Hamming şi Hanning), transformata Fourier discretă (FFT), transformata cosinus discretă (DCT).

Modului Lingvist genereză la ieşire un grafic de căutare (din engleză SearchGraph) şi poate fi configurat cu diverse implementări lingvistice şi este caracterizat de trei componente:

Modelul de limbă format din gramatici bazate pe grafice sau modele stocastice N-gram. Suportă formate precum: gramatică JSGF [JSGF], LMG, FST, model n-gram simplu, model trigram.
Dictionarul furnizează pronunţiile pentru cuvintele din modelul de limbă şi împarte cuvintele în secveţe formate din unităţi găsite în modelul acustic.

Modelul acustic o mapare între unităţile de vorbire şi un HMM, ce ia în calcul şi poziţia cuvântului şi contextul. Acesta permite varierea mai multor parametrii ce caracterizează un HMM printre care: mixturile gaussiene, matricile de tranziţii, înălţimea gaussienelor. Sphinx-4 este capabil să implementeze un singur model acustic ce încarcă şi foloseşte modelele acustice generate de antrenorul Sphinx-3.

Cel de-al treilea modul, blocul de decodare foloseşte parametrii rezultaţi din primul modul împreună cu SearchGraph pentru a genera ipotezele rezultate. Cea mai importantă componentă a sa o reprezintă managerul de căutare (din engleză SearchManager) şi poate folosi diverşi algoritmi de decodare precum Viterbi, A*, Bushderby, decodare paralelă [W01].

2.2 Spoken Term Detection [STD]

2.2.1 Introducere

Procesarea de informații a devenit o activitate principală în lumea actuală, comunicațiile vorbite constituind sursa majora a acestor informații. Procesul este o inițiativă pentru a facilita dezvoltarea tehnologiilor de găsire a informațiilor în arhivele de înregistrări. STD în cazul limbilor bine

documentate are la bază un sistem de recunoaştere automată a vorbirii dependent de o bază mare de date de antrenare, ce include atât înregistrări (pentru modelarea acustică), cât şi text (pentru modelarea de limbă). Acurateţea sistemului ASR pentru aceste limbi, precum şi tehnicile îmbunătăţite de indexare şi căutare asigură calitatea procesului STD.

2.2.2 Sarcina procesului şi arhitectura sistemului

Procesul constă în căutarea unui cuvânt rostit în interiorul unei baze de înregistrări, de asemenea rostite. Am dispus de două seturi de înregistrări multilingve: unul cu exemplele de întrebări şi celălalt set cu conţinutul în care are loc căutarea. Scopul unei asemenea evaluări este de a detecta rapid un termen, adică o secvenţă de cuvinte rostite consecutiv, în cadrul unor înregistrări.

Vom folosi arhitectura STD propusă în Campania NIST 2006 descrisă în figura 2.10, ce separă partea de indexare de cea de căutare şi nu efectuează sondarea direct pentru fiecare termen. Beneficiul acestei arhitecturi este acela de a permite măsurători uniforme asupra resurselor de operare: viteza de indexare, marimea indexului,viteza de căutare, precum şi efectuarea unei cautări mai rapide, deoarece indexarea o simplifică.

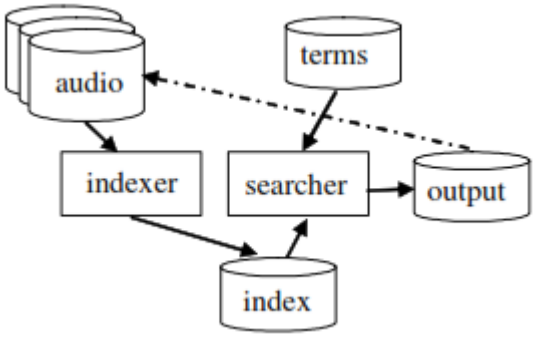


Figura 2.10 Arhitectura universală a sistemului STD [STD]

Un aspect important îl constituie timpii de căutare pentru un anumit cuvânt, care trebuie să fie mici (câteva secunde). Pentru a dezvolta acest sistem, ne bazăm pe sistemul de recunoaştere automată a vorbirii pentru limba română, dezvoltat anterior, pe care îl folosim pentru indexare, astfel încât toate înregistrările să fie transformate în şiruri de foneme. În timpul căutării, cuvantul rostit ţintă este transcris de sistemul ASR şi astfel vom căuta un rând de foneme asemanător cu cel pe care îl avem şi cu care depaşeşte un prag de similitudine.

2.2.3. Termenii utilizaţi. Fişierele de intrare şi ieşire ale sistemului

Un termen se referă la un cuvânt sau un grup de cuvinte care au un singur sens în orice context, astfel încât oriunde l-am găsi, semnificaţia este aceeaşi. Deşi un termen, de obicei are o singură formă de pronunţie, orice variaţie sau model de plural ar trebuie recunoscută în cazul căutării noastre. Termenii vor fi prezentaţi folosind ortografia limbii native: ASCII pentru engleză, UTF-8 pentru arabă şi GBK pentru limba mandarin chineză.

Termenii sunt constituiţi din unul sau mai multe cuvinte. Conceptul de cuvânt ia forme diferinte de la limbă la limbă.
Aparițiile de referinţă ale cuvintelor căutate pot fi găsite în fişierul RTTM furnizat, ce conţine informaţii precum numele înregistrării ce conţine termenul, canalul, timpul când începe exprimat în secunde, durata cuvântului.

Fişierul de ieşire va fi sub forma unui fişier text în format XML pentru fiecare combinaţie a categoriilor de limbi, ce va conţine următoarele informaţii:
-limba
-tipul sursei;
-indexul timpului de generare şi a măsurătorilor legate de dimensiuni;
-termenii căutaţi împreună cu următoarele informaţii pentru fiecare cuvânt: timpul de căutare şi pentru fiecare presupusă găsire a cuvântului aflăm: locaţia (timpul unde începe şi se termină în interiorul înregistrării), scorul de detecţie şi decizia binară luată.

Generarea la ieşire a deciziei luate împreună cu scorul de detecţie pentru fiecare posibilă apariţie are un beneficiu semnificativ asupra evaluării sistemului. Putem considera corectă ieşirea sistemului dacă termenii apar în transcriere ca o potrivire perfectă, corespunzând de asemenea şi timpii.

2.2.4. Metodologia evaluării

Performanţele unui system STD sunt caracterizate de două tipuri de erori: alarme false şi detecţii ratate şi pot fi evaluate, de asemenea prin două metode:

a)grafic folosind curba erorii de detectare a compromisului (DET) ce prezintă o vedere intuitivă a performanţei;
b)pentru un anumit punct de operare de pe curbă, se foloseşte valoarea ponderată a termenului (TWV), folosită pentru optimizarea performanţei sistemului.

2.2.4.1 Curba erorii de detectare a compromisului

Performanţele de bază ale detecţiei sunt caracterizate în principal de curbele standard ale erorilor de detecţie a compromisurilor DET (Detection Error Tradeoff) dintre probabilitatea de ratare (P_{Miss}), când un cuvânt nu este detectat deşi el apare şi probabilitatea de alarmă falsă (P_{FA}), în cazul în care detecţia sistemului nu corespunde cu existenţa cuvântului. Ambele probabilităţi sunt funcţii ce depind de pragul de detecţie Θ, după cum urmează:

$$P_{Miss}(cuvant, \Theta)=1-\frac{Ncorect(cuvant,\theta)}{Ntrue\,(cuvant)}$$

$$P_{FA}(cuvant, \Theta)=\frac{Nfals(cuvant,\theta)}{NNT\,(cuvant)}$$, unde

- N_{corect}(cuvânt, Θ) reprezintă numărul de detecţii corecte ale cuvântului cu un scor mai mare sau egal ca pragul Θ;
- N_{fals}(cuvânt, Θ) este numărul detecţiilor incorecte ale cuvântului cu un scor mai mare sau egal decât pragul Θ;

- N_{true}(cuvant) este numărul real al apariţiilor cuvântului căutat în context, iar N_{NT}(cuvant), numărul oportunităţilor pentru detecţiile incorecte ale cuvântului (=”Non-target”); N_{NT}(cuvânt) se alege în mod arbitrar, proporţional cu numarul de secunde ale vorbirii în setul folosit pentru testare:
- N_{NT}(cuvânt)= n_{tps}*$T_{vorbire}$- N_{true}(cuvânt), unde n_{tps} este numărul de încercări de vorbire pe secundă (setat arbitrat la 1).
- $T_{vorbire}$ reprezintă totalul discursului (exprimat în secunde).

Cele două probabilităţi vor fi calculate separat pentru fiecare termen şi după se vor media peste toţi termenii selectaţi cu aceeaşi pondere. Acest lucru este valabil doar pentru cuvintele cu un număr diferit de zero de apariţii în setul de înregistrări, astfel încât P_{FA} să poată fi definită. Curbele DET se evaluează ca o funcţie de limbă şi tipul sursei pentru diverse înregistrări şi query-uri.

2.2.4.2 Performanţele de detectare ale sistemului

A masura “valoarea” unui sistem se referă la utilitatea acelui sistem pentru un utilizator, deoarece un sistem perfect va raspunde întotdeauna corect la o cerinţă, fiecare raspuns greşit sau deruntant reducând această valoare alocată sistemului.

Performanţa de detectare a sistemului va fi masurată prin alocarea unui cost fiecarei recunoaşteri corecte şi un cost (o valoare negativă) fiecărei ieşiri incorecte. Se folosesc două definiţii ale valorii totale a sistemului numite: valoarea ponderată a apariţiilor şi valoarea ponderată a termenului (TWV).
Valoarea ponderată a apariţiilor se calculează prin acumularea valorilor pentru fiecare detecţie corectă din care se substrage costul pentru fiecare detecţie greşită, fiecare cuvânt contribuind în mod egal. Valorea ponderată a termenilor (Term Weighted Value(θ)) se evaluează iniţial prin calculul probabilităţilor de ratare şi alarma falsă, apoi cu ajutorul lor şi a unei probabilităţi aprior alese, de calcul a valorilor specifice fiecărui cuvânt, se face o medie ce include toţi termenii şi se determină valoarea totală a sistemului, după formula:

$$TWV(\theta)=1-medie\{P_{Miss}(cuvânt,\theta)+\beta * P_{FA}(cuvânt,\theta)\},\text{unde } \beta=\frac{C}{V}(Pr_{cuvânt}^{-1}-1);$$

Folosim raportul cost/valoare, C/V egal cu 0.1 şi probabilitatea anterioră a cuvântului $Pr_{cuvânt}^{-1}$ este 10^{-4}. Valoarea maximă posibilă va fi de 1.0 şi corespunde unui sistem “perfect”: fără ratări şi fără alarme false. Pot fi posibile şi valorile nule (când sistemul nu obţine la ieşire nimic) sau chiar valori negative.
Faţă de valoarea ponderată a apariţiilor, ce a termenilor prezintă un avantaj fiind mai puţin susceptibilă la frecvenţa de apariţie a cuvintelor, excluzându-le pe cele care nu au nici o apariţie. Ieşirea sistemul trebuie să includă de asemenea o indicaţie binară a detecţiei, aleasă astfel încât să maximizeze valoarea de ieşire a sistemului.

Întrucât curbele DET reprezintă performanţa pentru toate valorile posibile ale pragului θ, doar două puncte prezintă interes deoarece arată dacă pragul de decizie actual al sistemului este optim. Primul este valoarea ponderată actuală a termenilor (ATWV), iar al doilea punct este reprezentat de valoarea ponderată maximă a termenilor (MTWV), ce corepunde cu locul de pe curbă unde pragul θ coincide cu valoarea TWV maximă.

Metrica principală folosită pentru comparaţie a fost Actual Term Weighted Value (ATWV) şi reprezintă abilitatea sistemului de a preciza punctul optim de operare pe baza scorului. Diferenţa dintre ATWV şi MTWV indică beneficiile pe care le vom avea dacă alegem un prag de decizie mai bun.

2.2.5 Procesul de căutare

În cazul în care am obţine pentru sistemul de recunoaştere ASR o precizie de 100%, procesul de detectare a cuvântului rostit s-ar reduce la căutarea unui şir de caractere într-un conţinut textural şi ar constitui o problemă simplă. Deoarece nu suntem aproape de acest caz ideal, vom căuta în baza noastră de înregistrări un şir similar cuvântului căutat. Căutarea exactă a unui cuvânt are rezultate STD foarte slabe: o probabilitate de alarmă falsă (FAP) de 0.1% şi o probabilitate de ratare (MP) de 99%. Cum am precizat anterior, erorile în recunoaștera limbajului vorbit la nivel de fonem pot fi substituții, inserții și ștergeri. Aceste erori sunt numărate automat prin aplicarea algoritmului DTW, unul din cei mai vechi şi mai importanţi algoritmi folosiţi în recunoaşterea vocală. Numărul mare de cuvinte ce nu fac parte din vocabularul (OOV) ce par a face parte din termenii de interogare va creşte numărul de erori.

2.2.5.1 Etapele parcurse în sistemul de căutare

O metodă simplă de recunoaştere a unui cuvânt izolat este să fie comparat cu un număr predefinit de şabloane, dintre care se alege acela care se potriveşte cel mai bine.

Apar însă, în această situaţie, două probleme:
-ce formă trebuie să ia şabloanele;
-cum vom compara noi cuvântul cu semnalul vocal.

a)Prima etapă în procesul de recunoaştere este analiza vorbirii, în timpul căreia semnalul este procesat, astfel încât să extragem cele mai importante caracteristici, numite trăsături sau parametri. Precum am detaliat în partea de recunoaştere automată a vorbirii, am folosit coeficienţi cepstrali MFCC. Prin folosirea doar a celor mai importante caracteristici ale vorbirii, cantitatea de date utilizată pentru comparaţii este mult redusă şi prin urmare vom avea mai puţine comparaţii şi nevoie de un timp mai scurt.

b)A doua etapă a constat în clusterizarea acestor coeficienţi, adică procesul de grupare a unor obiecte în grupuri numite clusteri în aşa fel încât acestea să fie similare, găsind practic reprezentanţii unor grupuri omogene.

