Universitatea “Politehnica” din Bucureşti
Facultatea de Electronică, Telecomunicaţii şi Tehnologia Informaţiei

Recunoaştere vocală minimală pe sisteme embedded folosind inteligenţa artificială

Proiect de diplomă

prezentat ca cerinţă parţială pentru obţinerea titlului de
Inginer în domeniul *Electronică și Telecomunicații și Tehnologia Informației*
programul de studii de licenţă *Electronică Aplicată*

Conducător ştiinţific
S.L. Dr. Ing. Horia CUCU

Absolvent
Valentin PORUMBEL

2016

Cuprins

Lista figurilor

Lista tabelelor

Lista acronimelor

HMM - Hidden Markov Model – *modelul Markov cu stări ascunse*

EM - Expectation - Maximizationalgorithm

SS - Substracție Spectrală

PMC - parallel model combination

SS - spectral substraction

FFT - fast Fourier Transform

MFCC - mel frequency cepstral coefficients

DCT - discrete cosine transformation

RASTA - RelAtive Spectral

PLP - perceptual

IIR - infinite impulse response

GMM - Gaussian Mixture Model

DTW - dynamic time warping

WER - word error rate

SWER - single word error rate

CSR - command success rat

Introducere

Recunoașterea vocală de către un sistem electronic aduce în prim plan o veche problemă în domeniul tehnic, și anume limita autonomiei sistemelor pe care omul le poate construi. Interacțiunea dintre un om și un astfel de sistem este inevitabilă, pornind de la exemplul unui banal robinet prin care un operator, prin interacțiune fizică poate controla debitul unei pompe, soluțiile de recunoaștere vocală extind această funcționalitate. Astfel, sistemul descris anterior poate fi extins prin adăugarea unui controller care, bazându-se pe comenzile vocale primite de la un operator uman să realizeze funcția necesară, înlăturând astfel necesitatea unei interacțiuni fizice, aceasta lăsând loc unei interacțiuni acustice, realizabilă de la distanță.

Funcționalitatea sistemelor de recunoaștere vocală nu se oprește la trasmiterea de comenzi către un sistem, aceasta își dovedește utilitatea într-o gamă largă de aplicații, cum ar fi convertirea semnalului acustic în text sau sisteme de securitate bazate pe amprenta vocală a unui utilizator.

Algoritmii de inteligență artificială fac posibilă implementarea unor astfel de soluții, iar dezvoltarea sistemelor embedded extinde portabilitatea aplicațiilor care fac uz de recunoașterea vocală, oferind o soluție în același timp ieftină și fiabilă pentru acest gen de aplicații. Cu toate acestea, sistemele de tip embedded introduce o constrângere legată de capabilitatea acestora de a rula astfel de algoritmi, ceea ce impune elaborarea unor optimizări care să facă posibilă funcționarea satisfăcătoare.

Dificultatea care apare în recunoașterea vocală de către mașină este reprezentată de natura interdisciplinară a acestui domeniu. În continuare, vor fi enunțate pe scurt o parte dintre disciplinele pe care recunoașterea vocală le înglobează[1]:

- teoria semnalelor, furnizează modelul de analiză spectrală a semnalului vorbit, achiziționat de sistem și din care va trebui mai apoi extasă o serie de parametrii necesari algoritmului de recunoaștere a vorbirii;
- acustica, știința care explicitează relația dintre semnalul fizic și mecanismele fiziologice(tractul vocal al omului) care a produs vorbirea și de asemenea aparatul cu care acesta este perceput de om(mecanismul auditiv al omului);
- recunoașterea de caracteristici(pattern recognition/machine learning), algoritmii care fac posibilă încadrarea unor date achiziționate în modele ale căror parametrii sunt recunoscuți de mașină
- teoria informației, caracterizează procedurile de estimare a parametrilor modelelor statistice, și de "căutare" a unui anume cod într-un set finit de date, care corespunde cât mai bine informației recunoscute de sistem;

- lingvistica, relaționarea între sunete, cuvintele unei limbi(sintaxa), înțelesul cuvintelor (semantica), și sensul care derivă din înțeles.
- fiziologia, înțelegerea mecanismelor complexe facând parte din sistemul nervos central al omului, mecanisme care au rol în producerea vorbirii de către om și de asemenea în perceperea acesteia. Sistemele avansate de recunoaștere automată a vorbirii folosesc tehnici inspirate din această știință prin integrarea în sistemul de recunoaștere a unor rețele neuronale.
- informatica, în spatele logicii care controlează recunoașterea vorbirii stă cercetarea în domeniul algoritmicii și, de asemenea, avansul tehnologiilor hardware
- psihologia, se impune înțelegerea factorilor care ar permite utilizatorilor să integreze un sistem de acest tip în domeniul lor de activitate.

Lucrarea de față se adresează realizării unui sistem de recunoaștere vocală de dimensiuni reduse(vocabular limitat, sub 100 de cuvinte), care poate recunoaște comenzi ale utilizatorului, în urma cărora să execute o rutină specificată. Se va folosi o platformă embedded care limitează optiunile de implementare a sistemului prin constrângeri de tip hardware(putere de calcul).

