UNIVERSITATEA POLITEHNICA BUCUREŞTI
FACULTATEA DE ELECTRONICĂ, TELECOMUNICAŢII ŞI TEHNOLOGIA INFORMAŢIEI

APLICAŢII SOFTWARE ÎN DOMENIUL TEHNOLOGIEI VORBIRII

PROIECT DE DIPLOMĂ

Prezentat ca cerinţă parţială pentru obţinerea titlului de *Inginer*
în domeniul *Electronică, Telecomunicaţii şi Tehnologia Informaţiei*
Programul de studii *Calculatoare şi Tehnologia Informaţiei*

Conducători ştiinţifici
Ş.l. Dr. Ing. Horia CUCU
Prof. Dr. Ing. Corneliu BURILEANU

Absolvent
Alexandru Lucian GEORGESCU

Bucureşti
2016

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

Aprobat Director de Departament :

Prof. Dr. Ing. Sever PAŞCA

TEMA PROIECTULUI DE DIPLOMĂ
a studentului Georgescu E. Alexandru Lucian, grupa 443A

1. Titlul temei: **Aplicaţii software în domeniul tehnologiei vorbirii**

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

Serviciu web de autentificare prin voce

- Aplicaţie client: interfaţă grafică ce oferă funcţionalităţi de autentificare şi înregistrare a unor utilizatori noi în sistem. Aceasta va fi implementată folosind tehnologii Web: HTML, CSS, Javascript, Web Audio API.
- Baza de date: stochează informaţii despre utilizatorii aplicaţiei.
- Server de autentificare: serviciu de tip REST utilizând tehnologie Java ce realizează agregarea şi procesarea datelor, interacţionând cu aplicaţia client, cu baza de date şi cu serviciile de recunoaştere de vorbire şi vorbitor (ultimile două dezvoltate în cadrul unui alt proiect).

Aplicaţie pentru detecţia cuvintelor cheie

- Crearea unui sistem de recunoaştere automată a vorbirii folosind platforma de dezvoltare Kaldi şi evaluarea performanţei sistemului şi a resurselor de calcul necesare
- Implementarea unei aplicaţii client-server de detecţie a cuvintelor cheie folosind sistemul de recunoaştere a vorbirii menţionat mai sus. Aplicaţia este scrisă în limbajul de programare C şi foloseşte biblioteca de funcţii Gstreamer.

3. Proiectul se bazează pe cunoştinţe dobândite în principal la următoarele 3-4 discipline: Programare Obiect-Orientată, Tehnologii de Programare în Internet, Proiectarea bazelor de date, Interfeţe om-maşină.

4. Proprietatea intelectuală asupra proectului aparţine: Laborator Cercetare SpeeD, student Georgescu E. Alexandru Lucian

5. Locul de desfăşurare a activităţii: UPB

6. Realizarea practică rămâne în proprietatea: Laborator Cercetare SpeeD, student Georgescu E. Alexandru Lucian

7. Data eliberării temei: 01.10.2015

CONDUCĂTOR LUCRARE:
Ş.l. Dr. Ing. Horia CUCU

STUDENT:
Alexandru Lucian GEORGESCU

DECLARAȚIE DE ONESTITATE ACADEMICĂ

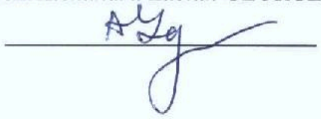
Prin prezenta declar că lucrarea cu titlul „*Aplicații software în domeniul tehnologiei vorbirii*", prezentată în cadrul Facultății de Electronică, Telecomunicații și Tehnologia Informației a Universității „Politehnica" din București ca cerință parțială pentru obținerea titlului de Inginer în domeniul Electronică, Telecomunicații și Tehnologia Informației, programul de studii Calculatoare si Tehnologia Informației este scrisă de mine și nu a mai fost prezentată niciodată la o facultate sau instituție de învățământ superior din țară sau străinătate.

Declar că toate resursele utilizate, inclusiv cele de pe Internet, sunt indicate în lucrare, ca referințe bibliografice. Fragmentele de text din alte surse, reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și fac referință la sursă. Reformularea în cuvinte proprii a textelor scrise de către alți autori face referință la sursă. Înțeleg că plagiatul constituie infracțiune și se sancționează conform legilor în vigoare.

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

București, *08.09.2016*

Absolvent *Alexandru Lucian GEORGESCU*



CUPRINS

LISTĂ DE FIGURI

LISTĂ DE TABELE

LISTĂ DE ACRONIME

ASR - Automatic Speech Recognition
KWS - Keyword Spotting
IVR - Interactive Voice Response System
MFCC - Mel Frequency Cepstral Coefficients
PLP - Perceptual Linear Prediction
HMM - Hidden Markov Models
GMM - Gaussian Mixture Model
FSG - Finite State Grammar
SER - Sentence Error Rate
WER - Word Error Rate
JSGF - Java Speech Grammar Format
FST - Finite State Transducer
LDA - Linear Discriminant Analysis
MLLT - Maximum Likelihood Linear Transform
SAT - Speaker Adaptive Traning
MMI - Maximum Mutual Information
CPU - Central Processing Unit
ECF - Experiment Control Files
RTTM - Rich Transcription Time Marked
TWV - Term Weighted Value
DET - Detection Error Tradeoff
HTML - HyperText Markup Language
CSS - Cascade Style Sheet
API - Application Programming Interface
REST - Representational State Transfer
HTTP -HyperText Transfer Protocol
XML - Extensible Markup Language
RTF - Real Time Factor

INTRODUCERE

MOTIVAŢIA LUCRĂRII

Comunicarea verbală este o abilitate pe care omul o deprinde în primii ani de viaţă. Deşi originea acestei particularităţi nu a fost încă elucidată şi este un subiect de dezbatere printre specialişti, dezvoltarea ei a fost cu siguranţă un pas extrem de important în evoluţia umană. Există studii ce atestă faptul că şi alte vieţuitoare posedă capacitatea de a comunica, dar delimitarea dintre acestea şi om este trasată de precizia şi complexitatea pe care numai comunicarea umană le prezintă.

În mod uzual, comunicarea este definită ca o acţiune ce permite schimbul de mesaje, idei şi gânduri, fiind esenţială în relaţiile interumane. Pentru ca acest proces să aibă loc, este necesară existenţa unor componente: emiţătorul, codul, canalul, informaţia şi receptorul. Până în istoria recentă a omenirii, acestea au impus unele limitări. Distanţa dintre emitor şi receptor nu putea să fie prea mare, limba folosită trebuia să fie cunoscută de către ambele părţi, iar canalul era reprezentat de aer. Avansul tehnologic, reprezentat mai ales de apariţia telefonului şi a radioului, au adus pentru prima dată modificări în acest scenariu clasic de comunicare, limitările fiind eliminate. Problema distanţei a fost depăşită, iar mesajele au devenit semnale electrice.

Dezvoltarea ulterioară a sistemelor de calcul a atras după sine un concept nou, acela de interacţiune om-maşină. Necesitatea acestui domeniu provine din dorinţa de a profita cât mai mult de avantajele utilizării calculatoarelor, asigurând în acelaşi timp un mod cât mai facil de a realiza acest lucru. Un exemplu în acest sens este apariţia interfeţelor grafice, acestea au înlesnit activitatea utilizatorilor mai puţin experimentaţi, oferind o interacţiune mult mai intuitivă decât interfeţele bazate pe comenzi text.

Având în vedere aceste considerente, nici comunicarea verbală nu a fost lăsată deoparte. Deoarece vorbirea este un proces natural, ce se desfăşoară fără depunerea unui efort considerabil, s-a dorit implementarea interacţiunii bazate pe voce. Sistemele de calcul au devenit capabile să interpreteze limbajul uman sau chiar să îl reproducă folosind tehnologia sintezei vocale.

O ramură importantă din acest domeniu este recunoaşterea automată a vorbirii (ASR). Aceasta presupune convertirea semnalului vocal în text. Tehnologia actuală a ajuns la un nivel de performanţă notabil, transcrierea fiind realizată într-un timp foarte scurt şi cu o acurateţe din ce în ce mai bună. Recunoaşterea automată a vorbirii este un subiect de mare interes la nivel mondial, în prezent existând mai multe grupuri de cercetare ce au ca scop găsirea unor noi algoritmi sau îmbunătăţirea celor existenţi. Preocuparea crescută poate fi motivată de diversitatea aplicaţiilor ce au la bază recunoaşterea vorbirii: comenzi vocale, indexare audio, transcrierea vorbirii spontane, sisteme biometrice, etc.

Am interacţionat pentru prima dată cu domeniul recunoaşterii vocale în timpul anului 3 de studii, atunci când am aflat de activitatea laboratorului de cercetare SpeeD (Speech and Dialogue), din cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei. Deşi este un domeniu foarte complex, mi-am dorit să cunosc mai mult, mai ales pentru că am avut ocazia să realizez unele proiecte cu caracater practic, în care noţiunile teoretice îşi dovedeau din plin aplicabilitatea. Aşadar, interesul manifestat faţă de recunoaşterea automată a vorbirii, împreună cu cel pentru dezvoltarea aplicaţiilor software, reprezintă motivaţia temei abordate în această lucrare.

OBIECTIVE

Lucrarea de faţă are ca scop expunerea sub formă de aplicaţii software a unor sisteme de recunoaştere automată a vorbirii (ASR) şi a unui sistem de detecţie a cuvintelor cheie (KWS). Se vor prezenta etapele de creare a unor astfel de sisteme în corelaţie cu noţiuni teoretice din domeniul tehnologiei vorbirii. Apoi se vor realiza diverse analize de performanţă pentru a stabili configuraţia optimă a sistemelor. În final, acestea vor fi utilizate la implementarea a două aplicaţii software. În mod concret, pentru realizarea acestor sarcini vor fi îndeplinite următoarele obiective:

1. Crearea modelelor (acustic, fonetic, ligvistic) pentru un sistem de recunoaştere de cifre în limba română. Evaluarea sistemului.
2. Crearea modelelor (acustic, fonetic, ligvistic) pentru un sistem de recunoaştere a frazelor în limba engleză coresponzătoare unui sistem interactiv cu răspuns vocal (IVR). Evaluarea sistemului.
3. Crearea unui sistem de detecţie a cuvintelor cheie ce se regăsesc în frazele IVR. Evaluarea sistemului.
4. Implementarea serverului de autentificare ca modul component al unui serviciu web de autentificare prin voce. Crearea aplicaţiei client şi a bazei de date ce stochează informaţii despre utilizatorii serviciului web.
5. Implementarea sistemului de detecţie a cuvintelor cheie sub forma unei aplicaţii client-server.

STRUCTURA LUCRĂRII

Capitolul 1 este o introducere teoretică ce are ca scop prezentarea fundamentelor ce stau la baza tehnologiei vorbirii.

Sunt descrise caracteristicile semnalului vocal, modul în care acesta este produs, dar şi perceput, precum şi procesul prin intermediul căruia este prelucrat în vederea obţinerii parametrilor vocali. Este prezentată arhitectura unui sistem de recunoaştere automată a vorbirii, algoritmii utilizaţi, metricile de evaluare, precum şi utilitarele ASR folosite.

A doua secţiune a capitolului prezintă principiile unui sistem de detecţie a cuvintelor cheie. Sunt prezentate câteva metode de căutare ale cuvintelor cheie, precum şi metricile de evaluare a sistemului.

Capitolul 2 cuprinde procesul de creare a două sisteme de recunoaştere automată a vorbirii: un sistem de recunoaştere a cifrelor în limba română şi un sistem de recunoaştere a frazelor în limba engleză, corespunzătoare unui sistem interactiv cu răspuns vocal. Se vor prezenta experimente şi rezultate din care pot fi desprinse concluzii despre performanţa sistemelor şi a algoritmilor folosiţi.

Capitolul 3 descrie procesul de creare a unui sistem de detecţie a cuvintelor cheie. Se vor analiza modelele de limbă potrivite pentru o astfel de sarcină, iar la final se va realiza evaluarea sistemului.

Capitolul 4 este dedicat serviciului web de autentificare prin voce. Se va prezenta arhitectura serviciului, precum şi modul de funcţionare din punctul de vedere al unui utilizator. Apoi se vor sumariza tehnologiile folosite şi se vor oferi detalii de implementare. Capitolul se încheie cu o descriere a dezvoltărilor ulterioare ce pot fi efectuate, însoţite de concluzii.

Capitolul 5 descrie aplicaţia de detecţie a cuvintelor cheie. În mod similar cu capitolul precedent, se va prezenta arhitectura aplicaţiei şi modul de funcţionare din perspectiva utilizatorului, iar ulterior se vor aduce la cunoştinţă tehnologiile folosite şi detaliile de implementare. În încheiere, se vor prezenta unele concluzii şi direcţii viitoare de dezvoltare.

CAPITOLUL 1

DOMENIUL TEHNOLOGIEI VORBIRII

1.1 STRUCTURA UNUI SISTEM DE RECUNOAŞTERE AUTOMATĂ A VORBIRII

Recunoaşterea automată a vorbirii (ASR) este o tehnologie ce permite unui sistem de calcul să transforme vorbirea în text. Intrarea unui astfel de sistem este reprezentată de un semnal vocal, în timp ce ieşirea sistemului este constituită dintr-o secvenţă de cuvinte corespunzătoare transcrierii.

Din punct de vedere probabilistic, acest proces implică determinarea celei mai probabile secvenţe de cuvinte *W**, dat fiind mesajul vorbit *X*:

$$W* = argmax\ P(W \mid X).$$

Conform regulii lui Bayes rezultă:

$$W* = argmax\ P(W \mid X) = argmax\ \frac{P(X \mid W) * P(W)}{P(X)} = argmax\ P(X \mid W) * P(W)$$

Probabilitatea mesajului vorbit $P(X)$ este independentă de secvenţă de cuvinte *W* şi poate fi ignorată. În acest fel, sarcina iniţială poate fi descompusă în alte două sarcini mai mici:

a. Estimarea probabilităţii mesajului vorbit, dată fiind secvenţă de cuvinte pronunţată $P(X|W)$.

b. Estimarea probabilităţii apriori a secvenţei de cuvinte $P(W)$.

Prima probabilitate poate fi determinată folosind un **model acustic**, în timp ce a doua probabilitate se obţine folosind un **model de limbă**.

Figura de mai jos ilustrează arhitectura unui sistem ASR:

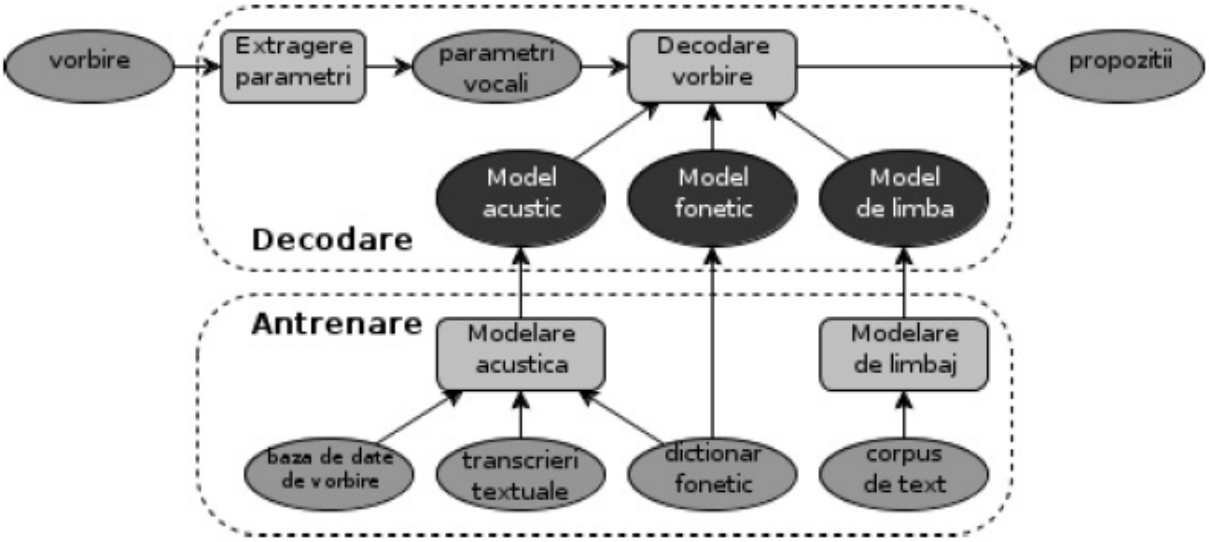


Figura 1.1 Arhitectura unui sistem ASR. Sursa: [1]

Un sistem ASR implică două faze distincte: o fază de antrenare şi o fază de decodare. Recunoaşterea vorbirii se realizează în faza de decodare, folosind parametrii vocali extraşi din semnalul vocal şi modelele (acustic, fonetic, lingvistic) dezvoltate în faza de antrenare.

Modelul acustic estimează probabilitatea unui mesaj vorbit, dată fiind secvenţă de cuvinte pronunţată. Acesta este construit pe baza unui set de înregistrări audio cât mai mari pentru care este cunoscută transcrierea text. El oferă o reprezentare statistică a fiecărui fonem. Fonemul este unitatea sonoră cea mai mică a limbii.

Modelul de limbă contribuie la îmbinarea cuvintelor astfel încât să poată fi obţinute propoziţii cu sens. În mod uzual se folosesc două tipuri de modele de limbă. Modelele probabilistice, obţinute prin calculul probabilităţilor de apariţie a cuvintelor sau de succesiune a lor, observând o cantitate cât mai mare de text. Al doilea tip este reprezentat de modelele de tip gramatică, acestea fiind modele bazate pe reguli, ce consistă într-un set de reguli ce definesc ordinea de apariţie a cuvintelor.

Modelul fonetic are rolul de a realiza legătura între modelul acustic şi modelul de limbă. Acesta este sub forma unui dicţionar ce specifică pronunţia cuvintelor. Fiecărui cuvânt din vocabular îi este asociată o secvenţă de foneme.

1.1.1 Producerea semnalului vocal

Cunoaşterea aparatului fonator uman oferă unele informaţii importante în vederea proiectării sistemelor de recunoaştere automată a vorbirii. Astfel, vorbirea este produsă de aerul eliminat din plămâni, modulat mai departe de către componentele tractului vocal, rezultând astfel un semnal în gama de frecvenţă audio. În 1960, Gunnar Fant a definit producerea vorbirii cu ajutorul unui model sursă-filtru. Aparatul respirator, prin aerul expirat, formează semnalul de excitaţie. Acesta ajunge în trahee, la capătul căruia se află laringele, cu rol de a modula presiunea aerului. Aici se găsesc corzile vocale, care prin depărtare formează o deschidere numită glotă. Sunetele vocalizate apar datorită vibraţiei cvasi-periodice a corzilor, în timp ce în cazul sunetelor nevocalizate nu există această vibraţie, ele fiind cauzate de turbulenţe la nivelul tractului vocal. Tractul vocal, ce are rolul unui filtru, cuprinde elementele sistemului fonator aflate deasupra corzilor vocale. Acesta este format din diferite cavităţi rezonatoare (faringiană,

nazală, bucală) şi organele articulatorii (limba, văl palatin, palatul dur, dinţii, buzele). Periodicitatea semnalului de excitaţie se numeşte frecvenţă fundamentală. Acesta este diferită în funcţie de caracteristicile fiecărei persoane. La bărbaţi, frecvenţa fundamentală medie este de 132 Hz, în timp ce la femei se află în jurul valorii de 223 Hz. [2]

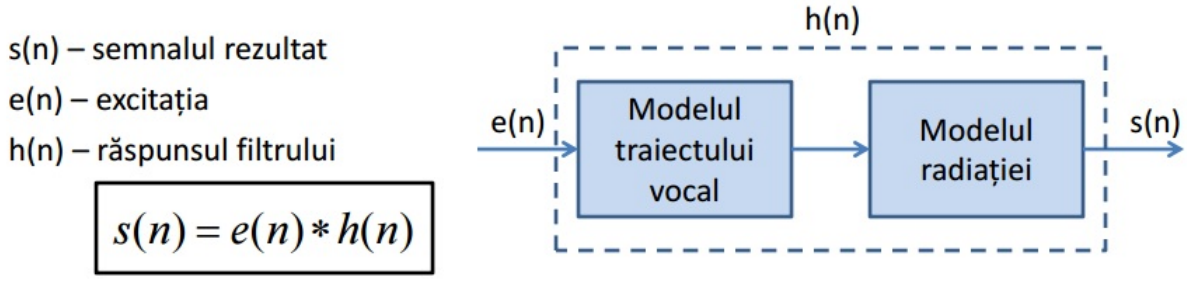


Figura 1.2 Modelul sursă-filtru al producerii vorbirii. Sursa:[3]

În funcţie de modul în care sunt articulate, fonemele se împart în vocale şi consoane (fricative, nazale, stopate). Vocalele sunt caracterizate de intensitatea crescută a energiei în raport cu alte foneme. Producerea lor are loc fără obstrucţii la nivelul tractului vocal. În schimb, producerea consoanelor este condiţionată de obstrucţionarea aerului de către diferite părţi componente ale tractului vocal.

1.1.2 Percepţia semnalului vocal

Aşa cum se poate observa în schema arhitecturii unui sistem ASR, semnalul vocal nu este utilizat în forma lui brută. Premergător etapei de antrenare, cât şi a celei de decodare, semnalul vocal este supus mai multor operaţii ce au ca scop obţinerea unui set de parametri vocali. Aceşti parametri conţin informaţii relevante pentru sarcina de recunoaştere automată a vorbirii, iar cei mai utilizaţi sunt coeficienţii mel-cepstrali (MFCC).

Aşa cum un sistem de sinteză de vorbire imită producerea semnalului vocal, un sistem de recunoaştere automată a vorbirii imita modul în care urechea umană percepe semnalul vocal. Pentru extragerea parametrilor vocali este necesar să se cunoască câteva detalii despre percepţie.

Sistemul auditiv uman intepretează frecvenţele într-un mod neliniar. Tonul fundamental este perceput liniar în gama de frecvente 0-1000 Hz [4]. Peste această valoare, scara devine logaritmică. În acest sens a fost dezvoltată scara mel, definită de următoarea formulă:

$$F_{mel} = \frac{1000}{\log(2)} \times \left[1 + \frac{F_{Hz}}{1000}\right].$$

F_{mel} este frecvenţa rezultată pe scara mel, iar F_{Hz} este frecvenţa normală.

Prin aplicarea scării mel, parametrii vocali vor fi grupaţi în jurul frecvenţelor joase, în zona tonului fundamental perceput. Astfel, semnalul va fi mult mai bine reprezentat din punct de vedere al sistemului auditiv uman.

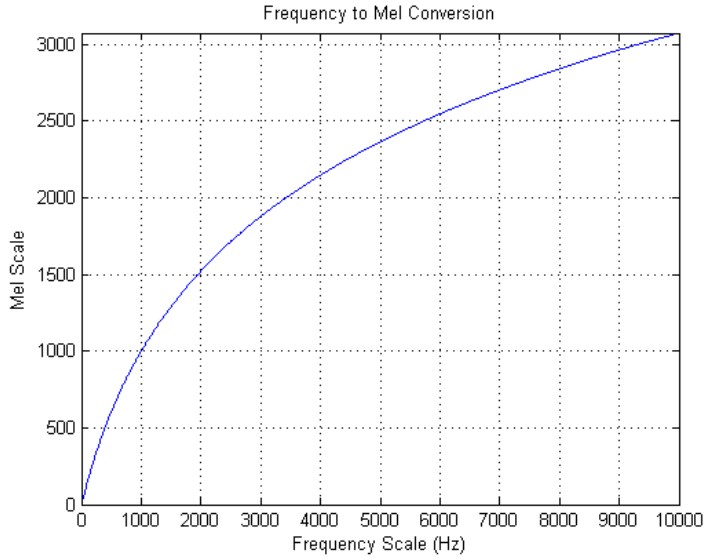


Figura 1.3 Scara Mel [5]

1.1.3 Parametri acustici

Figura de mai jos conţine procesul de extragere al parametrilor MFCC, aceştia sunt în număr de 7 . Se va face o descriere pentru fiecare parametru.

A. Într-o primă fază are loc procesul de preaccentuare care are rol de a creşte energia semnalului la frecvenţă înaltă.

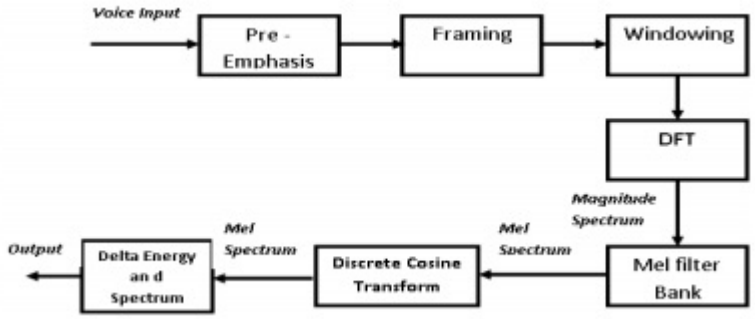


Figura 1.4 Metodă de extragere a MFCC. Sursa:[6]

B. Deoarece semnalul vocal nu este staţionar, analiza spectrală nu poate fi realizată pe întreg semnalul, ci pe porţiuni (ferestre) mai mici, cvasi-stationare cu o durată de 20-30 milisecunde. Ferestrele vor avea o perioadicitate de 10-15 milisecunde [7]. Ferestrele vor avea un timp de suprapunere pentru evitarea pierderii unor zone de semnal între acestea. Ferestele pot fi de mai multe tipuri, iar cea mai uzuală este detaliată la punctul următor. Orice tip de fereastră are un spectru de tip trece jos (un lob principal la frecvenţe joase şi mai mulţi lobi secundari atenuaţi diferit).

C. Fiecarea cadru obţinut la pasul anterior se va multiplica cu o fereastră Hamming. Scopul acestui pas este de corectare a discontinuităţilor care pot să apară.

$$W[n]=\begin{cases}0{,}54-0{,}46\cdot\cos\frac{2\pi n}{N-1}, & 0\le n\le N\\ 0, & in\ rest\end{cases}$$

D. Aşa cum am descris la secţiunea 1.1, semnalul vocal este determinat de convoluţia în timp a semnalului de excitaţie dat de aerul din plămâni şi răspunsul în timp al filtrului reprezentat de tractul vocal. Acesta din urmă fiind relevant pentru extragerea parametrilor MFCC. Deoarece cele două componente nu pot fi separabile în domeniul timp se va aplica transformată Fourier rapidă pentru a fi translatată în domeniul frecvenţă. Prin această transformare se ajunge ajunge la o relaţie care poate fi logaritmată putându-se astfel ajunge la o relaţie liniară de tip sumă care este separabilă.

$$S(n) = E(n) * h(n) \overset{T.F.}{\Leftrightarrow} S(\omega) = E(\omega) \cdot H(\omega)$$

$$S(\omega) = E(\omega) \cdot H(\omega) \overset{log}{\Rightarrow} \log(S(\omega)) = \log(E(\omega)) + \log(H(\omega))$$

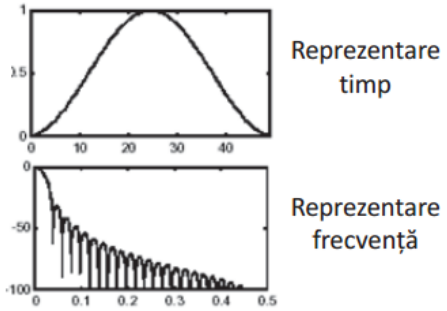


Figura 1.5 Reprezentarea în timp şi frecventă a semnalului. Sursa:[8]

E. Spectrul rezultat în urma aplicării transformatei Fourier se va netezi cu ajutorul unui banc de filtre mel triunghiulare. Este calculat spectrul mediu în jurul frecvenţei centrale a fiecărei benzi, iar fiecare filtru va ocupa o bandă mai largă în funcţie de ordinea din filtru, aşa cum este prezentat în figura următoare:

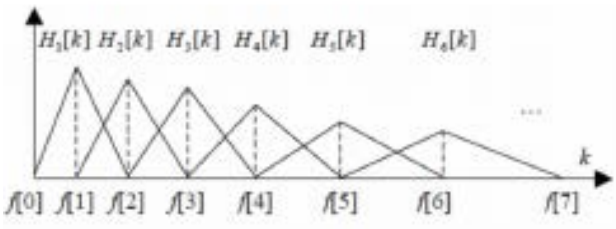


Figura 1.6 Bancul de filtre Mel. Sursa:[9]

F. Următorul pas este convertirea spectrului logaritmic pe scara MEL înapoi în domeniul timp folosind transformata cosinus discretă. Rezultatul acestei conversii fiind chiar coeficienţii MFCC. Transformata cosinus discretă are capacitatea de a concentra informaţia spectrală într-un număr mai mic de parametrii şi de a decorela aceste valori.

G. Dintr-o fereastră de analiză se extrag de obicei 13 parametrii MFCC dar uneori acest vector acustic este extins fiind adăugate şi derivatele funcţie de timp de ordinul I şi II (parametrii delta, delta+delta numiţi şi coeficienţi de viteză şi acceleraţie). În final se ajunge la un vector acustic cu 39 parametri.