Aceast algoritm se realizează parcurgând următoarele etape:

- Gruparea în K-means: partiţionăm vectorul de intrare în k seturi prin selecţie aleatoare sau pe baza distanţei maxime), așa cum este arătat și în figura 2.11 :

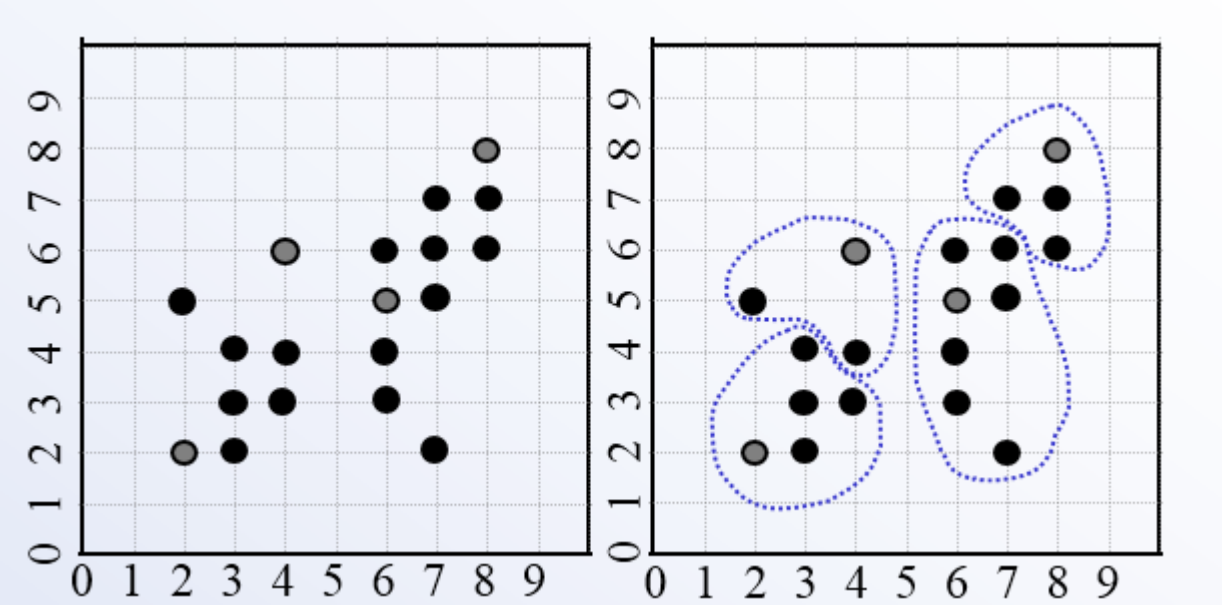
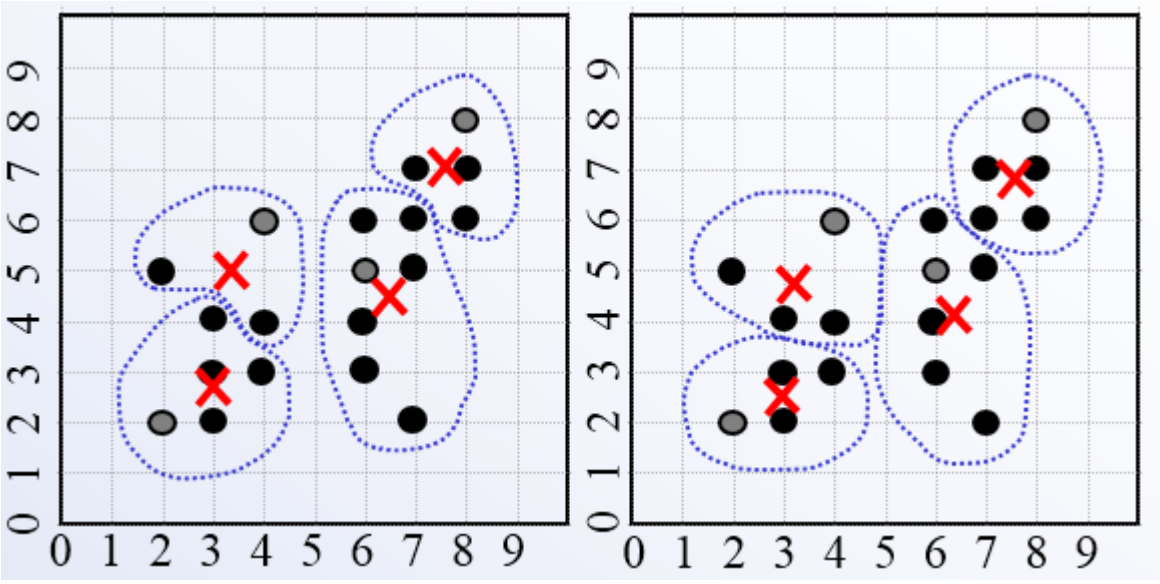


Figura 2.11 Gruparea punctelor în patru clustere

- Căutarea: pentru fiecare vector de antrenare găsim cel mai apropiat cuvânt de cod şi îi alocam grupului (clusterului) de care aparţine acel cuvânt.
- Updatarea centroidului: pentru fiecare grup, recalculăm centroidul, luând în calcul şi noua observaţie adaugată;
- Se repetă ultimii doi paşi până când distanţa medie scade sub un anumit prag sau nu se mai modifică, figura 2.12.



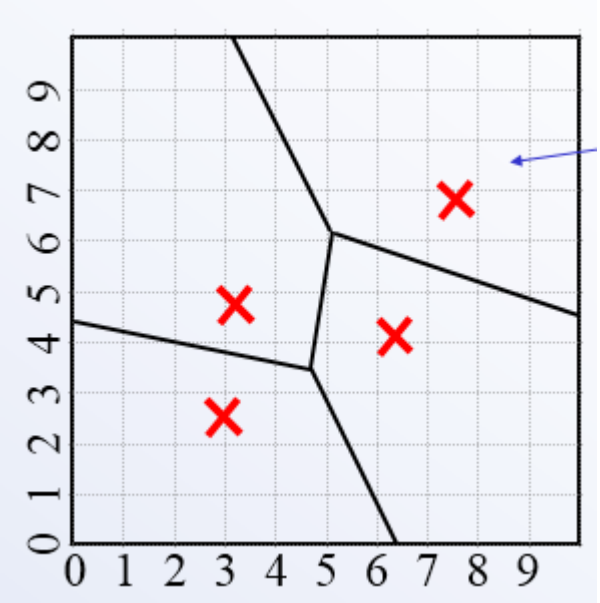


Figura 2.12 Calculul centroizilor fiecărui cluster până în momentul convergenței algoritmului

Algoritmul converge destul de repede în practică şi returnează cele mai bune clustere găsite prin execuţia a mai multor iteraţii. Soluţia depinde de numărul setului iniţial de clustere, ce trebuie definit înainte de a-l parcurge.

c)Împărţim fiecare înregistrare în ferestre proporţionale cu lungimea termenului căutat, de 1.5 ori mai mari.

d)Am folosit algoritmul DTW pentru a găsi alinierea neliniară optimă în timp pentru secvenţa de simboluri obţinute din procesarea textului şi secvenţa de segmente de voce înregistrate. Principiul de funcţionare este următorul: se creează o matrice având ca dimensiuni lungimile celor două secvenţe care trebuiesc comparate. În fiecare celulă a matricii se pune o valoare numerică dată de costul dintre elementele corespunzătoare ale celor două secvenţe, calculat cu ajutorul distanţei euclidiane.

Diferențele dintre formele de undă a unor cuvinte diferite pot fi văzute în figura 2.13.

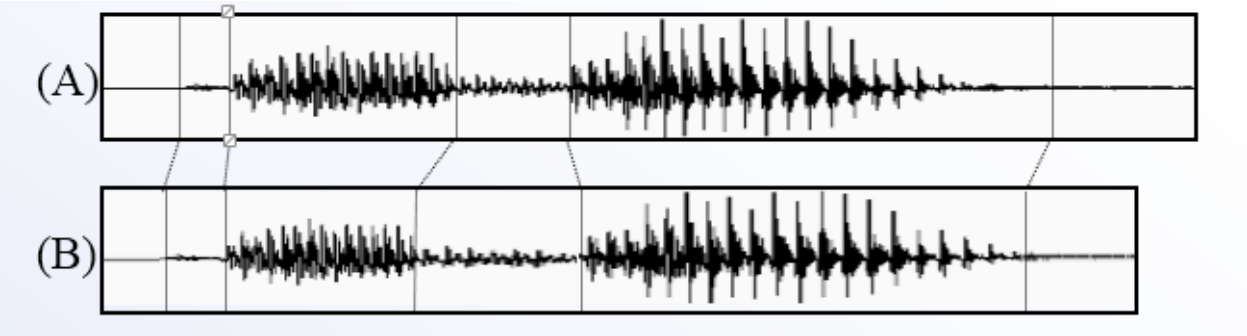


Figura 2.13 Formele de undă a două semnale diferite

Daca A şi B ar fi identice, drumul de cost minim ar fi o linie dreaptă ce trece prin 0 (diagonala principală). Deoarece cele două semnale nu sunt la fel, calea prin matrice arată cea mai bună potrivire între cadrele dintre A si B.

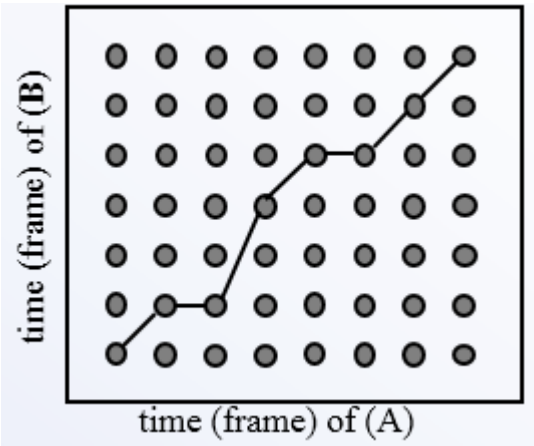


Figura 2.14 Rezultatul rulării algoritmului DTW

Vectorul de aliniere este reprezentat de această cale între fonemele de pe abscisă ce aparţin cuvântului original din înregistrare (cel de referinţă) şi cele ale cuvântului căutat de pe orizontală. Costurile au fost alese astfel încât să se poată acoperi toate tipurile de erori (substituiri, ştergeri şi inserţii). Tranziţiile orizontale corespund ştergerilor. Cele verticale, care corespund inserţiilor, au un cost mai mic decât cele orizontale, adică două foneme au fost recunoscute în locul uneia. De multe ori, fonemul recunoscut este cel corect şi încă unul, sau cel corect de două ori, ca în figura 2.15.

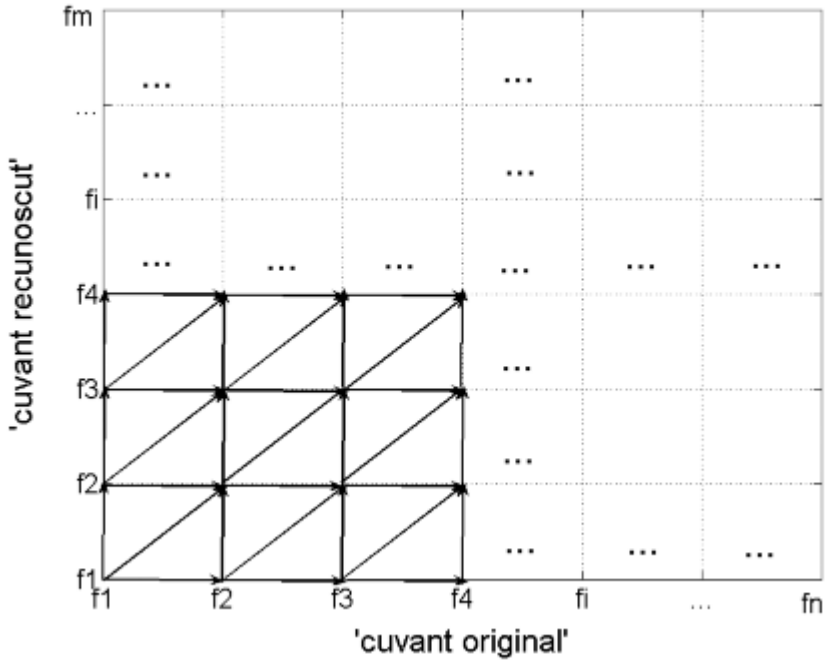


Figura 2.15 Aliniera cuvintelor cu algoritmul DTW [B01]

2.3 Detecția Genului

2.3.1 Introducere

Tehnologiile de recunoaştere a vorbitorului au fost studiate activ de câteva decenii, iar rezultatele au ajuns la crearea unor sisteme de ultimă generație, astfel încât telefoanele bazate pe servicii integrate cu recunoaştere a vorbirii, a vorbitorului sau a limbii vor suplimenta sau chiar înlocui serviciile cu operatori umani.

Scopul acestei aplicații este acela de a identifica genul vorbitorului pe baza vocii sale. Diferențele de gen în vorbirea umană se datorează diversității fiziologice precum lungimea tractului vocal, grosimea vocii sau stiluri de vorbire diferite, aspect evidențiat în figura 2.16.

Principiile recunoaşterii genului unui vorbitor sunt aceleaşi ca ale sistemului recunoaşterii automate ASR, constituit din două părţi: una de admitere în care este creat un model al vorbitorului cu ajutorul unui model de fundal şi o parte de recunoaştere în cadrul căreia luăm o decizie binară.

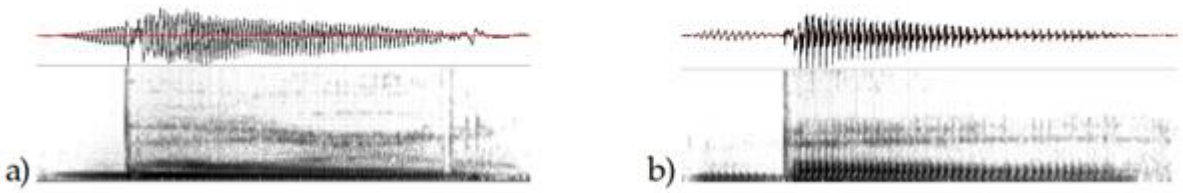


Figura 2.16 Cuvântul ‚minge' rostit de doi vorbitori diferiți: a) femeie și b) bărbat

Figura arată de ce simpla abordare de recunoaștere a semnalelor bazându-ne doar pe forma similară a sunetelor nu este de ajuns pentru sarcinile de recunoaștere a vorbirii. Jumătatea superioară a figurii arată forma de undă a semnalului brut, iar cea inferioară afișează versiunile prelucrate ale semnalelor evidențiind formatele specifice. Se poate observa că vorbirea este dependentă de persoană, lucru ce nu mai reprezintă o surpriză, deoarece vocile diferite sună diferit.

Principala caracteristică ce deosebește cele două genuri o reprezintă frecvența fundamentală F_0, ce are valoare tipică în cazul vorbirii masculine de 110Hz și 200Hz în cazul celei feminine. Înalțimea vocii în cazul copiilor este atât de variată, încât aceștia pot fi catalogați ca un al treilea gen. Valorile uzuale ale acestei frecvențe pentru persoanele cu vârsta cuprinsă între 20-70 ani sunt cuprinse între 80-170 Hz pentru barbați, 150-260 Hz pentru femei și 300-500 pentru copii.

Detectarea automată a genului unui vorbitor poate prezenta multiple avantaje:

- Identificarea genului și eliminarea componentelor caracteristice ce va conduce la o rată de compresie mai mare a semnalului, o cantitate mai mare de informații ce pot fi transmise și o economie asupra benzii ;
- Sortarea apelurilor telefonice după gen în cazul sondajelor;
- Facilitarea sistemelor de recunoaștere a vorbitorului prin împărțirea vorbirii în jumătăți, ce va conduce la reducerea calculelor și îmbunătățirea vitezei sistemului.

2.3.2 Uneltele folosite

Laboratoarele de informatică ale Universității din Maine (din limba franceză The Laboratoire d'Informatique de l'Université du Maine (LIUM)) au dezvoltat diverse pachete software scrise în limbajul de programare Java ce pot fi folosite în mai multe domenii precum: detecţia vorbirii/ liniştii, detecţia vorbitorului, calcularea coeficienţilor MFCC, segmentarea şi clusterizarea înregistrărilor.

LIUM oferă unelte atât pentru dezvoltatori ce îmbunătățesc performanțele WER, cât şi pentru utilizatori în cazul detecției spontane a vorbirii, cercetări în domeniul recunoașterii, traducere automată. LIUM_SpkDiarization conţine un set de unelte ce pot fi folosite pentru procesul cunoscut sub numele din engleză diarization, în care se pleacă din semnalul audio al utilizatorului pe care îl clusterizăm pe baza diverselor metrici. Setul a fost dezvoltat pentru campania de evaluare French ESTER2, cele mai bune rezultate obţinându-se pentru spectacolele de televiziune şi radio.

Segmentarea se referă la împărţirea fişierelor audio în diverse părţi omogene şi flexibile, iar procesul de diarization semnifică identificarea unică a vorbitorilor într-un fişier. Procesul poate îmbunătăți transcrierea automata, prezentând interes şi în indexarea documentelor multimedia.

2.3.3 Descrierea sistemului [Diarizare]

LIUM utilizează în aplicaţiile asociate vorbitorilor metode de grupare a înregistrărilor (în engleza diarization), proces ce constă, în terminologia NIST, în asocierea unor regiuni temporale din înregistrările audio cu etichetele corespunzătoare. Etichetele sunt formate din numele vorbitorilor, genul lor, tipul canalului utilizat (de banda largă sau banda îngustă), mediu din fundal (linişte, muzică, zgomot) sau orice altă caracteristică a semnalului. Scopul este acela de a detecta segmentele audio omogene, fiecare conţinând vocea unui singur vorbitor [S01].