Se va structura lucrarea în două părți, baza teoretică a modulului în care se vor prezenta blocurile care alcătuiesc un sistem de recunoaștere vocală, cu o scurtă descriere a modelelor pe baza cărora acestea funcționează și o parte în care se va detalia procesul de implementare a modelului adecvat aplicației curente.

Capitolul 1, “Noțiuni matematice legate de sistemele de recunoaștere vocală” va prezenta pe scurt aspectele matematice ale acestor sisteme, se va prezenta HMM, și algoritmii specifici de modelare ai acestuia

Capitolul 3 se adresează arhitecturii unui sistem de recunoaștere vocală, acompaniat de o scurtă descriere a blocurilor componente care vor fi tratate în detaliu în capitolele ulterioare

Capitolul 2 va face o scurtă trecere în revistă a algoritmilor tipici utilizați în recunoașterea vorbirii de către mașină, filtrare de zgomot(inverseaza cu arhitectura), extragerea parametrilor

Capitolul 4 va prezent resursele hardware și software utilizate în sistemul de recunoaștere vocală

Capitolul 1
Noțiuni matematice legate de sistemele de recunoaștere vocală

1.1 Modele Markov

Modelul Markov caracterizează un sistem probabilistic prin intermediul a două variabile aleatoare: starea sistemului și timpul de observare. Dacă se ia în considerare faptul că fiecare dintre aceste variabile aleatoare poate fi continuă sau discretă rezultă o mulțime de patru modele care se pot obține din combinațiile posibile ale celor două variabile aleatoare. Modelul Markov cu stări discrete și timp de observare discret se mai numește si Markov, analog, în cazul în care sistemul are starea discretă și timpul de observare continuu, se obține un proces Markov.

Orice model Markov este caracterizat de faptul că, dacă sistemul se află într-o anumită stare la momentul t_0, atunci probabilitatea ca acesta să se afle într-o anumită stare la momentul $t > t_0$, depinde numai de starea curentă(la momentul t_0), și nu depinde de modul în care sistemul a ajuns la starea cu pricina.

1.1.1 Procese Markov

Caracteristica principală a unui proces Markov este că apariția unei anumite stări este condiționată de un număr determinat de stări anterioare. Dacă numărul acestor stări este *n* atunci spunem că avem de a face cu un proces Markov de ordinul *n*.

De exemplu, fie un sistem probabilistic cu stări discrete și timp de observare continuu. Dacă presupunem că sistemul se află la momentele de timp $t_0, t_1, ..., t_k$ în stările $N(t_0), N(t_1), N(t_2), ..., N(t_k)$ și dacă el se comportă ca un sistem Markov de ordinul 1, atunci probabilitatea ca la momentul următor, t_{j+1} starea lui să fie $N(t_{j+1})$ este determinată doar de tarea lui din momentul anterior, și anume:

$$P\{N(t_{j+1}) = k \mid N(t_0) = n_0, N(t_1) = n_1, \ldots, N(t_j) = n\} = P\{N(t_{j+1}) = k \mid N(t_j) = n\} = p_k(t)$$

Astfel, putem spune că starea viitoare k depinde doar de starea prezentă n sau starea prezentă înglobează toată informația pe care o avem despre sistem la un anumit moment de timp. Se poate spune că procesele Markov sunt procese fără memorie, sau că viitorul este independent de trecut, dat fiind prezentul.

Putem înțelege intuitiv această proprietate a sistemelor Markov gândindu-ne la mișcarea Browniană, agitația termică a particulelor constituente ale sistemului generează o mișcare aleatoare a particulelor de polen(sau cenușă), traiectoria acestora fiind generată exclusiv de coliziunile care au loc la un anumit moment de timp.

1.1.2. Lanțuri Markov

Pe de altă parte, dacă pornim de la presupunerea existenței unui timp discret caracterizat de stări discrete ale sistemului(stări caracterizate de variabile aleatoare), avem un lanț Markov. Exemplul cel mai simplu care explicitează proprietățile unui lanț Markov este cel al unei variabile care poate la fiecare moment de timp o valoarea {-1} cu probabilitatea p, sau valoarea {1} cu probabilitatea 1 – p. Reprezentând grafic evoluția acestei variabile în timp, se obține figura 1.1.