1.1.4 Modele acustice

Modelul acustic are rolul de a estima probabiltatea că mesajul vorbit să fi fost generat de o secvenţă de cuvinte. În acest caz ar fi necesar să se realizeze observaţii pe baza unui set de cuvinte de la mai mulţi vorbitori creând astfel un model matematic pentru fiecare cuvânt în

parte. Totuşi acesta abordare nu este una eficientă. O unitate de vorbire de bază trebuie să aibă 3 caracteristici [10]:

- Precizie: este nevoie ca unitatea să repezinte realizarea acustică în contexte diferite;
- Antrenabilitate: datele de care dispunem trebuie să fie îndeajuns pentru o estimare cât mai exactă a unităţii;
- Generalizabilitate: posibilitatea construirii oricărui cuvânt având la bază un set de unităţi.

Deoarece cuvintele nu sunt antrenabile şi nici generalizabile se impune folosirea unor unităţi de bază sublexicale numite foneme. Acestea pot îndeplini cu uşurinţă cele trei caracteristici enumerate mai sus. Dar există totuşi un inconvenient: având în vedere sistemul articulator uman un fonem este dependent de contextul în care apare, el depinzând de fonemele vecine. Pentru înlăturarea acestuia inconvenient s-a ajuns la concluzia că fiind necesară trecerea la analiza fonemelor luate în context cu cel anterior şi următor ajungandu-se astfel la noţiunea de trifoneme.

Deoarece există foarte multe trifoneme ce conduc la un număr foarte mare de parametrii ce trebuie estimaţi, pentru a se ajunge la un optim între precizie şi antrenabilitate se procedează la gruparea trifoanemelor funcţie de stări. Fiecare astfel de grup de stări (stări Markov) au fost denumite senone, acestea fiind în momentul de faţă cele mai utilizate unităţi acustice de bază.

Având în vedere definiţia de mai sus a unui trifonem ca un fonem ce depinde de contex cu stările din vecinătatea sa el poate fi modelat folosind platforma HMM-GMM.

Modelul Markov Ascuns (HMM) este un automat cu stări finite, el este considerat ascuns deoarece observatorul nu cunoaşte secvenţa de stări ci doar vectorul de proprietăţi acustice al fiecărei stări (MFCC) generat de o funcţie de densitate de probabilitate.

În figura de mai jos este prezentată structura unui HMM.

Următorii parametri caracterizează un HMM:

Un set de stări:

$S = S_1S_2S_3....$

Un set de probabilităţi de tranziţie între stări de forma:

$P_{(Şi|\ Sj)}$ - reprezentând probabilitatea de tranziţie de la starea i la starea j;

Observaţii probabilistice:

$P(X|Ş_i)$ - fiind probabilitatea că observaţia x să fi fost generată de starea i.

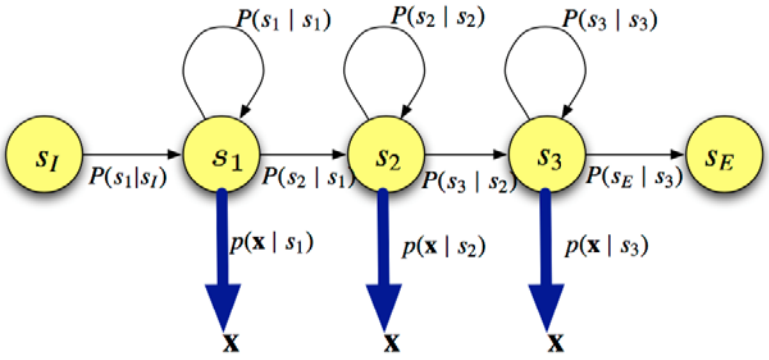


Figura 1.7 Structura unui HMM

Un GMM (model de mixturi Gaussiene) este probabilitatea ca observaţia x să fi fost generată de starea i. El este format dintr-o sumă de funcţii Gaussiene, fiecare caracterizat de o medie şi o dispersie proprie.

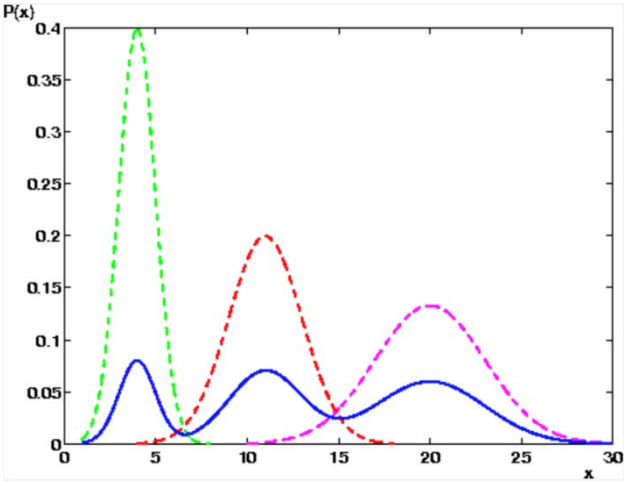


Figura 1.8 Mixtură de gaussiene. Sursa:[11]

Densitatea de probabilitate reprezentată cu albastru în figură de mai sus este dată de suma celorlate trei.

$P(x|\mu,\sigma^2) = N(x;\ \mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(x-\mu)^2}{2\sigma^2}\right)$

1.1.5 Modele de limbă

Scopul final de a obţine propoziţii cu sens din îmbinarea cuvintelor este dat de modelul de limbă. În mod uzual se folosesc două tipuri de modele de limbă.

a. Model probabilistic (de tip n-gram) care poate fi obţinut prin calculul probabilităţilor de apariţie a cuvintelor sau de succesiune a lor prin observarea unui volum cât mai mare de text. Probabilitatea este calculată în funcţie de istoricul cuvântului şi anume de secvenţa numărului finit de cuvinte care îl preced.

$$P(w) = P(w_1, w_{2,\dots,w_n}) = P(w_1) * P(w_2|w_1) * P(w_n|w_1, w_2, \dots, w_{n-1})$$

Astfel, probabilitatea unei secvenţe de cuvinte este descompusă în estimarea probabilităţilor unui singur cuvânt, dată fiind secvenţa de cuvinte anterioară acestuia.

Cele mai uzuale sunt modelele de tip 3-gram şi 2 gram unde este necesară o istorie de două şi respectiv un cuvânt. În cazul unul model 2-gram, calculul probabilităţii unei perechi e cuvinte se calculează în felul următor:

$$P(w_j|w_i) = \frac{count(w_i, w_j)}{\sum count(w_i, w)}$$

Probabilitatea de apariţie a perechii de cuvinte (w_i, w_j) este dată de numărul de apariţii ale cuvântului w_i urmat de cuvântul w_j, în raport cu numărul de apariţii ale aceluiaşi cuvânt w_i, urmat de alte cuvinte.

Pentru obţinerea unor probabilităţi cât mai corecte este necesară analiza unei cantităţi foarte mari de text.

b. Model bazat pe reguli, de tip gramatică cu stări finite (FSG). Constă într-un set de reguli ce definesc ordinea de apariţie a cuvintelor. Un astfel de model are forma unui graf, unde nodurile sunt cuvinte, iar arcele sunt tranziţii între cuvinte. Fiecare arc poate fi

caracterizat de o probabilitate de tranziţie. Drumurile grafului reprezintă secvenţe de cuvinte permise.

1.1.6 Modelul fonetic

Modelul fonetic are rolul de a conecta modelul acustic cu modelul de limbă. Acest model are forma unui dicţionar de pronunţii. Fiecărui cuvânt din vocabular îi este atribuită o secvenţă de foneme ce reprezintă pronunţia acelui cuvânt. În cazul existenţei a mai multor variante de pronunţie pentru un singur cuvânt, acestea vor fi specificate fiecare în parte.

Crearea unui model fonetic este o sarcină realizată de obicei manual, mai ales atunci când vocabularul este unul redus. Sarcina devine însă una complicată, dacă se doreşte crearea unui sistem de recunoaştere pe întregul vocabular al unei limbi. Se pot folosi unele aplicaţii software denumite *fonetizatoare*, ce au implementate reguli de pronunţie ale literelor în funcţie de contextul în care apar.

1.1.7 Metrici de evaluare

Evaluarea unui sistem de recunoaştere automată a vorbirii este posibilă prin transcrierea unei baza de date de evaluare. Pentru această bază de date sunt cunoscute transcrierile reale ale clipurilor audio, numite transcrieri de referinţă. Prin compararea acestora cu transcrierile generate de sistemul de recunoaştere automată a vorbirii, sunt calculate ratele de eroare la nivel de propoziţie şi la nivel de cuvânt.

Rata de eroare la nivel de propoziţie (SER) este dată de raportul dintre numărul de propoziţii transcrise eronat şi numărul total de propoziţii. O propoziţie este eronată dacă ea conţine cel puţin un cuvânt greşit.

$$SER[\%] = \frac{\#\ propozitii\ eronate}{\#\ total\ propozitii\ in\ referinta} \times 100$$

Această rată este concludentă în situaţiile în care transcrierea greşită a unui singur cuvânt conduce la anularea caracterului valabil al întregii propoziţii (de exemplu, transcrierea greşită a unei cifre dintr-un număr de telefon va face acel număr inutil, în timp ce transcrierea greşită a unui cuvânt dintr-o ştire difuzată la radio nu va afecta prea mult inteligibilitatea propoziţiei).

Rata de eroare la nivel de cuvânt se calculează prin utilizarea unui algoritm de aliniere a transcrierii ipotetice cu transcrierile de referinţă. Pot apărea erori de inserţie (apar cuvinte în plus faţă de referinţă), erori de substituţie (sunt transcrise alte cuvinte decât cele din referinţă) şi erori de ştergere (nu sunt transcrise cuvinte ce apar în referinţă). Formula de calcul este următoarea:

$$WER[\%] = \frac{\#Erori\ de\ insertie + \#Erori\ de\ substitutie + \#\ Erori\ de\ stergere}{\#\ Total\ cuvinte\ in\ referinta} \times 100$$

1.1.8 Utilitare pentru recunoaşterea automată a vorbirii

CMU Sphinx

CMU Sphinx este un utilitar cu sursă deschisă, scris în limbajul Java, utilizat pentru crearea sistemelor de recunoaştere automată a vorbirii. Acesta a fost dezvoltat în cadrul Carnegie Mellon University. Este o aplicaţie modulară, fiind formată din 3 blocuri principale:

- *Frontend*
- *Decoder*
- *Knowledge database*

Acesta arhitectură poate fi observată în imaginea următoare:

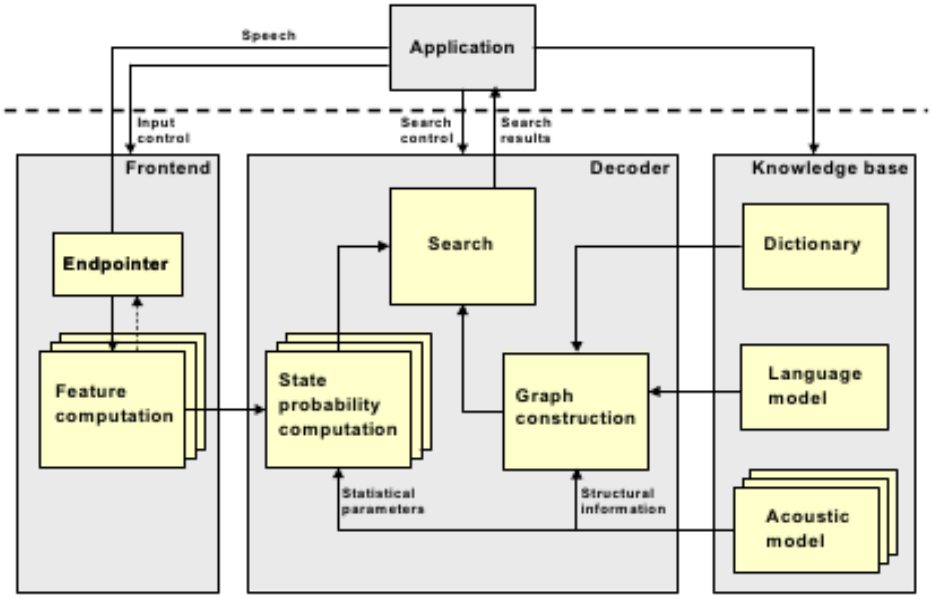


Figura 1.9 Arhitectură CMU Sphinx. Sursa:[12]

Blocul *Frontend* are rol în extracţia parametrilor vocali (MFCC sau PLP). Blocul *Knowledge base* conţine resursele acustice, fonetice şi ligvistice necesare sarcinii de recunoaştere. Blocul *Decoder* realizează transcrierea semnalului audio în text.

Modelele acustice folosite în CMU Sphinx sunt antrenate folosind platforma HMM-GMM. Modelarea fonetică permite utilizarea atât a fonemelor independente de context, cât şi a celor dependente (trifoneme). Modelele de limbă pot fi modele probabilistice, dar şi modele de tip gramatică, scrise în formatul JSGF (Java Speech Grammar Format).

Kaldi

Un alt utilitar folosit în crearea sistemelor de recunoaştere automată a vorbirii este Kaldi. Acesta a fost scris în limbajul C++, fiind de asemenea cu sursă deschisă.

Arhitectura Kaldi poate fi observată în următoarea imagine:

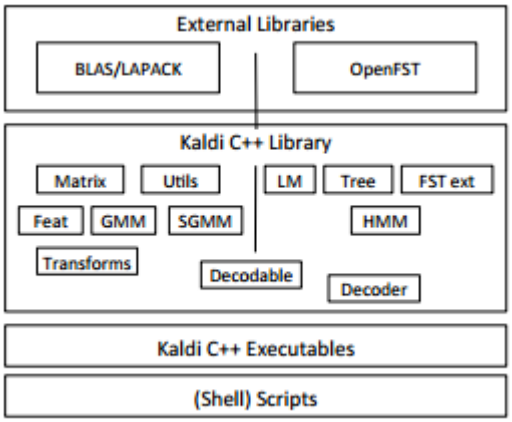


Figura 1.10 Arhitectură Kaldi. Sursa:[13]

Utilitarul depinde de două librării externe, OpenFST (pentru implementarea platformelor cu stări finite) şi BLAS/LAPACK (o colecţie de biblioteci matematice). Librăria Kaldi propriu-zisă conţine module de extragere a parametrilor (MFCC sau PLP), antrenarea modelelor acustice (bazate pe GMM şi mai nou pe reţele neuronale), compilarea modelelor de limba şi bineînţeles decodarea audio.

Kaldi permite crearea mai multor tipuri de sisteme acustice:

- *Monophone:* foloseşte foneme independente de context;
- *Triphone (TRI1)*: foloseşte foneme cu 3 stări, ce depind de contextul în care apar;
- *LDA+MLLT (TRI2)*: Este o alternativă la folosirea derivatelor coeficienţilor cepstrali. LDA (Linear Discriminant Analysis) are ca scop reducerea spaţiului de caracteristici, fiind păstrate numai cele care conţin informaţie discriminatorie.
- *LDA+MLLT+SAT (TRI3):* În plus faţă de modelul precedent, acesta realizează şi adaptarea la vorbitor (Speaker Adaptive Training).
- *LDA+MLLT+SAT+MMI(TRI3_MMI):* MMI (Maximum Mutual Information) tinde să maximizeze probabilitatea aposteriori a secvenţei rostite.

1.2 STRUCTURA UNUI SISTEM DE DETECŢIE A CUVINTELOR CHEIE

Procesarea informaţiei a devenit o preocupare majoră a zilelor noastre, iar procesarea limbajului vorbit este una dintre aceste provocări.

Detecţia cuvintelor cheie are ca scop determinarea existenţei unor cuvinte cheie într-o secvenţă de vorbire. Este o sarcină necesară în situaţia în care se doreşte indentificarea unor termeni rostiţi într-o cantitate foarte mare de date audio.

Un studiu din 2014 al companiei Domo, numit "Data Never Sleeps 2.0" [14], a arătat printre altele că în fiecare minut al unei zile, pe Youtube sunt încărcate 72 ore de conţinut audiovizual, după ce în 2012 cantitatea era de 48 ore de conţinut audiovizual. În numai doi ani, valoarea a crescut de 1.5 ori. Aşadar, în momentul actual, dacă tendinţa s-a păstrat, este posibil ca un număr de aproximativ 100 de ore de conţinut să fie urcate pe Youtube în fiecare minut. Aşadar, dacă ne raportăm la aceste cifre, căutarea termenilor vorbiţi în conţinutul audio este imposibilă să fie realizată prin ascultarea propriu-zisă a înregistrărilor. Folosind tehnica de procesare modernă, sistemele de detecţie a cuvintelor cheie sunt singurele care pot îndeplini această sarcină în timp util.

1.2.1 Căutarea prin lattice

Pentru înţelegerea acestei metode este mai întâi necesar să se cunoască definiţia unei lattice. Latticea se poate defini ca un graf G(N,A), unde N reprezintă nodurile şi A sunt arcele. În contextul recunoaşterii automate a vorbirii, rezultatul decodării poate fi strcturat sub forma unei lattice, unde fiecare arc are o anumită probabilitate şi reprezintă o secvenţă alternativă de cuvinte rostite. Alegând calea cu cea mai mare probabilitate se va obţine cea mai bună ipoteză a transcrierii, în acest fel obtinandu-se transcrierea textului în sine.

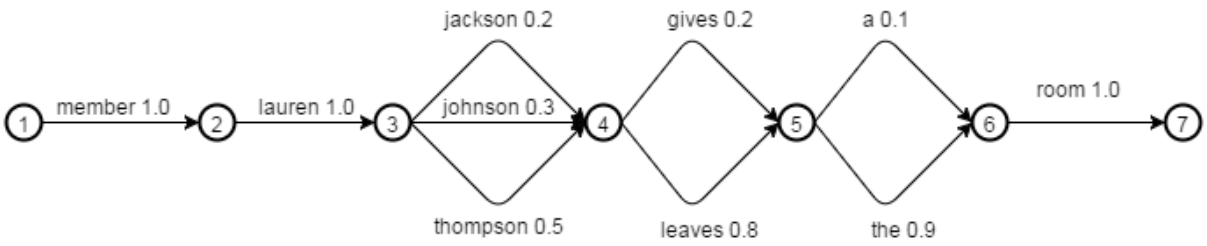


Figura 1.11 Exemplu de lattice

Această metodă este implementată sub forma unui modul KWS în utilitarul Kaldi. Kaldi realizează sarcină de KWS în mod offline în doi paşi:

- vorbirea este decodată cu un sistem ASR generând astfel o lattice de cuvinte;

- cuvintele cheie sunt căutate prin lattice.

În continuare se va detalia acest pas. Mai întâi este creată o listă de cuvinte cheie ce se doreşte a fi identificate apoi acestora li se asociază câte un ID şi sunt folosite pentru a se creea un automat cu stări finite (FST). De asemenea secvenţele audio rostite în care se realizează căutarea le sunt asociate un ID. Cu latticele obţinute la primul pas în felul următor:

ID-ul secvenţei rostite| momentul începerii | momentul de final | probabilitatea posterioară

În acest fel pentru fiecare secvenţă de cuvinte se va cunoaşte secvenţă audio corespunzătoare, momentele de timp între care se regăseşte şi probabilitatea. În final căutarea este realizată folosind automatul cu stări finite ce conţine cuvintele cheie, aplicată latticelor indexate. Principalul avantaj al acestei metode este reprezentat de posibilitatea de a seta un prag al probabilităţilor pe baza căruia cuvintele cheie găsite pot fi acceptate sau respinse. Fixarea unui prag jos va duce la acceptarea multor ipoteze crescând astfel şansa de a găsi cuvintele cheie căutate, dar în acelaşi timp probabilitatea unei alarme false este mărită. În schimb, fixarea unui prag înalt va asigura o rată scăzută a alarmelor false fiind posibil în acelaşi timp că apariţiile unor cuvinte cheie să nu fie contabilizate.

1.2.2 Căutarea prin transcriere text

Această metodă este una destul de simplă neavând de a face cu lattice şi cu probabilităţile arcelor pentru a lua o decizie referitoare la apariţia cuvintelor cheie. Este de asemenea o metodă care se realizează în doi paşi:

- secvenţa audio este decodată folosind un sistem ASR, generând o transcriere text;
- cuvintele cheie sunt căutate direct în transcrierea text.

Spre deosebire de metoda precedentă la acesta există doar două posibilităţi: cuvântul cheie să se regăsească sau nu în secvenţa rostită. Performanţa acestei metode depinde foarte mult de acurateţea sistemului ASR, aceasta fiind influenţată puternic de către modelul de limba şi modelul acustic folosit.

Pe baza rezultatelor experimentale se poate observa că un model de limbă foarte bogat produce apariţia unor probleme legate de resursele maşinii de calcul: memorie, CPU şi timpul de decodare.

1.2.3 Metrici de evaluare

Acest subcapitol prezintă unele metrici folosite pentru a măsura performanţa uni sistem KWS. Un sistem perfect trebuie să fie capabil să detecteze în cadrul secvenţei rostite, locul exact de apariţie al fiecărui cuvânt cheie, fără a fi generate false detecţii. Pentru a realiza evaluarea unui sistem KWS, este necesară mai întâi rularea sistemului de detecţie a cuvintelor cheie. Pe baza înregistrărilor audio existente în baza de date, a fişierului ECF (ce conţine indexarea fişierelor audio) şi a fişierului KWList (ce conţine cuvintele cheie căutate), este generat KWSList (fişierul cu detecţiile rezultate). Pentru fiecare cuvânt cheie sunt afişate fişierele audio în care acesta se găseşte, momentul de timp al apariţiei, durată, o probabilitatea de apariţie (cu o valoare normată între 0 şi 1) şi o decizie de interpretare a acestei probabilităţi (da/nu). De asemenea este necesar un fişier de referinţă (RTTM) care să conţină o listă cu toate apariţiile cuvintelor cheie în toate secvenţele audio rostite. Imaginea de mai jos prezintă schema de evaluare a sistemului KWS:

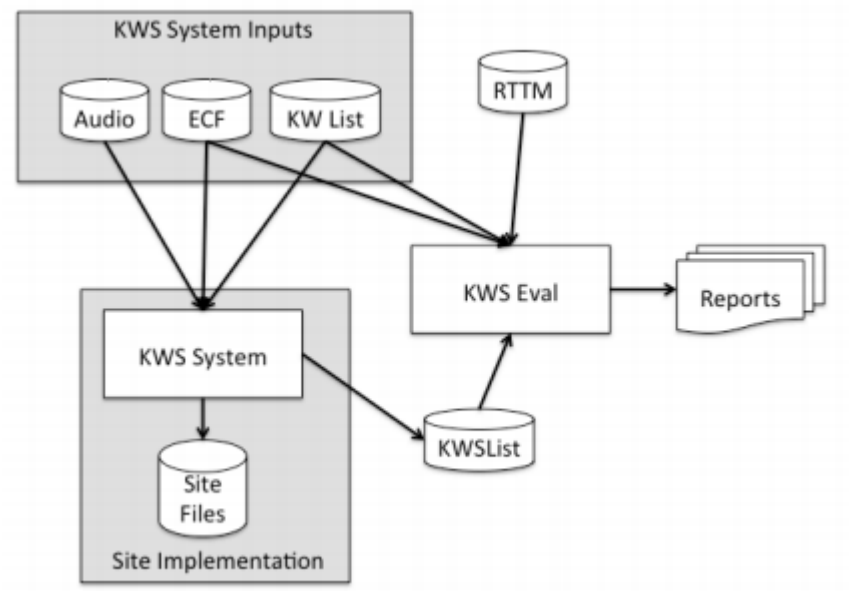


Figura 1.12 Schemă de evaluare a sistemului KWS. Sursa:[15]

Dacă existenţa unui cuvânt cheie în secvenţa rostită conduce la decizia "DA" a sistemul KWS, acest lucru va fi contabilizat ca o detecţie corectă altfel va fi contabilizat că o eroare de neidentificare. Dacă sistemul KWS ia decizia "DA" pentru un cuvânt cheie care în realitate nu apare în secvenţa rostită acest lucru va fi contabilizat că o eroare de alarmă falsă. Aceste decizii ale sistemului sunt bazate pe un prag: dacă probabilitatea de apariţie este mai mare sau egală decizia este "DA", altfel este "NU".

Probabilitatea erorii de neidentificare pentru un cuvânt cheie "q" şi un prag "ɵ"

$$Pmiss(q,\Theta) = \frac{Nmiss(q,\Theta)}{Nact(q)}$$

unde:

Nmiss(q,ɵ) = număr de erori de neidentificare pentru un cuvânt cheie "q" şi un prag "ɵ"

Nact(q) = numărul real de apariţii ale cuvântului "q"

Probabilitatea erorii de alarmă falsă pentru un cuvânt cheie "q" şi un prag "ɵ"

$$Pfa(q,\Theta) = \frac{Nfa(q,\Theta)}{Nnt(q)}$$

Nfa(q) = numărul de alarme false corespunzătoare unui cuvânt cheie "q" şi un prag "ɵ"

Nnt(q) = numărul maxim de alarme false

Nnt(q) = ntps * Taudio - Nact(q); ntps=1

Taudio = durata totală a secvenţelor vorbite (durata fişierelor audio)

Probabilităţile medii de eroare peste toate cuvintele cheie sunt date de formulele

$$Pmiss(\theta) = \frac{1}{|Q|}\sum Pmiss(q,\theta)$$

$$Pfa(\theta) = \frac{1}{|Q|}\sum Pfa(q,\theta)$$

Term-Weighted Value (TWV) este o combinaţie ponderată a probabilităţilor celor două erori de neidentificare respectiv de alarmă falsă, calculat cu următoarea relaţie:.

$$TWV(\theta) = 1 - (Pmiss(\theta) + \beta * Pfa(\theta))$$

Unde factorul β depinde de costul erorilor de neidentificare respectiv alarmă falsă.

Un alt parametru important este entropia normată Cnxe care măsoară cunoştinţele apriori pe care sistemul KWS le are asupra referinţei. Un sistem perfect ar trebui să aibe un parametru Cnxe care tinde la zero.

Curba de detecţie a erorii (DET curve) este un grafic unde axele sunt reprezentate de cele două probabilităţi de eroare (Pmiss(ɵ),Pfa(ɵ)), aceasta fiind un compromis între acestea.

CAPITOLUL 2

CONSTRUCŢIA UNOR SISTEME DE RECUNOAŞTERE AUTOMATĂ A VORBIRII

2.1 CREAREA SISTEMULUI "RODIGITS"

"Rodigits" este un sistem de recunoaştere automată a vorbirii continue cu vocabular redus în limba română. Scopul acestuia este de a realiza transcrierea unor clipuri audio a câte 12 cifre fiecare. Au fost dezvoltate două mari versiuni ale sistemul "Rodigits", atât cu ajutorul utilitarului CMU Sphinx, cât şi cu ajutorul utilitarului Kaldi.

Crearea acestui sistem a dus la îndeplinirea a două obiective propuse:

1. Obţinerea unei acurateţi cât mai mari pentru a putea folosi ulterior sistemul în cadrul aplicaţiei descrisă în capitolul 4, "Serviciul web de autentificare prin voce".
2. Realizarea unei comparaţii directe între cele două utilitare pentru recunoaşterea automată a vorbirii.

2.1.1 Baza de date audio "Rodigits"

Această bază de date de clipuri audio a fost folosită atât pentru antrenarea modelelor acustice ale sistemului, cât şi pentru evaluarea sa. La crearea înregistrărilor audio au participat studenţi din cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei, fiecare furnizând un set de 100 de clipuri audio a câte 12 cifre fiecare.

2.1.2 Antrenarea modelelor în vederea antrenării folosind utilitarul CMU Sphinx

Modelul fonetic

Acesta a fost creat prin transcrierea fonetică a fiecărei cifre din sistemul zecimal. Utilitarul CMU Sphinx foloseşte modele fonetice în format text, iar manualul acestuia recomandă utilizarea de foneme specifice pentru fiecare cuvânt. Fiecare cuvânt va trebui să modeleze în mod diferit fonemele în funcţie de poziţia la care acestea apar în cuvânt. Fişierul ce conţine modelul fonetic arată în felul următor:

```
zero z_zero e_zero r_zero o_zero
unu u_unu1 n_unu u_unu2
doi d_doi o_doi i3_doi
trei t_trei r_trei e_trei i3_trei
patru p_patru a_patru t_patru r_patru u_patru
cinci k1_cinci1 i_cinci1 n_cinci k1_cinci2 i_cinci2
şase s1_şase a_şase s_şase e_şase
şapte s1_şapte a_şapte p_şapte t_şapte e_şapte
opt o_opt p_opt t_opt
nouă n_nouă o1_nouă w_nouă a1_nouă
```

Modelul de limbă

Deoarece acest sistem de recunoaştere automată a vorbirii are un vocabular foarte redus, modelul de limbă folosit este unul bazat pe reguli, de tip gramatică cu stări finite (FSG).

Pentru a obţine gramatica într-un format compatibil cu utilitarul CMU Sphinx, se va crea un fişier în formatul Java Speech Gramar (JSGF):

```
#JSGF V1.0;
grammar rodigits;
public <numbers> = (zero | unu |doi | trei | patru | cinci |şase| şapte | opt |nouă)*;
```

În antetul gramaticii se poate observa identificatorul ce indică faptul că fişierul conţine o defnitie în format JSGF, precum şi numele gramaticii, "rodigits". În corpul gramaticii se regăseşte regula ce defineşte variabila "numbers". Separarea prin caracterul de disjuncţie logică "|" a cuvintelor ce reprezintă cifrele din sistemul zecimal, semnifică faptul că variabilei "numbers" îi poate fi atribuită una dintre aceste valori. Caracterul asterisc indică posibilitatea cuvintelor de a se repeta.

Automatul cu stări finite rezultat în urma compilării modelului de limba va fi sub forma unui graf, unde nodurile reprezintă cuvintele. Arcele grafului reprezintă tranziţiile între cuvinte şi indică modul în care este permisă parcurgerea grafului, implicit înlănţuirea cuvintelor.