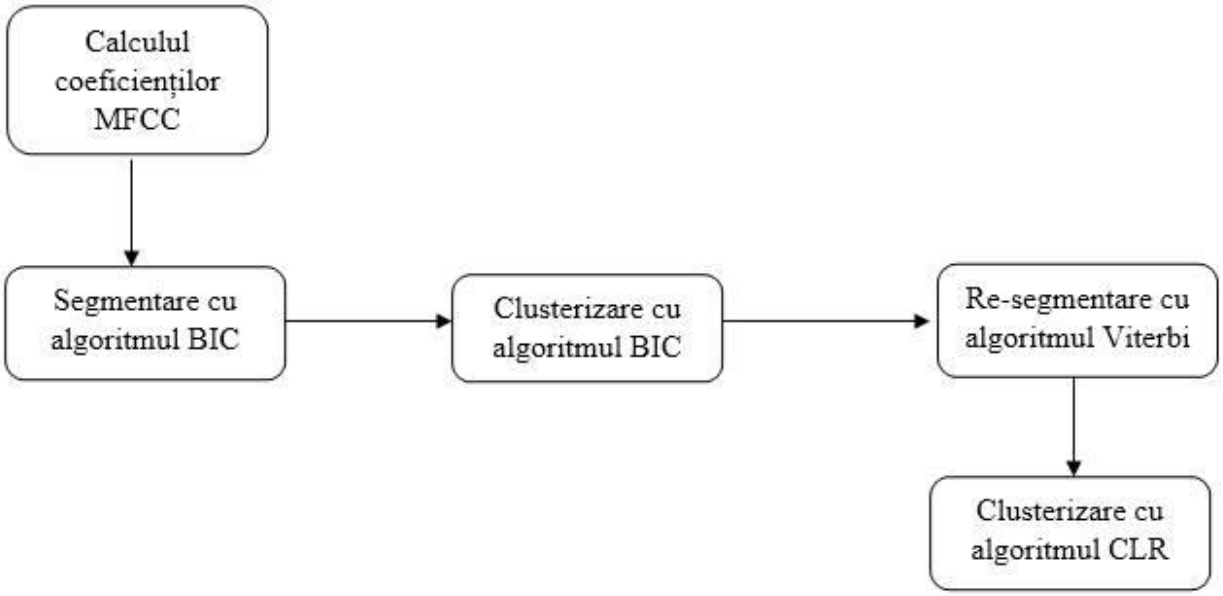


Figura 2.17 Arhitectura sistemului de diarization

Etapele care stau la baza sistemului sunt cele din figura 2.17:

1) Cadrele acustice sunt formate, în mod identic cu capitolul 2.1., şi rezultă 13 coeficienţi MFCC ce conţin şi energia C_0, după care îndepărtăm coeficientul C_0 şi calculăm derivatele de ordin întâi ale acestora.

2) Segmentarea este nucleul procesului de diarization şi are ca obiectiv divizarea fişierului audio în segmente omogene pentru vorbitori. Are loc în doua faze: prima detectează automat punctele de schimbare ce corespund marginilor segmentelor pe baza unui raport de probabilitate generalizat GLR (din engleza Generalized likelihood ratio), calculat cu ajutorul gaussianelor cu matrici de convergenţă complete. Aceste gaussiene sunt estimate pentru ferestre glisante de cinci secunde de-a lungul întregului semnal. Se estimează curba distanţei GLR pentru aceste ferestre, cele mai mari puncte corespunzând schimbării între vorbitori. Următoarea fază foloseşte distanţe BIC cu scopul de a asocia segmentele consecutive găsite ca aparţinând aceluiaşi vorbitor. Se pot folosi şi alte metode de măsurare precum KL2, divergența gausiană şi ΔBIC. BIC a fost introdus pentru prima dată de Chen si Gopalakrishnan [C01] și reprezintă un model de selecție, asimptotic, optimal, bazat pe criteriul Bayesian folosit pentru a decide care dintre cele N_c modele parametrice reprezintă cât mai bine M eșantioane ale datelor $x_i \in R^d$, i=1,2,……M . Eșantioanle x_i reprezintă vectori, d-dimensionali ce conțin trasături extrase la punctul anterior 1). În cazul segmentării vorbitorului, algoritmul se reduce doar la două modele angajate (N_c =2), adică se analizează doar două ferestre vecine X și Y în jurul timpul t_j, cu scopul de a decide dacă t_j este sau nu, un punct de schimbare al vorbitorului.

3) Toate segmentele sunt grupate în mulţimi (din engleza cluster), ce le sortează iniţial după nume şi apoi după momentul de început al înregistrării. Erori precum a avea două mulţimi distincte ce corespund aceluiaşi vorbitor sau asocierea segmentelor a doi vorbitori într-o singură mulţime au o penalitate destul de mare în metrica de evaluare NIST. Există mai mulţi algoritmi de grupare, secvenţiali sau ierarhici, cei din urmă divizându-se la rândul lor în două categorii (aglomeratici sau divizivi).

Algoritmul de clusterizare diviziv aglomerativ porneşte de la un număr N de clustere, fiecare cluster conţinând un singur segment, cu propriul lui model antrenat de o gaussiană cu matricea de covarianta completă. La fiecare pas sunt calculate distanţele dintre perechile de clustere, pe baza cărora clusterele care se asemănă cel mai mult sunt unite într-o mulţime nouă, căreia îi antrenăm propriul model. Algortimul se încheie când întâlnim un criteriu de stopare. Există diverse metrici şi criterii de stopare propuse în [Siu1991, Solomonoff1998, Chen1998a, Reynolds1998] . Unul dintre acestea este metrica Delta BIC (din engleza Bayesian Information Criterion) ce poate atât selecta clusterele în grupuri cât şi închide procesul, atunci când $\Delta BIC_{i,j} > 0$, i şi j reprezentând două clustere apropiate. Se pot folosi două tipuri de metrici Delta BIC, ce diferă prin factorul de penalizare:

Factorul de penalizare local P_l ce depinde de n_i si n_j, lungimile clusterului i, respectiv al clusterului j prin formula:

$$P_l = \frac{1}{2}\left(d + \frac{d(d+1)}{2}\right) + \log(n_i + n_j)$$

Factorul de penalizare global P_g ce foloseşte lungimea întregului semnal,n :

$$P_g = \frac{1}{2}(d + \frac{d(d+1)}{2}) + \log(n)$$

Experimentele au dovedit, că factorul local dă rezultate mai bune.

4) Cea de-a patru etapă constă într-o decodare cu algoritmul Viterbi pentru a genera o segmentare nouă, ce va îndepărta porțiunile de muzică sau zgomot. Clusterele create sunt modelate de un model Markov ascuns HMM cu o singură stare, reprezentate de un model de mixturi gaussiane cu opt componente. Segmentele de lungime mare sunt tăiate în punctele cu cea mai mică energie pentru a obţine doar segmente mai scurte de 20 secunde.

5) Ultimul pas constă în clusterizare folosind algoritmul CLR (Cross Likelihood ratio), cel de-al doilea algoritm ierarhic, aglomerativ. Trăsăturile folosite nu au fost normalizate cu scopul de a păstra informațiile mediului din fundal, ce ajută în diferențierea vorbitorilor. În momentul de față, fiecare cluster conție vocea unui singur utilizator, dar îi putem asocia acestuia mai multe. Pentru a putea efectua și ultima operație din proces, trebuie să eliminăm influența mediului din modelele clusterelor, prin normalizarea coeficienților. Astfel vom obține o relație de unu-la-unu între vorbitori și clustere.

La fiecare iterație, două clustere care maximizează rata scorului de încrucișare (CLR) [C01] sunt unite într-unul singur. Algoritmul se încheie când unitatea de măsură depășește un prag setat anterior.

2.3.4 Modelarea vorbitorului

Un sistem deterministic pas-cu-pas de diarization a fost dezvoltat de Meignier et al. și este folosit în detecția de gen și bandă. Acesta estimează automat numărul de vorbitori și se realizează având la bază modelele gaussiene GMM (cu 128 de componente diagonale) pentru fiecare din cele patru combinații de gen (feminin/masculin) și bandă îngustă/bandă largă. Fiecare cluster este etichetat în concordanță cu caracteristicele modelelor gaussiene care maximizează probabilitatea coeficienților conținuți. Fiecare model a fost extras dintr-o ora de înregistrare a fișierelor de antrenare date de ESTER. În urmă procesul de diarization au fost obținute segmente mai scurte de 20 de secunde, ce conțin vocea unui singur vorbitor, cu banda și genul cunoscut pentru fiecare segment în parte. Din fiecare înregistrare este eliminată liniștea și momentele de tacere.

După ce extragem vectori cu coeficienţi din înregistrările vorbitorului putem antrena un model al acestuia şi să îl stocăm în baza de date a sistemului. Întrucât folosim un model independent de vorbitor ce include o corespundenţă foarte mică sau chiar deloc între cadrele înregistărilor de referinţă şi cele de testare, alinierea la nivel de cadru nu este posibilă. Prin urmare, este utilă partiţionarea semnalului în foneme sau clase mari de foneme, adică structurarea fonetică a modelului fiecărui vorbitor.
În aplicaţiile de vorbire un rol important îl are adaptarea modelelor acustice la condiţiile noi de operare, deoarece există o varietate mare de vorbitori, de stiluri, medii de înregistrare [Gender].

Prin antrenarea unui GMM ne referim la estimarea parametrilor $\lambda = \{P_k, \mu_k, \Sigma_k\}^K{}_{k=1}$, pornind de la eşantionale de antrenare $=\{x_1, x_2, \ldots, x_T\}$, şi folosind estimarea probabilităţii maxime. Cu cât este mai mare valoarea acestei probabilităţi, cu atât putem afirma mai corect că vectorii necunoscuţi provin din modelul λ. Unul din algoritmii de bază este EM (din engleză Expectation-maximization), ce poate fi

folosit pentru a maximiza această probabilitate având anumite date iniţiale. Acesta porneşte iniţial de la o gaussiană învăţată pentru toate datele de antrenare, după care, interativ, gaussienele sunt împărţite şi antrenate până ajungem la numărul de componente dorit. În general, sunt suficiente cinci iteraţii pentru ca parametrii să conveargă, dar ne oprim în momentul în care câştigul de probabilitatea dintre două iteraţii este mai mic decât un anumit prag. În sistemele de recunoaştere independente de vorbitor, bazate pe GMM-uri, se foloseşte un model universal de fundal (universal background model-UBM) ce este antrenat pornind de la o bază de înregistrîri de zeci sau sute de ore, aparţinaând mai multor persoane. Modelul UBM este reprezentat de distribuţia vectorilor de coeficienţi, independent de vorbitor, iar în momentul în care un membru nou este introdus în sistem, parametrii acestui model sunt adaptaţi la trăsăturile sale. Modelul adaptat rezultat va fi folosit ca model al vorbitorului. Având datele necesare pentru a antrena un model universal, există mai multe abordări posibile:

- Prima metoda constă în unirea tuturor înregistrărilor și antrenarea lor cu un algoritm EM (Expectation-maximization), cu observaţia ca datele să fie repartizate echilibrat între subgrupe. De exemplu, dacă vrem un model independent de date este necesar ca înregistrările de voci feminine şi masculine să fie balansate, pentru a nu înclina spre direcţia dominantă.
- Cea de-a doua metodă se referă la antrenarea modelelor UBM pentru fiecare subgrupă în parte şi unirea modelului final. În acest caz, putem folosi şi înregistrări neechilibrate între grupuri.

Putem observa cele două abordări în figura 2.18 :

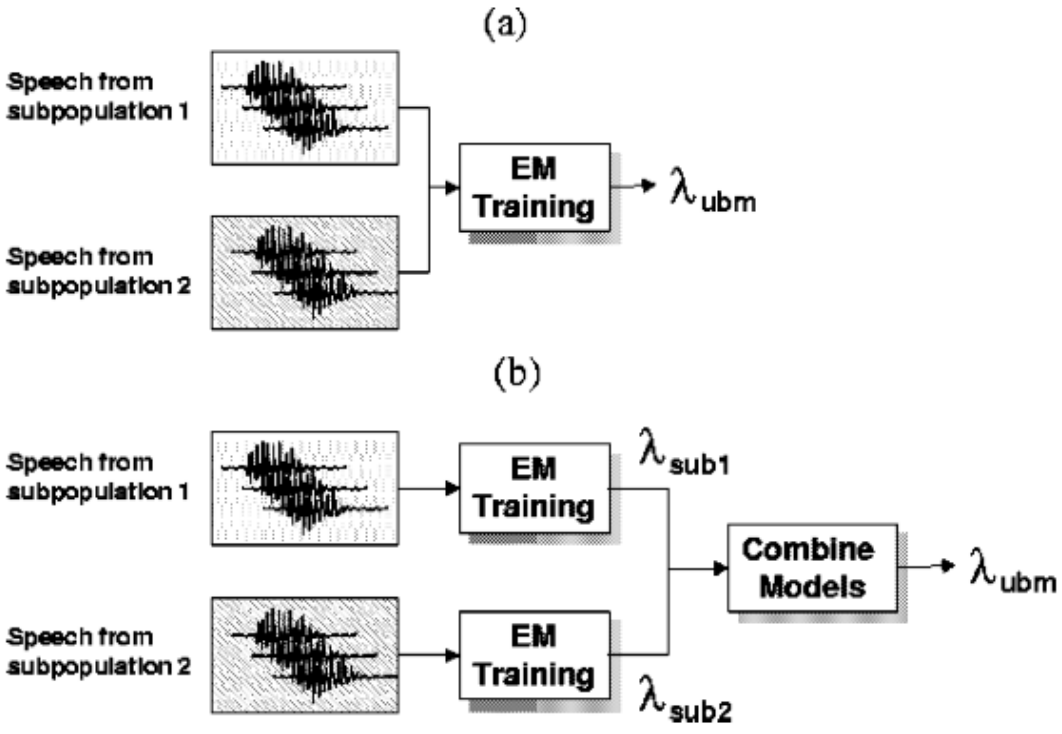


Figura 2.18 Abordările pentru crearea unui model UBM

Pentru aplicaţia noastră, este utilă antrenarea a două modele de fundal diferite: unul pentru vorbitorii feminini şi altul pentru cei masculini, situaţie în care vom antrena modelul noului venit în funcţie de genul său.

Putem adapta toţi parametrii sau doar o parte din ei pornind de la modelul de fundal.

Având eşantioanele $\chi=\{x_1, x_2,\ldots, x_T\}$ şi parametrii UBM (din engleza universal background model) $\lambda_{UBM} = \{P_k, \mu_k, \sum_k\}^K{}_{k=1}$, obţinem vectorii mediilor adaptate (μ_k`) cu metoda MAP (din engleza maximum a posteriori) ca o sumă ponderată a datelor de antrenare ale vorbitorilor şi media UBM:

$$\mu_k` = \alpha_k x_k+(1-\alpha_k)\mu_k \text{ ,unde} \quad (2.3.3)$$

$$\alpha_k = \frac{n_k}{n_k+r} \quad (2.3.4)$$

$$n_k = \sum_{t=1}^{T} P(k|x_t) \quad (2.3.5)$$

$$x_k=\frac{1}{n_k}\sum_{t=1}^{T} P(k|x_t)\, x_t \quad (2.3.6)$$

Pentru fiecare mixtură şi fiecare parametru s-a folosit un coeficient de adaptare, dependent de înregistrări α_k,cu $\mu_k$vectorul mediilor, r parametrul de legătură fxat si n_k un număr probabilistic.

În procesul de recunoaştere, modelul adaptat cu ajutorul algoritmului maxim a posteriori (MAP), ce implică derivarea unui model GMM specific vorbitorlui dintr-un model UBM, este asociat cu modelul UBM şi denumit în mod obişnuit *Gaussian mixture model-universal background model* sau modelul GMM-UBM.

Scorul de potrivire depinde atât de modelul ţintă (λ_{tinta}) cât şi de modelul de fundal (λ_{UBM}) prin intermediul logaritmului probabilităţii:

$$\text{Cost}(\chi, \lambda_{tinta}, \lambda_{UBM})=\frac{1}{T}\sum_{t=1}^{T}\{\log p(x_t \mid \lambda_{tinta})- \log p(x_t \mid \lambda_{UBM})\}$$

şi masoară diferenţa dintre cele două modele privind generarea observaţiilor $\chi=\{x_1, x_2,\ldots, x_T\}$;
Atunci când indentificarea vorbitorului se face într-un cadru strâns, se poate folosi doar GMM-uri fără a fi nevoie neapărat de modelul UBM, însă tehnica nu este la fel de precisă în modelarea trăsăturilor.