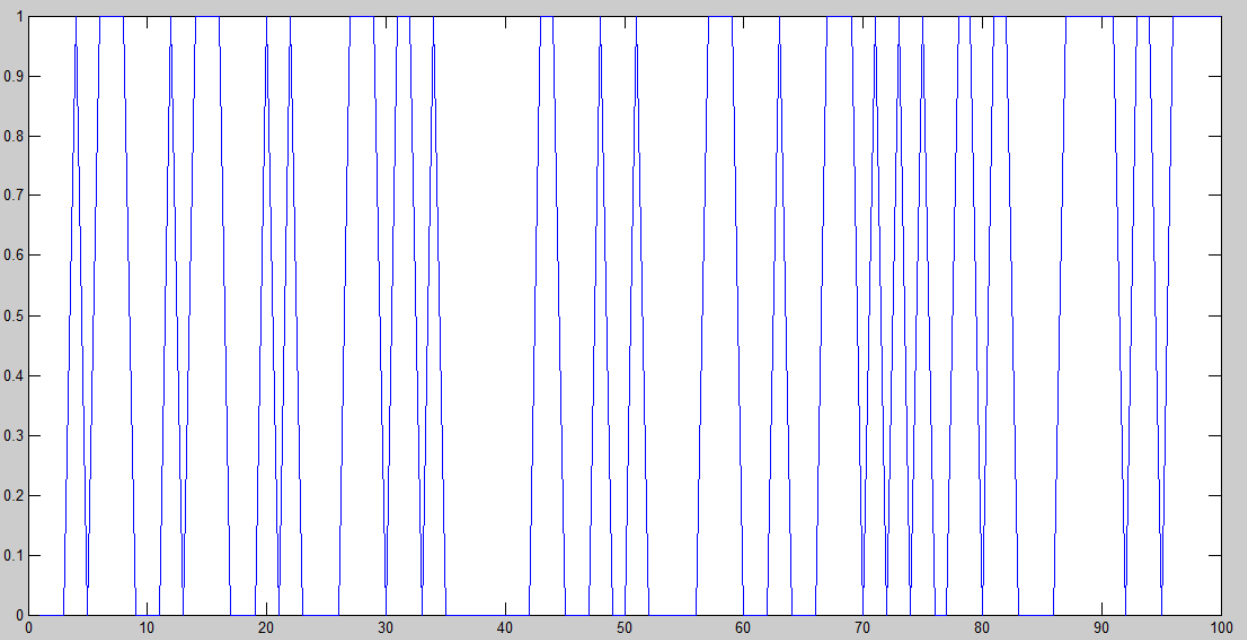

Figura 1.1. Reprezentarea valorilor unei variabile aleatoare modelată de un lanț Markov în Matlab

Pentru lanțurile Markov, procesul stochastic caracteristic sistemului poate fi descries prin secvența de stări pe care o parcurge sistemul. Pentru lanțurile Markov sunt caracterizate probabilitățile de tranziție dintr-o stare în alta, unde p_{ij} sunt probabilitățile ca sistemul să treacă din starea i în starea j. Dacă aceste probabilități nu depind de numărul de pași efectuați anterior și depind numai de starea inițială și starea finală, atunci lanțul se numește omogen.

1.2. Modelul Markov cu stări ascunse

De interes pentru sistemele de recunoaștere vocală este Modelul Markov Ascuns(HMM). În cadrul modelelor Markov mai simple din punct de vedere matematic, se presupune că starea sistemului este direct vizibilă observatorului(acesta deține informația completă asupra sistemului și astfel, probabilitățile de tranziție între stări reprezintă singurii parametrii ai modelului. Când vorbim despre HMM, nu avem informații despre stare, dar rezultatul proceselor(dependent de stare), este cunoscut.

Un lanț Markov este util în eventualitate în care se cere calculul probabilității asociat unui șir de evenimente care poate fi observat. În cazul recunoașterii vocale avem de a face cu evenimente acustice, acestea implică o prezență a unor cuvinte transparente fenomenului fizic, al căror sens reprezintă partea "ascunsă" ce stă la baza fenomenului acustic. Modelul Markov cu stări ascunse ne permite să vorbim în același timp despre fenomene observate(cum ar datele achiziționate de microfon) și evenimente ascunse(părțile de vorbire) pe care le considerăm ca fiind părți cauzale ale modelului probabilistic.

Pentru a da o definiție formală modelului Markov cu stări ascunse, trebuie să luăm în considerare următorii parametri[2]:

$Q = q_1 q_2 q_3 .. q_n$, mulțimea stărilor

$A = a_{11} a_{12} ... a_{n1} ... a_{nn}$, matricea de probabilistică de tranziie, unde a_{ij} reprezintă probabilitatea de tranziție din starea *i* în starea *j*

$O = o_1 o_2 ... o_T$, un șir de observații, fiecare provenită dintr-un vocabular $V = v_1, v_2, ..., v_V$

$B = b_i(o)$, un șir de estimare de observații, denumit și probabilitate de emisie, fiecare element reprezentând probabilitatea ca o observație o_t să fie generată pornind de la o stare *i*

q_0, q_F, stările de început și de final care nu sunt associate observațiilor

Analog unui lanț Markov de ordinul I, probabilitatea unei stări anume depinde numai de starea anterioară, numită și presupunerea Markov:

$$P(q_i \mid q_1 ... q_{i-1}) = P(q_i \mid q_{i-1})$$

Pe de altă parte, probabilitatea unei observații o_i depinde doare de starea care a produs observația q_i și nu de oricare altă stare sau de oricare altă observație, proprietate numită și independența rezultatului:

$$P(o_i \mid q_1 ... q_i, \ ... \ q_T, o_1, \ ..., \ o_i, \ ..., \ o_T) = P(o_i \mid q_i)$$

Se pune problema aplicațiilor care folosesc modele Markov cu stări ascunse și cum acesteea pot utiliza modelul matematic bazat pe acestea. *Rabiner(1989)*[3] propune analiza modelelor Markov din trei puncte de vedere, numite de acesta și "trei probleme fundamentale", acestea fiind:

Problema 1, sau probabilitatea: Dat fiind un HMM $\lambda = (A, B)$ și un șir de observații O, să se determine probabilitatea lui $P(O \mid \lambda)$.