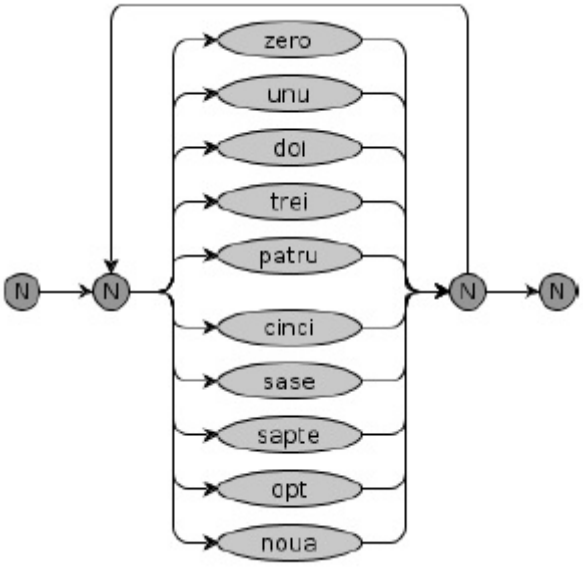


Figura 2.1 Modelul de limba de tip gramatică Rodigits (CMU Sphinx). Sursa:[16]

Nodurile notate cu "N" sunt noduri de infrastructură. Ele asigură intrarea şi de ieşirea din gramatica, precum şi tranziţiile înapoi.

Modelul acustic

Pentru crearea modelului acustic este necesară existenţa următoarelor resurse: baza de date de clipuri audio, transcrierea text a clipurilor audio şi modelul fonetic creat anterior. Pentru o mai bună acurateţe, acesta poate fi îmbunătăţit prin adăugarea unor elemente ce modelează fonemele nonverbale: linişte, zgomot, etc.

Deoarece s-a dorit crearea unui sistem de recunoaştere automată a vorbirii cât mai robust, au fost create mai multe modele acustice ce au fost testate şi optimizate treptat, până la obţinerea unor rezultate satisfăcătoare. Pentru a înţelege mai bine necesitatea parcurgerii acestor paşi intermediari, va fi prezentat pe scurt modul în care CMU Sphinx realizează antrenarea acustică. Folosind configuraţia standard a utilitarului, modelul acustic este construit cu unităţi de vorbire dependente de context de tip trifonem. Acestea sunt implementate cu ajutorul modelelor Markov ascunse cu 3 stări emisive, unde fiecărei stări îi corespunde o distribuţie de probabilitate de ieşire implementată cu ajutorul unei mixturi de Gaussiene. Datorită asemănării unor trifoneme, ele pot avea stari-model comune. Profitând de acest fapt şi având în vedere numărul foarte mare de stări, s-a recurs la gruparea lor pe bază de similaritate, un astfel de grup fiind numit senone. În mod implicit, CMU Sphinx foloseşte 200 de senone, fiecare senonă modelând parametrii acustici de ieşire cu ajutorul a 8 densităţi de probabilitate Gaussiene.

Prin modificarea configuraţiei standard, modelele dependente de context au suferit variaţii în ceea ce priveşte numărul de senone utilizat, precum şi numărul de densităţi Gaussiene. De asemenea, au fost efectuate teste şi cu modele independente de context, ce folosesc o singură densitate Gaussiana pentru fiecare fonem.

2.1.3 Experimente şi rezultate

Experiment 1

Acest prim experiment constă în crearea unui **model acustic dependent de vorbitor**. Au fost selectate din baza de date "Rodigits" numai cele 100 de fişiere audio proprii. Acestea au fost împărţite în felul următor: pentru etapa de antrenare au fost folosite 80 de fişiere (primele 50 şi ultimele 30), iar pentru etapa de testare s-au folosit restul de 20 de fişiere.

A. Rezultatele modelului indepedent de context:

WER[%]	22.9
SER[%]	100

Tabelul 2.1 Experiment 1. Rezultatele modelului independent de context

B. Rezultatele modelului depedent de context:

WER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	5.4	0.8	0	0
	200	14.2	7.1	10.4	12.1

Tabelul 2.2 Experiment 1. Rezultatele modelului dependent de context (WER)

SER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	35	10	0	0
	200	70	50	70	70

Tabelul 2.3 Experiment 1. Rezultatele modelului dependent de context (SER)

Se observă că modelul independent de context oferă o acurateţe cu certitudine mai slabă decât modelul dependent de context. Acesta din urmă realizează o transcriere perfectă în situaţia folosirii a 100 de senone şi un număr de densităţi gaussiene cel puţin egal cu 4.

Experiment 2

Acest experiment presupune **evaluarea modelului dependent de vorbitor cu date de la alţi 5 vorbitori necunoscuţi**, fiind alese câte 20 de înregistrări de la fiecare dintre aceştia.

A. Rezultatele modelului indepedent de context:

WER[%]	59.6
SER[%]	99

Tabelul 2.4 Experiment 2. Rezultatele modelului independent de context

B. Rezultatele modelului depedent de context:

WER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	69.8	63.2	65.2	67.6
	200	61.5	70.4	78.2	85.1

Tabelul 2.5 Experiment 2. Rezultatele modelului dependent de context (WER)

SER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	100	88	85	91
	200	99	99	100	100

Tabelul 2.6 Experiment 1. Rezultatele modelului dependent de context (SER)

Aşa cum era previzibil, rezultatele sunt destul de slabe. Acest lucru semnifică faptul că modelul dependent de vorbitor nu are capacitatea să se adapteze în cazul unor vorbitori necunoscuţi.

Experiment 3

Având în vedere concluzia de la experimentul precedent, s-a trecut la crearea unui **model independent de vorbitor**. Au fost selectaţi din baza de date "Rodigits" 50 de vorbitori. Cele 100 de fişiere ale fiecăruia au fost împărţite în felul următor: pentru etapa de antrenare au fost folosite câte 80 de fişiere de la fiecare vorbitor (primele 50 şi ultimele 30), în timp ce restul fişierelor au fost folosite în etapa de testare. Aşadar, **testarea se va face cu vorbitori cunoscuţi** sistemului, aceştia fiind identici cei implicaţi în procesul de antrenare.

A. Rezultatele modelului indepedent de context:

WER[%]	36.5
SER[%]	95.3

Tabelul 2.7 Experiment 3. Rezultatele modelului independent de context

B. Rezultatele modelului depedent de context:

WER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	16.9	10.9	5.1	4.3
	200	21	12.9	9	8.5

Tabelul 2.8 Experiment 3. Rezultatele modelului dependent de context 1 (WER)

SER[%]		**Densităţi Gaussiene per senone**			
		1	**2**	**4**	**8**
Senone	**100**	65.7	52.2	33.2	29.7
	200	77	65.8	51.6	45

Tabelul 2.9 Experiment 3. Rezultatele modelului dependent de context 1 (SER)

Se observă că în acest caz al antrenării şi testării folosind fişierele a 50 de vorbitori, modelul optim este cel cu 100 senone şi 8 densităţi gaussiene. Având în vedere rezultatele de până la acest punct, este evident faptul că modelele dependente de context ofertă o acurateţe superioară celor independente de context. Deoarece scăderea ratei de eroare se produce concomitent cu creşterea numărului de densităţi gaussiene, s-a trecut la testarea cu mai multe densităţi gaussiene. De asemenea, deoarece modelele ce folosesc 100 de senone au rezultate mai bune decât cele cu 200 de senone, testele următoare s-au efectuat folosind numai 100 senone.

WER[%]		**Densităţi Gaussiene per senone**			
		16	**32**	**64**	**128**
Senone	**100**	3.3	2.7	2.2	1.7

Tabelul 2.10 Experiment 3. Rezultatele modelului dependent de context 2 (WER)

SER[%]		**Densităţi Gaussiene per senone**			
		16	**32**	**64**	**128**
Senone	**100**	27.3	22.7	18.5	13.6

Tabelul 2.11 Experiment 3. Rezultatele modelului dependent de context 2 (SER)

Următoarele grafice prezintă tendinţa de scădere a ratelor de eroare în raport cu creşterea densităţilor gaussiene.

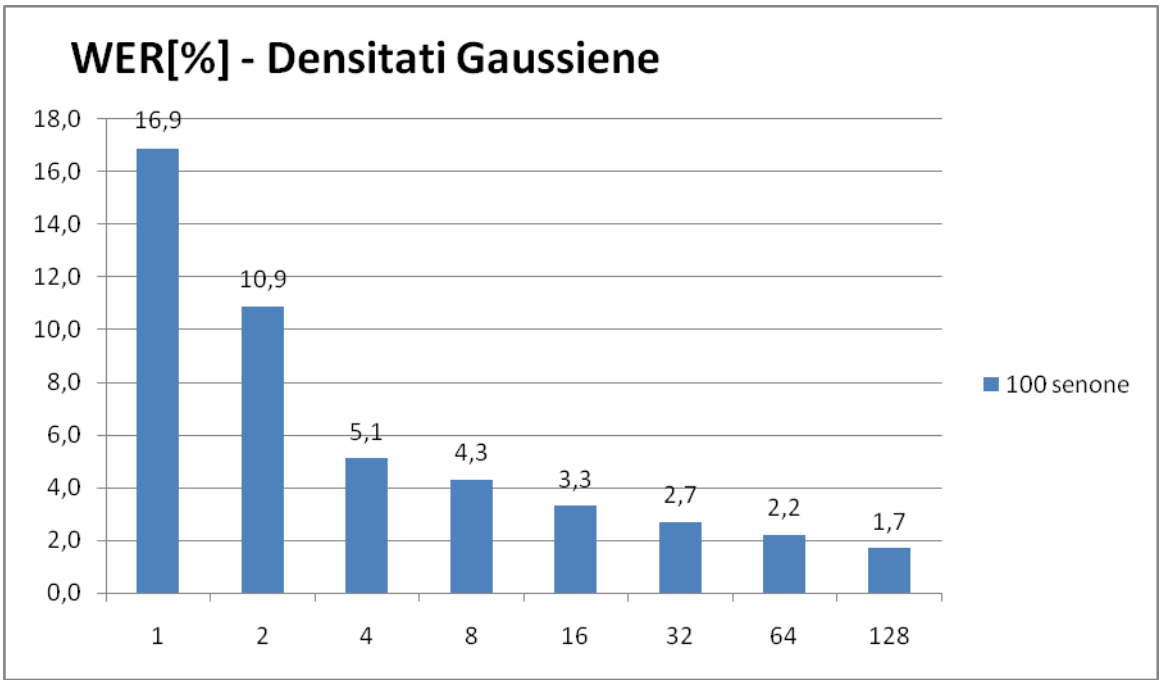


Figura 2.2 Raport WER - Densităţi gaussiene

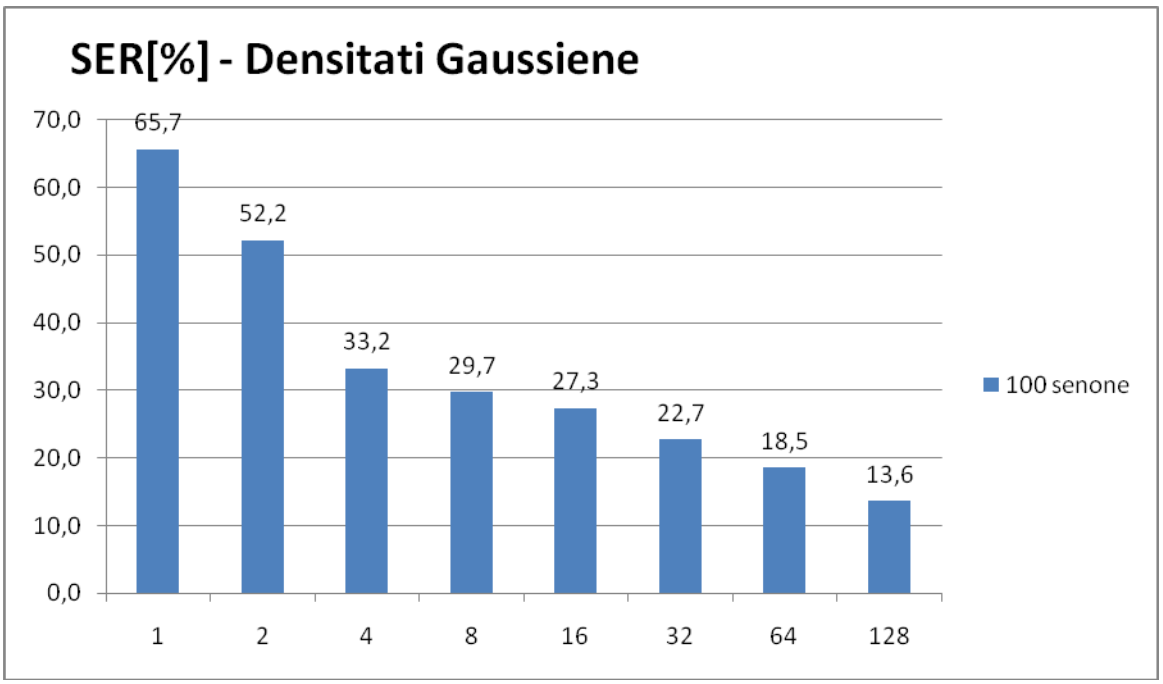


Figura 2.3 Raport SER - Densităţi Gaussiene

Mai departe, având în vedere performanţele crescute ale modelelor cu 100 de senone în comparaţie cu cele cu 200 de senone, s-a încercat o nouă scădere a acestei valori. Însă, aşa cum se poate observa din tabelul următor, acest test nu a condus la o îmbunătăţire a rezultatelor.

WER[%]		**Densităţi Gaussiene per senone**				
		1	**2**	**4**	**8**	**128**
Senone	**100**	16.9	10.9	5.1	4.3	1.7
	50	16.9	10.9	5.1	3.3	2

Tabelul 2.12 Variaţii ale densităţilor gaussiene şi senonelor în raport cu WER

În urma acestor experimente, s-a constatat faptul că **modelul optim este cel cu 100 senone şi 128 densităţi gaussiene**. Pentru a evidenţia caracterul independent de vorbitor al modelului, s-a efectuat testarea sa cu fişierele a **20 de vorbitori necunoscuţi.** Au fost obţinute următoarele rate de eroare:

- **WER: 1.1%;**
- **SER: 11%.**

Rezultatele obţinute sunt destul de interesante, mai ales din prisma faptului că eroarea pentru vorbitorii necunoscuţi (1.1%) este mai mică decât cea pentru vorbitorii cunoscuţi (1.7%). Pentru a afla cauza, s-a trecut la analiza erorii la nivel de cuvânt pentru fiecare vorbitor.

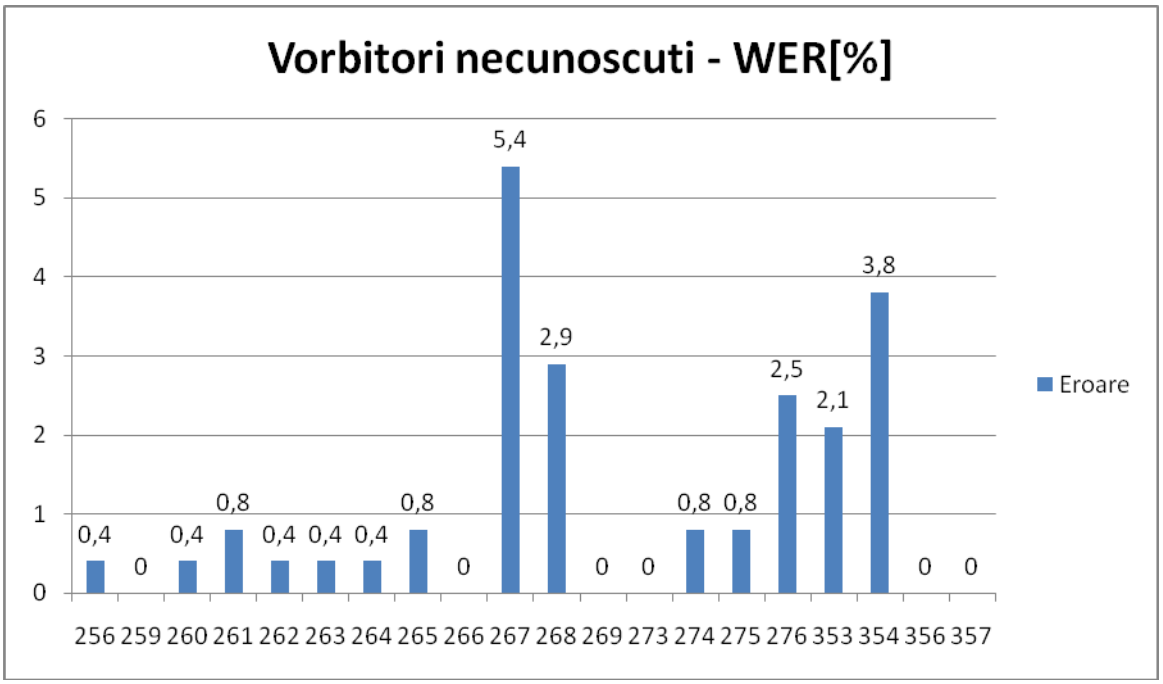


Figura 2.4 Evaluarea vorbitorilor necunoscuţi (WER)

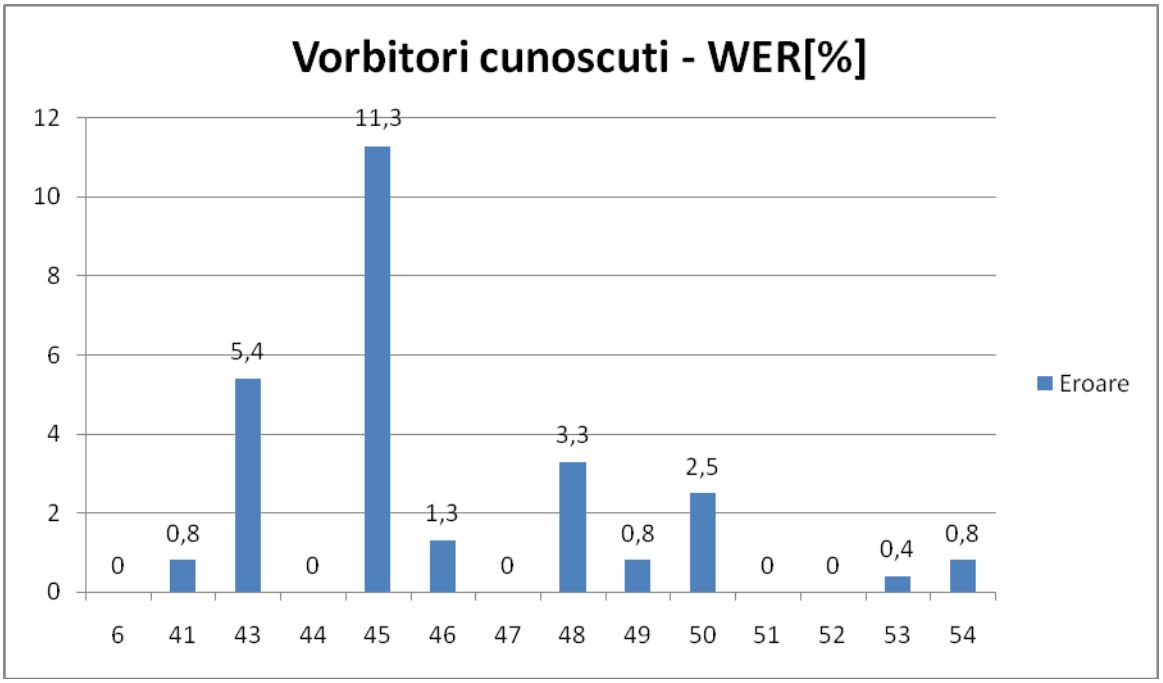


Figura 2.5 Evaluarea vorbitorilor cunoscuţi 1 (WER)

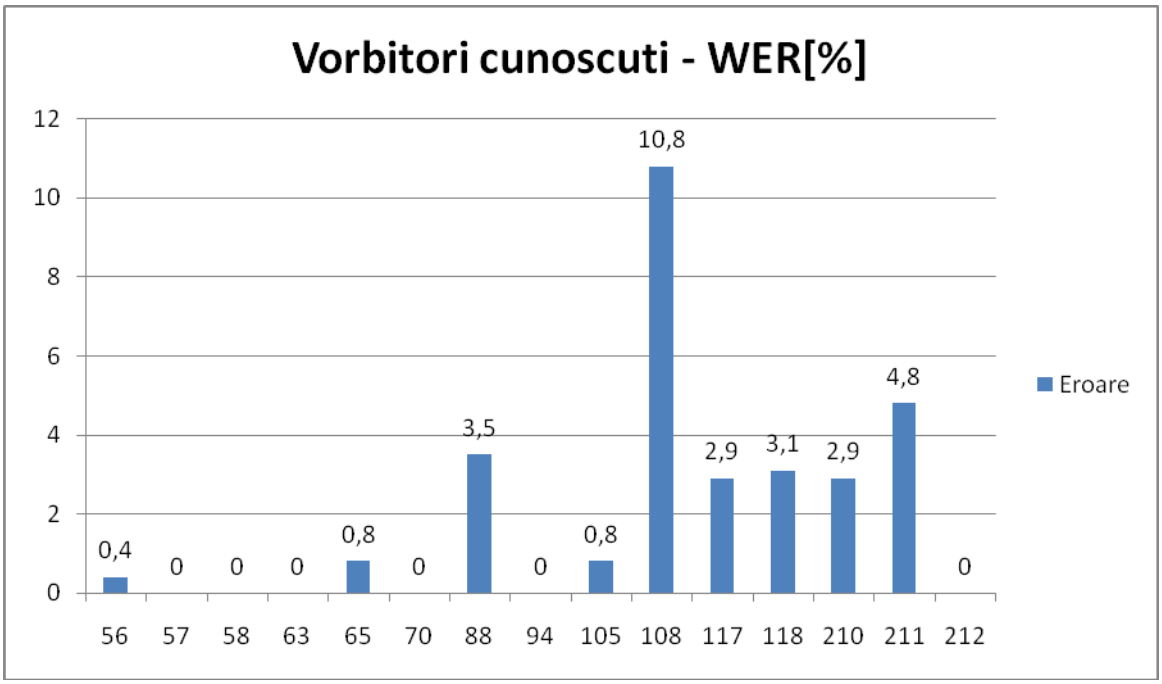


Figura 2.6 Evaluarea vorbitorilor cunoscuţi 2 (WER)

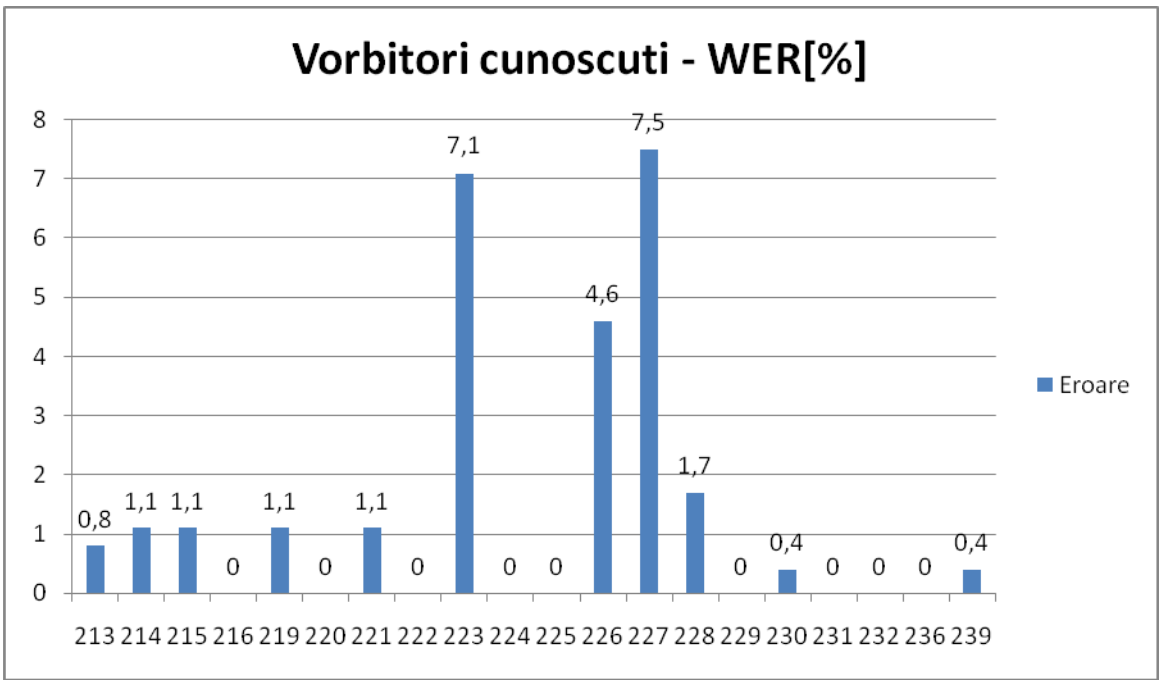


Figura 2.7 Evaluarea vorbitorilor cunoscuţi 3 (WER)

La vorbitorii necunoscuţi, cea mai mare rată de eroare este de 5.4% per vorbitor, în timp ce la vorbitorii cunoscuţi, apar doi vorbitori rate de eroare cu valori puţin peste 10%. În acest moment, rezultatele anterioare sunt explicabile. Prin ascultarea înregistrărilor audio de la cei doi vorbitori cu valori peste 10%, s-a constatat:

Vorbitorul 45 (masculin)

- Înregistrarea 45: rosteşte prima cifră înainte de începerea înregistrării;
- Înregistrarea 58: rosteşte o singură cifră;
- Pronunţia este foarte accentuată, nenaturală, iar silabele sunt alungite.

Vorbitorul 108 (feminin)

- Înregistrarea 24: lipseşte prima cifră;
- Înregistrarea 92: sunt rostite primele 4 cifre din grup, restul lipsesc;
- Pronunţă încet, uneori neclar, fiind foarte asemănător modul în care rosteşte cuvintele *şase* şi *şapte*.

2.1.4 Prepararea modelelor în vederea antrenării folosind utilitarul Kaldi

Modelul fonetic

Acesta a fost realizat prin transcrierea fonetică a cuvintelor, similar cu cel creat în cadrul proiectului folosind utilitarul CMU Sphinx. Se poate preciza faptul că de data aceasta nu a fost necesar să se precizeze poziţia fonemului în cuvânt.

```
<SIL> SIL
zero z e r o
unu u n u
doi d o i3
trei t r e i3
patru p a t r u
cinci k1 i n k1 i
şase s1 a s e
şapte s1 a p t e
opt o p  t
nouă n o1 w a1
```

Modelul de limbă

Au fost create două modele de limbă separate:

1. Un model de limbă **probabilistic de tip n-gram** realizat cu ajutorul utilitarului SRI-LM. Acesta a primit la intrare un corpus de text (în acest caz lista cuvintelor ce desemnează cifrele în limba română) şi a generat probabilitatea de apariţie a fiecărui cuvânt sau grup de cuvinte:

```
\data\
ngram 1=11

\1-grams:
-1     zero
-1     unu
-1     doi
-1     trei
-1     patru
-1     cinci
-1     şase
-1     şapte
-1     opt
-1     nouă
-99 <s>
-1 </s>

\end\
```

Înaintea fiecărui cuvânt se regăseşte probabilitatea să de apariţie, exprimată că logaritm în baza 10. Modelul a fost compilat sub forma unui automat cu stări finite (FST) cu ajutorul utilitarului OpenFST, iar graful corespunzător poate fi vizualizat mai jos:

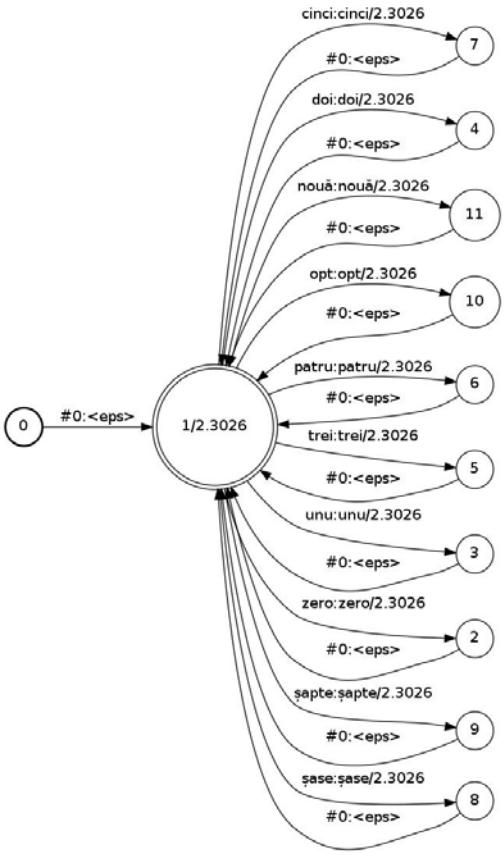


Figura 2.8 Modelul de limba 1-gram Rodigits (Kaldi)

Aşa cum se poate observa din imagine, există un nod de intrare în graf, urmat de un nod al stării iniţiale şi 10 noduri finale, corespunzătoare fiecărui cuvânt. Rostirea unui cuvânt va determina o tranziţie din nodul iniţial într-un nod final. Arcele inverse fac posibilă întoarcerea în nodul inţial, fiind astfel realizabilă o nouă tranziţie. Fiecare arc are asociată o probabilitate de tranziţie. În cazul curent ele sunt echiprobabile.