2.3.5 Decodarea

Problema decodării coincide cu problema recunoașterii. Deși la recunoaștere ne interesează doar modelul cel mai probabil, există situații (cum este cel al vorbirii continue) în care avem nevoie să cunoaștem întreaga secvența de stări. Prin vorbire continuă ne referim la cazul în care trebuie să găsim pentru o secvență de observații cea mai probabilă secvență de modele, necunoscând granițele între cuvinte. În acest scop, se utilizează algoritmul Viterbi, folosit de asemenea în capitolul 2.1.2.1 în sistemele de recunoaștere automată a vorbirii, unde înlocuim funcția de însumare cu una de maximizare. Acest algoritm a fost aplicat pentru prima oară în probleme de vorbire de Vintsyuk (1968). Fiecare stare a unui HMM [P01] reprezintă un cluster, iar funcțiile densității de probabilitate ale fiecarui astfel de cluster sunt modelate de GMM-uri. Procesul se rezumă la evaluarea modelelor HMM rezultate în urma transformării unui set de mixturi gaussiene, adică probabilitatea maximă de generare a secvenței de observații de la momentul 1 la momentul t:

$$\alpha_j(t)= \max_i\{\alpha_i(t-1)a_{ij}\}b_j(o_t) \quad (2.3.8), \text{ unde,}$$

$$\alpha_1(1)=1 \qquad (2.3.9)$$

$$\alpha_j(1)=a_{ij}\, b_j(o_1) \qquad (2.3.10)$$

a_{ij} este probabilitatea de a trece din stare i în stare j, iar $b_j(o_t)$ reprezintă probabilitatea să se emită simbolul o_t la starea j.

Relația care stă la baza algoritmului Viterbi este:

$$\mathcal{A}_j(t)= \max_i\{\mathcal{A}_i(t-1) + \log(a_{ij})\}+\log(b_j(o_t)) \qquad (2.3.11)$$

Acesta găsește drumul optim într-o rețea de noduri construită între ferestrele de timp și stările modelelor Markov, adică costul celei mai probabile căi (figura 2.19). Termenul $\log(b_j(o_t))$ din formula (2.3.11) reprezintă scorul nodului, iar termenul $\log(a_{ij})$ scorul tranziției.

Algoritmul este realizat astfel:

Pasul 1: Inițializarea

$$\mathcal{A}_1(1)= \pi_i b_i(o_1)$$

$$Bi(1) = 0 \qquad 1 \le i \le N$$

Pasul 2: Inducția

$$\mathcal{A}_j(t)= \max_i[\mathcal{A}_i(t-1)a_{ij}]b_j(o_t) \qquad 2 \le t \le T\,;\ 1 \le j \le N$$

$$B_j(t) = \arg\max_i[\mathcal{A}_i(t-1)a_{ij}] \qquad 2 \le t \le T;\ 1 \le j \le N$$

Pasul 3: Încheierea

$$\text{scorul cel mai bun} = \max(\mathcal{A}_j(t))$$

$$s_T^* = \arg\max_i B_i(t)$$

Pasul 4: Backtraking

$$s^*_t = B_{s^*t+1}(t+1) \quad t=T-1,T-2,\ldots,1$$

$S^*= (s^*_1,s^*_{2,\ldots}...s^*_T)$ este cea mai bună secvența de stări

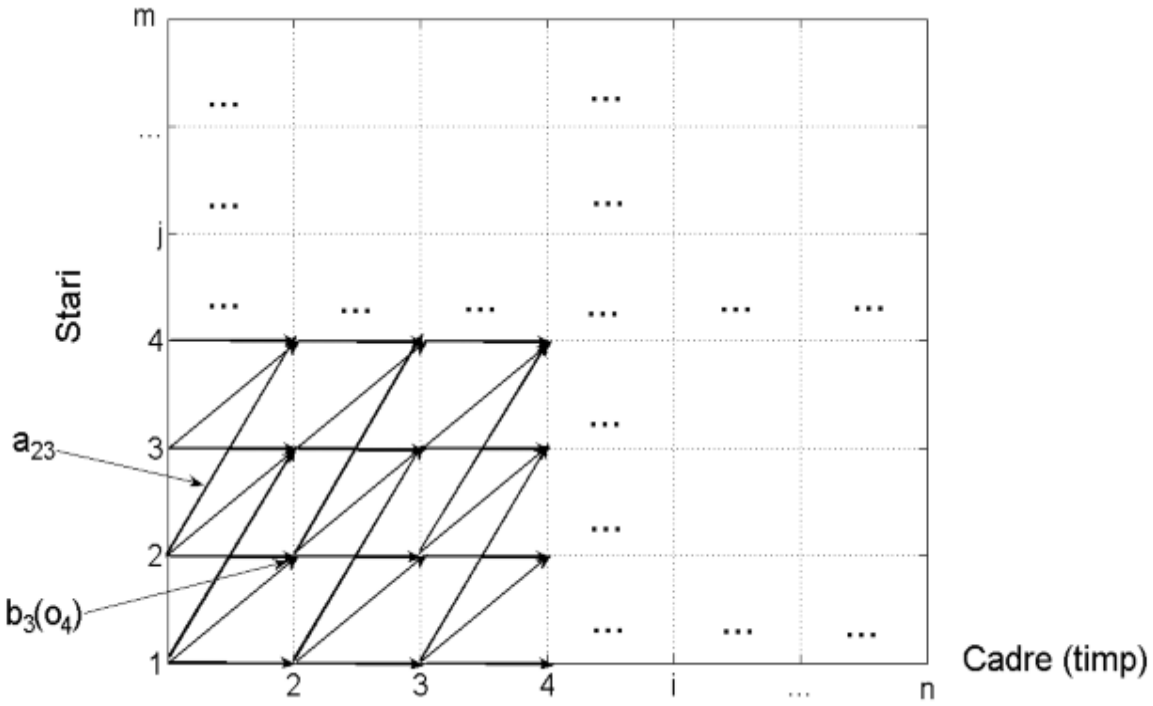


Figura 2.19 Algoritmul Viterbi [B01]

Întrucât foarte multe probabilități au valori mici, se aplică un logaritm asupra acestora ($\log P(w)$), ceea ce conduce la adunarea costurilor și nu înmulțirea probabilităților. În final, drumul va consta în alegerea celui mai mare cost și nu a celei mai mari probabilități.

Algoritmul Viterbi realizează implicit alinierea semnalului vocal cu modelele recunoscute. Rezultatul decodării va consta în găsirea secvenței de stări HMM ce a generat semnalul de voce și returnarea granițelor care indică segmentele ocupate de fiecare din cele patru modele în semnal vocal. Complexitatea acestuia depinde de numărul de stări pe care le au HMM-urile, de aceea se preferă soluția cu numărul de stări cel mai mic. Calea cea mai bună de-a lungul stărilor poate fi definița și printr-un proces de diarization ce folosește numele GMM-urilor pentru a înlocui numele grupurilor, iar stările aliniate reprezintă segmentele din fiecare grup.

2.3.6 Evaluarea performanțelor

Performațele sistemului de detecție a genului sunt măsurate printr-o asociere de una-la-unu între ID-urile vorbitorilor de referință și cei ipotetici. Putem analiza și performanțele proceselor intermediare ce compun sistemul final. Astfel, metrica de bază în cazul acțiunii de diarization o reprezintă rata de eroare DER (din engleză diarization error rate), suma erorilor de ratare, alarma falsă sau eroare a vorbitorilor. Pentru performanțele metodelor de clusterizare se analizează puritatea grupului, adică raportul între numărul de cadre ale vorbitorului dominant și numărul total de cadre din cluster și acoperirea grupului care se ocupă de dispersia vocii unui anumit vorbitor de-a lungul clusterelor.

CAPITOLUL 3

EXPERIMENTE

3.1. Recunoaşterea secvenţelor audio ce conţin cifre

3.1.1. Construcţia unui sistem de recunoaştere a vorbirii continue [ASR]

Sistemul de recunoaştere a clipurilor audio ce conţin cifre este un sistem de recunoaştere a vorbirii continue cu vocabular redus ce conţine doar zece cifre. Din figura 2.1 a arhitecturii generale a unui sistem de recunoaştere automată a vorbirii (RAV) reiese faptul că avem nevoie de două procese [Speech]:

a) extragerea de parametrii vocali din semnalul vocal (folosind mesajul vorbit) utilizaţi apoi în procesul de recunoaştere;

b) utilizarea unor modele (acustic, fonetic şi lingvistic) dezvoltate anterior.

Pentru cazul nostru, sistem de recunoaştere cu vocabular redus, se folosesc modelele de limbă de tip gramatical cu stări finite, deci nu avem nevoie de resurse textuale pentru a le forma.

La baza construcţiei unui asemenea sistem stau următorii paşi:

- **Înregistrarea unei baze de date de clipuri audio**

Pentru a forma modelul acustic avem nevoie de un set de clipuri audio înregistrate.Astfel am înregistrat pentru fiecare vorbitor diferit din cei 69 folositţi, 100 de clipuri audio a câte 12 cifre fiecare.

- **Crearea dicţionarului fonetic şi a listei de foneme**

Dicţionarul fonetic specifică modul în care sunt pronunţate cuvintele unei limbi. El va conţine, pentru sarcina noastră, cele zece cifre ale sistemului zecimal: zero, unu, doi, trei, patru, cinci, şase, sapte, opt, nouă.Sistemul de recunoaştere CMU Sphinx foloseşte dicţionarele fonetice în format text , cu câte un cuvânt pe fiecare linie, însă transcrierea fonetică a unui cuvânt apare pe aceeaşi linie. Fonemele le scriem separate prin spaţiu. Lista de foneme se crează prin scrierea tuturor fonemelor ce vor fi modelate de modelul acustic pe câte o linie, în ordine alfabetică.

- **Antrenarea modelului acustic**

Necesită existenţa următoarelor resurse:
-baza de înregistrări audio ce conţine vorbire;
-dicţionarul fonetic cu toate cuvintele;
-un dicţionar cu elemente ce nu sunt foneme (muzică, tuse, râset etc).

Se crează o listă cu clipurile audio ce vor fi folosite în procesul de antrenare şi un fişier cu transcrierea textuală a acestor înregistrări.Vom folosi primele 50 şi ultimele 30 din clipurile audio ale fiecărui vorbitor.Pentru orice proiect CMU Sphinx există un fişier de configurare unde modificăm parametrii modelului acustic şi locaţia resurselor necesare.Avem nevoie apoi de coeficienţii MFCC corespunzători clipurilor de antrenare.Astfel creăm 80 de fişiere cu coeficienţi cepstrali pentru fiecare vorbitor şi antrenăm modelul.

- **Crearea modelul de limbă (gramaticii)**

Vom construi pentru o sarcină de transcriere a secvenţelor audio ce conţin cifre o gramatică cu stări finite.Presupunem că vorbitorul poate rosti numai cuvintele zero, unu, doi, trei, patru, cinci, şase, şapte, opt, nouă.Definirea grafului va începe cu definirea nodurilor de intrare şi ieşire din gramatică, notate cu N (null), deoarece trecerea prin ele nu generează afişarea niciunui cuvânt.Creăm apoi nodurile ce corespund cuvintelor şi le legăm de nodurile de intrare şi ieşire cu arce direcţionate.A fost creată o gramatică dacă se pronunţă o cifră o singură dată.Adaugăm o tranziţie de la nodul de ieşire la cel de intrare şi încă două stări de null, pentru ca celelalte două acum au arce care intră şi ies din ele.

Figura 3.1 prezintă grafic gramatica cu stări finite pentru sarcina propusă. Modelul e format din 14 noduri, dintre care 10 reprezintă cuvintele (cifrele de la zero la nouă).Tranziţiile arată modul în care se parcurge gramatica şi ce este permis ca secvenţe de cuvinte:

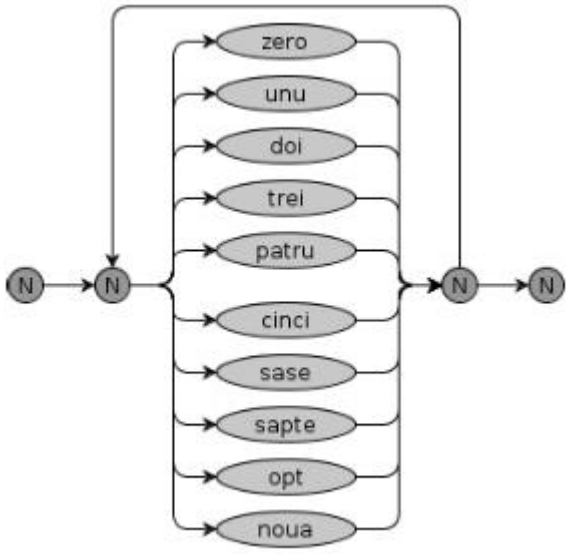


Figura 3.1 Gramatica cu stări finite a sarcinii de recunoaştere de cifre

Cifrele şi succesiunile de cifre din înregistrările noastre apar cu probabilităţi aproape egale şi din această cauză se foloseşte gramatica cu stări finite (FSG-Finite State Grammar) ce se poate implementa cu ajutorul formatului Java Speech Grammar (JSGF) [JSGF].

```
#JSGF V1.0;
grammar rodigits;
public<numbers>=(zero|unu|doi|trei|patru|cinci|şase|şapte|opt|no
uă)*;
```

Tranziţiile, nodurile şi probabilităţile de tranziţie ale gramaticii cu stări finite sunt specificateîn formatul intern folosit de CMU Sphinx, FSG astfel:

```
FSG_BEGIN <rodigits.numbers>
NUM_STATES 18
START_STATE 0
FINAL_STATE 1
TRANSITION 0 2 1.000000
TRANSITION 2 4 0.500041
TRANSITION 2 5 0.500041
TRANSITION 3 1 1.000000
TRANSITION 4 3 1.000000
TRANSITION 5 7 0.100000 nouă
TRANSITION 5 8 0.100000 opt
TRANSITION 5 9 0.100000 şapte
TRANSITION 5 10 0.100000 şase
TRANSITION 5 11 0.100000 cinci
TRANSITION 5 13 0.100000 patru
TRANSITION 5 14 0.100000 trei
```

```
TRANSITION 5 15 0.100000 doi
TRANSITION 5 16 0.100000 unu
TRANSITION 5 17 0.100000 zero
TRANSITION 6 2 1.000000
TRANSITION 6 3 1.000000
TRANSITION 7 6 1.000000
TRANSITION 8 6 1.000000
TRANSITION 9 6 1.000000
TRANSITION 10 6 1.000000
TRANSITION 11 12 1.000000
TRANSITION 12 6 1.000000
TRANSITION 13 6 1.000000
TRANSITION 14 6 1.000000
TRANSITION 15 6 1.000000
TRANSITION 16 6 1.000000
TRANSITION 17 6 1.000000
FSG_END
```

- **Evaluarea sistemului de recunoaştere şi interpretarea rezultatelor**

Având la dispoziţie cele 3 componente fundamentale ale unui sistem de recunoaştere (modelul acustic, modelul de limbă şi modelul fonetic) putem decoda clipurile audio de evaluare şi comparare a transcrierii textuale rezultate cu transcrierea textuală de referinţă în vederea evaluării sistemului de recunoaştere de secvenţe audio ce conţin cifre.

După rularea procesului de decodare vom obţine rezultatele pentru fiecare configuraţie în parte.Raportul de evaluare afişează pe fiecare linie rezumatul rezultatelor pentru diverşi vorbitori din baza de date de evaluare, iar pe coloane avem informaţii precum: id-ul vorbitorului, numărul propoziţiilor şi a cuvintelor de evaluare, procentul de cuvinte substituite/ şterse/ inserate, procentul de cuvinte recunoscute corect, rata de eroare la nivel de propoziţie, precum şi la nivel de cuvânt.
La secţiunea Sentence Recognition Performance se oferă mai multe detalii cu privire la erorile la nivel de propoziţie, iar la secţiunea Word Recognition Performance se oferă mai multe detalii cu privire la erorile la nivel de cuvânt.Următoarele cinci secţiuni prezintă statistici despre perechile de cuvinte confundate cel mai des (Confusion Pairs), cuvintele cele mai frecvent inserate (Insetions), şterse (Deletions), substituite (Substitutions) şi cuvintele recunoscute fals cel mai des (Falsely Recognized).
În final, sunt prezentate în raport, frazele ipotetice aliniate de la frazele de referinţă corespunzatoare, precizând tipurile de erori pentru fiecare frază.

- **Optimizarea modelului acustic**

Modelul acustic creat de noi cu CMU Sphinx este construit cu trifoneme (fonem căruia i se precizează vecinii din stânga şi dreapta), fiecare trifonem fiind implementat ca HMM cu trei stări emisive.Fiecare astfel de model-stare poartă denumirea de senone şi poate fi comună mai multor trifoneme în funcţie de

asemănarea dintre ele.Modelul în stare inițială are numărul de senone egal cu 200, iar numărul de densităţi de probabilitate pentru fiecare GMM egal cu 8.

Pentru a obţine acest model acustic cu 200 de senone şi 8 densităţi Gaussiene per senone s-a trecut prin mai multe etape ce includ:

-modele fonetice independente de context;
-modele fonetice dependente de context cu o singură densitate Gaussiana per senone, apoi cu 2 şi respectiv 4 densităţi;

Am configurat procesul de decodare pentru a folosi pe rând fiecare din cele 4 modele şi am centralizat rezultatele astfel încât să avem o evaluare finală şi comparativă a diverselor configuraţii.