Problema 2, sau decodarea stărilor: Dat fiind un șir de observații O, și un HMM $\lambda = (A, B)$, să se obțină mulțimea optimă de stări ascunse, Q.

Problema 3, sau antrenarea: Fie un șir de observații, O, și o mulțime de stări posibile din HMM, să se deducă parametrii A și B.

Fiecare dintre aceste 3 abordări ale HMM sunt descrise de câte un algoritm specific, și anume, în ordine: algoritmul Forward, decodarea Viterbi, și algoritmul Backward-Forward.

1.2.1 Algorimul Forward

Scopul algoritmului Forward este de a calcula probabilitatea $p(x_t, y_{1:t})$, unde $x(t)$ a fost notat x_t, iar mulțimea $(y(1), y(2), ..., y(t))$ s-a notat ca $y_{1:t}$. Calcularea $p(x_t, y_{1:t})$ în mod direct ar presupune "marginalizarea" asupra tuturor mulțimilor de stări $\{x_{1:t-1}\}$, numărul cărora crește exponențial cu evoluția lui t. Pentru un HMM cu N stări ascunse, și un șir de T observații, există N^T șiruri posibile de stări ascunse. Dar fiind că pentru sisteme reale, ambii parametrii sunt reprezentați de o valoare considerabilă, probabilitatea $p(x_t, y_{1:t})$ presupune calcularea probabilității fiecărei stări ascunse a șirului și apoi sumarea acestora. În locul acestui tip de algoritm cu o complexitate puternic exponențială se alege folosirea algoritmului Forward, a cărui complexitate este de ordinul $O(N^2T)$. În pseudocod, algoritmul poate fi reprezentat în felul următor:

```
construiește matricea de probabilitate forward[N + 2, T]
pentru fiecare stare q de la 1 la N
     forward[s,1] = a0,s * bs(o1)
pentru fiecare pas de timp t de la 2 la T
     pentru fiecare stare s' de la 1 la N
          forward[s,t] = Σ de la s' = 1 la N
                         forward[s', t - 1]* as',s * bs(ot)
forward[qF, T] = Σ de la s = 1 la N forward[s, T]as, qF
```

Prin urmare, algoritmul Forward face uz de avantajele programării dinamice bazându-se pe una dintre proprietățile de bază ale HMM, și anume independența observațiilor pentru a realiza calculul în mod recursiv.

Astfel, fie:

$$\alpha_t(x_t) = p(x_t, y_{1:t}) = \Sigma\, p(x_t, x_{t-1}, y_{1:t})$$

Dezvoltând ecuația, se poate scrie:

$$\alpha_t(x_t) = \Sigma\, p(y_t \mid x_t, x_{t-1}, y_{1:t})\, p(x_t \mid x_{t-1}, y_{1:t-1})\, p(x_{t-1}, y_{1:t})$$

Dat fiind că y_t este independent de orice altă variabilă înafară de x_t, și x_t este independent de orice alt parametru înafară de x_{t-1}, ecuația devine:

$$\alpha_t(x_t) = p(y_t \mid x_t)\, \Sigma\, p(x_t \mid x_{t-1})\, \alpha_{t-1}(x_{t-1})$$

Prin urmare, dat fiind că $p(y_t \mid x_t)$ și $p(x_t \mid x_{t-1})$ sunt cunoscute prin parametrii specifici modelului, și anume probabilitatea de emisie și de probabilitățile de emisie, $\alpha_t(x_t)$ se poate calcula cu ușurință, evitându-se astfel dificultățile apărute din cauza complexității exponențiale de calcul.

1.2.2. Decodarea Viterbi

Pentru orice model care conține variabile ascunse, cum ar fi HMM, sarcina deteminării șirului care stă la baza șirului de variabile observabile se numește *decodare*. Algoritmul Viterbi reprezintă un exemplu de programare dinamică, al cărei scop este găsirea șirului optim de stări ascunse, numit și calea Viterbi, care rezultă din secvența unor evenimente observate, în mod special în contextul HMM. Reprezentând problema în mod formal, se obține următorul enunț:

Pentru șirul de observații generat de $x_1, ..., x_n$, să se găsească

$$\arg\max p(x_1 \ldots x_n, y_1, \ldots, y_n),$$

unde *arg max* se consideră pe tot șirul $y_1, \ldots, y_{n+1}$ astfel încât y_{n+1} = STOP, reprezentând momentul final de timp. Pentru rezolvarea acestei probleme, abordarea de tipul "brute force se dovedește ineficientă.

Astfel, se impune utilizarea algoritmului Viterbi. Acest algoritm seamănă foarte mult cu algoritmul Forward, în schimb folosește maximul căilor descoperite în timp ce algoritmul Forward utilizează suma acestora pentru a-și calcula rezultatul. În schimb se folosesc "backpointers" deoarece, în timp ce Forward trebuie să gasescă o probabilitate, Viterbi calculează o probabilitate și o cale probabilă. Șirul optim de stări se calculează ținând cont de calea stărilor ascunse care au condus către fiecare stare în parte. În cele din urmă, se obține calea dorită prin procedeul de "backtracing".