2. Un **model bazat pe reguli de tip gramatică cu stări finite** realizat cu ajutorul utilitarului Thrax.

```
numbers= "[2]" | "[3]" | "[4]" | "[5]" | "[6]" | "[7]" | "[8]" | "[9]" | "[10]" | "[11]";
export rodigits=Optimize[ numbers+];
```

Kaldi nu lucrează direct cu cuvinte în format text, ci foloseşte o tabelă de indexare a acestora. Simbolurile "[2]"..."[11]" reprezintă indecşii cuvintelor şi sunt separate prin caracterul de disjuncţie logică "|", semnificând faptul că variabilă "numbers" poate lua oricare dintre aceste valori. Simbolul plus indică posibilitatea cuvintelor de a se repeta. În urma compilării a rezultat graful următor:

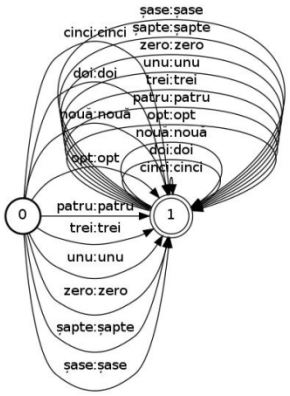


Figura 2.9 Modelul de limba de tip gramatică Rodigits (Kaldi)

O prima rostire va determina o tranziţie din starea #0 în starea #1. Următoarele rostiri vor avea ca efect tranziţii repetate din starea #1 în starea #1.

Modelul acustic

Au fost folosiţi mai mulţi algoritmi disponibili în Kaldi, fiecare generând un tip de sistem acustic diferit. Au fost păstrate valorile standard ale parametrilor acustici.

2.1.5 Experimente şi rezultate. Compararea utilitarelor

Pentru efectuarea experimentelor, au fost folosiţi 85 de vorbitori din baza de date "Rodigits", fiecare având un set de 100 de fişiere. Acestea au fost împărţite în felul următor:

- Setul de antrenare:
  - 60 de vorbitori, 80 de fişiere de la fiecare;
- Setul de testare:
  - Vorbitori cunoscuţi: cei 60 folosiţi la antrenare, restul de 20 de fişiere de la fiecare;
  - Vorbitori necunoscuţi: 25 de vorbitori, 20 de fişiere de la fiecare.

Experiment 1

În urma antrenării şi testării folosind utilitarul Kaldi au fost obţinute următoarele rezultate:

		Model de limba N-gram		Model de limbă FST	
Tipul modelului acustic	**Parametri acustici**	**WER[%]**		**WER[%]**	
		Vorbitori cunoscuţi	**Vorbitori necunoscuţi**	**Vorbitori cunoscuţi**	**Vorbitori necunoscuţi**
MONOPHONE		1.5	1.68	2.31	2.5
TRI1	**LeavesTri1=100**	2.21	2.05	3.46	3.12
	GaussTri1=256				
TRI2 LDA+MLLT	**LeavesMLLT=2500**	0.56	1.37	0.76	2.25
	GaussMLLT=15000				
TRI3	**LeavesSAT=100**	1.13	0.62	1.53	0.97
LDA+MLLT+SAT	**GaussSAT=256**				
SGMM2	**GaussUBM=128**	0.43	0.28	0.7	0.73
	LeavesSGMM=100				
	GaussSGMM=128				

Tabelul 2.13 Experiment 1. Rezultate Rodigits (Kaldi)

Se observă că anumite modele acustice din Kaldi obţin o acurateţe mai bună decât celelalte, SGMM2 fiind în această situaţie cel mai bun. Acelaşi lucru se poate spune despre modelul de limba n-gram, în comparaţie cu cel de tip gramatică. Deşi ambele modele de limbă oferă probabilităţi egale de tranziţie între cuvinte, totuşi rezultatele sunt diferite. În momentul actual nu se cunoaşte cauza, acest lucru fiind investigat.

Experiment 2

Utilizând în mod identic setul de antrenare, precum şi cele de testare descrise anterior, experimentul a fost replicat folosind utilitarul CMU Sphinx, parametrii acustici fiind cei determinaţi că optimi în secţiunea 3.1.3.

		Model de limbă FST	
Tipul sistemului acustic	**Parametri acustici**	**WER[%]**	
		Vorbitori cunoscuţi	**Vorbitori necunoscuţi**
HMM-GMM	**Densităţi gaussiene=128**	0.8	1.5
	Senone=100		

Tabelul 2.14 Experiment 2. Rezultate Rodigits (CMU Sphinx)

Comparând rezultatele din cele două experimente, cel puţin pentru configuraţiile experimentale prezentate, Kaldi pare să ofere o acurateţe mai bună în comparaţie cu CMU Sphinx.

2.2 CREAREA SISTEMULUI "MOLINA"

"Molina" este un sistem de recunoaştere automată a vorbirii continue cu vocabular relativ redus (215 cuvinte unice) în limba engleză. Scopul acestuia este de a realiza transcrierea unor clipuri audio ce conţin fraze care imită răspunsul unui sistem interactiv cu răspuns vocal. În continuare, se vor prezenta detalii referitoare la achiziţia şi structura bazei de date de evaluare, procedurile

de creare a modelelor componente (acustic, fonetic, lingvistic), dar şi rezultate privind acurateţea şi performanţele sistemului.

2.2.1 Baza de date audio "Molina"

Aceasta reprezintă baza de date de evaluare a sistemului ASR. Pentru crearea ei a fost selectată o listă de 75 de fraze, similare sau chiar identice cu cele ale sistemului IVR Molina [17]. Aceste fraze simulează răspunsul unui sistem IVR şi au ca scop ghidarea utilizatorilor acestuia, oferindu-le informaţiile dorite. Mai jos se regăsesc câteva exemple:

0001: Welcome to the Interactive Voice Response System for Molina Healthcare of Wisconsin. Please listen carefully because our options have changed.

0002: Press 1 for Eligibility Information.

0003: Press 2 for Claim Status.

0004: Press 3 for an authorization Status.

0005: If you would like to hear these chioces again, press 9.

Este important de specificat faptul că frazele au un format specific, acestea fiind alcătuite din **părţi statice** şi **părţi dinamice**. Frazele care diferă numai prin părţile lor dinamice, în timp ce părţile statice sunt identice, se consideră că aparţin aceluiaşi **tip de frază**:

Fraza 1: If your credit is five thousand dollars, press one.

Fraza 2: If your credit is seventy three dollars, press two.

Tip frază: If your credit is NUMBER dollars, press UNITS.

În exemplul dat, cele două fraze conţin câte două părţi dinamice, marcate prin culori diferite. Atât "five thousand" cât şi "seventy three" aparţin părţii dinamice denumite în mod generic "NUMBER" (ce poate reprezenta orice număr), în timp ce "one" şi "two" aparţin părţii dinamice denumite în mod generic "UNITS" (ce poate reprezenta orice cifră). În baza de date "Molina" se disting 42 de tipuri de fraze.

La crearea înregistrărilor audio din această bază de date au participat 5 membri ai Laboratorului de Cercetare "SpeeD" din cadrul Facultăţii de Electronică, Telecomunicaţii şi Tehnologia Informaţiei, fiecare furnizând un set de 75 fraze. S-a folosit o aplicaţie web ce afişează fiecare frază şi permite utilizatorului să înregistreze, să asculte şi să reînregistreze în cazul în care este necesar. Aşadar, baza de date conţine un număr de 375 fraze şi are o durată totală de 42 minute.

2.2.2 Antrenarea modelelor

Modelul acustic

A fost obţinut prin antrenarea unei bazei de date în limba engleză numită "TED-LIUM"[11] ce conţine atât fişierele audio cât şi transcrierile de la conferinţele "TedTalks" [13]. Tabelul de mai jos conţine caracteristicile acestei baze de date:

Caracteristică	**Antrenare**	**Dezvoltare**
Număr de conversaţii	774	19
Durata totală	118 h, 4 m, 48 s	4 h, 12 m, 55 s

• **Vorbitori masculini** • **Vorbitori feminini**	81 h, 53 m, 7s 36 h, 11 m, 41 s	3 h, 13 m, 57 s 58 m, 58 s
Durata medie	9 m, 9 s	13 m, 18 s
Număr de vorbitori unici	666	19
Număr de segmente	56.6 K	2 K
Număr de cuvinte	1.7 M	47 K

Tabelul 2.15 Caracteristicile bazei de date TED-LIUM

Sistemul de recunoaştere automată a vorbirii "TED-LIUM" este unul din proiectele oferite ca exemplu de către creatorii utilitarului Kaldi. Prin descărcarea acestuia, utilizatorului îi sunt oferite toate componentele necesare pentru crearea sistemului ASR (baze de date de antrenare, de dezvoltare şi de testare împreună cu transcrierile aferente, modelul fonetic şi modelul de limba). De asemenea, este pus la dispoziţie un fişier cu diverse comenzi asupra modulelor din utilitar, iar prin simpla rulare a lui se va realiza antrenarea sistemului, rezultând mai multe modele acustice, fiecare model fiind obţinut cu ajutorul unui algoritm ce corespunde unui tip de sistem acustic diferit. La final este realizată decodarea fişierelor de test şi se evaluează performanţa sistemului.

Modelul de limbă corespunzător sistemului "TED-LIUM" este unul de tip probabilistic, creat pe baza trascrierilor conferinţelor TedTalks şi conţine probabilităţile de apariţie pentru:

- 1-gram: 19794 termeni;
- 2-gram: 1377200 termeni;
- 3-gram: 3178194 termeni.

Tabelul de mai jos prezintă o comparaţie între rezultatele oficiale (obţinute de creatorii utilitarului) şi rezultatele personale ale erorii la nivel de cuvânt.

Tipul sistemului acustic	**Setul de date**	**Rezultate WER[%]**	
		Rezultate oficiale	**Rezultate personale**
TRI1	**dev**	36.0	36.0
	test	34.7	34.6
TRI2	**dev**	32.1	31.8
	test	29.8	29.9
TRI3	**dev**	27.4	27.4
	test	24.6	24.9
TRI3 MMI	**dev**	24.0	24.3
	test	21.6	21.3

Tabelul 2.16 Rezultatele antrenării bazei de date TED-LIUM

Având în vedere că valorile obţinute prin rularea programului sunt foarte apropiate de cele ale creatorilor utilitarului, acest lucru indică faptul că proiectul oferit ca exemplu a rulat corespunzător. În felul acesta au fost obţinute câteva modele acustice pentru limba engleză ce vor fi folisite mai departe în sistemul Molina ASR.

Modelul fonetic

A fost creat prin transcrierea fonetică manuală a fiecărui cuvânt ce se găseşte în baza de date de evaluare. Modelul poate conţine mai multe transcrieri ale unui cuvânt dacă pentru acesta există mai multe tipuri de pronunţie.

Mai jos este un extras din transcrierea fonetică creată:

a AH
access AE K S EH S
after AE F T ER
again AH G EH N
amount AH M AW N T
an AE N
and AE N D
and AH N D
another AH N AH DH ER
any EH N IY
are ÂÂ R
ashley AE SH L IY
asked AE S K T
asked AO S K T

De asemenea, în compunerea modelului fonetic se regăseşte o listă a tuturor fonemelor existente precum şi o listă a fonemelor nonverbale.

Fonemele limbii engleze	
AA	L
AE	M
AH	N
AO	NG
AW	OW
AY	OY
B	P
CH	R
D	S
DH	SH
EH	T
ER	TH
EY	UH
F	UW
G	V
HH	W
IH	Y
IY	Z
JH	ZH
K	

Fonemele nonverbale (linişte, zgomot, sunete necunoscute, etc):

SIL
BRH

CGH
NSN
SMK
UM
UHH

Modelul de limbă

Deoarece aste sistem de recunoaştere automată a vorbirii are un vocabular relativ redus, modelul de limbă utilizat este unul creat special pentru această sarcină, fiind un model bazat pe reguli, de tip gramatică cu stări finite (FSG).

Acesta a fost creat folosind utilitarul Thrax, iar mai jos este exemplificată procedura de creare.

Modelul de limba de tip gramatică LM1-v1
f1="[214][201][197][121][211][180][195][100][144][110][155][219][166][135][64][52][163][160][109][67]";
f2="[170][2][100][88][119]";
f3="[170][20][100][72][194]";
f4="[170][27][100][41][49][194]";
……
f75="[115][224][76][123][35][85][170][2][115][150][170][20]";
fragment=f1|f2|f3|f4|f5|……|f75;
export molina=Optimize[fragment];

Variabilele f1, f2, f3 ...f75 iau ca valoare o serie de simboluri numerice ce reprezintă indexul fiecărui cuvânt din vocabularul bazei de date de evaluare. Astfel, aceste variabile compun frazele bazei de date. Caracterul "|" reprezintă operatorul de disjuncţie logică, acesta indică faptul că fraza decodată de sistem poate fi oricare dintre frazele f1, f2, ...f75.

Acest model de limba a fost compilat sub forma unui automat de stări finite (FST) folosind următoarele comenzi specifice Thrax:

thraxmakedep molina.grm
make
farextract Molina.far

Automatul cu stări finite rezultat poate fi vizualizat sub forma unui graf. Deoarece acesta este destul de complex, în imaginea de mai jos se regăseşte un fragment ce reprezintă primele 5 fraze:

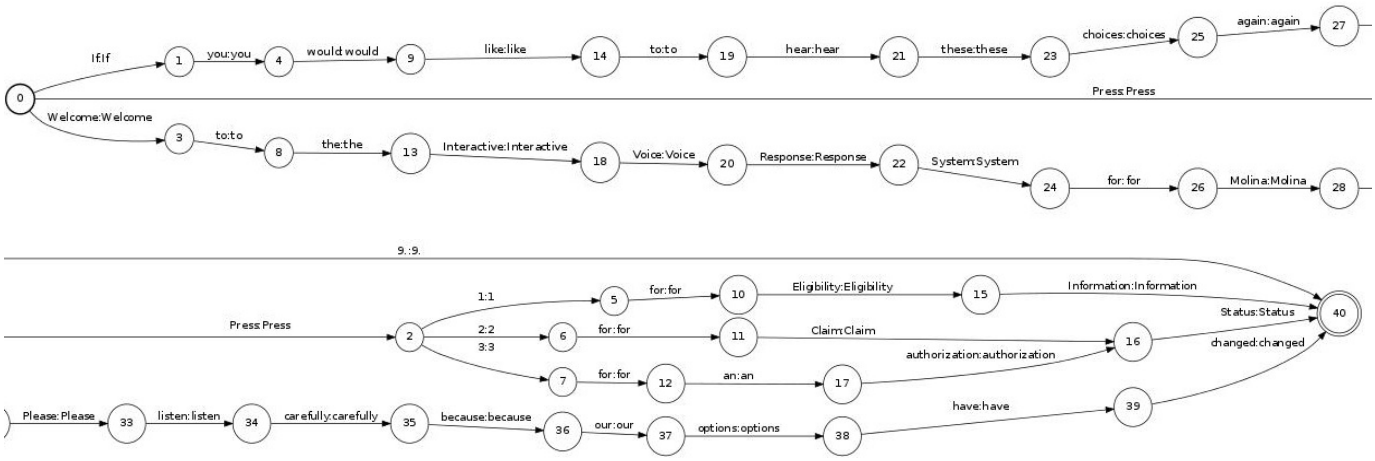


Figura 2.10 Graful corespunzator modelului de limba de tip gramatică Molina

Optimizarea modelului de limbă precedent

Modelul de limbă creat poate fi considerat ca fiind unul "static", unde fiecare frază este definită prin şirul de cuvinte componente. Mai mult, fiecare cuvânt este considerat de sine stătător, fără a se lua în calcul faptul că unele cuvinte sau grupuri de cuvinte pot fi obţinute prin concatenarea altor unităţi mai mici. Ţinând cont de acest fapt, dar şi de formatul specific al frazelor, s-a realizat trecerea la o nouă versiune a modelului de limbă bazat de reguli, un model dinamic, unde numerele, datele calendaristice, numele proprii să fie obţinute prin concatenarea cuvintelor componente. Mai precis, au fost create subgramatici pentru părţile dinamice ale frazelor, iar mai departe, frazele au fost definite prin concatenarea părţilor dinamice cu părţile statice. În tabelul următor sunt prezentate subgramaticile obţinute: O astfel de subgramatica pentru cazul numerelor formate din două cifre este prezentată mai jos:

```
units="[217]"|"[137]"|"[195]"|"[190]"|"[75]"|"[69]"|"[172]"|"[166]"|"[53]"|"[125]";
tens="[194]"|"[186]"|"[74]"|"[67]"|"[174]"|"[168]"|"[55]"|"[127]";
twoDigits=tens units;
```

Variabilă "units" poate fi orice cuvânt reprezentat de cifrele din intervalul [0;9], în timp ce variabilă "tens" poate fi orice cuvânt reprezentat de numerele din mulţimea [20; 30; 40; 50; 60; 70; 80; 90] (cuvintele sunt reprezentate prin simbolul echivalent din tabela de indexare a cuvintelor specifică utilitarului Kaldi). Variabilă "twoDigits" rezultă prin concatenarea celorlalte două variabile şi va corespunde tuturor numerelor cuprinse în intervalul [20;99]. În mod similar au fost realizate subgramatici pentru toate părţile dinamice existente. Tabelul următor reprezintă un extras din noua versiune de gramatică:

Modelul de limba de tip gramatică LM1-v2
f2 = "[104]" UNITS "[44]" "[38]" "[63]" ;
f3 = "[104]" UNITS "[44]" "[20]" "[122]" ;
f4 = "[104]" UNITS "[44]" "[33]" "[66]" "[122]" ;
.....
f19 = "[88]" FIRST_NAME LAST_NAME "[124]" DATE "[63]" "[30]" "[38]" "[44]" "[103]" "[104]" "[103]" "[93]" PLAN_NAME ;
.....
f42 = "[62]" "[141]" "[293]" "[63]" NUMBER "[34]" "[104]" UNITS "[62]" "[96]" "[104]" UNITS ;

```
phrases = f1 | f2 | f3 | f4 | f5 | f6 | f7 | f8 | f9 | f10 | f11 | f12 | f13 | f14 | f15 |f16 | f17 | f18 | f19 |
f20 | f21 | f22 | f23 | f24 | f25 | f26 | f27 | f28 | f29 | f30 | f31 | f32 | f33 | f34 | f35 | f35 | f36 | f37 |
f38 | f39 | f40 | f41 | f42 ;
export molina_lm = Optimize [ phrases+ ] ;
```

Concluzie: Aşa cum se va observa în secţiunea următoare, modelul de limba LM1-v1 va oferi o acurateţe mai bună decât modelul de limba LM1-v2. Acest lucru este normal, deoarece primul model impune o foarte mare stricteţe din punct de vedere al îmbinării cuvintelor în fraze, în timp ce al doilea permite o variabilitate mai mare. Totuşi, avantajul celui de-al doilea model este conferit de posibilitatea de a modifica sau de a adăuga o nouă subgramatica într-un mod mult mai facil.

2.2.3 Experimente şi rezultate

Experiment 1

Folosind modelele explicitate mai sus şi baza de date de evaluare Molina s-a obţinut următoarele rezultate de evaluare ale sistemului:

	Modelul de limba de tip gramatică LM1-v1	
Tipul sistemului acustic	**WER[%]**	**SER[%]**
MONOPHONE	1.56	3.73
TRI1	2.75	4.8
TRI2 **LDA+MLLT**	1.81	3.2
TRI3 **LDA+MLLT+SAT**	1.1	2.67
TRI3 **MMI**	0.84	2.13

Tabelul 2.17 Experiment 1. Rezultate Molina (LM1-v1)

Se poate observa că cele mai bune rezulatate s-au obţinut pentru modelul TRI3 MMI (triphone maximum mutual information) ce urmăreşte să maximizeze probabilitatea aposteriori a rostirii corecte.

În concluzie, frazele din baza de date de evaluare Molina au fost decodate cu o eroare la nivel de cuvânt de 0,84%, în timp ce eroarea la nivel de frază fiind de 12-13%.

Experiment 2

Au fost realizate teste de performanţă ale sistemului de calcul în timpul decodării bazei de date Molina folosind sistemul ASR Molina creat. Testele au fost create folosind două modele de limbă diferite. Primul este un model de limbă bazat pe reguli atât pentru fraze cât şi pentru părţile dinamice din acestea (a se vedea secţiunea de mai sus). Iar al doilea este un model de limba 3-gram pentru limba engleză împrumutat din proiectul exemplu Tedlium. Pentru aceste două modele s-a variat numărul de procese şi numărul de fire de excuţie, au fost create unele

proceduri pentru măsurarea consumului de memorie, a timpului de decodare, valoarea media de solicitare a CPU, factorul de timp real, accelerarea şi consumul de energie.

Pentru calculul valorii medii a solicitării CPU şi a memoriei a fost creat un script care prin realizarea unor apeluri la comandă "top", din sistemul de operare Linux, la fiecare secundă, pe toată durata procesului de decodare, oferă gradul de utilizare al memoriei şi al CPU. Aceste valori au fost însumate şi împărţite la durata totală a decodării.

Factorul de timp real (RTF) este un raport între durata decodării şi durata secvenţei audio decodate. Accelerarea arată de câte ori este mai rapidă sarcină de decodare atunci când sunt folosite mai multe fire de execuţie în comparaţie cu cazul în care este folosit unul singur.

Consumul de energie este calculat folosind utilitarul PowerAPI. În timpul decodării au fost monitorizate toate procesele corespunzătoare acestei sarcini şi au fost extrase eşantioane la interval de o secundă ce indică puterea instantanee consumată. A fost calculat consumul mediu de energie precum şi valorile instantanee maxime atinse.

Observaţie: La începutul sarcinii de decodare în timp ce decodorul din Kaldi este iniţializat (sunt încărcate modelele acustic, fonetic şi de limbă) consuml de energie creşte gradual. Apoi în timpul decodării propriu -zise are un caracter constant, iar când decodarea se termină şi latticele obţinute sunt evaluate, consumul de energie începe să scadă.

Sumar experiment:

Baza de date folosită: *Molina*

Durata totală a bazei de date: *2557 secunde, aproximativ 42 minute*

Model acustic folosit: *TRI3 MMI antrenat pe bază de date TED-LIUM*

Model de limbă folosit: *LM1-v2 (model bazat pe reguli special creat pentru baza de date Molina) şi LM2 (3-gram împrmutat din proiectul exemplu TED-LIUM)*

WER[%] folosind LM1-v2: *1.76*

WER[%] folosind LM2: *60.99*

LM1-v2	Număr procese	Număr fire de execuţie	Memorie [MB]	Timp decodare [s]	CPU [%]	RTF	Accelerare	Media consumului de putere [W]	Punct maxim de consum [W]
Test nr.									
1	1	1	48	171	4.43	0.067		2.73	2.8
2	1	4	50	48	16.1	0.018	3.56	10.34	11.3
3	1	8	81	28	29.13	0.0109	6.10	18.67	22.5
4	1	16	98	21	51	0.0082	8.14	31.69	45
5	1	24	116	21	67	0.0082	8.14	41.86	62
6	1	32	123	22	61	0.0086	7.77	40.4	61
7	2	1	89	96	8.39	0.0375		5.04	5.5
8	2	4	90	30	28.85	0.0117	3.20	18.16	22.5
9	2	8	148	22	51.3	0.0086	4.36	32.71	45
10	2	16	190	21	70.72	0.0082	4.57	46.87	64
11	2	24	227	21	71	0.0082	4.57	44.5	64.5
12	2	32	258	21	72	0.0082	4.57	45.88	65

Tabelul 2.18 Evaluarea consumului de resurse (LM1-v2)

LM2	Număr procese	Număr fire de execuţie	Memorie [GB]	Timp decodare [s]	CPU [%]	RTF	Accelerare	Media consumului de putere [W]	Punct maxim de consum [W]
Test nr.									
1	1	1	4.8	3778	4.37	1.477		2.6	2.8
2	1	4	4.86	1095	16.4	0.428	3.45	10.8	13.8
3	1	8	4.95	644	31.49	0.251	5.86	20.7	25
4	1	16	5.37	466	58	0.182	8.10	38.4	47.2
5	1	24	5.7	441	76	0.172	8.57	51	66
6	1	32	6	447	80	0.174	8.45	53	66
7	2	1	9.54	1919	8.53	0.75		5.45	6.63
8	2	4	9.42	648	31.43	0.253	2.96	20.6	26
9	2	8	9.59	478	59.24	0.186	4.01	40	49
10	2	16	10.47	441	89.75	0.172	4.35	56.17	65
11	2	24	11.22	448	84	0.175	4.28	56.15	66
12	2	32	11.94	455	86	0.177	4.21	57.11	65

Tabelul 2.19 Evaluarea consumului de resurse (LM2)

Concluzie: Se observă că la un anumit moment, chiar dacă numărul de fire de execuţie creşte, timpul de decodare nu este îmbunătăţit. Acest lucru este probabil datorat faptului că peste un anumit număr de fire de execuţie creat acestea nu mai sunt necesare. Kaldi foloseşte un număr optim de fire de execuţie în funcţie şi de volumul datelor ce urmează a fi procesate.

Modelul de limbă bazat pe reguli (LM1-v2) fiind creat special pentru frazele din baza de date Molina a dus la un consum de resurse mult mai bun decât la modelul de limba 3-gram (LM2).

De asemenea, se văd mari diferenţe în rata de eroare la nivel de cuvânt. Un model de limbă bazat pe reguli este mult mai indicat dacă se cunoaşte formatul frazelor şi este relativ uşor de creat atunci când se cunoaşte transcrierea bazei de date care se testează.

CAPITOLUL 3

CONSTRUCŢIA SISTEMULUI DE DETECŢIE A CUVINTELOR CHEIE

Capitolul curent are ca scop construcţia unui sistem de detecţie a cuvintelor cheie (KWS) folosind modulul special creat pentru această sarcină din utilitarul Kaldi. Acesta foloseşte metoda căutării prin lattice, descrisă în secţiunea 2.1, iar aşa cum am menţionat în respectiva secţiune este necesar mai întâi existenţa unui sistem ASR care să genereze aceste lattice. Sistemul KWS astfel creat trebuie să fie capabil să detecteze cuvinte cheie în limba engleză care se pot regăsi în frazele bazei de date Molina.

3.1 ANALIZA ASUPRA MODELELOR DE LIMBĂ FOLOSITE PENTRU DETECŢIA CUVINTELOR CHEIE

3.1.1 Modelul de limba 3-gram

O abordare a sarcinii de KWS este de a crea un sistem cu două componente:

- un modul ASR care decodează materialul audio generând lattice;

- un modul care caută cuvintele cheie în latticele indexate sau în mod direct prin transcrierea text.

În acest caz, importanţa sistemul ASR este foarte ridicată. O transcriere cât mai precisă a semnalului audio va asigura o bună performanţă a întregului sistem KWS. Deoarece s-a dispus de un model de limba 3-gram şi un model acustic, ambele dezvoltate pentru un sistem de recunoaştere în limba română a vorbirii continue cu vocabular extins (RVC-VE) s-a decis realizarea unui prim experiment folosind aceste modele. Scopul a fost observarea modului în care Kaldi se comportă atunci când este folosit un model 3-gram de o dimensiune considerabilă.

1-gram - 53700 termeni;

2-gram - 16427743 termeni;

3-gram - 23674351 termeni.

Pentru antrenarea modelului acustic s-a folosit o bază de date de vorbire citită în limba română şi o bază de date de vorbire înregistrată din emisiuni TV.

Pentru testarea acestui sistem ASR s-a folosit o bază de date mixtă, unele fişiere provenind din baza de date de vorbire în limba română iar altele din baza de date de vorbire înregistrată din emisiuni TV. Fişierele folosite pentru testare nu au fost cele utilizate la antrenare. În tabelul de mai jos sunt prezentate datele în urma evaluării sistemului.

WER[%] **Tipul sistemului acustic**	**Omega test**	**News all**	**News clean**	**News noise**
TRI1	12.37	29.77	23.04	34.97
TRI2 **LDA+MLLT**	11.34	28.94	22.39	33.74
TRI3 **LDA+MLLT+SAT**	9.75	27.56	20.47	33.07
TRI3 **MMI**	9.01	26.49	19.41	32.3

Tabelul 3.1 Rezultatele evaluării sistemului în limba română

"News all" reprezintă toate fişierele corespunzătoare emisiunilor de ştiri şi constă în două părţi:

- "news clean" (fişiere cu vorbire curată fără zgomot de fond)
- "news noise" (fişiere cu vorbire cu zgomote de fundal: râsete, muzică, etc.)

Se observă că performanţa este mai bună cu cât tipul modelului acustic este mai avansat. Rezultatele fişierelor fără zgomot sunt mai bune decât cele cu zgomot.

Mai departe sunt prezentate performanţele sistemului ASR în limba română cu model de limba 3-gram.

Test nr.	Număr procese	Număr fire de execuţie	Memorie	Timp de decodare	RTF	Comentariu
1	1	1	24.3 GB	72 min.	0.218	
2	1	4	24.3 GB	24 min.	0.072	
3	1	24	24.3 GB	12 min.	0.036	
4	2	1	47.04 GB	--	--	Creşte memoria swap

Tabelul 3.2 Consumul de memorie în cazul modelului de limba 3-gram

S-a încercat varierea numărului de procese cât şi al numărului de fire de execuţie şi s-au notat valorile corespunzătoare consumului de memorie, al timpului de decodare şi al factorului de timp real. S-a putut observa că resursele sunt partajate între firele de execuţie ale aceluiaşi proces dar sunt multiplicate în cazul existenţei mai multor procese. Acest lucru se poate observa în tabel unde consumul de memorie este identic pentru primele 3 teste iar pentru ultimul test fiind aproape dublu. În oricare din cazuri consumul de memorie este mult prea mare pentru folosirea modelului de limba 3-gram.