Pentru decodarea clipurilor audio de evaluare folosind metode fonetice independente de context, modificăm fişierul de configurare al proiectului, specificând alte modele acustice.Am folosit modelele cu 200 de senone şi am variat numărul de densităţi gaussiene per senone.

Pentru a verifica dacă numărul de 200 de senone este optim sau nu pentru sarcina noastră de recunoaştere a secvenţelor audio ce conţin cifre, vom antrena un model acustic cu un număr de 100 de senone folosite la modelarea fonemelor ce formează cuvintele. Variem şi acum numărul de densităţi Gaussiene şi evaluăm 4 modele acustice.

Rezultate privind evaluarea sistemului dependent de vorbitor

Tabel 3.1 Ratele WER obținute pentru un sistem dependent de vorbitor

WER[%]		Densităţi Gaussinene per senone			
		1	2	4	8
SENONE	100	2.5	1.7	1.7	1.3
	200	6.3	7.5	15	22.8

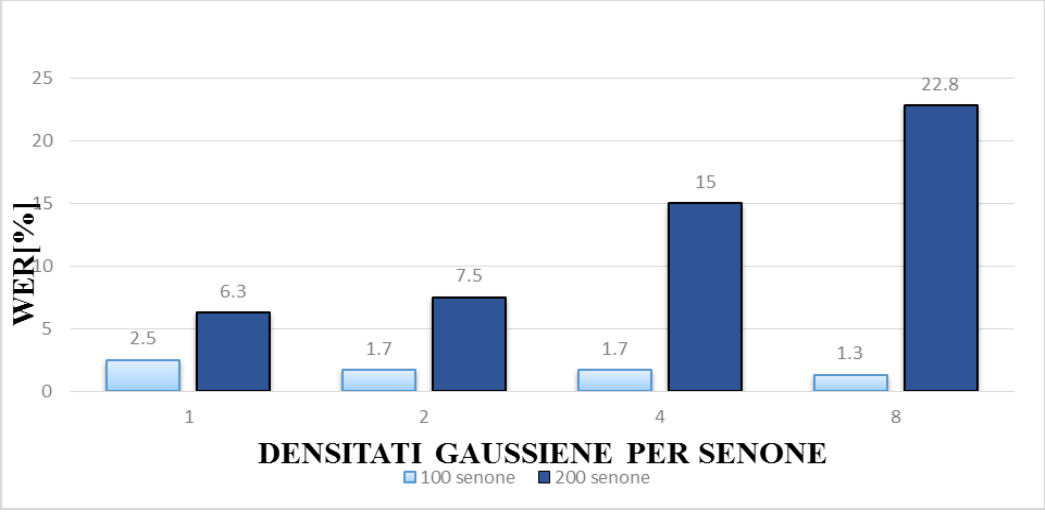


Figura 3.2 Reprezentarea grafică a ratelor WER obținute pentru un sistem dependent de vorbitor

Tabel 3.2 Ratele SER obținute pentru un sistem dependent de vorbitor

SER[%]		Densităţi Gaussinene per senone			
		1	2	4	8
SENONE	100	20	10	10	10
	200	30	40	60	75

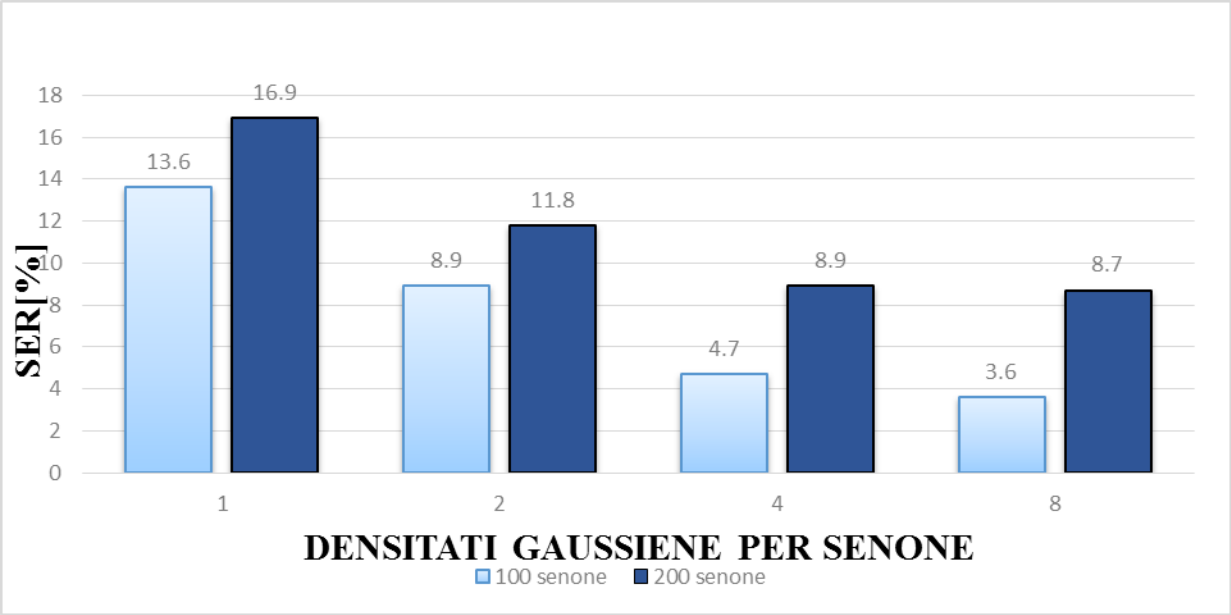


Figura 3.3 Reprezentarea grafică a ratelor SER obținute pentru un sistem dependent de vorbitor

Pentru un sistem dependent de vorbitor, dar independent de context am obținut o rata de eroare pe cuvânt de 3.3% şi o rata de eroare pentru propoziții SER de 30%.

3.1.2 Construcţia unui sistem de recunoaştere independent de vorbitor

Necesitatea unui astfel de sistem reiese din erorile foarte mari obţinute în urma evaluării sistemului creat anterior pe fişierele de evaluare ale altor colegi.Rezultatele sunt de înţeles, deoarece modelul acustic din proiect a fost antrenat doar cu vocea noastra şi cea mai bună recunoaştere o va face tot pentru noi.Am decis să creăm un alt proiect, independent de vorbitor în acest caz, folosind nu doar clipurile audio înregistrate de noi, ci toate clipurile din baza de date.

Pentru crearea acestuia ne-am folosit de :

a) pentru antrenare: 80 de fişiere audio de la 69 de vorbitori (5520 fişiere audio);
b) pentru testare: 20 de fişiere audio de la aceiaşi 69 de vorbitori (1380 fişiere audio);

Urmăm aceiaşi paşi parcurşi în cazul anterior, dependent de vorbitor, cu un număr de 200 şi respectiv 100 de senone.Pentru fiecare situaţie folosim metode fonetice independente de context şi dependente cu 1/2/4/8 densităţi Gaussiene per senone.

Rezultate privind evaluarea sistemului independent de vorbitor (vorbitori cunoscuţi)

Tabel 3.3 Ratele WER obținute pentru un sistem independent de vorbitor

WER[%]		Densităţi Gaussinene per senone			
		1	2	4	8
SENONE	100	13.6	8.9	4.7	3.6
	200	16.9	11.8	8.9	8.7

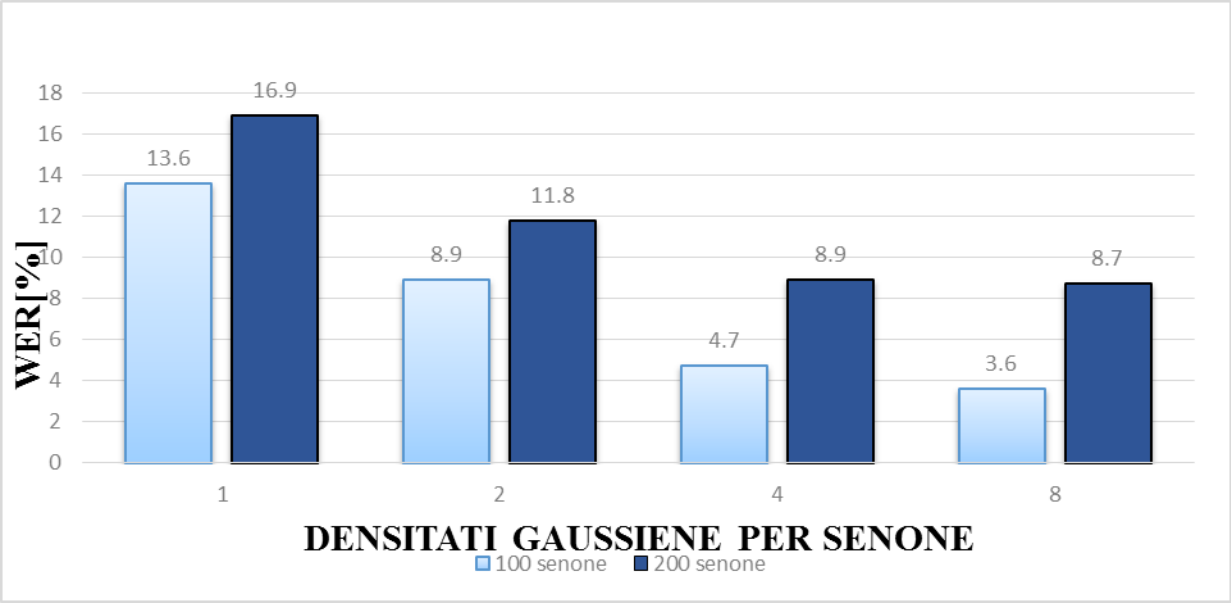


Figura 3.4 Reprezentarea grafică a ratelor WER obținute pentru un sistem independent de vorbitor

Tabel 3.4 Ratele SER obținute pentru un sistem independent de vorbitor

SER[%]		Densităţi Gaussinene per senone			
		1	2	4	8
SENONE	100	57.8	42.5	27	21.7
	200	67.2	56	48	46.4

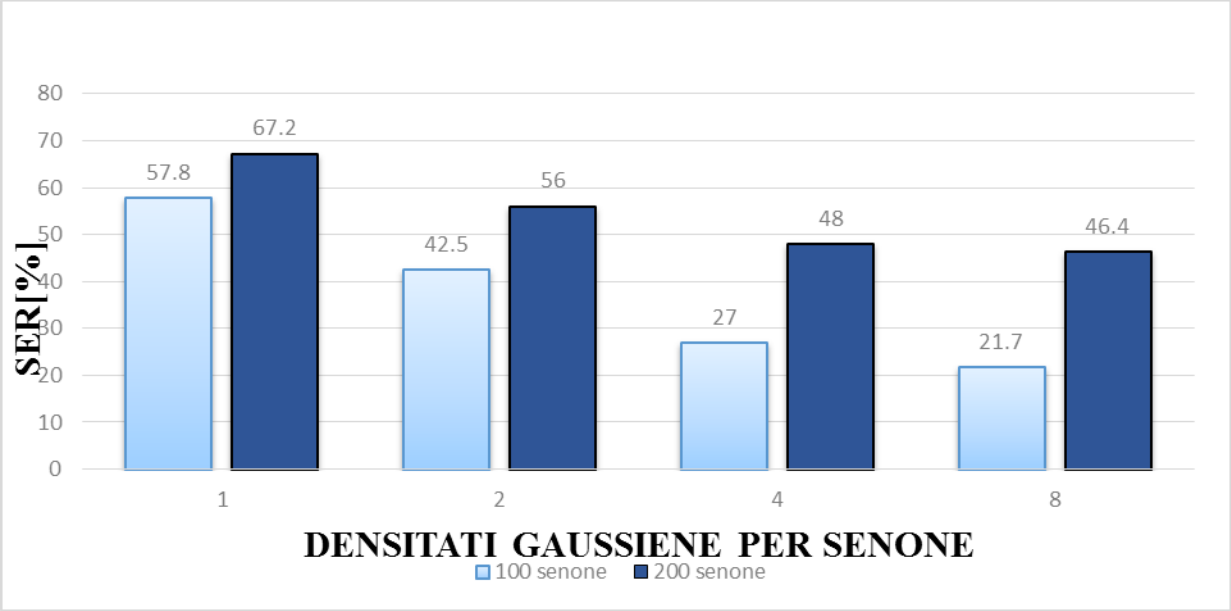


Figura 3.5 Reprezentarea grafică a ratelor SER obținute pentru un sistem independent de vorbitor

Comparând rezultatele obţinute, observăm un trend de scădere al erorii pentru 100 de senone şi decidem să extindem numărul densităţilor Gaussiene, trecând prin 16, 32, 64 şi în final 128.

Tabel 3.5 Ratele WER obținute pentru un sistem independent de vorbitor, 100 de senone si un număr mai mare de densități gaussiene

WER[%]		Densităţi Gaussinene per senone			
		16	32	64	128
SENONE	100	3.5	2.9	2.4	2.4

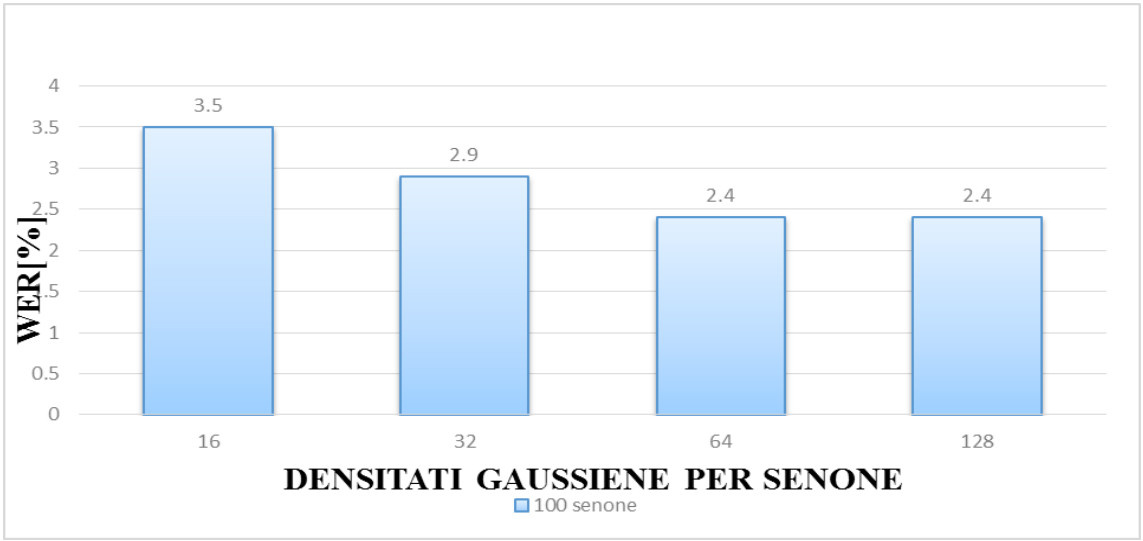


Figura 3.6 Reprezentarea grafică a ratelor WER obținute pentru un sistem independent de vorbitor, pentru 100 de senone

Tabel 3.6 Ratele SER obținute pentru un sistem independent de vorbitor, 100 de senone si un număr mai mare de densități gaussiene

SER[%]		Densități Gaussinene per senone			
		16	32	64	128
SENONE	100	21.8	17.9	17.2	16.7

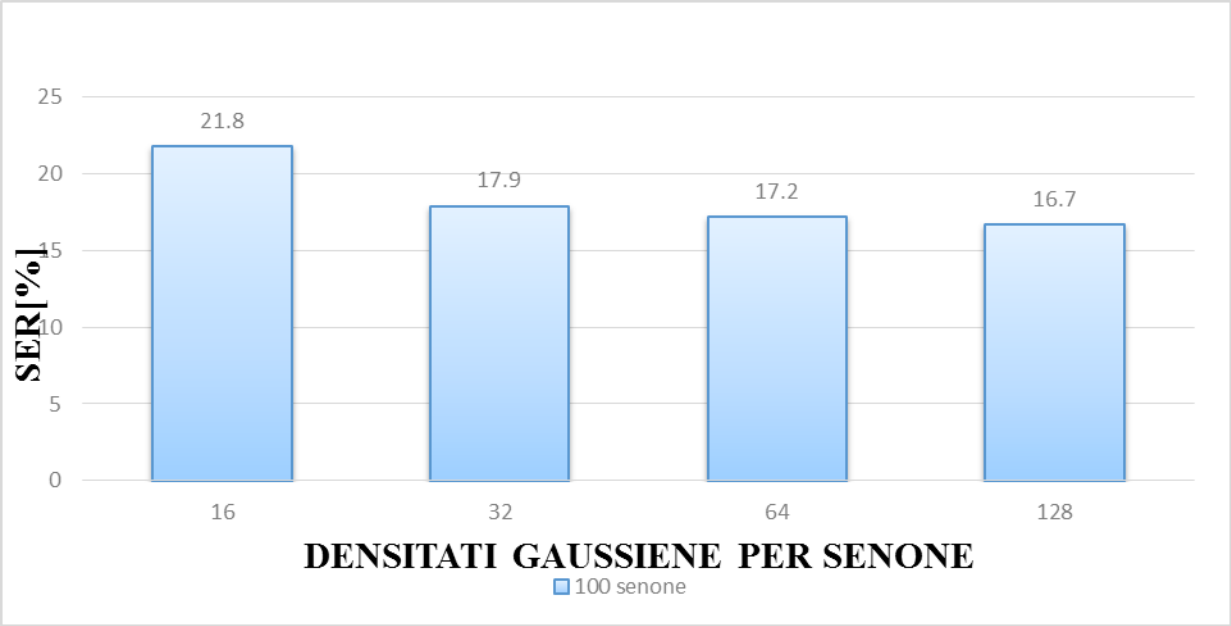


Figura 3.7 Reprezentarea grafică a ratelor SER obținute pentru un sistem independent de vorbitor, pentru 100 de senone

3.1.3 Concluzii

În urma evaluărilor, am selectat modelul optim ca fiind cel cu 128 densităţi Gaussiene şi 100 de senone. Numărul mare al densităților Gaussiene oferă o aproximare mai bună a modelului.