Algoritmul poate fi exprimat sub următoarea formă, în pseudocod:

```
creează matricea de probabilitate a căii viterbi[N + 2, T]
pentru fiecare stare de la 1 la N
     viterbi[s, 1] = a0, s * bs(o1)
     backpointer[s, 1] = 0
pentru fiecare pas de timp t de la 2 la T
     pentru fiecare stare s de la 1 la N
          viterbi[s, t] = max (s' = 1 la N) din
                          viterbi[s', t - 1] * as', s * bs(ot)
          backpointer[s, t] = arg max(s' = 1 la N) din
                              viterbi[s', t - 1] * as', s
viterbi[qF, T] = max viterbi[s, T] * as, qF
backpinter[qF, T] = arg max Viterbi[s, T] * as, qF
returnează "backtrace-ul" prin urmărirea backpointer-ului către
stările anterioare, începând cu backpointer[qF, T]
```

Prin urmare, la un moment de timp *t*, dar fiind că s-a calculat probabilitatea de a fi în fiecare stare la momentul *t-1*, probabilitatea Viterbi se calculează luând cea mai probabila extensie a stărilor care conduc către "celula" curentă. Pentru o stare dată, q_j la momentul t, valoarea lui $v_t(j)$ se calculează ca fiind:

$$v_t(j) = max\ v_{t-1}(i)\ a_{ij}\ b_j(o_t)$$

Cei trei factori prin intermediul cărora se calculează probabilitatea Viterbi la un moment *t*, sunt:

$v_{t-1}(i)$, - probabilitatea Viterbi de la momentul anterior
a_{ij}, - probabilitatea de tranziție din starea q_i în starea q_j
$b_j(o_t)$ - probabilitatea de observare a stării dată de simbolul o_t, dată fiind starea curentă *j*

1.2.3. Algoritmul Forward-Backward(Algoritmul Baum – Welch)

Algoritmul Baum-Welch se consideră a fi o aplicație a algoritmului EM care se pretează modelului Markov cu stări ascunse, devenind astfel și algoritmul standard pentru antrenarea HMM(Baum, 1972), acesta permițând în același timp antrenarea probabilităților de tranziție și de emisie ale modelului.

Modelul Markov cu stări ascunse descrie probabilitatea de conjuncție a colecției de variabile aleatoare discrete "ascunse" și observate. Acesta, după cum s-a enunțat anterior, se bazează pe prezumția că.a *n*-a variabilă "ascunsă", dată fiind cea de a *n-1* -a variabilă "ascunsă", este independentă de variabilele anterioare, și că variabilele care descriu observația curentă depinde numai și numai de starea "ascunsă" curentă.

Algoritmul Baum-Welch se folosește de principiul algoritmului EM pentru a estima verosimilitatea maximă a parametrilor pentru un HMM, dat fiind un set de parametrii observați. Pentru a înțelege intuitiv această problemă, putem aborda cazul unui lanț Markov, care poate fi văzut ca un HMM degenerat pentru care parametrul *b* este 1 pentru toate simbolurile observate și 0 pentru toate celelalte. Astfel, trebuie antrenată numai matricea probabilităților de tranzitie, A.

Astfel, se poate estima verosimilitatea maximă a probabilității a_{ij} pentru o tranziție anume între stările *i* și *j* prin numărarea cazurilor în care aceasta s-a desfășurat, apoi împărțirea acestui număr la numărul total de tranziții din *i*:

$$a_{ij} = \frac{N(i,j)}{\Sigma N(i,q)}$$

Acest calcul este posibil pentru un lanț Markov deoarece deținem informații complete asupra tuturor parametrilor sistemului. În cazul HMM, informația pe care o cunoaștem nu mai este completă, prin urmare nu putem ști calea pe care a sistemul a urmat-o ca urmare a datelor de intrare. Algoritmul Baum-Welch abordează această problemă folosindu-se de două euristici. În primul rând, numărul de tranziții este estimat în mod iterativ, se va porni de la o estimare inițială a probabilităților de tranziție și de emisie. In al doilea rând, estimarea probabilităților se obține

prin calcularea probabilității Forward(procedeu descris anterior) pentru o anumită observație iar apoi prin medierea acestei probabilități pe toate căile care au contribuit calculului probabilității Forward cu pricina.

De asemenea, acest algoritm definește și o probabilitate Backward(de unde vine și numele algoritmului, Forward-Backward). Probabilitatea Backward *(β)* este probabilitatea de a face observații de la momentul $t + 1$, presupunând că sistemul se află în starea *i* la momentul *t*.

$$\beta_t(i) = P(o_{t+1}, o_{t+2}, ..., o_T \mid q_t = i, \lambda)$$

Calculul acestei probabilități se face asemănător cu cel al probabilității Forward. Pașii pe care îi urmează algoritmul Backward sunt următorii:

1. Inițializarea

$$\beta_T(i) = a_{iF}, 1 \leq i \leq N$$

2. Recursivitatea

$$\beta_T(i) = \Sigma\, a_{ij}\, b_j\, (o_{t+1})\, \beta_{t+1}(j), 1 \leq i \leq N, 1 \leq t < T$$

3. Finalizarea

$$P(O|\, \lambda) = \alpha_T(q_F) = \beta_1(q_0) = \Sigma\, a_{0j}\, b_j(o_1)\, \beta_1(j)$$

În continuare, trebuie estimată probabilitatea de tranziție, de forma

$$a_{ij} = \frac{\text{număr estimat de tranziții din } i \text{ în } j}{\text{număr estimat de tranziții din } i}$$