Concluzie: Sarcină de detecţie a cuvintelor cheie nu poate fi realizată printr-o metodă care să presupună obţinerea transcrierii complete. O transcriere completă folosind modelul de limba 3-gram nu este deloc potrivită din punct de vedere al consumului de memorie. Mai departe s-a dorit găsirea unei alternative mai eficiente.

3.1.2 Alternativă la modelul de limba 3-gram

Având în vedere concluziile din subcapitolul precedent s-a căutat o metodă prin care transcrierea să conţină numai cuvintele cheie căutate. Astfel au fost create câteva modele de limbă noi care conţin doar aceste cuvinte. S-au realizat următoarele teste din tabelul de mai jos:

Baza de date folosită pentru testare: Omega test
Durata totală a bazei de date: 329 minute
Model acustic folosit pentru decodare: Tri2 (antrenat pe limba română)
Număr de procese: 1
Număr de fire de execuţie: 24

Test nr.	Model de limbă	Număr de cuvinte	Memorie	Timp de decodare	RTF	Comentariu
1	gramatica: disjuncţie logică	10	144 MB	2 min.	0.006	
2	gramatica: disjuncţie logică	1000	672 MB	3 min.	0.009	
3	gramatica: disjuncţie logică +umplutură	12+ filler	6.24 GB	5 min.	0.38	Au fost decodate mai puţine fişiere (13 min.)
4	1-gram	53700	768 MB	10 min.	0.03	

Tabelul 3.3 Consumul de meorie în cazul modelului de limbă alternativ

Pentru primele două teste s-a folosit modele de limbă diferite bazate pe reguli ce conţin 10, repectiv 1000 de cuvinte cheie.

Cuvinte = cuvânt 1 | cuvânt 2 | cuvânt 3 | cuvânt 4 | ..... | cuvânt 10;

Modelul de limba a fost creat prin înşiruirea cuvintelor cheie utilizând operatorul de disjuncţie logică între acestea.

Desigur că transcrierea nu este relevantă în acest caz şi nu poate fi calculată o rată de eroare la nivel de cuvânt deoarece toate cuvintele ce se găsesc în baza de date de testare vor fi transcrise printr-unul din cuvintele din modelul de limbă. Nici evaluarea de detecţie a cuvintelor cheie nu ar avea sens deoarece probabilitatea erorilor de alarmă falsă va fi foarte mare în timp ce probabilitatea de neidentificare va fi nulă. Singurul scop al acestor teste a fost pentru observarea consumului de memorie care a fost mult mai scăzut decât în cazul modelelor 3-gram.

Cel de-al treilea test este similar cu primele două, diferenţa constând în folosirea unui cuvând de "umplutură" pe lângă cuvintele cheie căutate.

În ultimul test s-a folosit un model de limba n-gram, mai exact s-au păstrat doar termenii 1-gram din modelul 3-gram în limba română folosit anterior. Acest test a avut ca scop compararea consumului de memorie în cazul folosirii modelului 1-gram faţă de 3-gram. Consumul de memorie scade de aproximativ 30 ori (de la 24,3GB la 768MB), dar în schimb eroare la nivel de cuvânt creşte de aproximativ 3 ori (32% faţă de 11,34%).

Concluzie: se poate spune că modelul de limba cu "filler" utilizat în cadrul celui de-al treilea test poate fi o alternativă viabilă în cadrul unui sistem KWS. Chiar dacă consumul de memorie este mai mic decât în cazul modelului 3-gram este destul de mare în comparaţie cu primele două teste din tabel. Acest fapt poate fi explicat prin modul în care modelul de limba cu "filler" este tratat din punct de vedere fonetic. Cuvântul de "umplutură" apare în modelul fonetic ca fiind transcris prin fiecare fonem în parte. Considerând că un fonem are 3 stări, atunci cuvântul de "umplutură" va fi expandat ajungând la o dimensiune egală cu numărul de foneme la puterea 3. Deoarece limba română are 36 de foneme vor exista 36^3=46656 versiuni ale cuvântului de "umplutură".

3.2 CREAREA REFERINŢEI

Pentru a evalua acurateţea unui sistem de detecţie a cuvintelor cheie este necesar crearea unui model de referintă ce indică momentul de timp al apariţiei pentru fiecare cuvânt cheie din baza de date de evaluare. Această operaţie s-a realizat manual. A fost aleasă o listă de 20 termeni ce se găsec în baza de date Molina, au fost ascultate înregistrările audio şi pentru fiecare fişier s-a notat momentul de timp şi durata apariţiei fiecărui cuvânt cheie. În acest fel a fost etichetată baza de date şi a fost creat un fisiser de referinţă respectând standardul RTTM. Mai jos este prezentat un extract din acest fişier:

SPEAKER 006_42_0001 1 0.000 8.6 <NA> <NA> SELF <NA>
LEXEME 006_42_0001 1 1.058 0.824 word-0002 lex SELF <NA>
LEXEME 006_42_0001 1 5.409 0.397 word-0011 lex SELF <NA>
SPEAKER 006_42_0002 1 0.000 3.35 <NA> <NA> SELF <NA>
LEXEME 006_42_0002 1 0.000 3.35 NO_KEYWORD lex SELF <NA>

Într-un rând din acest fişier se găsesc următoarele câmpuri: tipul obiect (speaker, lexeme, segment, non-lex şi non-speech), numele fişierului audio, numărul de canale, momentul de început, durata, câmp ortografic, numele vorbitorului şi scorul de încredere. În exemplul de mai sus în fişierul 006_42_0001 se regăseşte conţinut audio de la un vorbitor. Acesta a fost înregistrat pe un singur canal, timpul de început fiind 0.000 şi are o durată totală de 8.6 secunde. Datele care nu sunt disponibele sunt marcate cu <NA>. Următoarele două rânduri fac de asemenea referire la fişierul 006_42_0001 acesta indică faptul că fişierul audio conţine cuvinte cheie şi anume word-0002 şi word-0011 (acestea fiind ID-uri asociate cuvintelor cheie reale). Momentul de început este la secunda 1.058 respectiv 5.409 şi durează 0.824 secunde respectiv 0.397 secunde. Acest fişier RTTM trebuie să conţină informaţii despre toate fişierele audio din baza de date fie că acestea conţin sau nu cuvinte cheie. Scopul acestui fişier de referinţă este de a oferii informaţii precise cu privire la apariţia cuvintelor cheie în baza de date. Prin compararea acestui fişier cu fişierul rezultat la ieşirea sistemului KWS este posibilă evaluarea acurateţii de detecţie a cuvintelor cheie.

Un exemplu de fişier rezultat la ieşirea sistemului KWS este prezentat mai jos:

<detected_termlist termid="word-0008" term_search_time="1" oov_count="0">
<term file="006_42_0044" channel="1" tbeg="1.71" dur="1.24" score="1" decision="YES"/>
<term file="266_42_0044" channel="1" tbeg="1.62" dur="0.86" score="1" decision="YES"/>
<term file="356_42_0044" channel="1" tbeg="1.25" dur="0.83" score="1" decision="YES"/>
<term file="300_42_0044" channel="1" tbeg="1.43" dur="0.86" score="0.998178" decision="YES"/>
<term file="324_42_0044" channel="1" tbeg="1.88" dur="1.29" score="0.627089" decision="YES"/>
</detected_termlist>

Pentru fiecare cuvânt cheie căutat este specificat fişierul audio în care acesta apare împreună cu momentul de timp al apariţiei, durata şi probabilitatea.

3.3 EXPERIMENTE ŞI REZULTATE

Acest subcapitol prezintă evaluarea sistemului KWS având la bază sistemul ASR Molina a cărei construcţie a fost prezentată în secţiunea 3.2. Au fost alese 20 de cuvinte cheie ce se doresc a fi

găsite. Pentru această sarcină a fost folosit modulul KWS al utilitarului Kaldi (al cărui mod de funcţionare a fost prezentat în secţiunea 2.1). În urma procesului de KWS a rezultat o listă în formatul celei prezentate în secţiunea 4.2. Această listă împreună cu fişierul de referinţă a fost modificată astfel încât să aibă un format specific corespunzător utilitarului folosit pentru evaluarea cuvintelor cheie în cadrul MediaEval. Prin folosirea acestui utilitar pus la dispoziţie de către membrii Laboratorului de Cercetare Speed s-a obţinut o curbă DET precum şi probabilităţile de neidentificare şi de alarmă falsă (concepte prezentate în secţiunea 2.3)

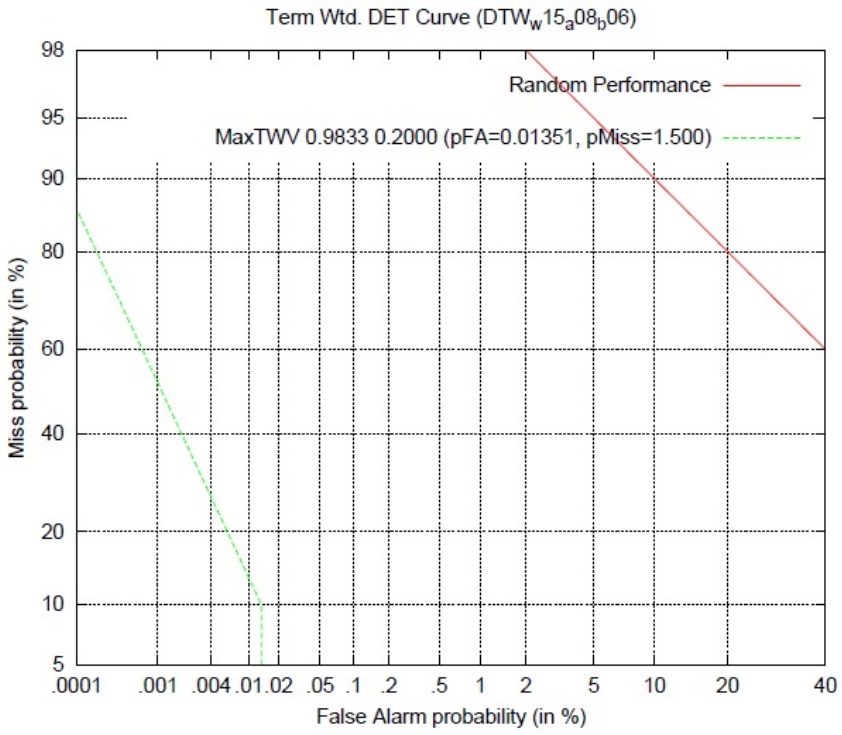


Figura 3.1 Evaluarea sistemului de detecţie a cuvintelor cheie

Se poate observa compromisul dintre probabilitatea de neidentificare (1.5%) şi cea de alarmă falsă (0.01351 %). În funcţie de scopul sistemului KWS obţinut se poate selecta un prag care prin nivelul valorii impuse conduce la următoarele două posibilităţi:

- un prag de nivel scăzut va avea drept rezultat o probalitate mare de alarme false şi una mică pentru cele neidentificate;
- un prag de nivel ridicat va conduce spre o probabilitate mică de alarme false şi una mare pentru cele neidentificate.

S-a dorit extinderea testelor pentru evaluarea sistemului KWS prin căutarea a mai mult de 20 cuvinte cheie. Pentru aceasta a fost extins model de limbă bazat pe reguli (LM1), fiind adăugate foarte multe nume proprii ce nu se găsec în baza de date Molina (aproximativ 700 nume şi 500 prenume). În acest fel a fost creată oportunitatea ca sistemul să introducă noi erori de transcriere. Au fost pregătite 3 scenarii de test:

- s-au căutat câte 10 cuvinte cheie (2 dintre acestea existând în transcrieri) iar 8 nu existau, dar ele pot apărea ca erori de transcriere din cauza noului model de limbă;
- s-au căutat câte 100 cuvinte cheie (20 dintre acestea existând în transcrieri) iar 80 nu existau;
- s-au căutat câte 1000 cuvinte cheie (20 dintre acestea existând în transcrieri) iar 980 nu existau;

Din păcate, din cauza faptului că utilitarul de evaluare nu permite căutarea unor cuvinte ce nu apar în referinţă, aceste teste nu au putut fi efectuate. S-a încercat folosirea unui alt utilitar de evaluare, F4DE, dar şi acesta prezintă aceeaşi problemă. În prezent, problema este investigată.

CAPITOLUL 4

SERVICIUL WEB DE AUTENTIFICARE PRIN VOCE

Serviciul web de autentificare prin voce este o aplicaţie ce oferă utilizatorilor posibilitatea de înregistrare şi autentificare pe bază de parolă şi recunoaştere de vorbire şi amprentă vocală.

4.1 DESCRIERE ŞI ARHITECTURĂ

Proiectul este compus dintr-un ansamblu de module componente: trei module principale expuse sub forma unor servicii REST (Representational State Transfer), fiecare realizând o sarcină precisă, o aplicaţie client şi o bază de date. Comunicaţia dintre acestea este realizată prin intermediul mesajelor de tip "cerere/răspuns" specifice protocolului HTTP (Hyper Text Transfer Protocol). Datorită acestui fapt, sistemul poate lucra în mod distribuit.

Componentele aplicaţiei sunt:

- Serviciul de autentificare;
- Serviciul de recunoaştere de vorbitor (nu este parte a acestei lucrări);

- Serviciul de recunoaştere de vorbire (nu este parte a acestei lucrări);
- Aplicaţia client;
- Baza de date.

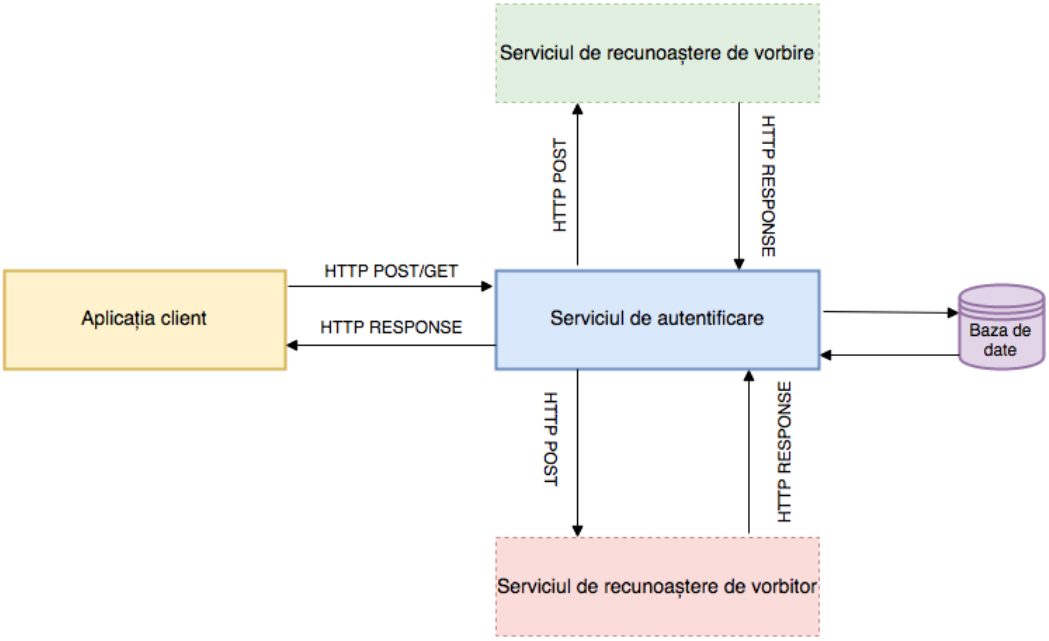


Figura 4.1 Arhitectura serviciului web de autentificare prin voce

Aplicaţia client

În etapa de înregistrare, utilizatorul trebuie să completeze un formular web cu datele sale personale. Tot în această etapă, trebuiesc create cel puţin 10 înregistrări audio a câte 12 cifre fiecare, şirurile de cifre fiind generate în mod aleator şi afişate în aplicaţia client. Atât datele personale cât şi înregistrările audio vor fi trimise către serviciul de autentificare. Clientul va primi un mesaj informativ cu privire la starea contului (dacă a fost creat sau nu).

În etapa de autentificare, utilizatorul va completa un formular cu adresa de poştă electronică şi parola şi va înregistra o singură secvenţă audio. Aceste date vor fi de asemenea trimise serviciului de autentificare. Dacă autentificarea a fost realizată cu succes, aplicaţia client va fi redirecţionată spre pagina personală a utilizatorului, în caz contrar primind un mesaj informativ cu privire la eroarea întâmpinată.

Serviciul de autentificare

Este elementul central al aplicaţiei, deoarece realizează agregarea şi procesarea datelor, interacţionează cu aplicaţia client, cu baza de date şi cu serviciile de recunoaştere de vorbitor şi vorbire.

În etapa de înregistrare, datele personale şi înregistrările audio primite de la aplicaţia client vor fi inserate în baza de date. Apoi, aceleaşi înregistrări audio vor fi trimise către serviciul de recunoaştere de vorbitor, respectiv serviciul de recunoaştere de vorbire. Primul va realiza un model specific vorbitorului, utilizat ulterior la autentificare. Cel de-al doilea va returna transcrierile înregistrărilor audio. Pe baza acestora şi a transcrierilor de referiţă (cele ce se presupun că au fost citite de către utilizator, afişate în aplicaţia client la înregistrare), se va calcula rata de eroare la nivel de cuvânt pentru fiecare înregistrare, precum şi media erorilor. Baza de date va fi actualizată cu aceste valori. La final, aplicaţiei client îi va fi returnat un mesaj privitor la acţiunea de înregistrare.

În etapa de autentificare, se vor primi de la aplicaţia client: numele de utilizator (reprezentat de adresa de poştă electronică), parola şi o înregistrare audio. Indiferent dacă autentificarea va avea

loc sau nu, tentativa de autentificare va fi stocată în baza de date. Înregistrarea audio va fi trimisă către serviciul de recunoaştere de vorbitor, respectiv serviciul de recunoaştere de vorbire. Se vor realiza 3 verificări:

1. Parola să fie identică cu cea oferită la înregistrare;
2. Serviciul de recunoaştere de vorbitor trebuie să confirme că vocea din înregistrare corespunde modelului utilizatorului pretins;
3. Serviciul de recunoaştere de vorbire va returna transcrierea înregistrării şi se va calcula rata de eroare la nivel de cuvânt. Aceasta nu trebuie să depăşească un anumit prag sau cel mult să se regăsească în jurul valorii de eroare medie, specifică utilizatorului şi calculată la înregistrare.

Serviciul de recunoaştere de vorbitor

În etapa de înregistrare, acesta primeşte de la serviciul de autentificare setul de fişiere audio ale utilizatorului, pe baza cărora va crea un model specific acestuia, amprenta sa vocală.

În etapa de autentificare, primeşte de la serviciul de autentificare fişierul audio al utilizatorului, iar prin comparaţie cu amprenta vocală deja existentă, va decide dacă utilizatorul este cel pretins sau un impostor.

Implementarea acestui serviciu nu este parte a lucrării de faţă, mai multe detalii pot fi consultate în lucrarea [14].

Serviciul de recunoaştere de vorbire

Atât în etapa de înregistrare, cât şi în cea de autentificare, acesta primeşte de la serviciul de autentificare înregistrările audio ale utilizatorului. Având implementat un sistem de recunoaştere automată a vorbirii, specalizat în recunoaşterea cifrelor în limba română, acesta întoarce transcrierile fişierelor audio. Rolul acestui serviciu este de a verifica dacă utilizatorul a rostit într-adevăr şirurile de cifre afişate în aplicaţia client.

Implementarea acestui serviciu nu este parte a lucrării de faţă, mai multe detalii pot fi consultate în lucrarea [14].

Baza de date

Toate operaţiile asupra acesteia sunt efectuate de către serviciul de autentificare. Baza de date stochează datele personale ale utilizatorilor înregistraţi, fişierele audio împreună cu transcrierile acestora şi ratele de eroare calculate. De asemenea, stochează informaţii referitoare la toate tentativele de autentificare, reuşite sau nu.

4.2 FUNCŢIONALITATE DIN PERSPECTIVA UTILIZATORULUI

Etapa de înregistrare

În această etapă, aşa cum a fost prezentat şi anterior, utilizatorul trebuie să completeze un formular cu datele personale: nume, prenume, adresa de poştă electronică, parola şi o fotografie. Seturi a câte 12 cifre sunt afişate în pagină. Pentru înregistrarea audio a unui set, utilizatorul va trebui mai întâi să accepte accesul aplicaţiei la microfon. Apoi, prin apăsarea butonului *Start*, va începe captarea audio, iar utilizatorul va rosti setul de cifre. Pentru încheierea captării audio, se va apăsa din nou butonul *Start* (devenit acum *Stop*, fiind un buton ce poate lua două valori, în funcţie de starea sistemului de înregistrare audio: pornit sau oprit). În acest moment, utilizatorul poate asculta înregistrarea audio, iar în cazul în care constată că au intervenit erori de pronunţie sau alte probleme, poate recurge la refacerea înregistrării. În momentul în care consideră că înregistrarea audio este corectă, se poate trece la crearea următoarei înregistrări, prin apăsarea butonului *Next*. Totodată, interfaţa se va actualiza, un nou set de cifre fiind afişat. Apăsarea butonului *Submit* va avea ca efect trimiterea formularului şi a înregistrărilor la server. Acesta va

fi disponibil începând cu momentul în care s-a realizat înregistrarea audio a cel puţin 10 seturi de cifre. Pentru a obţine o amprenta vocală cât mai puternică, se recomandă crearea a cât mai multor înregistrări audio. La final, utilizatorul va primi un mesaj informativ cu privire la rezultatul acţiunii de înregistrare.

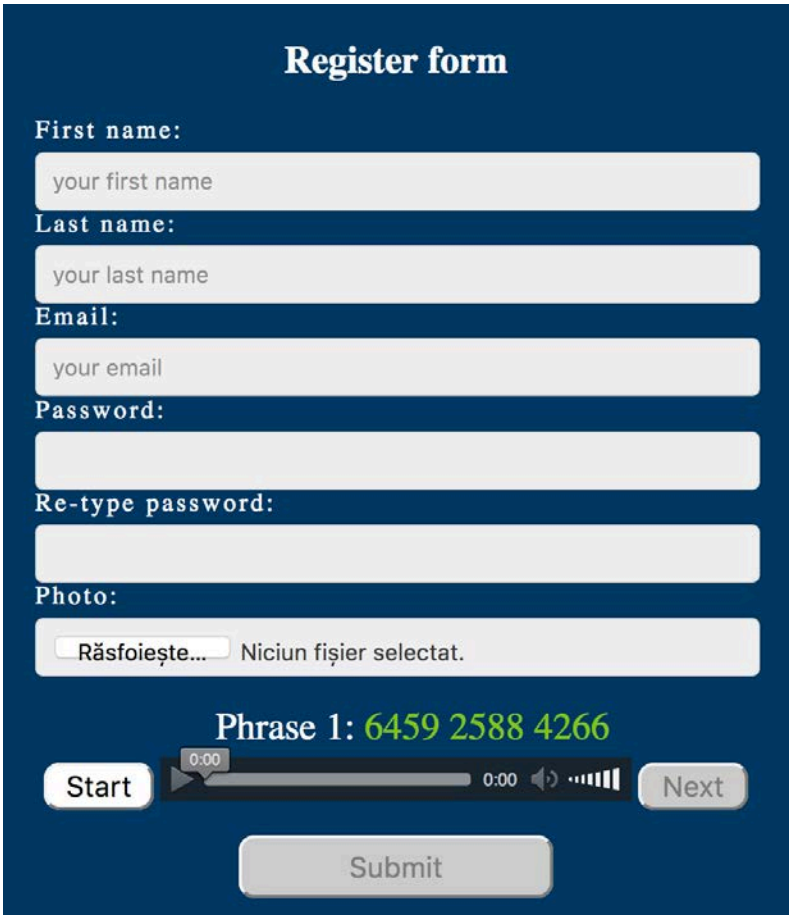


Figura 4.2 Formularul de înrolare

Etapa de autentificare

În această etapă, utilizatorul completează un formular cu adresa sa de poştă electronică, parola şi înregistrază un singur clip audio. Butonul *Submit* va deveni disponibil, iar datele se vor trimite la serviciul de autentificare unde vor fi realizate cele 3 verificări. În cazul în care una dintre acestea va eşua, utilizatorul va primi un mesaj ce confirma eroarea întâmpinată. Altfel, va fi redirecţionat către contul său personal.

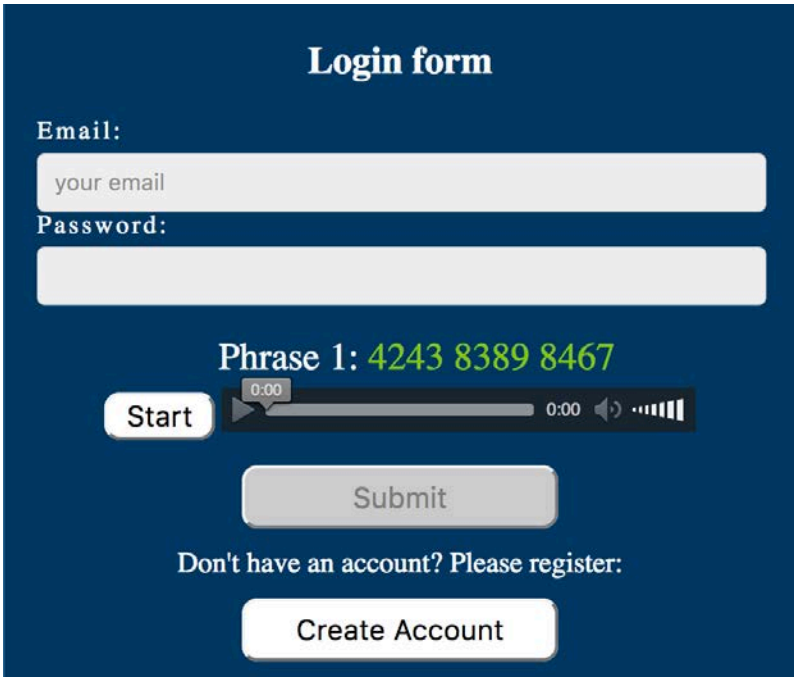


Figura 4.3 Formularul de autentificare

Etapa de accesare a contului personal

În urma autentificării realizate cu succes, aplicaţia client va afişa contul personal al utilizatorului. Aici se regăsesc datele sale personale oferite în etapa de înregistrare. Totodată, pot fi aduse modificări asupra numelui, parolei şi a fotografiei de profil. Pentru părăsirea contului şi încheierea sesiunii, se va apăsa butonul *Logout*.

Figura 4.4 Pagina contului personal

4.3 SCENARII DE UTILIZARE

Scenariile de utilizare aduc în discuţie câteva situaţii ce pot avea loc în cadrul etapei de autentificare. Se va descrie modul în care aplicaţia tratează încercarea unui utilizator autorizat de a-şi accesa contul, dar şi eventualele tentative de acces fraudulos şi felul în care acestea sunt prevenite, accentuând rolul serviciilor de verificare de vorbitor şi vorbire.

Scenariul 1

Un utilizator autorizat încearcă să îşi acceseze contul personal creat anterior conform procedurii de înregistrare. În urma completării formularului de autentificare şi a creării clipului audio, serviciul de autentificare va constata că parola este cea corectă, serviciul de recunoaştere a vorbitorului va constata că vocea utilizatorului corespunde cu amprenta sa vocală, iar în urma apelării serviciului de recunoaştere de vorbire se va obţine transcrierea text a secvenţei audio, ce va certifica faptul că a fost rostit setul de cifre cerut.

- Parola corectă ✓
- Vocea din clipul audio corespunde amprentei vocale ✓
- Utilizatorul a pronunţat setul de cifre cerut ✓

} Autentificare reuşită ✓

Scenariul 2

Un utilizator neautorizat care cunoaşte adresa de poştă electronică şi parola unui alt utilizator, încearcă să acceseze fraudulos contul personal al acestuia. În urma completării formularului de autentificare şi a creării clipului audio, serviciul de autentificare va constata că parola este cea corectă, în schimb serviciul de recunoaştere a vorbitorului va constata că vocea utilizatorului nu corespunde amprentei sale vocale. Acest fapt va avea ca efect respingerea autentificării, chiar dacă a fost rostit corect setul de cifre cerut.

- Parola corectă ✓
- Vocea din clipul audio corespunde amprentei vocale ☒
- Utilizatorul a pronunţat setul de cifre cerut ✓

} Autentificare reuşită ☒

Scenariul 3

Acest scenariu este similar cu cel precedent, singura diferenţă fiind faptul că utilizatorul neautorizat deţine de această dată un fişier audio cu vocea utilizatorului autorizat. Deşi amprenta vocală va corespunde cu cea a utilizatorului autorizat, autentificarea va fi din nou respinsă, deoarece fişierul nu va conţine rostirea setului de cifre afişat în pagină. Având în vedere faptul că un set nou este generat la fiecare tentativă de autentificare, precum şi numărul mare de combinaţii în care pot apărea cele 12 cifre, şansele sunt extrem de mici că transcrierea returnată de serviciul de recunoaştere de vorbire să fie compatibilă cu setul de cifre afişat.

- Parola corectă ✓
- Vocea din clipul audio corespunde amprentei vocale ✓
- Utilizatorul a pronunţat setul de cifre cerut ☒

} Autentificare reuşită ☒

Scenariul 4

Acest senariu prezintă o metodă prin care utilizatorul neautorizat poate accesa cu succes contul utilizatorului autorizat. Acest fapt reprezintă o vulnerabilitate reală aplicaţiei, fiind indicate totodată câteva soluţii ce se pot adopta în această privinţă.