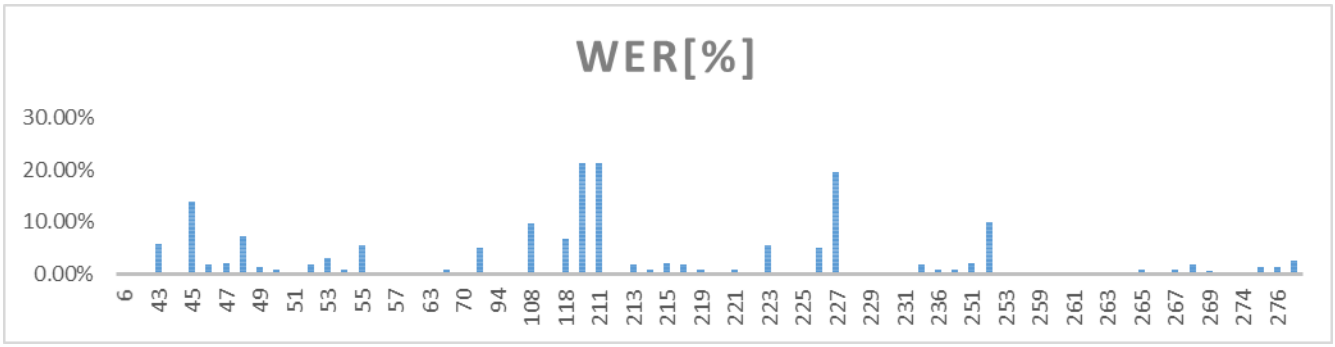


Figura 3.8 Reprezentarea grafică a ratelor WER pentru diverşi vorbitori, obținute pentru un sistem independent de vorbitor

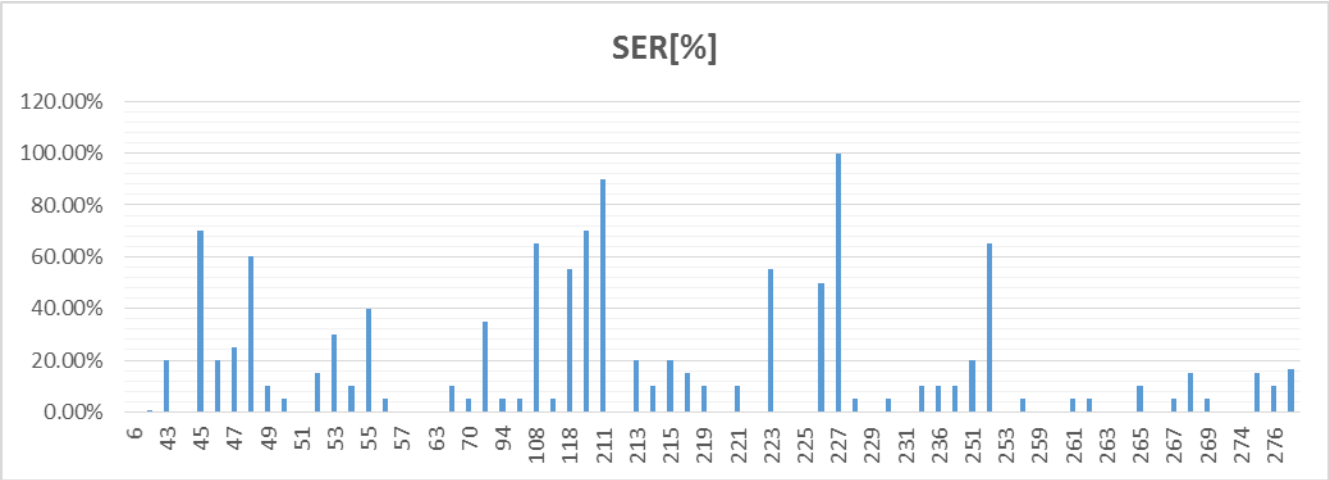


Figura 3.9 Reprezentarea grafică a ratelor SER pentru diverşi vorbitori, obținute pentru un sistem independent de vorbitor

Pentru obţinerea unui rezultat bun într-un sistem bazat pe modele Markov ascunse cu densităţi continue, este necesar să folosim îmbinarea dintre distribuţiile Gaussiene şi modelele(fonetice) dependente de context.

3.2 MediaEval 2013: Sarcina de căutare a cuvintelor vorbite (Spoken Web Search)

Atribuţiile pentru MediaEval 2013 au constat în căutarea unui cuvânt rostit în interiorul unei baze de înregistrări, de asemenea rostite.

Procesul s-a bazat pe două seturi ce conţineau înregistrări multilingve: un set cu exemple de termeni căutaţi (unul sau mai multe exemple pentru un termen) şi un set de documente audio unde vor avea loc căutarile şi a implicat ca fiecare apariţie a unui cuvânt cerut să fie identificată. Întrucât atât termenii cât şi înregistrările audio pot conţine diferite limbi, sistemele de căutare trebuie să fie independente de limbă.

Fişierele audio din baza de date a proiectului au fost înregistrate în diverse limbi (engleză, română, albaneză, cehă, bască, precum şi 4 limbi africane) cu accente şi condiţii acustice diferite (unele realizate în cameră cu ajutorul microfoanelor, în timp ce altele pe stradă cu telefonul mobil). Dimensiunea bazei de înregistrări este în jur de 20 ore, iar numărul de cuvinte căutate este de aproximativ 400, ele fiind proporţionale cu numărul de limbi utilizate.

3.2.1 Abordarea sarcinii

Abordarea acestei sarcini are la bază sistemul de recunoaştere automată a vorbirii pentru limba română, dezvoltat anterior.

Conform arhitecturii sistemului propusă de NIST [STD], avem două componente: o etapă de indexare, ce rulează transparent faţă de utilizator şi una de căutare. În faza de indexare folosim sistemul ASR, ce preia o înregistrare şi returnează un şir de foneme, reprezentând transcrierea fonetică a acesteia. Pasul

următor, acela de a găsi termenii daţi se realizează prin căutarea transcrierii acestora în transcrierea fonetică a bazei de înregistrări. Scopul este acela de a găsi locurile în care asemănarea dintre şirurile de foneme (ale cuvintelor şi bazei avute) depăşeşte un anumit prag.

3.2.2 Sistemul de recunoaştere automată a vorbirii (ASR) pentru limba română

Modelul acustic pentru acest sistem a fost creat având la bază 64 ore de înregistrări apartinând unor diversi vorbitori. Cea mai bună performanţă a fost o rată de eroare a cuvintelor (WER) de 18% pentru un discurs clar şi continuu, în cazul în care a fost folosit un model de limbă antrenat cu 170 milioane de cuvinte, bazat pe N-grame (rezultatul a fost preluat din experimenţele anterioare realizate de echipa de cercetare).

Deoarece înregistrările sunt aleatorizate, nu a fost furnizată nici o informaţie privind limba vorbită în fisier. Pentru a reduce neconcordanţa dintre aceasta bază de înregistrări şi baza de antrenare a sistemului ASR, am filtrat înregistrările în limba română la 8kHz şi am reantrenat modelul acustic.
Baza de înregistrări avută nu conţine niciun tip de transcriere. Din acest motiv, este mai fezabil să folosim recunoaşterea de cuvinte, însă folosim recunoaşterea de foneme. Limba română are doar 26 foneme, deşi numărul mare de limbi conţinute în înregistrări triplează acest număr. Deoarece multe sisteme de recunoaştere automată a vorbirii folosesc clusterizarea fonemelor prin gruparea acestora în clase, în funcţie de similitudinea acustică, putem avea mai puţine clase decât foneme.

Pentru realizarea acestui lucru folosim modelele Markov ascunse (HMM) cu 3 stări per foneme, cu 4000 de senone, 8 densităţi Gaussiene, cu parametrii de voce MFCC. Modelele au fost antrenate să dividă spațiul trăsăturii vorbirii în clase de foneme, iar fonemele care nu aparțin limbii romane vor fi clasificate în una din aceste clase. Daca toate aceste instanțe, considerate ca neaparținând fonemelor din limba română ar fi recunoscute ca foneme românești, atât baza de înregistrări cât și cuvântul căutat ar avea acelaşi simbol. Rezultatele experimentale arată însă contrariul, adică faptul că multe foneme ocupă o regiune a vorbirii ce se suprapune doar parțial cu subspațiile fonemelor limbii române. Pentru baza de date a sistemului ASR din limba română avem o rată de eroare a fonemelor (PhER) de 36.8% (testată pe date în limba română).

3.2.3 Metodologia căutării

Algoritmul DTWSS (DTW String Search) foloseşte DTW pentru a alinia cuvântul căutat în conţinut. Deoarece lungimea înregistrării o depăşeşte pe cea a query-ului, căutarea trebuie să se realizeze doar în partea proporţională cu lungimea query-ul căutat numită “fereastra glisantă”, în care noi am împărţit înregistrarea. Aceaste ferestre nu pot fi oricât de mari, deoarece fonemele unui termen căutat pot fi eventual găsite în interiorul ferestrei, deşi este o potrivire falsă. Cuvântul este considerat găsit atunci când scorul algoritmului DTW depăşeşte un anumit prag.

Pentru fiecare fereastră, alinierea la un scor s, complementar costului (egal cu valoarea PhER) :

$$S = (1\text{-PhER}) \quad (1)$$

unde s este scorul de similitudine, atunci când detecţia se bazează pe un prag determinat empiric. Procesul de segmentare a înregistrărilor în ferestre glisante şi marcajele de timp fac procesul de

localizare a unui cuvânt posibil. Această metodă trebuie să ia în calcul şi query-urile foarte scurte şi implicit răspândirea potrivirilor DTW în context. Formula poate îmbunătăţi căutarea, dacă este folosită sub formă:

$$s = (1 - PhER)(1 + \alpha \frac{L_Q - L_{Qm}}{L_{QM} - L_{Qm}})(1 + \beta \frac{L_W - L_S}{L_Q}) \quad (2)$$

unde, L_Q este lungimea cuvântului căutat, L_{QM}=17 şi L_{Qm}=4 sunt lungimile maximă respectiv minimă a unui query găsit în setul de date, L_W este lungimea ferestrei glisante, α şi β sunt parametrii configurabili, L_S este lungimea termenului găsit în înregistrare.

Penalizările în formula (1) sunt date de presupunerea că două cuvinte de lungimi diferite se pot potrivi într-un anumit conţinut prin aceeaşi valoare PhER, iar query-ul cu lungimea cea mai mare este mai probabil să fie cel corect. Ponderea de raspândire a potrivirilor în cazul algoritmului DTW se explică prin faptul că, cu cât sunt mai compact aliniate fonemele, cu atât este mai mare probabilitatea ca ele să aparţină aceluiaşi termen. Introducerea parametrilor ajustabili (α şi β) reprezintă o abordare nouă, valoarea lor optimă fiind obţinută empiric.

3.2.4.Rezultate experimentale

Rezultatele obţinute cu metoda primară, algoritmul DTW, pe setul de înregistrări de evaluare sunt arătate în Tabelul 1. Evaluarea se bazează pe valuarea ATWV care ia în calcul şi locaţia cuvântului căutat. Rezultatele au fost obţinute pentru o lungime a ferestrei glisante mai mare de 1.5 ori decât cea a cuvântului căutat. Cu cât este mai scurt query-ul, cu atât cresc şansele ca, cuvinte diferite, dar similare să obţină scoruri mai mari.

Tabel 3.7 Rezultatele algoritmului DTWSS

ATWV	α=0	α=0.2	α=0.4	α=0.6	α=0.8	α=1	α=1.2
β=0	0	0.009	0.026	0.06	0.094	0.106	0.105
β=0.2	0	0.009	0.026	0.061	0.094	0.106	0.105
β=0.4	0	0.009	0.026	0.064	0.094	0.107	0.106
β=0.6	0	0.009	0.027	0.065	0.096	0.107	0.105
β=0.8	0	0.009	0.028	0.067	0.096	**0.108**	0.105
β=1	0	0.009	0.031	0.07	0.097	**0.108**	0.105
β=1.2	0	0.009	0.031	0.071	0.098	0.107	0.105

Experimentele au fost făcute pentru a determina valoarea optimă a parametrilor α şi β, pentru metoda de căutare DTWSS, pornind de la formula 1(α=0; β=0) am determinat baza. Se observă că este mai bine să îi dăm parametrului α o valoare mai mare. În mod similar, cu cât este mai mare raspândirea potrivirii dată de DTW, cu atât este mai mică probabilitatea ca acela să fie cuvântul căutat. Există totuşi o valoare optimă pentru α şi β, explicate pe baza că, crescând α peste o anumită valoare, cuvintelor scurte li se alocă scoruri mari doar pentru că sunt de lungime mică, deşi ele ar putea avea o valoare

PhER mare. Similar, o valoare mare a lui β înseamnă că potrivirile DTW vor primi un scor mai mare deşi scorul lor real DTW (1-PhER) este mai mic.
Am ales următoarele combinaţii (α, β): (1; 0.8), (1, 1).

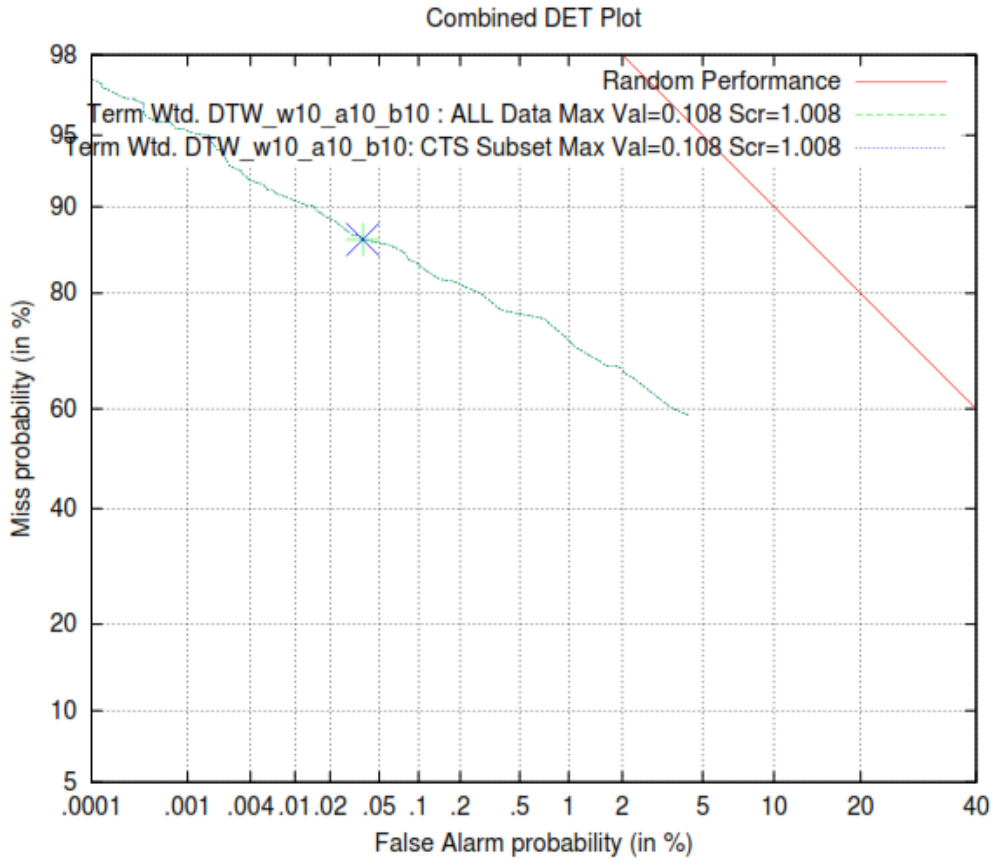


Figura 3.10 Rezultatele rulării principale cu baza de evaluare

Figura 3.9 arată curbele erorii de detectare a compromisului (DET) folosind metoda DTWSS, obţinută pentru baza de înregistrări de evaluare.