Pentru a calcula aceast termen, trebuie definită o probabilitate ca sistemul să se afle în starea *i* la momentul de timp *t* și în starea *j* la momentul de timp *t+1*, dată fiind secvența de observații, și evident, modelul.

$$\xi_t(i, j) = \frac{\alpha_t(t) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\alpha_T(q_F)}$$

De asemenea, este nevoie de o formulă de calcul pentru probabilitatea de observare, această probabilitate fiind, de fapt, probabilitatea unui anume simbol v_k din vocabularul *V*, dată fiind o stare *j*:

$$b_j(v_k) = \frac{\text{estimarea numărului de momente în care ne aflăm în starea } j \text{ și observăm simbolul } v_k}{\text{estimarea numărului de momente în care ne aflăm în starea } j}$$

În acest scop, se definește probabilitatea de a ne afla în starea *j* la momentul *t*, pe care o vom numi $\gamma_t(j)$:

$$\gamma_t(j) = P(q_t = j \mid O, \lambda)$$

Într-un final, se poate calcula *b*:

$$b_j(v_k) = \frac{\sum_{t=1 s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Unde notația $\sum_{t=1 s.t. O_t = v_k}^{T}$ reprezintă "sumă după toate momentele de timp t pentru care simbolul observat a fost v_k.

Dați fiind acești termeni, se pot reestima probabilitățile de tranziție A și probabilitățile de emisie B dintr-un șir de observații O, dată fiind o aproximare anterioară a lui A și B. Aceste reestimări stau la baza algoritmului Forward-Backward.

Amintit fiind mai devreme algoritmul EM, algoritmul Baum-Welch pornește de la o aproximare inițială a parametrilor HMM λ = (A, B), iar folosind pașii de tip *expectation*(E-step) și *maximization*(M-step) se calculează ξ și γ, respectiv în pasul M se folosesc cei doi parametrii calculați anterior pentru a recalcula probabilitățile A și B. Desigur, algoritmul poate fi reprezentat în pseudocod astfel:

```
inițializează A și B (aproximarea inițială)
itrează până la convergență
     expectation
          γt(j) = αt(i)βt(j) / αT(qF)

          ξt(i, j) = αt(t)aijbj(ot+1)βt+1(j) / αT(qF)
     maximization
          aij = Σ_{t=1}^{T-1} ξt(i,j) / Σ_{t=1}^{T-1} Σ_{k=1}^{N} ξt(i,j)
          bj(vk) = Σ_{t=1 s.t. Ot=vk}^{T-1} γt(j) / Σ_{t=1}^{T-1} γk(j)
returnează A, B
```

Deși principiul acestui algoritm este de a antrena parametrii A și B fără supraveghere, în practică trebuie acordată o importanță deosebită condițiilor inițiale.

Capitolul 2

Algoritmi utilizați în recunoașterea vocală de către un sistem electronic

2.1 Filtrare de zgomot

2.1.1 Substracția Spectrală (SS)

Recunoașterea vocală de către un sistem electronic presupune, printre altele, achiziția unui semnal acustic, în care este codată informația referitoare la cuvintele care se doresc a fi transmise. Extragerea parametrilor semnalului sonor presupune de asemenea algoritmi de antrenare al căror rezultat poate fi puternic afectat de distorsiuni apărute în proba analizată de sistem. Prin urmare se impune ca aceste eșantioane de vorbire să fie colectate într-un mediu lipsit de zgomot sau de alți factori perturbatori care eventual ar putea declanșa o eroare în ceea ce privește prelucrarea datelor. Din păcate, situațiile reale în care un astfel de sistem de recunoaștere vocală ar putea funcționa nu se pretează ipotezei enunțate anterior, ceea ce ne conduce la ideea că trebuie aplicată o metodă de atenuare sau chiar eliminare completă a părții din semnal care ne incomodează.

Una dintre metodele propuse în literatură este PMC(parallel model combination), aceasta folosește o aproximare *log normală*, pentru a transforma statisticile cepstrale în statistici liniar specrale astfel încât parametrii acustici să poată fi extrași cu ușurință din semnalul curățat[4], adaptând astfel semnalul la orice fel de zgomot. Cu toate acestea, PMC impune ca zgomotul de fundal să fie antrenat dacă acesta s-a schimbat(nu corespunde cu ceea ce sistemul cunoaște). Concluzia fiind că ar fi de preferat o metodă care să reducă componenta zgomotoasă a semnalului acustic.

Substracția Spectrală (SS) este una dintre metodele propuse în mod convențional pentru a contrabalansa dezavantajele PMC[5]. Desigur, și această metodă prezintă un dezavantaj, și anume faptul că reduce performanțele de recunoaștere ale sistemului din cauza distorsionării spectrului din cauza calibrării necorespunzătoare necesității semnalului.

Substracția Spectrală se bazează pe eliminarea unui spectru mediat din semnalul analizat, și anume partea care se consideră a fi cea zgomotoasă[6]. Spectrul zgomotului este de obicei preluat în momentele în care semnalul este absent, presupunându-se în același timp că zgomotul reprezintă un proces cvasistaționar, astfel încât spectrul acestuia să nu se schimbe semnificativ între timpii de eșantionare. Datorită estimărilor care se fac asupra spectrului instantaneu de putere, metoda introduce probleme în ceea ce privește distorsionarea semnalului reconstruit, fapt care devine o liabilitate o dată cu scăderea raportului semnal-zgomot. SS însă se dovedește a fi atractivă în ceea ce privește complexitatea de calcul.