Similar cu utimele două scenarii anterioare, se presupune că utlizatorul neautorizat cunoaşte adresa de poştă electronică, parola aferentă, iar în plus acesta deţine un fişier audio cu vocea utilizatorului autorizat, acesta fiind suprins în timp ce rosteşte un şir de cifre (de exemplu, câteva numere de telefon). Până la acest punct, condiţiile iniţiale sunt identice cu cele din *Scenariul 3*. Diferenţa este dată de o nouă abordare în încercarea de a elimina verificarea dată de serviciul de

recunoaştere de vorbire. Fişierul audio ce conţine cifre rostite de utilizatorul autorizat va fi împărţit în fişiere mai mici, obtinandu-se astfel câte un fişier audio în care se va regăsi rostirea unei singure cifre. În acest moment, prin redarea fişierelor în ordinea apariţiei cifrelor în set, transcrierea returnată de serviciul de recunoaştere de vorbire va corespunde cu setul de cifre afişat, oricare ar fi acesta.

- Parola corectă ✓
- Vocea din clipul audio corespunde amprentei vocale ✓
- Utilizatorul a pronunţat setul de cifre cerut ✓

} Autentificare reuşită ✓

Soluţii: În momentul actual, serviciul de recunoaştere de vorbire este capabil să realizeze transcrierea a 10 cuvinte, acestea fiind cifrele sistemului zecimal. Deoarece vocabularul este unul redus, fraudarea sistemului este destul de posibilă, aşa cum s-a prezentat în ultimul scenariu. Ca soluţie, aplicaţia client ar trebui configurată astfel încât să afişeze un set de caractere alfanumerice sau chiar grupuri de cuvinte, iar sistemul de recunoaştere de vorbire să fie extins astfel încât să realizeze transcrieri în această nouă situaţie. Fraudarea ar deveni foarte puţin probabilă, utilizatorul neautorizat va avea nevoie de fişiere audio ce conţin rostirea de către utilizatorul autorizat a cât mai multor cuvinte sau sunete. De asemenea, mixarea acestora într-un singur clip audio ar fi diferită din punct de vedere al articulării vorbirii, în comparaţie cu un clip audio creat prin utilizarea normală a aplicaţiei.

4.4 TEHNOLOGII UTILIZATE

HTML

HTML (HyperText Markup Language) este un limbaj de marcare utilizat pentru crearea paginilor ce pot fi afişate într-un navigator web. Conţinutul unei astfel de pagini poate fi organizat prin folosirea etichetelor şi a atributelor.

Etichetele definesc elemente HTML şi au un format specific, denumirea elementului regasindu-se între caracterele "<", ">". De regulă, orice etichetă deschisă atrage după sine şi închiderea ei. De exemplu, deschiderea etichetei * * ce defineşte un bloc, o secţiune a paginii web, va exista în pereche cu etichetă ce marchează închiderea să, * *. Elementele HTML pot fi imbricate, între etichetele de deschidere şi închidere ale unui element se pot afla alte elemente. Codul unei pagini web minimale trebuie să fie cuprins între etichetele *<html>* şi *</html>*. Apoi, în interiorul acestora, se vor regăsi alte două elemente. Primul element, *<head> </head>*, va conţine la rândul său elemente ce definesc titlul paginii, precum şi elemente ce fac referinţă la paginile de stil sau la scripturile incluse. Al doilea element, *<body> </body>*, va conţine elemente ce alcătuiesc pagina propriu-zisă. Alte elemente uzuale sunt cele ce definesc paragrafe, imagini, legături, tabele, etc.

Atributele oferă informaţii suplimentare asupra elementelor. De regulă, se folosesc atribute ce definesc numele elementului, atribute ce stabilesc un identificator unic al elementului, sau atribute ce plasează elementul într-o clasă de elemente. În funcţie de elementul căruia aparţin, atributele pot oferi detalii despre dimensiuni (în cazul blocurilor sau al imaginilor), despre fonturi şi culori (în cazul paragrafelor). În exemplul următor, elementul * * marchează introducerea unui paragraf, iar *align* şi *id* sunt atributele sale:

* Acesta este un exemplu *.

CSS

De la bun început, scopul elementelor HTML a fost de a structura paginile web. Mai apoi, a devenit necesară formatarea aspectului şi au apărut elemente ca ** sau atribute ce modifică dimensiunile, culoarea, spaţierile, etc. Aceste lucruri însă au făcut din dezvoltarea paginilor un proces dificil.

CSS (Cascading Style Sheets) descrie modul în care elementele HTML ar trebui să fie afişate. CSS a apărut din necesitatea de a formată conţinutul paginilor web într-un mod organizat şi eficient. La începutul paginii HTML se va introduce în interiorul elementul <*head*> </*head*> un rând de forma: <*link rel="stylesheet" type="text/csş" href="style.css"*>. Acesta realizează o referinţă către fişierul *style.css* ce va conţine formatarea elementelor. Fiecare element poate fi referit prin identificatorul său unic, clasa din care face parte sau chiar denumirea elementului. În exemplul de mai jos este prezentată formatarea unui element de tip bloc referit prin identificatorul sau, numit *container*:

```
#container{
Margin-top: 20px;
Color: white;
Font-size: 24 px;
}
```

Dintre avantajele folosirii CSS pot fi amintite:

- Eficienţă crescută: o singură secvenţă de cod CSS poate fi aplicată unui grup de elemente HTML;
- Mentenanţa simplificată: o singură modificare în pagina de stil se va aplica în toate documentele HTML unde este inclusă;
- Datorită partajării codului CSS, paginile se vor încărca mai repede;
- CSS oferă mult mai multe posibilităţi de stilizare în comparaţie cu aspectul realizat cu ajutorul atributelor;
- CSS poate realiza compatibilitatea cu diverse dispozitive: acelaşi document HTML poate fi prezentat în mod diferit, astfel încât aspectul să fie optim, în funcţie de tipul dispozitivului, dimensiunea afişajului, performanţele de calcul, etc.

Javascript

Javascript este un limbaj de programare bazat pe obiecte, folosit mai ales pentru introducerea unor funcţionalităţi în paginile web. Este un limbaj de scripting ce nu necesită compilare, codul fiind interpretat. Oferă interactivitate documentelor HTML şi poate fi inglobal direct în acestea, regasindu-se deseori în secţiunea <*head*> </*head*>, cuprins între etichetele <*script type="text/javascript"*> </*script*> sau poate fi adăugată o referinţă către un fişier extern <*script src="javascript_code.js"*></*script*>.

Javascript interacţionează cu evenimente generate de utilizator sau sistem. Un astfel de eveniment poate fi apăsarea unui buton, terminarea încărcării unei pagini sau editarea unui câmp de text. Pot fi utilizate mai multe gestionare de evenimente: *onclick* (declanşează execuţia unei funcţii atunci când un element HTML este apăsat), *onload* şi *onunload* (declanşează execuţia unei funcţii atunci când utilizatorul accesează sau părăseşte pagina), *onchange* (folosit de obicei odată cu validarea câmpurilor de intrare), *onmouseover*, *onmouseout*, *onmousedown*, *onmouseup* (declanşează execuţia unei funcţii în funcţie de poziţia mouse-ului faţă de un element HTML), *onsubmit* (va apela o funcţie atunci când se acţionează butonul *Submit* al unui formular). Intructiunea *document.write()* este modalitatea prin care poate fi adăugat un conţinut variabil în pagină HTML. Instrucţiunea *document.getElementById().value* permite citirea sau modificarea valorii unui element HTML.

Fiind un limbaj ce permite programarea pe obiecte, Javascript permite atât utilizarea obiectelor deja încroporate, cât şi crearea unor obiecte proprii. Fiecare obiect este caracterizat de proprietăţi şi metode specifice. Câteva obiecte standard ce se regăsesc în limbajul Javascript: *String*, *Date*, *Array*, *Math*, *RegExp*, *Number*, *Cookie*. Alte obiecte importante sunt: *Window* (reprezintă o fereastră deschisă în navigator), *Screen* (conţine informaţii despre ecranul utilizatorului), *Location* (conţine informaţii despre url-ul curent).

Ajax

Ajax (Asyncronous Javascript and XML) este o tehnologie ce permite actualizarea elementelor dintr-o pagină web în mod dinamic, fără întreruperi pentru încărcări de pagină sau reîmprospătări de ecran. Cererile şi răspunsurile serverului nu mai trebuie să corespundă cu anumite acţiuni ale untilizatorului, ci pot avea loc în orice moment convenabil pentru aplicaţie. Navigatorul permite ca utilizatorul să continue folosirea paginii, în timp ce cererile trimise serverului se vor efectua în mod asincron.

Caracteristic Ajax este obiectul *XMLHTTPRequest*, acesta oferind posibilitatea de a trimite cereri HTTP fără ca finalizarea acestora să corespundă neapărat cu încărcarea unei pagini noi. Obiectul este caracterizat de mai multe meode şi proprietăţi. Metodă *open()* pregăteşte comunicarea cu serverul, în timp ce metodă *send()* timite cererea. Proprietatea *readyState* poate lua valori numerice în intervalul [0;4] ce semnifică starea cererii: neinitializata, în încărcare, încărcată, interactivă şi finalizată. Proprietatea *onreadystatechange* determina ce manager de eveniment va fi apelat când se modifică proprietatea *readyState* a obiectului. Proprietatea *response* reprezintă datele returnate de către server.

Web Audio API

Web Audio API (Application Programming Interface) este un API Javascript de nivel înalt specializat în prelucrarea şi sintetizarea audio în aplicaţii web. Scopul acestuia este de a include capacităţi de mixare, procesare şi flitrare, atât de specifice aplicaţiilor audio moderne.

API-ul este construit în jurul conceptului de *AudioContext*. Acesta este reprezentat de un graf orientat ce defineşte modul în care fluxul audio trece de la sursă spre destinaţie. Fiecare nod prin care trece poate inspecta sau modifică proprietăţile fluxului. Figură următoare prezintă un astfel de graf, caracterizat de mai multe noduri cu rol de sursă, noduri intermediare cu rol de filtru, precum şi un nod destinaţie.

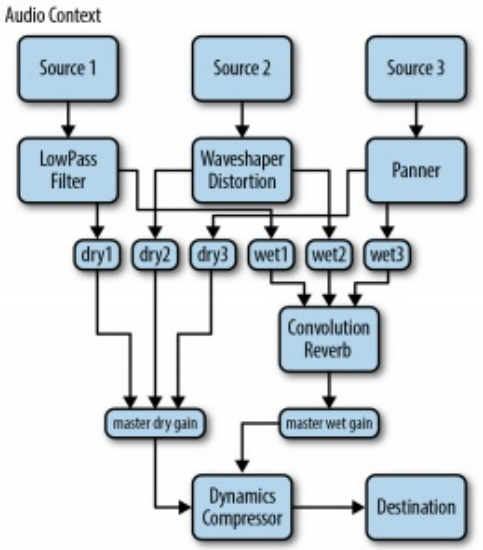


Figura 4.5 Model de Context Audio (Web Audio API). Sursa:[21]

Sursa poate fi reprezentată de un fişier audio sau un microfon, în timp ce destinaţia constă într-un dispozitiv de redare audio. În cazul mai multor surse, cum este cazul şi în imaginea prezentată, semnalele audio vor fi pur si simplu îmbinate. Nodurile oferă posibilitatea de a fi conectate şi deconectate, iar asupra fluxului audio se pot aplica diverse filtre şi analizoare de semnal.

Java

Java este un limbaj de programare de nivel înalt, obiect-orientat, dezvoltat în special pentru a avea cât mai puţine dependinţe posibile. Iniţiatorul proiectului este James Gosling, iar prima versiune, Java 1.0 a fost lansată public în 1995 în cadrul companiei Sun Microsystems. Acesta a

fost conceput astfel încât să fie scris o singură dată şi de a rula oriunde ("write once, run anywhere"). Aplicaţiile Java sunt compilate ca bytecode (fişiere de tip clasa) ce pot rula pe orice JVM (Java Virtual Machine), indiferent de arhitectura sistemului de calcul. Din acest punct de vedere, Java este un limbaj portabil.

Java este considerat un limbaj simplu, foarte multe caracteristici ale sale fiind derivate din C/C++ şi este bazat în totalitate pe conceptele POO (programării obiect-orientata). A fost conceput pentru aplicaţii ce lucrează în mod distribuit, fiind utilizate deoseori protocoale de comunicaţii pentru apelarea la distanţă a funcţiilor şi accesarea resurselor. O altă caracteristică este dată de posibilitatea de a rula simultan pe mai multe fire de execuţie, aplicaţiile având un grad de interactivitate crescut. Codul Java este sigur, acesta fiind verificat înainte de execuţie, pentru a se confirma faptul că a fost compilat de un compilator standard. Erorile de acces la memorie sunt eliminate, datorită faptului că nu se folosesc pointeri.

În prezent, Java a ajuns la versiunea 8 şi este unul dintre cele mai utilizate limbaje de programare, aflându-se la baza multor aplicaţii şi servicii web.

Arhitectura REST

REST (Representational state transfer) este un stil arhitectural care defineşte un set de principii ce fac mult mai scalabilă şi flexibilă dezvoltarea aplicaţiilor web. În anul 2000, Roy Fielding a introdus pentru prima dată acest concept, definind un set de constrângeri ce stau la baza seriviciilor REST:

- Adresabilitate: datele şi informaţiile sunt privite în mod abstract sub forma unor resurse, iar fiecare resursă poate fi accesată printr-un URI (Uniform Resurce Identifier).
- Interfaţa uniformă: între clienţi şi servere există o interfaţă bine definită şi simplificată, bazată pe protocolul HTTP, ce permite manipularea resurselor.
- Lipsa stării: orice cerere HTTP are loc într-o izolare completă, fiind independente faţă de cererile precedente. Toate informaţiile necesare îndeplinirii unei cereri sunt incluse în această.
- Client-server: interfaţă uniformă ce separă aceste două entităţi. Fiecare are un rol specific, fiind posibilă dezvoltarea în mod independent, atât timp cât interfaţa rămâne comuna. Clientul nu este preocupat de stocarea datelor, portabilitatea codului său fiind îmbunătăţită. În schimb, serverul nu este preocupat de interfaţarea cu utilizatorul.
- Sistem stratificat: în orice moment, clientul poate fi conectat direct la serverul final sau la un intermediar. Serverele intermediare îmbunătăţesc scalabilitatea sistemului.

Serviciile REST au la bază protocolul HTTP. Acesta presupune un schimb de mesaje de tip cerere/răspuns. Clientul iniţiază un mesaj de tip cerere către server în care specifică metoda HTTP folosită, locaţia resursei, antetul şi corpul mesajului. Răspunsul serverului conţine un cod de răspuns, un antet şi un corp al mesajului. Cele mai importante metode HTTP sunt:

- GET: folosită pentru a interoga serverul în vederea obţinerii unei informaţii specifice. Este o operaţie idempotentă şi sigură; de oricâte ori va fi apelată, rezultatul va fi acealaşi iar starea serverului nu va fi influenţată.
- PUT: solocita serverului să stocheze coprul mesajului transmis la locaţia specificată în mesajul HTTP. Este folosită pentru a insera sau modifca date.
- DELETE: folosită pentru a şterge resurse.
- POST: folosită pentru crearea unei resurse. Solicită serverului să accepte şi să stocheze datele cuprinse în corpul mesajului.

În urma unei cereri HTTP se generează un cod de răspuns, mai jos fiind amintite principalele clase în care acestea sunt grupate:

- 1xx: clasa codurilor de informare;

- 2xx: clasa specifică cererilor ce au avut loc cu succes;
- 3xx: clasa codurilor de redirecţionare. Clientului îi este indicat să realizeze o acţiune suplimentară pentru finalizarea cererii;
- 4xx: clasa codurior ce indică erori generate de client;
- 5xx: clasa codurilor ce indică erori generate de server.

Jersey

Jersey este o platformă de dezvoltare a serviciilor REST folosind limbajul Java. Jersey oferă programatorilor o serie de adnotări ce pot fi asociate claselor şi metodelor cu scopul de a defini resursele şi acţiunile ce pot fi efectuate pe baza acestora. Adnotările se scriu înaintea definirii unei clase sau metode. Cele mai întâlnite sunt:

- *@Path()*: valoarea acestei adnotări este un URI relativ ce desemnează calea către clasa sau metoda căreia îi este asociată.

  Exemplu: *http://localhost/VoiceApp/login/auth*. Acest URI este format din: protocol, adresa serverului, numele serviciului web, numele clasei şi numele funcţiei ce va prelua cererea HTTP.

- *@GET, @ POST, @PUT, @DELETE*: aceste adnotări specifică tipul cererii HTTP ce poate fi prelucrată de către o funcţie.
- *@Consumes:* această adnotare specifică tipul media al resursei primite de către server.
- *@Produces:* această adnotare specifică tipul media al resursei ce va fi trimise înapoi clientului sub forma unui răspuns HTTP.

Jersey oferă suport atât pentru crearea aplicaţiilor server, cât şi pentru aplicaţiile client.

4.5 IMPLEMENTAREA APLICAŢIEI CLIENT

Aplicaţia client oferă utilizatorului o interfaţă grafică intuitivă cu ajutorul căreia datele sale pot fi preluate sau afişate. A fost dezvoltată cu ajutorul tehnologiilor HTML, CSS şi Javascript. Interacţiunea aplicaţiei client cu serverul este realizată cu ajutorul tehnologiei Ajax. Funcţionalitatea de înregistrare audio a fost posibilă prin inserarea în aplicaţie a componenţei software RecorderJS [22]. Următoarea diagrama prezintă structura aplicaţiei client, fiind prezentate fişierele componente şi legătura dintre acestea.

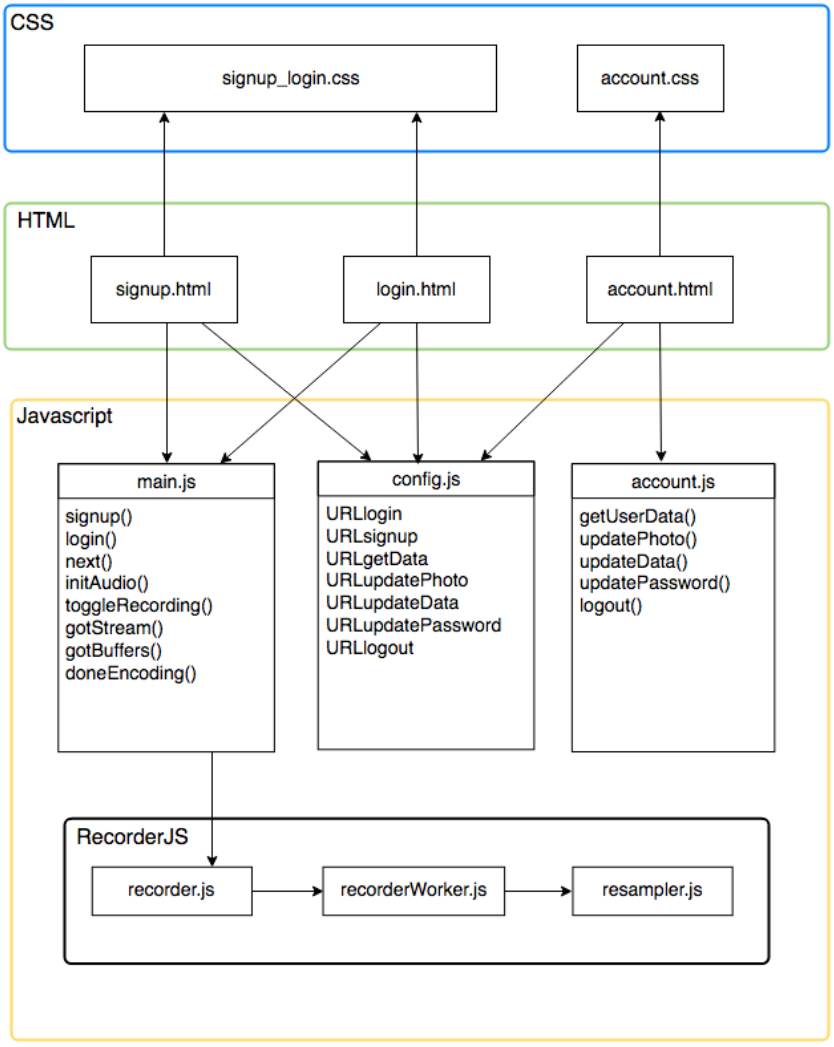


Figura 4.6 Structura aplicaţiei client

HTML

Paginile corespunzătoare procesului de înrolare şi autentificare (*signup.html* şi *login.html*) conţin o referinţă către fişierul *signup_login.css* în care este definit aspectul elementelor HTML, precum şi fişierele *main.js* şi *config.js*. Cele două pagini HTML o structură similară, fiind formate dintr-un formular ce conţine câmpuri de intrare, necesare pentru completarea datelor personale. Datele personale, cât şi înregistrările audio, vor fi preluate de către funcţia *signup()*, respectiv *login()*, definite în fişierul *main.js*.

```
<form name="userData"  method="post" onsubmit="signup(); return false;">
    <label>First name: </label><input type="text" name="firstName" id="firstName" placeholder="your first name" required>
    <label>Last name:</label><input type="text" name="lastName" id="lastName" placeholder="your last name" required>
    <label>Email:</label><input type="email" name="email" id="email" placeholder="your email" required>
    <label>Password: </label><input type="password" name="password" id="password" required >
    <label>Re-type password:</label><input type="password" name="retypedPassword" id="retypedPassword" required>
    <label>Photo:</label><input type="file" name="photo" id="photo" size="45" required >
```

În afara formularului, paginile conţin un bloc corespunzător procedurii de creare a clipurilor audio. Aici se regăsesc: butonul de *Start/Stop*, ce activează funcţia *toggleRecording()*, butonul *Next*, prin a cărui apăsare se execută funcţia *next()*, elemente HTML pentru redare audio. Toate acestea pot fi observate în imaginea următoare:

```
<div id="container">
<div id="wrap">
        <div id="numbers" align="center"> </div>
        <div id="phrase"></div>
</div>
    <div id="controls" align="center">
    <table>
        <tr><td><button type="button" id="toggle" onclick="toggleRecording(this);">Start</button></td>
        <td><audio id="fisier" controls="controls" preload="none"></audio></td>
        <td><button id="nextAudio" type="button" onclick="next()">Next</button></td>
    </table>
        <a id="save" href="#"></a>
        <div id="recordingState"></div>
    </div>
</div>
```

Pagina *account.html* conţine câteva blocuri în care vor fi afişate datele utilizatorului autentificat, precum şi câmpuri de intrare în care se pot efectua modificări asupra datelor personale (nume, parola, fotografie de profil). După acest pas, prin apăsarea butonului *Update* va fi executată una din funcţiile *updateData(), updatePassword()* sau *updatePhoto()*, definite în fişierul *account.js*. Modificările vor fi trimise spre server în vederea actualizării bazei de date.

CSS

Cele două fişiere CSS ale aplicaţiei conţin detalii referitoare la formatarea elementelor HTML. Acestea sunt referite pe baza denumirii elementului, identificatorul unic sau numele clasei din care face parte. Culorile de fundal, distanţele dintre elemente şi dimensinile lor, aspectul textului sunt câteva dintre proprietăţile definite în paginile de stil. Următorul exemplu reprezintă formatarea butonului de trimitere a formularului de înrolare:

```
#submitSignup
{
        border-radius: 10px;
        width:200px;
        height:40px;
        background-color:#CCCCCC;
        font-size:18px;
        margin-bottom:10px;
        margin-top:10px;
}
```

Javascript

Fişierul *main.js* este inclus în paginile *signup.html* şi *login.html* şi are rolul de a prelua datele din acestea şi de a le trimite la server. De asemenea, în fişierul *main.js* este creat un obiect de tip *Recorder*, definit în cadrul modulului RecorderJS. Acesta captează semnalul audio şi îl salvează ca obiect *Blob*.

Funcţiile *login()* şi *signup()* citesc valorile elementelor HTML cu ajutorul instrucţiunii *document.getElementById()*. Aceste valori vor fi adăugate într-un formular de tipul *FormData()*. Un obiect *XMLHttpRequest()* va deschide o conexiune către server şi va trimite o cerere HTTP ce conţine formularul creat. Va fi citit codul de răspuns ce indică starea cererii, precum şi răspunsul propriu zis. Secvenţa următoare conţine definiţia funcţiei *login()*:

```
function login()
{
        var email = document.getElementById("email").value;
        var password = document.getElementById("password").value;

        var r=new XMLHttpRequest();
        var form=new FormData();
        form.append("email",email);
        form.append("password",password);
        form.append("audioFile",bloblogin);
```

```
        r.open("POST",URLlogin,true);
        r.onreadystatechange=function()
          {
               if (r.readyState===4)
                     if(r.status===200)
                       {
               window.location.replace("account.html");
                        }
                 else
                              alert(r.response);
          };
      r.send(form);
}
```

Fişierul *account.js* este responsabil de opţiunile oferite utilizatorului în momentul în care acesta este autentificat şi îşi vizitează pagina personală. Sunt definite funcţii de obţinere a datelor personale de la server, precum şi de actualizare a acestora. Sunt folsite obiecte de tip XMLHttpRequest(), iar elementele HTML sunt accesate într-un mod similar cu cel prezentat anterior.

Fişierul *config.js* este un fişier de configurare, acesta conţine adresele URL corespunzătoare diverselor metode din serviciul de autentificare spre care se îndreaptă cererile HTTP.

RecorderJS este un modul software dezvoltat în Javascript ce utilizează Web Audio API. În fişierul *recorder.js* este definit obiectul *Recorder*, capabil să capteze audio în format WAV. Fişierul *resampler.js* defineşte o funcţie ce permite modificarea frecvenţei de eşantionare a semnalului, fiind posibilă şi alegerea numărului de canale luate în considerare. În cazul de faţă, se realizează conversia la 16 KHz şi este folosit un singur canal deoarece aceasta este configuraţia audio necesară serviciilor de recunoaştere de vorbire şi vorbitor. Pentru a nu fi afectată performanţă paginii şi interacţiunea cu utilizatorul, toate aceste operaţii sunt executate în fundal de către un obiect de tip *Worker()*.

4.6 IMPLEMENTAREA BAZEI DE DATE

Baza de date aferentă serviciului de autentificare rulează în cadrul unui server MySQL şi este compusă din 3 tabele: *users*, *audio*, *login*.

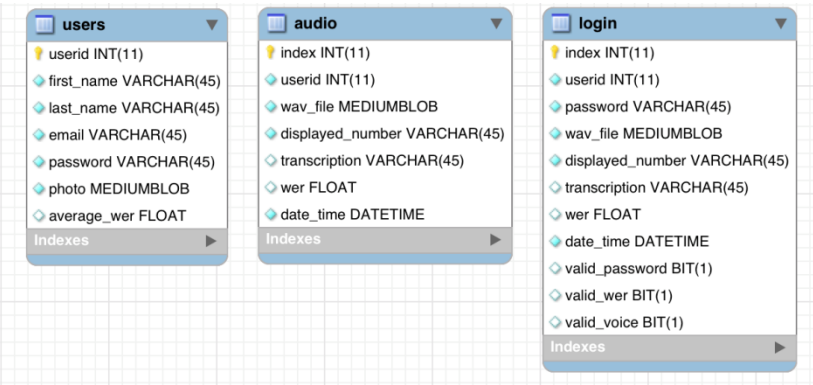


Figura 4.7 Tabelele bazei de date

Primul tabel conţine datele personale ale utilizatorilor şi valoarea medie a erorii la nivel de cuvânt, obţinută în urma transcrierii clipurilor audio create în etapa de înrolare. În următoarea secvenţă SQL se poate observa instrucţiunea de creare a tabelului, tipurile de date ale câmpurilor, cheile primară şi unică.

```
CREATE TABLE `users` (
```

```
  `userid` INT(11) NOT NULL AUTO_INCREMENT,
  `first_name` VARCHAR(45) NOT NULL,
  `last_name` VARCHAR(45) NOT NULL,
  `email` VARCHAR(45) NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `photo` MEDIUMBLOB NOT NULL,
  `average_wer` FLOAT NULL,
  PRIMARY KEY (`userid`),
  UNIQUE INDEX `email_UNIQUE` (`email` ASC));
```

Al doilea tabel conţine clipurile audio provenite de la toţi utilizatorii înrolaţi în sistem. Pentru fiecare clip se cunoaşte vorbitorul, secvenţa de cifre afişată în aplicaţia client, secvenţa de cifre rezultată în urma transcrierii realizate de serviciul de recunoaştere de vorbire, rata de eroare la nivel de cuvânt şi momentul de timp la care a fost creat.

```
CREATE TABLE `audio` (
  `index` INT NOT NULL AUTO_INCREMENT,
  `userid` INT NOT NULL,
  `wav_file` MEDIUMBLOB NOT NULL,
  `displayed_number` VARCHAR(45) NOT NULL,
  `transcription` VARCHAR(45) NULL,
  `wer` FLOAT NULL,
  `date_time` DATETIME NOT NULL,
  PRIMARY KEY (`index`));
```

Al treilea tabel este dedicat tentativelor de autentificare, reuşite sau nu. Acesta conţine identificatorul numeric al utilizatorului, parola completată, secvenţa de cifre afişată şi cea transcrisă, rata de eroare la nivel de cuvânt, momentul de timp, precum şi 3 câmpuri ce pot lua valori binare în funcţie de rezultatul celor 3 verificări efectuate la autentificare.

```
CREATE TABLE `login` (
  `index` INT NOT NULL AUTO_INCREMENT,
  `userid` INT NOT NULL,
  `password` VARCHAR(45) NOT NULL,
  `wav_file` MEDIUMBLOB NOT NULL,
  `displayed_number` VARCHAR(45) NOT NULL,
  `transcription` VARCHAR(45) NULL,
  `wer` FLOAT NULL,
  `date_time` DATETIME NOT NULL,
  `valid_password` BIT(1) NULL DEFAULT b'0',
  `valid_wer` BIT(1) NULL DEFAULT b'0',
  `valid_voice` BIT(1) NULL DEFAULT b'0',
  PRIMARY KEY (`index`));
```

Toate acţiunile asupra bazei de date au loc prin intermediul procedurilor stocate. A fost aleasă această metodă pentru o mai bună organizare a codului aplicaţiei. Au fost create 19 proceduri stocate:

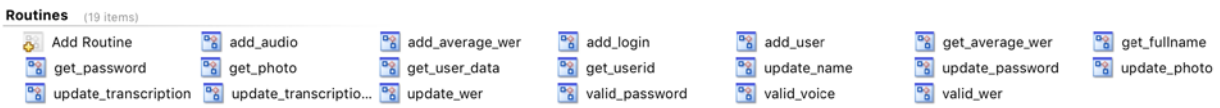


Figura 4.8 Lista procedurilor stocate

Următoarea secvenţă SQL prezintă instrucţiunea de creare a procedurii stocate corespunzătoare acţiunii de extragere a identificatorului numeric al utilizatorului:

```
DELIMITER $$
CREATE PROCEDURE `get_userid`(ÎN arg_email VARCHAR (45), OUT arg_userid INT)
BEGIN
      SELECT userid INTO arg_userid from users WHERE email=arg_email;
END $$
DELIMITER ;
```

Procedura este definită sub forma unei funcţii cu două argumente. Adresa de poştă electronică reprezintă argumentul de intrare, pe baza acestuia se va selecta din tabel identificatorul numeric corespunzător, livrat apoi ca argument de ieşire.