3.2.5 Concluzii:

Sistemul de recunoaştere a termenilor (STD) s-a conturat ca un process în doi paşi. Am folosit un ASR în limba română pentru recunoaşterea de foneme, în cazul indexării bazei de înregistrări, în timp ce algoritmul DTW se utilizează în căutarea unui cuvânt dat. Rezultatele se îmbunătaţesc dacă scorul deciziei este reglat în concordanţă cu lungimea cuvântului şi raspândirea de potriviri. Metoda aleasă este scalabilă pentru baze mari de date, deoarece am redus căutarea la o comparaţie de siruri de foneme.

CAPITOLUL 4

APLICAŢII DEMONSTRATIVE

4.1 Recunoaştere de cifre

4.1.1 Descriere

Aplicaţia demo de recunoaştere de cifre în limba română este o aplicaţie în Java ce preia cifrele rostite de utilizator, cu ajutorul unui microfon şi le afişează textual în interfaţa grafică, cum este prezentat în figura 4.1.

Modelul acustic şi modelul lingvistic folosite pentru aceasta aplicaţie sunt cele create anterior, care au dat cele mai bune rezultate de recunoaştere, în urma testării a diverse configuraţii.

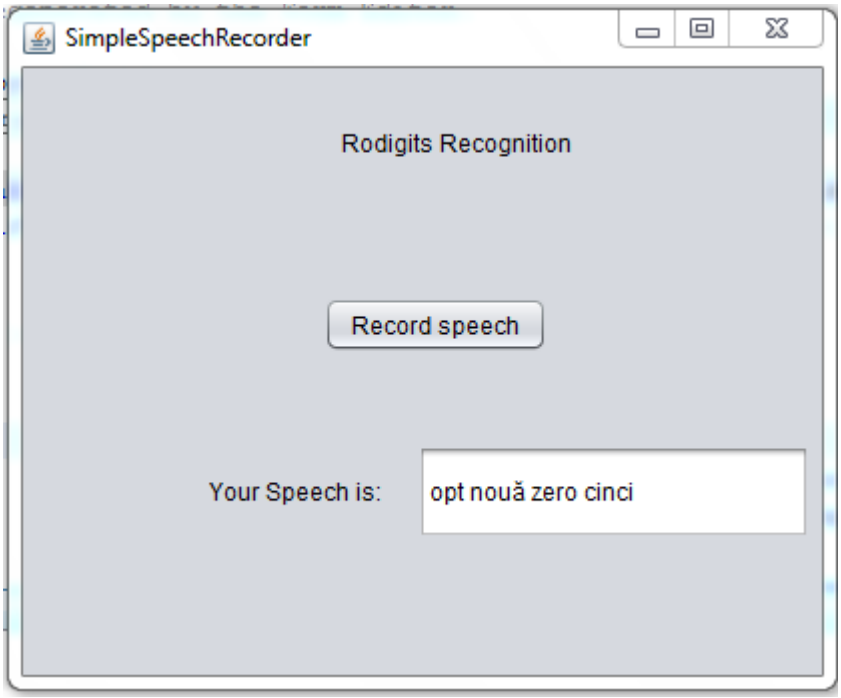


Figura 4.1 Interfața aplicației demonstrative

4.1.2 Interfaţa utilizatorului

Aplicaţia are o interfaţă grafică simplă construită cu ajutorul cadrului de lucru swing javax. Prin apăsarea butonului „Record speech" se începe înregistrarea utilizatorului, textul butonului se schimbă în „Stop recording", iar în momentul în care apăsăm pe „Stop recording", oprim înregistrarea şi începe procesul de recunoaştere. Programul detectează automat sfârşitul mesajului rostit şi face recunoaşterea de cifre, dar este recomandată apăsarea butonului de stop pentru a diminua zgomotele din jur, ce pot influenţa negativ rezultatele recunoaşterii. După ce recunoaşterea este terminată, câmpul de text „Your Speech is:" va afişa şirul de cifre depistat. Se pot face mai multe recunoaşteri consecutive prin reapăsarea butonului de înregistrare (Record speech), moment în care caseta de afişare se goleşte şi procesul se reia.

4.1.3 Implementare

Pentru dezvoltarea proiectului am folosit uneltele puse la dispoziţie în mod gratuit de CMU Sphinx, ce oferă posibilitatea de a crea sisteme de recunoaştere a vorbirii continue, cu un vocabular larg, independente de vorbitor și mediul de dezvoltare Netbeans IDE. Sphinx-4 conţine un set de clase ce folosesc Java Speech API (JSAPI) ca un motor în recunoaşterea de vorbire. NetBeans reprezintă un suport integrat pentru dezvoltarea proiectelor ce includ o interfață grafică cu utilizatorul și oferă multe facilități.

Pentru început, am setat calea spre atributele necesare sarcinii noastre, specificând modelul acustic, dicţionarul creat, fişierul cu gramatica şi sursa de unde va proveni mesajul vorbit. Primele trei atribute au fost setate folosind obiectul de configurare, care a fost apoi pasat recunoscătorului de cifre.

Recunoaşterea s-a bazat pe înregistrarea live a vocii utilizatorului cu ajutorul unui microfon şi pentru a realiza acest lucru am folosit o clasă a librariei Sphinx-4 numită LiveSpeechRecognizer împreună cu metodele sale. După ce microfonul este pornit, programul încearcă să recunoască ceea ce utilizatorul spune (cu metoda recognizer.startRecognition(true)). Cifrele recunoscute prin apelul metodei recognizer.getResult() sunt afişate în caseta de text corespunzătoare.

Pentru a putea realiza recunoaşterea în mod transparent utilizatorului şi a permite ca interfaţa să rămână activă, astfel încât să putem realiza şi alte acţiuni în acest timp, se folosesc mai multe fire de execuţie. SwingWorker este o clasă abstractă, ideală pentru a detaşa unele sarcini de aplicaţie şi doar să putem monitoriza statusul lor sau să le putem închide când dorim. Fiecare sarcină a unui fir de execuție este reprezentată de o instanță a lui javax.swing.SwingWorker.

Se pot furniza trei tipuri diferite de fire de execuție:

- Firul curent în cadrul căruia este apelată metoda execute( ), ce programează un SwingWorker pentru a executa un fir și returnează răspunsul imediat;
- Firul lucrător (din engleza Worker) ce apelează doInBackground ( ) și se ocupă de toate activitățile din fundal. Cu această metodă putem actualiza progresele aplicației.
- Event Dispatch Thread (EDT): toate activitățile Swing legate au loc aici și se folosesc metodele process și done( ).

Am utilizat metoda doInBackground( ) , în interiorul căruia pornim procesul de recunoaştere a cifrelor şi recepţionăm rezultatele. Pentru pornirea execuției codului apelăm metoda execute( ). În momentul în care utilizatorul apasă pe butonul de înregistrare, SwingWorker-ul începe recunoaşterea în fundal, apoi eliberează firul de execuţie pentru a împiedica blocarea aplicaţiei. Când este finalizată operația din fundal, automat este invocată metoda done( ) pentru a putem recupera rezultatul. Apelul metodei done() se poate face doar în firul de fundal numit event dispatching thread, singurul fir valid ce poate modifica starea componentelor de pe interfața grafică.

Atunci când calculul este terminat şi rezultatul este disponibil, îl preluăm cu ajutorul metodei getResult() și astfel, putem să îl afișăm pe interfață.

4.1.4 Concluzii

Scopul principal al acestei aplicații demo a constat în a vedea cum sunt recunoscute secvențele de cifre rostite, indiferent de utilizator sau de modul său de pronunție și beneficiile pe care o astfel de aplicație le poate aduce tehnologiei. Există, bineînțeles, numeroase dezvoltări ce pot mări gradul de utilizare și complexitate al acestui sistem.

4.2 Detecția de gen

4.2.1 Descriere

Aplicaţia demo de detectare de gen este un program scris în limbajul Java ce preia o înregistrare audio a utilizatorului şi returnează genul acestuia. O populație de indivizi poate fi partiționată în diverse subcategorii pe baza mai multor criterii, lucru ce interesează din multiple motive. Detecția automată a

genului prezintă mai multe aplicații: în contextul detecției vorbitorului, genul îmbunătățește performanțele prin limitarea spațiului de căutare, reprezintă o unealtă utilă în sistemele de analiză a semnalelor, în sistemele de indexare, cunoașterea genului vorbitorului este o indicație folosită în notare.

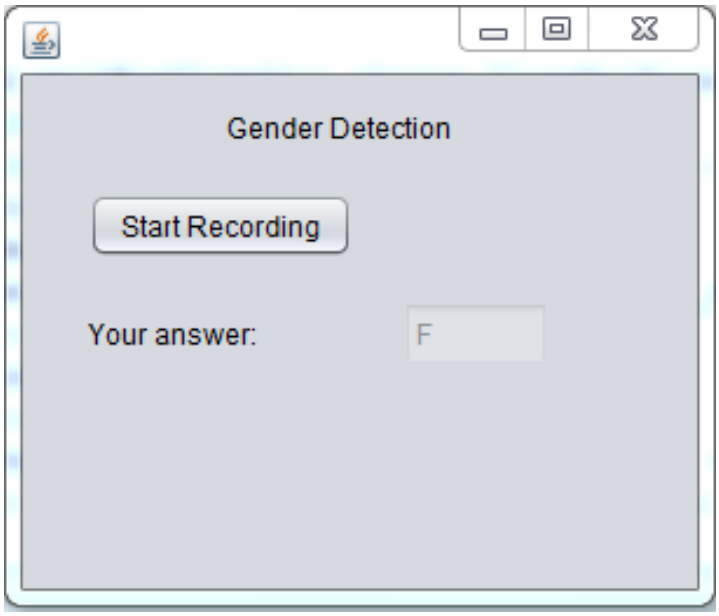


Figura 4.2 Interfaţa aplicaţiei demonstrative

4.2.2 Interfaţa utilizatorului

Aplicaţia, în mod asemănător cu cea anterioră, are o interfaţă grafică simplă construită cu ajutorul cadrului de lucru swing javax, ce are la bază o conexiune de tip client-server. Apăsând butonul „Start Recording”, începe înregistrarea vocii utilizatorului, eticheta se schimbă în „Stop Recording”, ce ne permite să oprim înregistrarea în momentul în care ne dorim acest lucru şi să vedem rezultatul returnat de aplicaţie. Butonul revine în stare activă de „Start Recording”, astfel încât putem să ne înregistrăm de mai multe ori consecutiv, fără a fi nevoie să pornim întreg procesul. Fişierul audio rezultat este transmis serverului, care îl prelucrează şi returnează răspunsul detectat, ce va fi afişat în caseta de text corespunzătoare, precedată de eticheta „Your answer:”, figura 4.2.

4.2.3 Implementarea

Pentru dezvoltarea proiectului de recunoaştere a genului am utilizat o conexiune client-server, în cadrul căreia fiecare clasă are rolul său. Figura 4.3 prezintă o diagramă de secvență simplificată a aplicatiei:

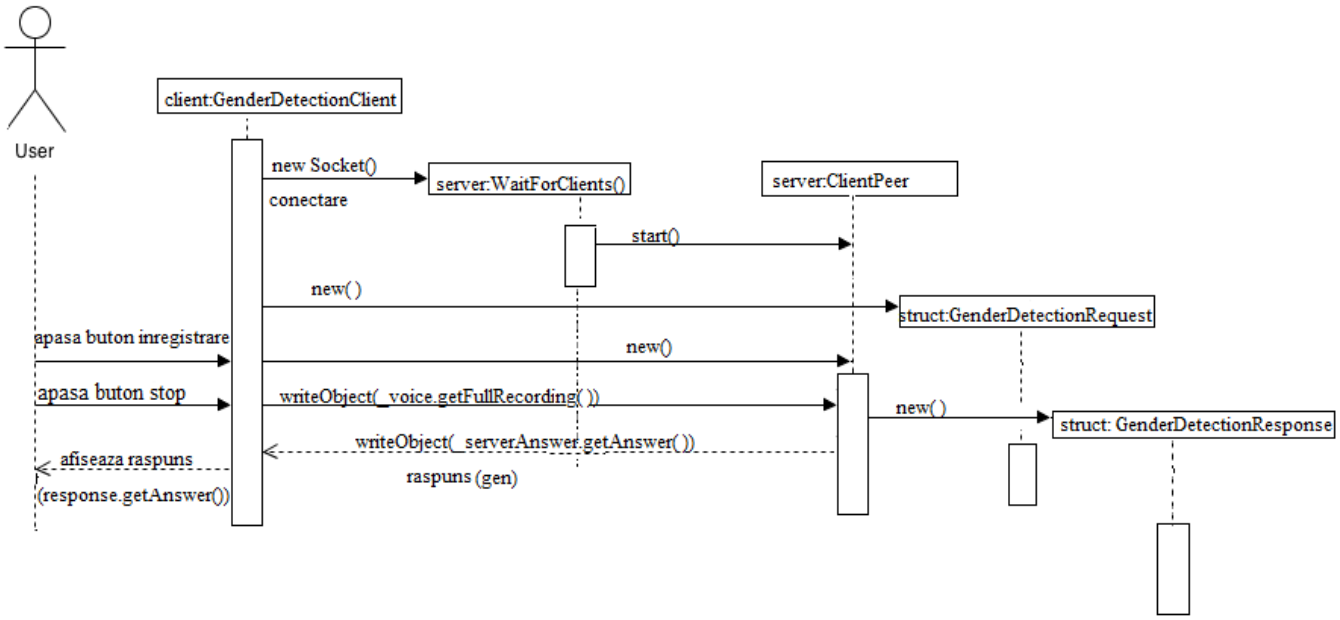


Figura 4.3 Diagrama de secvență a aplicației demonstrative

Principalele clase folosite pentru executarea sarcinii de recunoaștere a genului sunt prezentate în figura 4.4 :

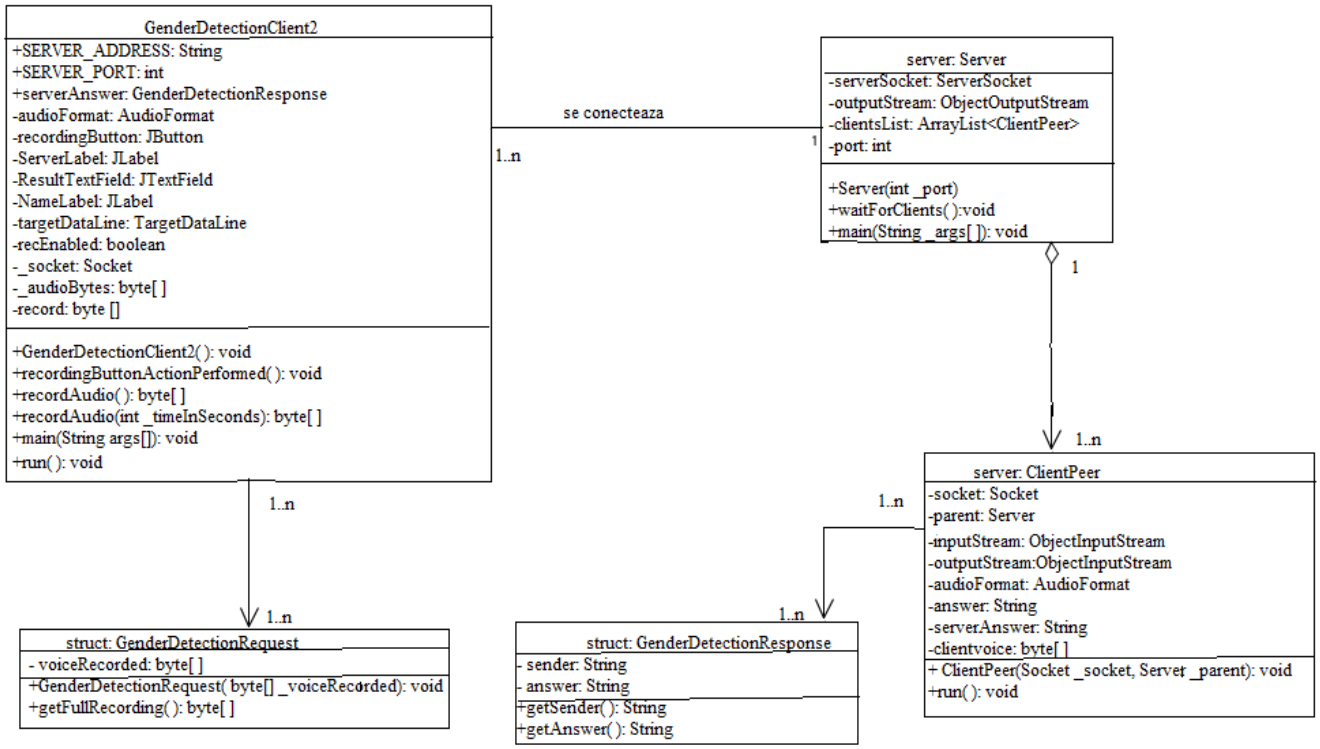


Figura 4.4 Diagrama UML (de clase) a aplicației demonstrative

Figura descrie structura internă a sistemului prin identificarea claselor, a atributelor şi a relaţiilor dintre clase. Fiecare clasă este reprezentată sub forma unui dreptunghi cu trei compartimente pentru: numele clasei, compartimentul din mijloc afișează atributele clasei, iar cel inferior operațiile efectuate.