In aplicațiile precum recunoșterea vocală, singurul semnal disponibil este cel zgomotos(spre deosebire de aplicații unde pe lângă semnalul care trebuie prelucrat mai este disponibil și un canal pe care se recepționează exclusiv zgomotul ambiental, caz în care se simplifică mult sarcina de filtrare, fiind posibilă compararea directă a semnalelor).

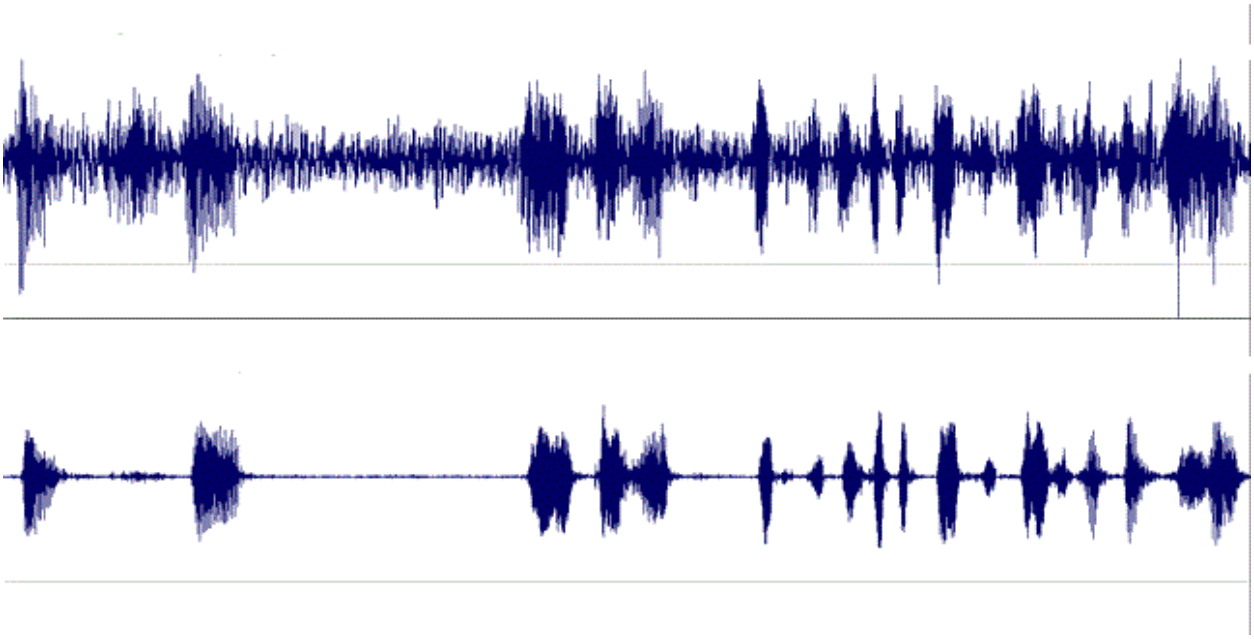

Figura 2.1:Sus, semnal achiziționat de microfon, jos, semnalul filtrat

Semnalul zgomotos poate fi exprimat în domeniul timp în următorul mod:

$$y(m) = x(m) + n(m), \text{ sau}$$
$$Y(f) = X(f) + N(f)$$

Prin SS, semnalul este descompus segmente de N eșantioane ferestruite(de obicei cu o fereastră Hann sau Hamming) iar apoi transformate prin intermediul unei transformări Fourier discrete în benzi spectrale(fereistruirea se impune pentru menținerea fidelității semnalului la transformarea în domeniul frecvență, i.e. se evită perioade incomplete ale semnalului care ar distorsiona spectrul prin introducerea unor componente de frecvențe mult mai mari decât frecvența Nyquist).

Astfel, semnalul asupra căruia se aplică fereastra are forma:

$$y_w(m) = w(m)y(m) = w(m)[x(m) + n(m)] = x_w(m) + n_w(m)$$

sau în domeniul frecvență

$$Y_w(f) = W(f) * Y(f) = X_w(f) + N_w(f)$$

În cele din urmă, ecuația care caracterizează SS se poate exprima în felul următor:

$$\left|\hat{X}(f)\right|^b = |Y(f)|^b - \alpha\overline{|N(f)|^b}$$

unde $\left|\hat{X}(f)\right|^b$ reprezintă estimarea care se face asupra semnalului initial, iar $\overline{|N(f)|^b}$ se referă la media spectrală a zgomotului.