4.7 IMPLEMENTAREA SERVICIULUI DE AUTENTIFICARE

Serviciul de autentificare a fost dezvoltat în mediul de programare Eclipse Marş, utilizând limbajul Java şi platforma de dezvoltare Jersey, specifică serviciilor REST. Este compus din mai multe clase organizate în pachete. În următoarea imagine este prezentată diagramă UML a claselor:

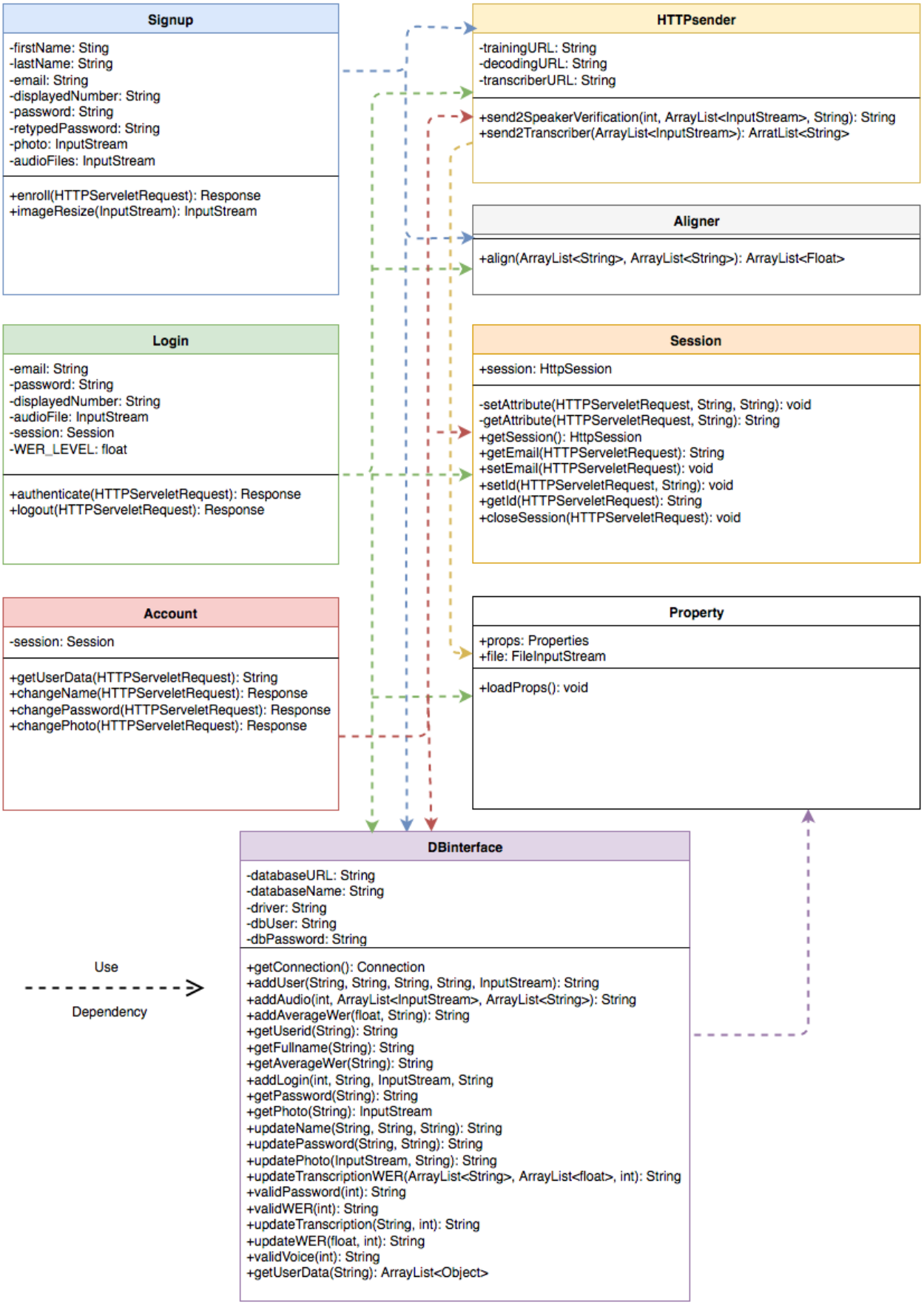


Figura 4.9 Diagramă UML a claselor

Clasa Dbinterface.java

Această clasă funcţionează ca o interfaţă între serviciul REST propriu zis şi baza de date, unde fiecărei proceduri stocate îi corespunde o funcţie definită în clasa. Următoarea secvenţă prezintă funcţia corespunzătoare extragerii identificatorului numeric al unui utilizator, cunoscandu-se adresa sa de poştă electronică:

```
public String getUserid(String email){
            con = getConnection();
            try{
                  CallableStatement statement = null;
                  String query = "{call get_userid(?,?)}";
                  statement = (CallableStatement) con.prepareCall(query);
                  statement.setString(1, email);
                  statement.registerOutParameter(2, java.sql.Types.INTEGER);
                  int queryResult=statement.executeUpdate();
                  userid = statement.getInt(2)+"";
                  statement.close();
                  con.close();
            }catch(SQLException e){
                  return Finals.NONEXISTENT_EMAIL;
            }
            return userid;
      }
```

După stabilirea conexiunii cu baza de date, se utilizează un obiect *CallableStatement*, căruia îi vor fi atribuiţi parametri de intrare şi ieşire, conform procedurii stocate apelate. În urma execuţiei, este extrasă valoarea parametrului de ieşire.

Clasa HTTPsender.java

Prin cele două funcţii ale sale, clasa realizează comunicarea serviciului de autentificare cu celelalte două servicii, cel de recunoaştere de vorbire, respectiv recunoaştere de vorbitor. Către acestea sunt trimise prin protocolul HTTP clipurile audio primite de la aplicaţia client. În continuare sunt prezentate principalele secvenţe de cod ale uneia dintre funcţii:

```
public static ArrayList<String> send2Transcriber(ArrayList<InputStream> file){

            ClientConfig config = new DefaultClientConfig();
            Client client = Client.create(config);
            WebResource service =
client.resource(UriBuilder.fromUri(transcriberURL).build());
.....................................................
FormDataMultiPart formDataMultiPart = new FormDataMultiPart();
            for (int i = 0; i < numberOfFiles; i++){
                  formDataMultiPart.field("audioFile",
file.get(i),MediaType.APPLICATION_OCTET_STREAM_TYPE);
            }
.....................................................
clientResp =
service.type(MediaType.MULTIPART_FORM_DATA_TYPE).post(ClientResponse.class,
formDataMultiPart);
.....................................................
String response=clientResp.getEntity(String.class);
.....................................................
}
```

Este creat şi configurat un obiect *Client* capabil să execute o cerere HTTP către serviciul de recunoaştere de vorbire. Apoi este creat obiect de tip formular ce conţine mai multe câmpuri, fiecare câmp fiind denumit *audioFile* şi având atribuit un clip audio. În urma execuţiei cererii, răspunsul întors (transcrierea fişierelor) este stocat într-un obiect *String*.

Clasa Session.java

Cererile HTTP sunt de regulă complet independente. Totuşi, uneori este nevoie ca anumite informaţii să fie reţinute de la o cerere la alta. Acest lucru este realizabil prin intermediul sesiunilor. Atunci când un utilizator vizitează pentru prima dată o pagină web, se creează o sesiune caracterizată de un identificator numeric unic. Acesta este păstrat atât pe parte de server, cât şi pe parte de client, într-un fişier *cookie*. O sesiune poate stoca atribute cu scopul de a le citi mai târziu. După ce un utilizator s-a autentificat, este de dorit ca el să fie recunoscut pe parcursul cererilor următoare. Astfel, se va salva într-o sesiune adresa sa de poştă electronică şi identificatorul numeric corespunzător ce se regăseşte în baza de date. Dacă ulterior va executa o acţiune care necesită în prealabil autentificare, se va consulta informaţia salvată în sesiune pentru a stabili identitatea utilizatorului. Clasa *Session.java* implementează metode de manipulare a atributelor unei sesiuni.

```
private void setAttribute(@Context HttpServletRequest request, String numeAtribut,
String valoareAtribut){
              session = request.getSession(true);
              session.setAttribute(numeAtribut, valoareAtribut);
       }

private String getAttribute(@Context HttpServletRequest request, StringnumeAtribut){
              session = request.getSession(true);
              Object result = session.getAttribute(numeAtribut);
              if (result != null) {
                     return result.toString();
              }else {
                     return null;
              }
       }
public void closeSession(@Context HttpServletRequest request){
              session = request.getSession(true);
              session.invalidate();
       }
```

Clasa Aligner.java

Este o interfaţă pentru clasa *NISTAlign.java*, disponibilă în utilitarul CMU Sphinx. Pe baza secvenţei de cifre afişată în aplicaţia client şi a secvenţei de cifre obţinute în urma transcrierii unui clip audio, această clasă calculează rata de eroare la nivel de cuvânt.

Clasa Property.java

Este o clasă prin intermediul căreia sunt iniţializate anumite constante din serviciul web cu valorile lor corespunzătoare aflate într-un fişier de configurare.

Clasa Signup.java

```
@Path("signup")
public class Signup {
.....................................................
       @POST
       @Path("enroll")
       @Consumes(MediaType.MULTIPART_FORM_DATĂ)
       public Response enroll(@Context HttpServletRequest request){

List<FileItem> items = new ServletFileUpload(new
DiskFileItemFactory()).parseRequest(request);

              for (FileItem item : items) {
                            String fieldname = item.getFieldName();
                            switch (fieldname) {
```

```
                        case "audioFile":
                               audioFiles.add(item.getInputStream());

                               break;
                        case "photo":
                               photo = imageResize(item.getInputStream());
                               break;
                        case "displayedNumber":
                               displayedNumber.add(item.getString());
                               break;
                        case "lastName":
                               lastName = item.getString();
                               break;
                        case "firstName":
                               firstName = item.getString();
                               break;
                        case "email":
                               email = item.getString();
                               break;
                        case "password":
                               password = item.getString();
                               break;
                        case "retypedPassword":
                               retypedPassword = item.getString();
                               break;
                    }
                }
.....................................................................
                return Response.status(Status.OK).build();
        }
```

În urma unui HTTP POST către calea specificată în adnotări, va fi executată funcţia *enroll*. Prin analiza conţinutului cererii, va fi identificat un formular cu mai multe câmpuri. Acesta conţine datele personale ale utilizatorului şi clipurile audio înregistrate. Utilizând un obiect de tip *DBInterface*, datele vor fi introduse în baza de date, iar cu ajutorul clasei *HTTPsender* clipurile audio vor fi trimise la serviciile de verificare de vorbire şi vorbitor. Dacă toate operaţiile decurg cu succes, aplicaţia client va primi înapoi un mesaj ce confirmă acest lucru.

Clasa **Login.java** o structură similară cu aceasta şi nu va mai fi prezentată.

Clasa Account.java

Dispune de o metodă de obţinere a datelor utilizatorului şi 3 metode de modificare a datelor acestuia.

```
@Path("/account")
public class Account {
@GET
@Path("/getUserData")
@Produces("application/xml")
public String getUserData(@Context HttpServletRequest request) throws{
.....................................................................
                String email=session.getEmail(request);
                if(email == null){
                        return "you are not logged!";
                }
.....................................................................
                DBinterface db_manager = new DBinterface();
                userData=db_manager.getUserData(email);
```

```
            firstName=(String) userData.get(0);
            lastName=(String) userData.get(1);
            photo=(byte[]) userData.get(2);
            String photoBase64 = new sun.misc.BASE64Encoder().encode(photo);
                String xmlOutput="<user>"
                   + "<firstName>"+firstName+"</firstName>"
                   +"<lastName>"+lastName+"</lastName>"
                   +"<photo>"+photoBase64+"</photo>"
                   +"<email>"+email+"</email>"
                   +"</user>";
                return xmlOutput;
    }
```

Funcţia *getUserData()* este apelată în urma unei cereri HTTP GET către calea specificată în adnotări şi întoarce un răspuns în format XML. Deoarece extragerea datelor personale este o operaţie ce necesită ca utilizatorul să fie autentificat, acest lucru se va verifica prin citirea sesiunii. Aceasta ar trebui să conţină adresa de poştă electronică a utilizatorului. Apoi, prin utilizarea unui obiect *DBInterface*, datele personale ale utilizatorului vor fi extrase din baza de date şi returnate aplicaţiei client.

Funcţiile de modificare a datelor personale sunt apelate în urma unor cereri HTTP POST, fiind asemănătoare cu funcţia *enroll* din clasa *Signup.java*.

4.8 CONCLUZII ŞI DEZVOLTĂRI ULTERIOARE

Aplicaţia oferă posibilitatea autentificării pe bază de parolă şi recunoaştere de vorbire şi amprentă vocală, reuşind să îndeplinească sarcina pentru care a fost concepută. Precizia cu care unui utilizator îi este acceptată sau respinsă încercarea de autentificare este strict proporţională cu eficienţa sistemelor de recunoaştere de vorbire şi vorbitor. Totodată, aplicaţia poate fi considerată un punct de plecare în vederea realizării serviciilor de tip REST.

Pentru dezvoltările viitoare ale proiectului sunt vizate următoarele ţinte:

- Implementarea unor soluţii de eliminare a vulnerabilităţii descrisă în secţiunea 4.3 Scenarii de utilizare (Scenariul 4).
- Extinderea aplicaţiei sub forma unui proiect pilot ce presupune provocarea cât mai multor utilizatori de a se înrola în aplicaţie. Crearea unui sistem competiţional în care utilizatorii să fie încurajaţi să încerce autentificări neautorizate. În acest fel, aplicaţia va fi testată intens şi vor putea fi observate alte posibile probleme. Cu această ocazie se vor obţine baze de date de vorbire de mari dimensiuni, iar sistemul de recunoaştere de vorbire (recunoaşterea secvenţelor de cifre) ar putea fi îmbunătăţit.
- Îmbunătăţirea performanţelor aplicaţiei din punct de vedere al timpului de răspuns. Au fost efectuate teste pe o maşină cu sistem de operare OS X, procesor Intel i5 2.4 GHz şi 8 GB memorie. S-a observat că sistemul de recunoaştere de vorbire este mai lent, încetinind astfel aplicaţia mai ales în etapa de înrolare, ce presupune transcrierea a cel puţin 10 clipuri audio.
  Deşi implementarea serviciului de recunoaştere de vorbire nu este subiectul acestei lucrări, am încercat îmbunătăţirea sa. În implementarea iniţială, transcrierea clipurilor se face secvenţial. Mai mult, resursele necesare (model acustic, fonetic, lingvistic) şi obiectul ce realizează transcrierea (de tip *recognizer*, specific CMU Sphinx) sunt încărcate în momentul în care se primesc clipurile audio. Deoarece încărcarea lor are o durată ce nu poate fi neglijată, resursele necesare precum şi obiectul *recognizer* au fost încărcate în memorie la momentul pornirii aplicaţiei şi ţinute în aşteptare până la apariţia unui client. A fost creat un grup de fire de execuţie, egal cu numărul de nuclee al procesorului (4). La înrolarea unui utilizator, fiecare dintre cele 10 clipuri audio este

procesat în paralel, pe un fir de execuţie separat, folsind unul dintre obiectele *recognizer* preîncărcate. Au fost obţinute următoarele valori experimentale:

- crearea unui obiect *recognizer*: 0.6 s;
- transcrierea unui singur clip audio: 0.6 (valoare medie);
- crearea modelului (amprentei vocale) a unui utilizator: 1.42 s;
- verificarea identităţii (amprentei vocale) a unui utilizator: 0.63 s;

De asemenea, au fost obţinute şi valori ce compara cele două implementări ale serviciului de recunoaştere de vorbire:

- transcrierea secvenţială a 10 clipuri audio, folosind un singur obiect *recognizer*, încărcat la momentul execuţiei: 5.84 s;
- transcrierea paralelă cu 4 fire de execuţie a 10 clipuri audio, folosind obiecte *recognizer* preîncărcate: 3.86 s.

Se observă că implementarea îmbunătăţită conferă un câştig în ceea ce priveşte timpul de execuţie. Acesta este însă destul de mic din cauza faptului că sarcina de transcriere are o durată mică şi relativ comparabilă cu încetinirea apărută din cauza comutării între firele de execuţie.

Aşadar, sunt investigate în continuare noi metode de accelerare a timpului de execuţie al aplicaţiei.

CAPITOLUL 5

APLICAŢIA DE DETECŢIE A CUVINTELOR CHEIE

Aplicaţia înglobează într-un proiect software sistemul de recunoaştere automată a vorbirii "Molina", dezvoltat cu ajutorul utilitarului Kaldi şi prezentat pe larg în secţiunea 2.2. Scopul aplicaţiei este de a efectua 3 sarcini principale:

- Obţinerea transcrierii text a unui clip audio din baza de date Molina;
- Identificarea posibilielor apariţii ale cuvintelor cheie în clipul audio, dată fiind în prealabil o listă a cuvintelor cheie căutate;
- Verificarea răspunsului unui sistem interactiv cu răspuns vocal; aşa cum s-a prezentat în secţiunea 2.2.1, baza de date Molina este formată din clipuri audio cu un format special, ce imită răspunsul oferit de un sistem interactiv cu răspuns vocal. Clipurile sunt compuse din părţi statice şi părţi dinamice, considerându-se că aparţin aceluiaşi tip de frază dacă

diferă numai prin părţile lor dinamice, părţile statice fiind identice. Astfel, verificarea răspunsului unui sistem interactiv cu răspuns vocal constă în determinarea tipului de frază căruia aparţine un clip audio din baza de date, dată fiind în prealabil o listă a tipurilor de fraze posibile.

5.1 DESCRIERE ŞI ARHITECTURĂ

Aplicaţia a fost dezvoltată în limbajul C sub forma a două module, client şi server, capabile să comunice prin protocolul TCP.

Modulul client permite utilizatorului să citească un fişier audio de pe disc şi să-l trimită serverului sub forma unui flux audio. Modulul server primeşte fluxul audio de la client şi realizează transcrierea lui text, identifica cuvinte cheie şi tipul frazelor transcrise având în vedere mai multe categorii de fraze ce pot constitui ieşirea unui sistem IVR. Toate acestea vor fi afişate în timp real în consola serverului. Imaginea de mai jos reprezintă arhitectura aplicaţiei:

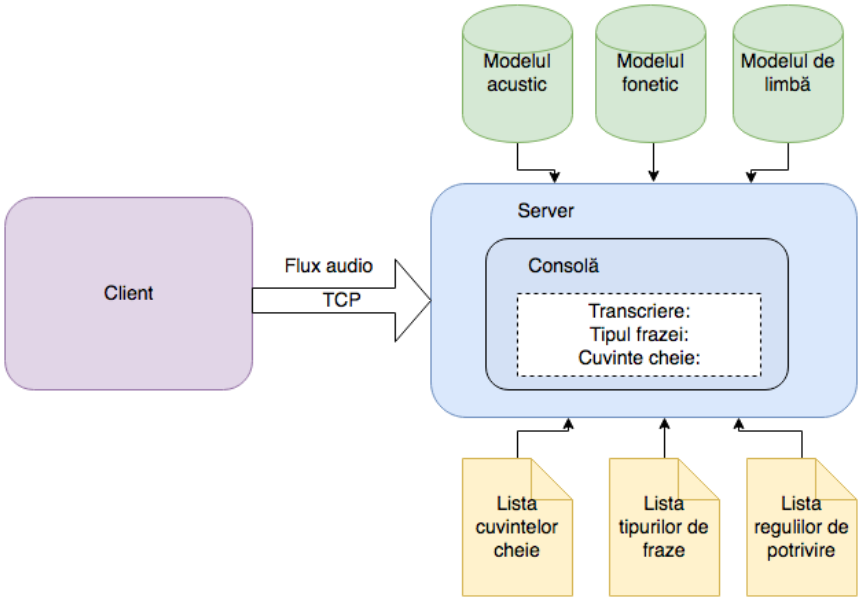


Figura 5.1 Arhitectura aplicaţiei de detecţie a cuvintelor cheie

Se pot observa cele două module (client şi server). Serverul încarcă cele 3 modele (acustic, fonetic şi lingvistic) corespunzătoare sistemului ASR Molina dar şi 3 liste:

- o listă ce conţine cuvintele cheie căutate;
- o listă ce conţine tipurile de fraze ce pot constitui ieşirea unui sistem IVR.;
- o listă cu reguli care ajută la categorisirea frazelor în tipurile din lista precedentă.

Serverul astfel configurat va asculta pe un anumit port un flux de date audio şi va scoate la ieşire transcrierea acestuia, tipul de fraze din care face parte precum şi cuvintele cheie identificate.

5.2 FUNCŢIONALITATE DIN PERSPECTIVA UTILIZATORULUI

Mai întâi, pentru rularea modulului client este necesar ca Gstreamer 1.0 să fie instalat în sistemul de operare. Modulul dispune de un fişier de configurare în care se stabileşte adresa IP a maşinii pe care se află modulul server, precum şi portul pe care se va trimite fluxul audio.

Pentru rularea modulului server, suplimentar faţă de cazul precedent este necesar utilitarul Kaldi. Similar, acest modul dispune de un fişier de configurare în ca re se stabilesc căile către

resursele folosite, precum şi adresa IP a maşinii pe care rulează, împreună cu portul pe care va aştepta conexiuni.

Aplicaţia rulează direct din linia de comandă, nefiind disponibilă o interfaţă grafică. Figura următoare ilustrează două terminale. În cel de sus rulează modulul server, în timp ce în cel de jos rulează modulul client.

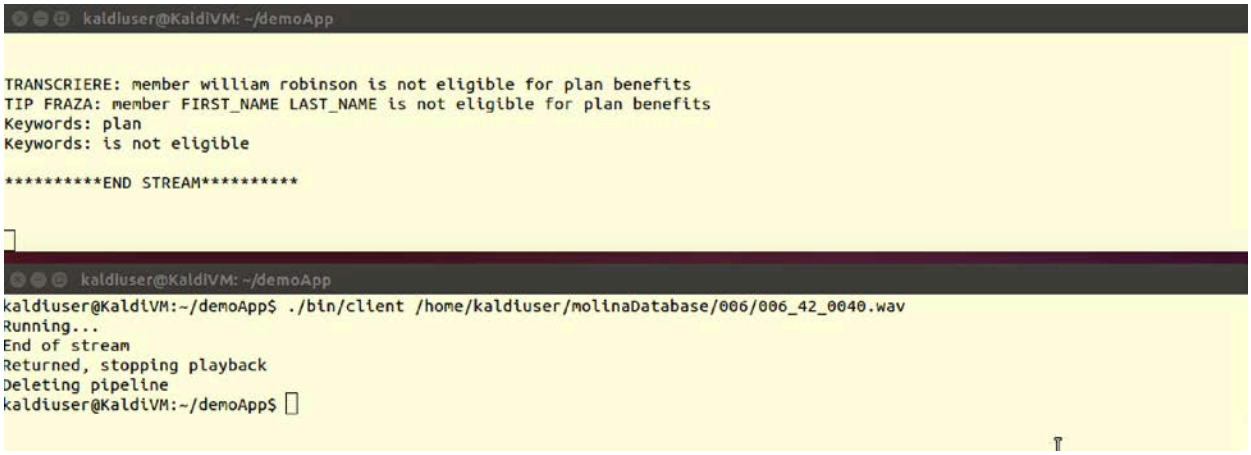


Figura 5.2 Rularea aplicaţiei de detecţie a cuvintelor cheie

Prima dată trebuie pornit modulul server. Acesta îşi va încărca resursele necesare şi va aştepta o conexiune pe portul configurat. Modulul client se apelează având ca argument calea către un clip audio.

Se poate observa rezultatul afişat în urma rulării aplicaţiei.

5.3 TEHNOLOGIA GSTREAMER

Gstreamer este o platformă de dezvoltare multimedia cu sursă deschisă care a fost implementată în mai multe limbaje de programare, iar în cazul de faţă fiind folosită biblioteca de funcţii din Limbajul C. Aceasta funcţionează pe majoritatea sistemelor de operare fiind folosită cu preponderenţă în sistemele Linux. Gstreamer este specializat în prelucrarea fluxurilor de date audio/video.

Un bloc Gstreamer are aspectul unui graf, ce poartă denumirea de *pipeline*, fiind format din elemente înlănţuite, unde datele curg dinspre elementul sursa (producător) către elementul final (consumator). De obicei, elementele implementează diverse filtre de semnal. Fiecare element conţine un port de intrare în element, numit *sinkpad*, în timp ce portul prin care datele părăsesc elementul se numeşte *sourcepad*. De asemenea, fiecare bloc deţine mai multe proprietăţi ce pot fi configurate. Mai multe detalii despre un anumit element Gstreamer pot fi obţinute prin comandă *gst-inspect-1.0 <nume-element>* tastata în terminal. Imaginea următoare ilustrează cele două blocuri *pipeline* ale aplicaţiei, cel corespunzător modulului client şi cel corespunzător modulului server.

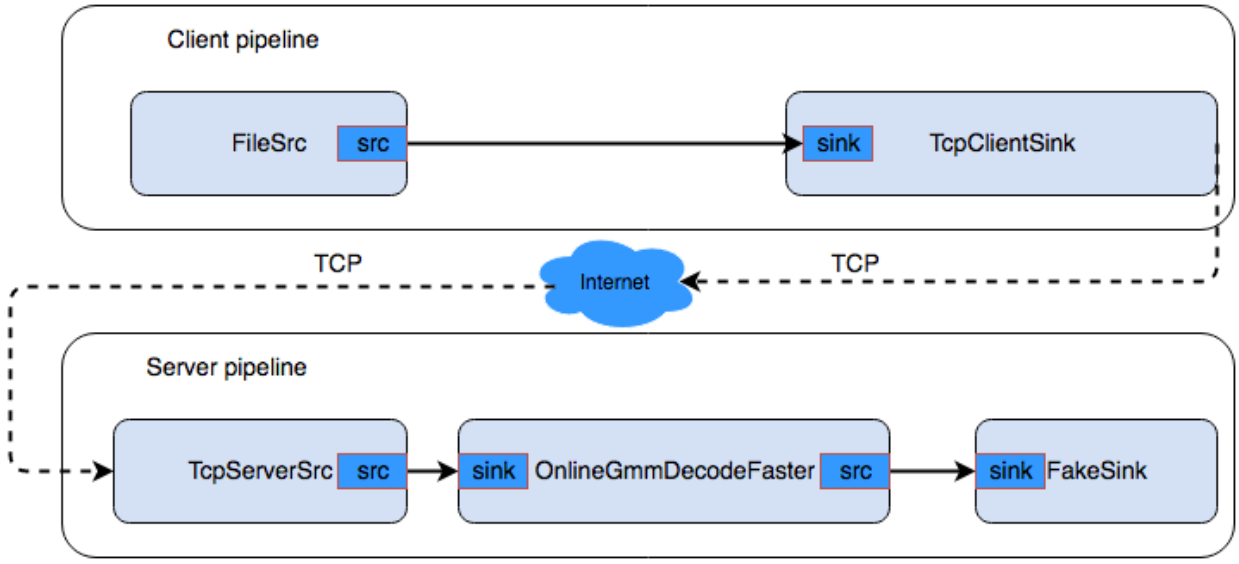


Figura 5.3 Pipeline-ul GStreamer

FileSrc este un element sursă, acesta citeşte un fişier de pe disc şi îl poate trimite mai departe prin portul *src*. Cele mai importante proprietăţi ale sale sunt: *location* (calea către fişierul ce va fi citit), *blocksize* (dimensiunea în octeţi a unui bloc de date prelucrat), *name* (elementului îi poate fi atribuit un nume).