4.2.3.1 Clientul

Clientul se conectează la server, înregistrează vocea sub forma unei înşiruiri de octeţi, îi transmite serverului, recepţionează şi afişează răspunsul. Toate aceste acțiuni au loc în mod transparent utilizatorului, prin intermediul librăriei SwingWorker, descrisă în capitolul anterior 4.1.3, ce furnizează updatarea interfeței în tip ce rulează, în spate, diverse sarcini. În acest timp, serverul aşteaptă conexiunile clienţilor, recepţionează fişierele audio trimise de aceştia, rulează partea de detectare şi transmite răspunsul înapoi la client.

Pentru a realiza o aplicaţie client se va folosi clasa Socket ce permite stabilirea unei conexiuni cu un server existent precum şi folosirea conexiunii pentru schimb bidirecţional de date, de tip TCP. Porturile și soclurile reprezintă mecanismul prin care se realizează legătura cu un server. Porturile reprezintă o poarta la care sunt primiți clienții pentru un anumit serviciu. Clientul trebuie să ştie numele maşinii severului pe care rulează precum şi numărul portului pe care serverul ascultă. La nivelul său, socket-ul se realizează specificând adresa IP şi portul serverului.

De exemplu la aceeași adresa se pot oferi diferite servicii, acestea oferindu-se la porturi diferite. Același calculator (cu o singură adresa IP) poate să ne ofere oricâte servicii dorește. Clienții care apelează la serviciile acestui calculator vor utiliza aceeași adresă, indiferent la care serviciu apelează, și toți clienții care doresc utilizarea unui anumit serviciu vor utiliza același port. Un soclu este de fapt un nod abstract de comunicație, reprezentând o interfață de nivel scăzut pentru comunicarea în rețea. Ele permit comunicarea între procese aflate pe același calculator sau pe calculatoare diferite din rețea. Java suportă trei tipuri de socluri. Clasa *Socket* utilizează un protocol orientat pe conexiune (TCP) și fluxuri de date (din engleză stream) pentru a trimite și a recepționa mesaje.

Un client ce dorește o comunicare cu serverul, trebuie să cunoscă trei lucruri:
- adresa server
- portul server utilizat pentru comunicare
- protocolul de comunicatie utilizat de server

4.2.3.2 Serverul

În aplicaţia Server, pentru început, se crează un obiect de tip ServerSocket care ascultă pe un anumit port. Constructorul pentru ServerSocket aruncă o excepţie dacă nu poate asculta pe portul respectiv.
Dacă însă, se conectează pe portul său, acceptă conexiunea de la client şi creează un fir de execuţie nou, căruia îi încredinţează socketul de legătură. Este necesară adăugarea referinţei către fiecare client în parte într-o listă, pentru a putea transmite răspunsul către adresa corespunzătoare. Serverul continuă să asculte cererile celorlalţi clienţi care vor să se conecteze și îi acceptă cu ajutorul metodei accept().

Pe fiecare fir de execuţie (numit în aplicaţia curentă ClientPeer), în mod independent, este creat un flux de intrare pe care se primeşte şirul de octeţi corespunzători vocii clientului și un flux de ieșire pentru a

returna răspunsul. Fluxul primit îl transformăm într-un fişier audio şi pe baza acestuia este format un fişier .seg ce conţine informaţii precum: numele înregistrării, canalul utilizat, începutul cadrului, durata în milisecunde, genul utilizatorului (în cazul în care deţinem informaţii de referinţă ce pot fi folosite la verificare), banda (8kHz sau 16kHz) şi mediu în care a avut loc înregistrarea. Extragerea coeficienţilor cepstrali MFCC se poate realiza prin metodele puse la dispoziţie de LIUM sau de Sphinx-4 (sphinx_fe), când se precizează şi caracteristice pe care le dorim. Următorul pas îl reprezintă alegerea modelelor de gen folosite.

LIUM estimează modelele de gen prin una din metodele: EM sau MAP pornind de la un set de înregistrări şi etichetele respective. În cazul modelării folosind metoda MAP (din engleza maximum a posteriori), copiem modelul UBM-ul (din engleza universal background model) , pentru fiecare vorbitor, obținut din mixarea GMM-urilor dependente de gen și banda. Pentru fiecare cluster, doar mediile sunt adaptate, conform formulei (2.3.3).

Decodarea va genera la ieșire un fișier cu rezultate ce conține informații precum: numele înregistrării, detecția sistemului, scorurile obținute pentru cele patru modele de gen și banda și referința (în cazul în care dispunem de ea pentru o verificare a performanțelor). După calcularea rezultatului, serverul este capabil să trimită răspunsul său clientului, iar acesta va fi afișat în câmpul de text rezervat. Utilizatorul are acum permisiunea de a își reînregistra vocea pentru o noua detecție.

4.2.4 Dezvoltări ulterioare

Există mai multe direcții în care aplicația poate fi îmbunătățită. Acestea se pot reflecta fie în interfață prin prezentarea mai multor informații sau prin permiterea unui control mai mare al utilizatorului, fie în codul pe care se bazează.

Unul din pașii doriți constă în modificarea și testarea caracteristicelor folosite în antrenarea modelelor vorbitorilor și în decodare, pentru a observa care combinație obține cele mai bune rezultate. Totodată, se pot investiga și cauzele erorilor de detecție obținute.

O alta îmbunătățire adusă aplicației ar fi detecția genului în diverse condiții de fundal ale înregistrării precum: zgomot, reflexii ale vocii, ce pot îngreuna decizia sistemului.

4.2.5 Concluzii

Scopul principal al acestei aplicații a fost acela de a recunoaște genul vorbitorului și de a observa cum arată o astfel de aplicație, ce poate fi folosită în diverse domenii reale. Un exemplu ar consta în adaptarea muzicii de așteptare la telefon sau furnizarea unor informații statistice pe baza distribuțiilor de gen. Am folosit sistemul LIUM, care este de nouă generație în domeniul vorbirii, bazat pe segmentare urmată de o clasificare. Scorurile obținute pentru cele patru categorii de combinații gen-bandă au prezentat diferențe foarte mici, lucru ce a condus la unele detecții de gen eronate.

CAPITOLUL 5

CONCLUZII

5.1 Concluzii generale

Scopul principal al acestei lucrări a fost acela de a prezenta diversele forme sub care sistemele de recunoaştere a vorbirii pot influenţa. Această sarcină a fost împărţită în trei sub-parţi: un sistem de recunoaştere a cifrele în limba română, o căutare a termenilor rostiţi într-un context şi un sistem de detecţie a genului vorbitorului, fiecare cu subdiviziunile specifice. Prima parte a fost îndeplinită cu succes, pentru cea de-a doua am obţinut rezultate pe baza experimentelor, în timp ce ultima încă mai prezintă o rată de detecţie greşită destul de semnificativă.

Lucrarea prezintă pașii urmați pentru îndeplinirea fiecărei atribuții. În capitolul 1 a fost descrisă motivația unor astfel de aplicații, în timp ce Capitolul 2 a prezentat detaliat cele trei parți componente ale lucrării.

Secțiunea 2.1 s-a axat pe prezentarea sistemele de recunoaștere automată a vorbirii, împreună cu uneltele puse la dispoziția utilizatorilor pentru prelucrarea datelor și obținerea rezultatelor.

Subcapitolul 2.2 a detaliat sarcina Spoken Web Search, de găsire a unui termen, împreună cu planul de evaluare NIST.

Ultime secțiune 2.3 a dezvoltat problema recunoașterii genului, parcurgând pașii necesari pentru rezolvarea acesteia și modalități de evaluare a performanțelor.

Capitolul 3 s-a focusat pe rezultatele experimentale obținute în competiția de recunoaștere a termenilor rostiți MediaEval 2013 (Spoken Web Search) și în detecția de gen. Tot în cadrul acestui capitol, am prezentat metodologia algoritmului de căutare DTW pentru MediaEval și rezultatele optime obținute prin varierea parametrilor α și β în algoritmul DTWSS (din engleză Dynamic Time Warping String Search). De asemenea, experimentele pentru recunoașterea de cifre rostite în limba română m-au ajutat în găsirea celor mai bune modele acustice și de limbă, ce au dat cele mai bune rate de eroare.

Capitolul 4 a prezentat două aplicații demonstrative cu interfața grafică pentru utilizatori. Prima a fost dezvoltată pentru îndeplinirea sarcinii de recunoaștere a cifrelor și a presupus folosirea setului de instrumente CMU Sphinx, în timp ce a doua prezintă recunoașterea genului și detaliază o arhitectură de timp client-server pe care s-a bazat.

Aceste două aplicații pot fi folosite pentru a demonstra performanțele unor sisteme de recunoaștere.

5.2 Contribuții personale

Atribuțiile personale ale autorului ce au dus la îndeplinirea obiectivelor acestei lucrari pot fi sumarizate astfel:

1)Proiectul recunoașterii de cifre

a) Crearea unui sistem de recunoaștere a cifrelor dependent de vorbitor.
b) Crearea unui sistem de recunoaștere a cifrelor independent de vorbitor.
c) Efectuarea experimentelor de optimizare a configurației unui sistem de recunoaștere automată a cifrelor din limba română.
d) Realizarea unei aplicații cu interfața grafică pentru recunoașterea cifrelor captate într-o înregistrare cu ajutorul microfonului.

2) Proiectul de găsire a termenilor vorbiți (MediaEval 2013)

a)Crearea unor fișiere ce conțin numele înregistrărilor în care se efectuează căutările și respectiv a înregistrărilor cu termenii căutați;
b)Efectuarea experimentelor de optimizarea a configurației unui sistem de recunoaștere automată a vorbirii pentru limba română;
c) maparea fonemelor din limbile în care au fost realizate înregistrările cu foneme din limba română;
d) adaptarea modelului acustic în limba română la datele acustice furnizate;
e)crearea modelelor de limbă trigrame;
f)realizarea experimentelor pentru optimizarea parametrilor din cadrul algorimului DTWSS;
g)eliminarea termenilor formați din mai puțin de patru caractere, pentru a obține rezultate
h)Rularea scriptului de evaluare furnizat de NIST STD

i)Reprezentarea curbelor DET pentru toate combinațiile configurate ale parametrilor algoritmului de căutare DTWSS.

3)Proiect de recunoaștere a genului

a)Crearea unor fișiere folosite în procesul de antrenare și evaluare a metodelor de detectare a genului.
b)Antrenarea cu ajutorul unei baze de înregistrări a unor modele pentru genul masculin și feminin.
c)Realizarea unor experimente având ca scop comparația diverselor modele de gen și metode de decodare.
d)Crearea unei aplicații cu interfața grafică de tip client-server pentru detectarea genului utilizatorului.
e)Dezvoltarea sistemului se bazează pe modelele de gen furnizate de LIUM.

BIBLIOGRAFIE

[ASR] http://speed.pub.ro/speed3/wp-content/uploads/2013/04/Indrumar-de-proiect-PCDTV-v4.pd (Accesat la 12.05.2014)

[B01] Buzo, A., "Automatic Speech Recognition over Mobile Communication Networks," Teza de doctorat, Universitatea Politehnica din Bucuresti, România, 2011, http://speed.pub.ro/academic/research-and-development-project-in-spoken-language technology.

[C01] S. S. Chen and P. S. Gopalakrishnan, Clustering via the Bayesian information criterion wi applications in speech recognition, in Proc. IEEE Int. Conf. Acoustics Speech, and Sign Processing, vol. 2, pp. 645 - 648, Seattle, USA, May 1998.

[C02] Cucu, H., "Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian," Teza de doctorat, Universitatea Politehnica din Bucuresti, România, 2011.

[D01] N. Deshmukh, A. Ganapathiraju, J. Hamaker, J. Picone, and M. Ordowski, "A public doma speech-to-text system," in *Proceedings of the 6th European Conference on Speec Communication and Technology*, vol. 5, Budapest, Hungary, Sept.1999, pp. 2127–2130.

[Diarizare] http://archives.limsi.fr/RS2005/chm/tlp/tlp1/index.html (Accesat la 21.06.2014)

[Gender] http://www.itl.nist.gov/iad/mig/publications/proceedings/darpa98/html/bn30/bn30.html (Accesat la 15.04.2014)

[H01] Huang, X., Acero, A., Wuen-Hon, H., Spoken Language Processing – A Guide to Theory, Algorithm, and System Development, Prentice Hall, 2001.

[H02] Hwang, M.Y., X.D. Huang, "Acoustic Classification of Phonetic Hidden Markov Models," Eurospeech 1991, 1991.

[J01] Daniel Jurafsky, James H. Martin, Speech and Language Processing- An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition, 1999.

[JSGF] Java Speech Grammar Format, http://java.sun.com/products/java media/speech/forDevelopers/JSGF/index.html. (Accesat la data de 01.06.2014)

[L01] X. X. Li, Y. Zhao, X. Pi, L. H. Liang, and A. V. Nefian, "Audio-visual continuous speech recognition using a coupled hidden Markov model," in *Proceedings of the 7th International Conference on Spoken Language Processing*, Denver, CO,Sept. 2002, pp. 213–216.

[P01] Poritz, A., "Hidden Markov models: a guided tour," ICASSP 1988, pp. 7–13.

[R01] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition," Proceedings of the IEEE, vol. 77, no. 2, pp. 257-286, 1989.

[S01] George Saon, Jen-Tzung, „Large-Vocabulary Continuous Speech Recognition System" în IEEE Signal Processing Magazine, nr.6/2012, pp.19-25.

[Speech] http://en.wikipedia.org/wiki/Speech_recognition (Accesat la 15.06.2014)

[Sphinx] Toolkitul CMU Sphinx si tutorialele asociate, http://cmusphinx.sourceforge.net/wiki/ (Accesat la 21.06.2014)

[STD] http://www.multimediaeval.org/mediaeval2013 (Accesat la 20.06.2014)

[W01] Willie Walker, Paul Lamere, Philip Kwok, Bhiksha Raj,Rita Singh, Evandro Gouvea, Peter Wolf, and Joe Woelfel, „Sphinx-4: A Flexible Open Source Framework for Speech Recognition**”,** November 2004

[Y01] S. Young, "The HTK hidden Markov model toolkit: Design and philosophy," Cambridge University Engineering Department, UK, Tech. Rep. CUED/F-INFENG/TR152, Sept. 1994.