TcpClientSink este un element ce implementează un client capabil să trimită date peste reţea prin protocolul TCP. Acesta are un singur port, *sink*, prin care primeşte date de la elementele precedente din *pipeline* şi le trimite mai departe spre o anumită adresă IP şi un port specific. Cele mai importante proprietăţi ale sale sunt *host* (adresa IP către care trimite datele) şi *port* (portul pe care va trimite datele).

TcpServerSrc etste un element ce implementează un server care poate primi date din reţea prin protocolul TCP. Prin portul *src* va trimite mai departe datele recepţionate către alte elemente din *pipeline*. Similar cu elementul client, acestuia îi pot fi configurate proprietăţile *host* şi *port*.

OnlineGmmDecodeFaster este un element ce implementează un modul de transcriere în timp real, specific utilitarului Kaldi. Acesta dispune de două porturi, unul de intrare prin care primeşte fluxul audio (în formatul S16LE, 1 canal, 16 KHz) şi altul de ieşire prin care furnizează transcrierea text. Proprietăţile sale trebuiesc configurate stabilind calea către resursele unui sistem de recunoaştere automată a vorbirii dezvoltat cu Kaldi. Dintre proprietăţile sale pot fi amintite: *model*, *fst*, *lda-mat*, *word-syms*, etc. Specific elementelor Gstreamer, *OnlineGmmDecodeFaster* are capabilitatea de a emite semnale ce vor declanşa execuţia unor funcţii. În cazul de faţă, în momentul în care este obţinută transcrierea text a unui flux audio, semnalul *hyp-word* este emis, fapt ce va declanşa apelul funcţiei cu următorul antet:

*void word_output (GstElement *asr, gchar *word, gpointer dată)*, unde *asr* este o referinţă la elementul *OnlineGmmDecodeFaster*, iar *word* reprezintă transcrierea text generată.

Deoarece în Gstreamer orice date produse trebuiesc consumate, elementul ***FakeSink*** are rolul de fals consumator, transcrierea text primită fiind redirecţionată către linia de comandă.

5.4 DESCRIEREA ALGORITMULUI

Secţiunea curentă va descrie paşii executaţi de către aplicaţie pentru a îndeplini cele 3 sarcini propuse: transcrierea clipurilor, identificarea cuvintelor cheie, identificarea tipului de frază. Următoarea diagramă UML de stare ilustrează paşii ce vor fi prezentaţi:

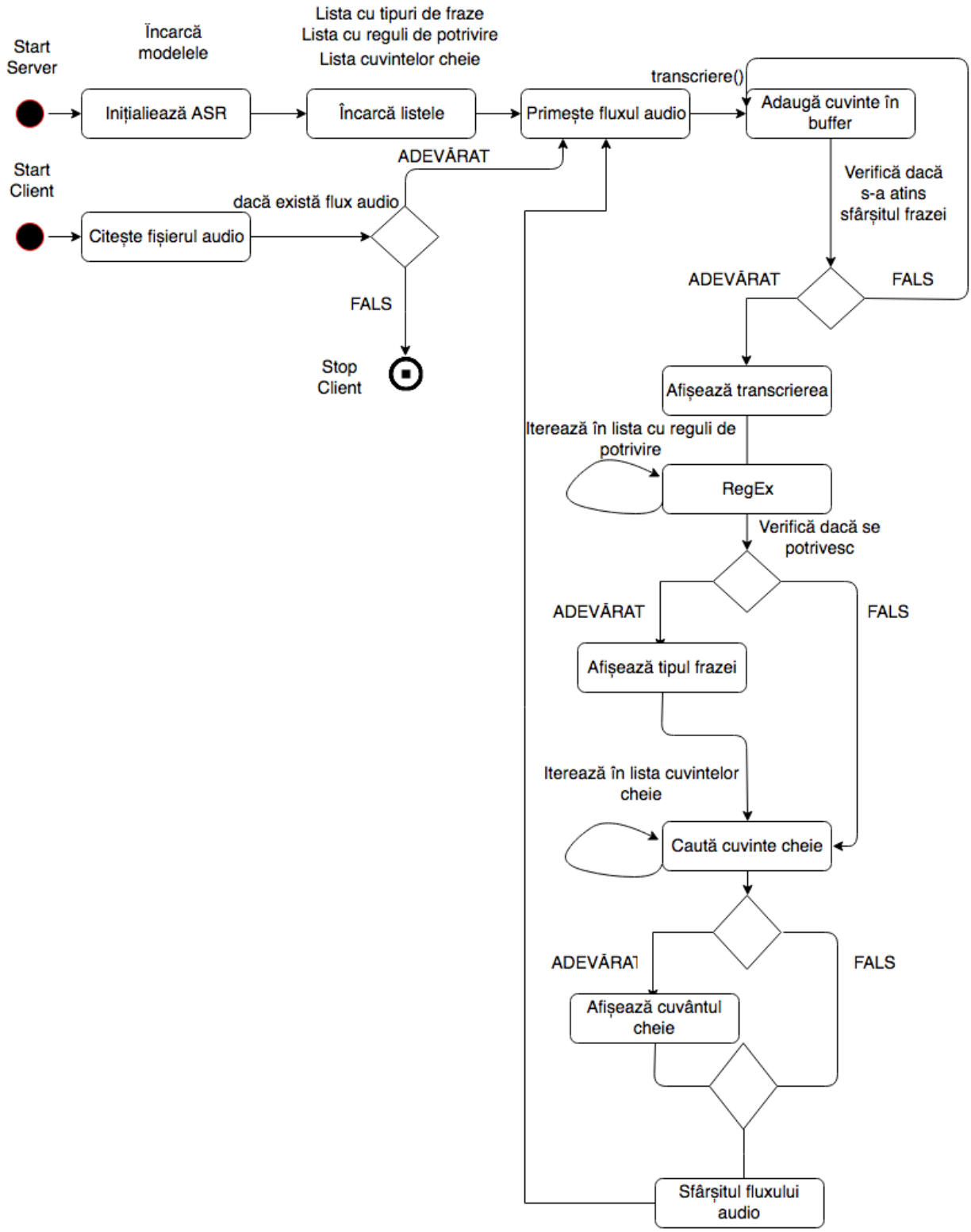


Figura 5.4 Diagramă UML de stare a aplicaţiei de detecție a cuvintelor cheie

Modulul server

1. Citeşte fişierul de configurare.
2. Încarcă resursele ASR (modelul acustic, graful de decodare, lista cuvintelor, lista fonemelor, matricea cu caracteristici LDA-MLLT).

3. Încarcă listele (lista tipurilor de fraze, lista regulilor de potrivire, lista cuvintelor cheie).
4. Aşteaptă conexiunea clientului. Pentru fiecare client conectat:
   4.1. Preia fluxul audio. Începe transcrierea fluxului. Pentru fiecare cuvânt transcris:
      4.1.1. Adaugă cuvântul într-un şir în care se va forma fraza.
      4.1.2. Verifică dacă sistemul de transcriere a returnat simbolul de delimitare a frazelor (*<#s>*).
   4.2. Când delimitatorul de frază este returnat, şirul va conţine transcrierea integrală a clipului, aceasta fiind afişată.
   4.3. Se încearcă potrivirea şirului ce conţine transcrierea integrală cu fiecare regulă de potrivire.
   4.4. Dacă o potrivire a fost realizată cu succes, tipul frazei este afişat.
   4.5. Fiecare cuvânt cheie este căutat în transcrierea text.
   4.6. Cuvintele cheie identificate sunt afişate la consolă.
   4.7. Şirul cu transcrierea text este resetat.
   4.8. Algoritmul se întoarce la pasul 4 (aşteaptă o nouă conexiune din partea clientului).

Modulul client

1. Citeşte un fişier audio de la disc.
2. Dacă fişierul nu este gol, clientul îl timite la server.
3. Atunci când tot şirul de octeţi ai fişierului au fost trimişi, aplicaţia client se opreşte.

5.5 VERIFICAREA UNUI SISTEM INTERACTIV CU RĂSPUNS VOCAL

La începutul capitolului s-a reamintit în ce constă verificarea unui sistem interactiv cu răspuns vocal. Această sarcină a fost îndeplinită folosind o bibliotecă C ce implementează **expresii regulate**. O expresie regulată este un şir de caractere text ce permite crearea unui model cu ajutorul căruia se vor face potriviri, se va manipula sau se va indentifica o secvenţă text.

Următorul exemplu constituie un **tip de frază** specific bazei de date Molina:

member FIRST_NAME LAST_NAME is not eligible for plan benefits plan name PLAN_NAME

Cuvintele scrise cu majuscule sunt părţile dinamice ale frazei, în timp ce restul cuvintelor scrise cu litere mici sunt părţile statice.

Folosind expresiile regulate, pentru acest tip de frază a fost creată următoarea **regulă de potrivire**:

member .+? is not eligible for plan benefits plan name .+? <#s>

Operatorul punct (".") va realiza potrivirea cu orice caracter, mai puţin caracterul de linie nouă. Operatorul "+" potriveşte cu una sau mai multe apariţii ale caracterului precedent. Operatorul "?" potriveşte cu cel mult o apariţie a caracterului precedent. Cei trei operatori luaţi împreună, ".+?", vor potrivi orice caracter de un număr nelimitat de ori, dar în acelaşi timp de cât mai puţine ori posibil. Restul elementelor din expresia regulată (cuvântul "member" sau grupul de cuvinte "is not eligible for plan benefits") trebuie să se regăsească în mod identic pentru a fi posibilă potrivirea. Grupul de caractere *<#s>* este generat de către sistemul de transcriere a semnalului vocal în text şi reprezintă o parte statică o frazei, marcând sfârşitul acesteia.

Considerând următoarele fraze ca fiind ieşirea unui sistem interactiv cu răspuns vocal, se va încerca potrivirea lor cu expresia regulată definită anterior:

1. *member John Smith is not eligible for plan benefits plan name United Healthcare Plan <#s>*
2. *member Lauren Thompson is eligible for plan benefits plan name United Healthcare Plus <#s>*

3. *member Nicole Brown is not eligible <#s>*

Prima frază se va potrivi cu expresia regulată deoarece părţile statice sunt identice, în timp ce părţile dinamice reprezintă o expansiune de caractere luată de oricâte ori. Cea de-a doua frază nu va realiza potrivirea deoarece lipseşte cuvântul "not" din cea de-a doua parte statică a frazei. Nici ultima frază nu va realiza potrivirea deoarece lipseşte partea statică "beneftis for plan name" şi ultima parte dinamică, "PLAN_NAME".

5.6 IMPLEMENTARE

Ambele module au fost implementate în limbajul C sub sistemul de operare Linux, folosind librăria GNU C (glib), specifică sistemelor Unix, ce asigură înaltă performanţă şi portabilitate.

Modulul server

Pentru a funcţiona corespunzător, acesta are nevoie de următoarele:

- Lista cuvintelor cheie căutate (câte unul pe fiecare rând);
- Lista regulior de potrivire a frazelor (câte una pe fiecare rând, similar cu cea explicată în secţiunea 5.5);
- Lista tipurilor de fraze (câte una pe fiecare rând, similar cu cea oferită ca exemplu în secţiunea 5.5);
- Resursele sistemului de recunoaştere: modelul acustic, graful de decodare, lista cuvintelor, lista fonemelor, matricea cu caracteristici LDA-MLLT;
- Un fişier de configurare, format din mai multe secţiuni: secţiunea cailor spre liste şi spre resursele sistemului de recunoaştere, secţiunea informaţiilor referitoare la conexiune (IP, port), secţiune ce păstrează numărul cuvintelor din listă de cuvinte cheie, respectiv numărul tipurilor de fraze şi al regulilor de potrivire.

Codul modulului este structurat într-un singur fişier ce conţine funcţia principală, împreună cu definiţiile altor câteva funcţii apelate de către aceasta.

Funcţia *read_config_file()* este funcţia care citeşte fişierul de configurare. Elementele din acesta sunt identificate în funcţie de numele lor şi numele secţiunii. Fie exemplu următor:

```
HOST_IP = g_key_file_get_string(gKeyFile,"connection","HOST_IP",&error);
HOST_PORT = g_key_file_get_integer(gKeyFile,"connection","HOST_PORT",&error);
```

Celor două variabile declarate global le vor fi atribuite valorile din fişierul de configurare ce corespund secţiunii *connection*, mai exact valorile elementelor *HOST_IP* şi *HOST_PORT*.

Funcţia *read_resources()* încarcă în memorie cele 3 liste necesare modulului. Conţinutul listelor este citit şi plasat în câte o structură de tip matrice.

Funcţia *bus_call()* reprezintă un sistem de gestiune a mesajelor de eroare ce pot apărea într-un *pipeline* Gstreamer.

Funcţia *search_for_keywords()* ia fiecare cuvânt din lista cuvintelor cheie şi îl caută în transcrierea rezultată.

Funcţia *word_output()* este o funcţie *callback*, fiind executată atunci când se declanşează în funcţia principală semnalul *hyp-word*, specific apariţiei unei noi transcrieri. Cât timp nu au fost generate caracterele ce marchează sfârşitul frazei (*<#s>*), fiecare nou cuvânt transcris este adăugat într-o listă în care se va forma fraza. Când fraza a fost transcrisă integral, se va încerca determinarea tipului ei prin apelul funcţiei *test_matching()*.

Funcţia *int test_matching(gchar *buffer)* primeşte ca argument transcrierea integrală a frazei. Prin folosirea bibliotecii *regex.h* se va încerca potrivirea cu fiecare regulă de potrivire din fişier, în acest fel fiind determinat tipul frazei. Instrucţiunea *regex = g_regex_new (pattern, 0, 0, &err);* va compila regulă de potrivire ("pattern"), în timp ce instrucţiunea *g_regex_match (regex, buffer, 0, &matchInfo);* va testa potrivirea cu transcrierea frazei regăsită în variabilă *buffer*.

Funcţia *main()* este funcţia principală a programului. Aceasta începe prin declararea unor obiecte specifice librăriei Gstreamer. Astfel, cele 3 elemente Gstreamer ale serverului sunt declarate, împreună cu *pipeline-ul*, un container în care acesta va rula şi elementul *bus* de gestiune a mesajelor de eroare.

```
gint main (gint   argc, gchar *argv[])
{
    GMainLoop *loop;
    GstElement *pipeline, *src, *asr, *fakesink;
    GstBus *bus;
```

Se va iniţializa lista ce va conţine transcrierea frazelor generată de sistemul ASR.

```
    transcriptionBuffer= g_string_new(NULL);
```

Se vor încărca cele 3 liste necesare: cuvintele cheie, regulile de transcriere şi tipurile de fraze.

```
    read_config_file();
    phraseTypes=read_resources(PHRASE_TYPES_PATH, PHRASE_TYPES_NR);
    matchRules=read_resources(MATCH_RULES_PATH, PHRASE_TYPES_NR);
    keywords=read_resources(KEYWORDS_PATH, KEYWORDS_NR);
```

Containerul Gstreamer este iniţializat.

```
    gst_init (&argc, &argv);
    loop = g_main_loop_new (NULL, FALSE);
```

Este creat un obiect de tip pipeline şi este iniţializat obiectul bus.

```
    pipeline = gst_pipeline_new ("pipeline");
    bus = gst_pipeline_get_bus (GST_PIPELINE (pipeline));
    gst_bus_add_watch (bus, bus_call, loop);
    gst_object_unref (bus);
```

Sunt create cele 3 elemente Gstreamer descrise în secţiunea 5.3.

```
    src = gst_element_factory_make ("tcpserversrc", "source");
    asr=gst_element_factory_make("onlinegmmdecodefaster", "asr");
    fakesink=gst_element_factory_make("fakesink","fakesink");
```

Sunt configurate proprietăţile elementelor.

```
    g_object_set (G_OBJECT (src), "host", HOST_IP, NULL);
    g_object_set (G_OBJECT (src), "port", HOST_PORT , NULL);

    g_object_set (G_OBJECT (asr), "fst", FST_PATH, NULL);
    g_object_set (G_OBJECT (asr), "lda-mat", LDA_PATH, NULL);
    g_object_set (G_OBJECT (asr), "model", AM_PATH, NULL);
    g_object_set (G_OBJECT (asr), "word-syms", WORDS_PATH, NULL);
    g_object_set (G_OBJECT (asr), "silence-phones", SILENCE_PHONES, NULL);
    g_object_set (G_OBJECT (asr), "max-active", MAX_ACTIVE, NULL);
    g_object_set (G_OBJECT (asr), "beam", BEAM,NULL);
    g_object_set (G_OBJECT (asr), "acoustic-scale", ACOUSTIC_SCALE,NULL);
    g_object_set (G_OBJECT (asr), "inter-utt-sil", INTER_UTT_SIL,NULL);
    g_object_set (G_OBJECT (asr), "silent", FALSE, NULL);
    g_object_set (G_OBJECT (fakesink), "dump", FALSE, NULL);
```

Elementele sunt adăugate în pipeline şi sunt create legăturile dintre acestea.

```
    gst_bin_add_many (GST_BIN (pipeline), src, asr, fakesink, NULL);
    gst_element_link_many (src, asr,fakesink, NULL);
```

Semnalul generat de elementul ce realizează transcrierea (asr) este conectat la funcţia ce va fi apelată la declanşarea sa.

```
    g_signal_connect (asr, "hyp-word", G_CALLBACK (word_output), NULL);
```

Pipeline-ul este pornit într-o buclă infinită şi permite prelucrarea a oricâte fluxuri audio primite secvenţial.

```
    while(1)
    {
        gst_element_set_state (pipeline, GST_STATE_PLAYING);
        g_main_loop_run (loop);
        gst_element_set_state (pipeline, GST_STATE_NULL);
    }
    gst_object_unref (GST_OBJECT (pipeline));
    return 0;
}
```

Modulul client a fost creat într-un mod similar cu modulul server. Elementele Gstreamer ale modulului client au fost descrise în secţiunea 5.3.

5.7 CONCLUZII ŞI DEZVOLTĂRI ULTERIOARE

Aplicaţia demonstrativă oferă funcţionalităţile pentru care a fost concepută. Pentru fluxurile audio ce imită răspunsul unui sistem IVR sunt realizate transcrierile, sunt identificate cuvintele cheie şi este determinat tipul de frază.

În momentul actual, aplicaţia foloseşte pentru transcrierea fluxurilor audio modele de tip HMM-GMM antrenate în Kaldi. Pentru dezvoltările ulterioare ale proiectului se urmăreşte integrarea în aplicaţie a modelelor acustice antrenate cu reţele neuronale. Astfel de modele au fost deja obţinute cu succes, iar acurateţea transcrierilor este superioară faţă de modelele bazate pe GMM.

Tot în cadrul dezvoltărilor viitoare se va avea în vedere crearea unei interfeţe standard a serverului în raport cu aplicaţiile client. Cunoscând acesta interfaţă, acestea vor fi capabile să se conecteze la server pentru a obţine atât transcrieri în timp real ale fluxurilor ce provin de la microfon sau alte surse, cât şi a fişiere audio ce vor fi încărcate pe server. Totodată, foarte importantă este paralelizarea aplicaţiei, fiind posibilă manipularea cererilor venite simultan de la mai mulţi clienţi.

Pentru sarcina de identificare a cuvintelor cheie este important de menţionat momentul de timp al apariţiei acestora în raport cu momentul de început al fluxului audio. Vor fi aduse modificări ale elementului Gstreamer *OnlineGmmDecodeFaster*, în vederea obţinerii acestei facilităţi.

CONCLUZII FINALE

Această lucrare a avut ca scop descrierea implementării unor aplicaţii software ce includ sisteme de recunoaştere a vorbirii şi detecţie a cuvintelor cheie. S-a dorit evaluarea acurateţii sistemelor şi a performanţelor acestora.

Capitolul 1 a reprezentat o introducere teoretică în domeniul tehnologiei vorbirii. Au prezentate detalii despre producerea şi percepţia semnalului vocal, extragerea parametrilor acustici, precum şi resursele necesare unui sistem de recunoaştere automată a vorbirii. Au fost sumarizate informaţii despre utilitarele folosite şi metricile de evaluare. De asemenea, au fost prezentate principalele metode de detecţie a cuvintelor cheie, precum şi metricile de evaluare ale unui astfel de sistem.

Capitolul 2 s-a axat pe construcţia unor sisteme de recunoaştere automată a vorbirii.

În prima secţiune a capitolului, s-a descris crearea unui sistem de recunoaştere a cifrelor în limba română. Au fost create modelele necesare şi s-au efectuat mai multe teste şi optimizări. S-a constatat faptul că un model acustic dependent de context este mult mai performant decât unul independent de context. De asemenea, sistemul independent de vorbitor este capabil să realizeze transcrieri corecte chiar şi în cazul vorbitorilor necunoscuţi. Prin optimizări succesive s-a constatat că modelul optim este cel cu 100 senone şi 128 densităţi gaussiene. S-au evaluat ratele de eroare la nivel de cuvânt pentru fiecare vorbitor în parte şi s-au constatat sursele ce pot provoca erori. Prin replicarea experimentelor folosind ambele utilitare de recunoaştere automată a vorbirii, a fost realizată o comparaţie directă între acestea. S-a dovedit faptul că utilitarul Kaldi oferă o acurateţe mai bună faţă de CMU Sphinx.

În a doua secţiune a capitolului s-a prezentat crearea unui sistem de recunaostere automată a vorbirii cu vocabular redus în limba engleză. Au fost create modelele necesare şi s-au efectuat teste de acurateţe, dar şi teste privind consumul resurselor. În cadrul celor din urmă, s-a constatat că un model de limba de tip gramatica este mai eficient decât un mod probabilistic de tip n-gram.

Capitolul 3 a prezentat crearea unui sistem de detecţie a cuvintelor cheie folosind un modul din utilitarul Kaldi. S-a efectuat o analiză asupra consumului de resurse cauzat în special de către modelul de limbă. S-a constatat că un model de limbă bazat pe reguli, ce conţine numai cuvintele cheie căutate este mult mai eficient decât un model n-gram. Apoi, baza de date de testare a fost etichetată, fiind marcate apariţiile cuvintelor cheie. S-a evaluat sistemul de detecţie obţinut. Din păcate, nu au fost duse la capăt toate scenariile de testare propuse. Acest lucru nu a fost posibil din cauza problemelor apărute la utilizarea aplicaţiilor de evaluare.

Capitolul 4 s-a concentrat pe implementarea software a serviciului de autentificare prin voce. A fost descrisă funcţionalitatea aplicaţiei şi arhitectura. Au fost oferite detalii de implementare a aplicaţiei client, dar şi a serviciului de autentificare şi a bazei de date. Aplicaţia finală şi-a atins scopul propus, fiind posibilă înrolarea şi autentificarea utilizatorilor pe bază de recunoaştere de vorbire şi vorbitor. S-au constatat însă unele limitări cauzate de serviciul de transcriere de cifre, fapt ce determină creşterea timpului de aşteptare la crearea contului. Au fost propuse şi analizate noi metode de implementare ale serviciului de transcriere a cifrelor. În continuare, acest aspect se caută a fi îmbunătăţit. Nu în ultimul rând, a fost descoperită o eventuală problemă de securitate la autentificare. Aceasta a fost descrisă pe larg în secţiunea corespunzătoare modurilor de utilizare ale aplicaţiei. Au fost propuse câteva metode de rezolvare a probemei.

Capitolul 5 a descris implementarea unei aplicaţii software de detecţie a cuvintelor cheie. Aceasta a fost realizată cu succes, îndeplinind şi funcţionalităţi de transcriere text a fluxurilor audio primite, precum şi de idenitificare a tipului de frază corespunzător. Totuşi, aplicaţia este una demonstrativă, ce nu oferă foarte multe opţiuni utilizatorului. Modulul server suporta un singur client la un anumit moment de timp. Modulul client este şi el unul limitat, permite numai citirea unui fişier audio de pe disc şi trimiterea lui la server, nefiind posibilă transcrierea unui flux captat de la microfon. Totuşi, aplicaţia este funcţională în limitele descrise şi poate fi un bun punct de plecare în cazul unor viitoare dezvoltări.

CONTRIBUŢII PERSONALE

Într-o mare măsură, lucrarea de faţă a reuşit să îndeplinească obiectivele propuse. Contribuţiile personale ale autorului ce au condus la atingerea obiectivelor pot fi sumarizate după cum urmează:

1. **Sistemele de recunoaştere automată a vorbirii**
   1.1 Sistemul Rodigits: recunoaşterea cifrelor în limba română
   - 1.1.1 Crearea modelelor (acustic, fonetic, lingvistic);
   - 1.2.2 Crearea sistemului dependent de vorbitor;
   - 1.2.3 Crearea sistemului independent de vorbitor;
   - 1.2.4 Evaluarea acurateţii şi optimizarea sistemului;
   - 1.2.5 Analiza rezultatelor, depistarea posibilelor surse de eroare;
   - 1.2.6 Replicarea proiectului în vederea realizării comparaţiei celor două utilitare.

   1.2 Sistemul Molina: recunoaşterea vorbirii în limba engleză cu vocabular redus
   - 1.2.1 Crearea bazei de date audio de evaluare;
   - 1.2.2 Crearea modelelor (acustic, fonetic, lingvistic);
   - 1.2.3 Evaluarea acurateţii sistemului;
   - 1.2.4 Evaluarea consumului de resurse la momentul decodării.
2. **Sistemul de detecţie a cuvintelor cheie**

   2.1 Analiza limitării generată de consumul resurselor în cazul modelului de limba 3-gram
   2.2 Testarea unui model de limbă alternativ
   2.3 Etichetarea manuală a cuvintelor cheie din baza de date de evaluare
   2.4 Obţinerea rezultatelor în urma detecţiei
   2.5 Evaluarea sistmeului de detecţie a cuvintelor cheie
3. **Serviciul web de autentificare prin voce**

   3.1 Implementarea aplicaţiei client
   3.2 Implementarea bazei de date
   3.3 Implementarea serviciului de autentificare
   3.4 Analiza limitării cauzată de timpul de execuţie al serviciului de recunoştere de cifre
4. **Aplicaţia de detecţie a cuvintelor cheie**

   4.1 Implementarea modulului server
   4.2 Implementarea modulului client

BIBLIOGRAFIE

[1] Horia Cucu, "Proiect de cercetare-dezvoltare în Tehnologia Vorbirii", Indrumar de proiect, Universitatea "Politehnica" Bucureşti.

[2] Keith Allan, "Linguistic Meaning", Routlege, 2014.

[3] Anca Popescu, Carmen Pătraşcu, Interfeţe Om-Maşină, Laborator 3, http://lpsv.pub.ro/images/IOM/CURS_-_IOM/IOM_3_AP.pdf

[4] The mel frequency scale and coefficients, http://kom.aau.dk/group/04gr742/pdf/MFCC_worksheet.pdf

[5] Jordan Critten, Parker Evans, Speaker recongition, ECE 576 FINAL PROJECT, https://courses.cit.cornell.edu/ece576/FinalProjects/f2008/pae26_jsc59/pae26_jsc59/

[6] Lindasalwa Muda, Mumtaj Begam and I. Elamvazuthi, "Voice Recognition Algorithms using Mel Frequency Cepstral Coefficient (MFCC) and Dynamic Time Warping (DTW) Techniques", JOURNAL OF COMPUTING, VOLUME 2, ISSUE 3, MARCH 2010, ISSN 2151-9617

[7] Roberto Togneri, Daniel Pullella, „An overview of speaker identification: Accuracy and Robustness Issues", IEEE Circuits and systems magazine, 2011

[8] Anca Popescu, Carmen Pătraşcu, Interfeţe Om-Maşină, Laborator 3,

http://lpsv.pub.ro/images/IOM/Lab_02_Analiza_de_timp_scurt_a_semnalului_vocal.pdf

[9] Andi Buzo, "Automatic speech recognition over mobile telecommunication channels", PhD Thesis, Universitatea "Politehnica" Bucureşti, Oct 2011.

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

[11], Mixture Model, http://www.en.wikipedia.org/Mixture_model

[12], CMU Sphinx architecture, http://www.di.uniba.it/~loglisci/FELIPE/

[13], Daniel Povey, "The Kaldi Speech Recognition Toolkit", Microsoft Research, 2011

[14], [14, Josh James, Data never sleeps 2.0, https://www.domo.com/blog/2014/04/data-never-sleeps-2-0/

[15], KWS 15 Keyword Search Evaluation Plan

https://www.nist.gov/sites/default/files/documents/itl/iad/mig/KWS15-evalplan-v05.pdf

[16] Horia Cucu, "Proiect de cercetare-dezvoltare în Tehnologia Vorbirii", îndrumar de proiect, Universitatea "Politehnica" Bucureşti.

[17] Molina Healthcare IVR system,

http://www.molinahealthcare.com/providers/wi/medicaid/forms/PDF/forms_WI_10_IVR_reference_guide.pdf

[18], Anthony Rousseau, Paul Deléglise, Yannick Estève, "TED-LIUM: an Automatic Speech Recognition dedicated corpus", Laboratoire Informatique de l'Université du Mâine (LIUM), 2012

[19], Ted Conference, https://www.ted.com/

[20] Alina Bănică,"Serviciu web de autentificare prin voce. Sisteme de recunoaştere de cifre conectate şi verificare de vorbitor", Proiect de Licenţă, Universitatea "Politehnica" Bucureşti, 2015

[22] Boris Smus, "Web Audio API", O'Reilly, 2013

[23] RecorderJS, https://github.com/mattdiamond/Recorderjs