Ultimul termen al ecuației se obține din eșantioanele în care semnalul sonor este absent și numai semnalul zgomotos este preluat de sistem.

$$\overline{|N(f)|^b} = \frac{1}{K}\sum_{i=0}^{K-1}|N_i(f)|^b$$

unde $|N_i(f)|$ reprezintă spectrul cadrului i, presupunând că există un număr de N cadre într-un interval de timp după care se face eșantionarea zgomotului. Iar pentru reconstruirea semnalului în domeniul timp, se aplică o transformată Fourier discretă semnalului $\left|\hat{X}(f)\right|^b$ ținând cont de faza semnalului zgomotos [θ_Y(k)]. Acceptând prezumția că zgomotul perceput de ureche se datorează în principal distorsiunilor apărute în spectrul de amplitudini, și că efectele distorsiunii spectrului de faze nu influențează radical procesul de percepție, se poate scrie ecuație semnalului rezultat în domeniul timp:

$$\hat{x} = \sum_{k=0}^{N-1}\left|\hat{X}(k)\right|e^{j\theta_Y(k)}e^{-j\frac{2\pi}{N}km}$$

Datorită variațiilor în spectrul zgomotului, SS poate da rezultate negative pentru spectrul de puteri sau cel de amplitudini, probabilitatea acestui eveniment este cu atât mai mare cu cât raportul semnal-zgomot scade. Astfel, se impune o post-procesare asupra rezultatelor folosind o funcție de tipul(unde *fn[Y(f)]* este o funcție care în cea mai simplă formă a sa, poate reprezenta chiar pragul de zgomot)

$$T\left[\left|\hat{X}(f)\right|\right] = \begin{cases}\left|\hat{X}(f)\right|, & \left|\hat{X}(f)\right| > \beta|Y(f)| \\ fn[|Y(f)|], & in\ rest\end{cases}$$

2.1.2 Implementarea Substracției spectrale

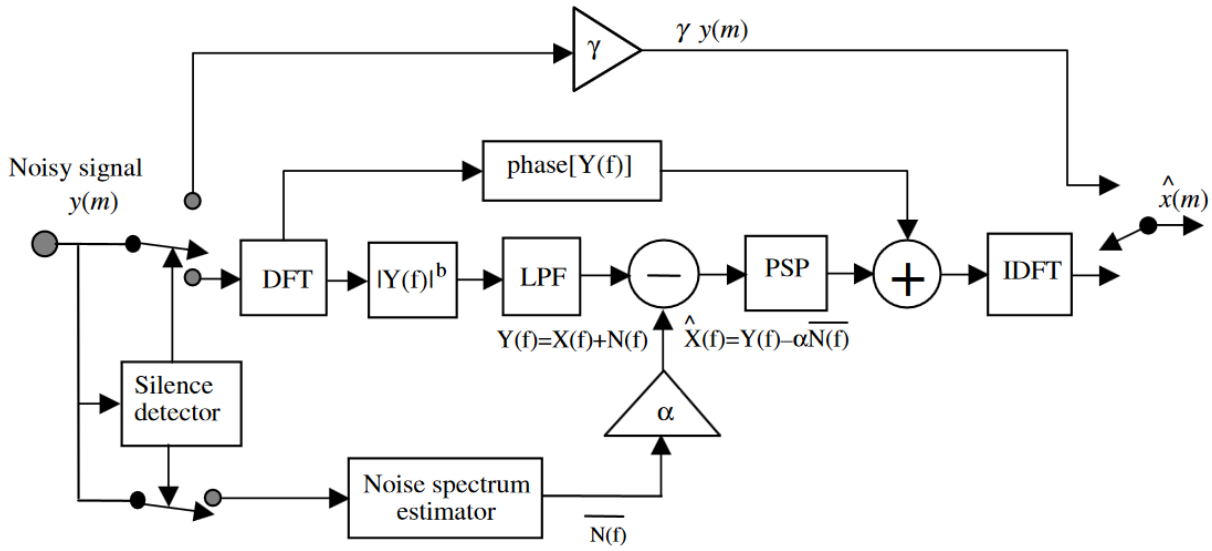

Figura 2.2: Schema bloc a algoritmului

Pentru a putea implementa funcționalitatea acestui algoritm în sistemul nostru, avem nevoie de următoarele elemente

(a) Un detector de liniște pentru perioadele de inactivitate ale semnalului(eșantionul pentru care se face medierea spectrala a zgomotului este actualizat în această perioadă
(b) Transformare Fourier Discretă pentru transformarea în domeniul frecvență, și scalare
(c) Un FTJ pentru reducerea variației zgomotului, scopul său fiind reducere distorsiunilor apărute în procesarea semnalului datorită variațiilor în semna
(d) Un bloc de post procesare care implementează funcția *T[]* definită anterior
(e) Transformarea Fourier Inversă pentru întoarcerea în domeniul timp
(f) Un atenuator pentru atenuarea zgomotului în momentele de "liniște"

Semnalul achiziționat de microfon este trecut printr-un buffer și împărțit în mulțimi de N eșantioane pentru a fi procesat prin DFT, fiecare bloc fiind ferestruit printr-o fereastră Hanning. După aplicarea SS, spectrul de amplitudini este combinat cu faza semnalului de zgomot și transformat înapoi în domeniul timp.

Alegerea duratei de timp pe care se face analiza spectrala devine un compromis între rezoluția în timp a semnalului și rezoluia spectrală. În acest scop, se obișnuiește folosirea unui bloc de lungime 5-50 de milisecunde, iar dată fiind o frecvență de eșantionare de 20 kHz, acest lucru se traduce într-o valoare a lui N între 100 și 1000 de eșantioane. Dat fiind că rezoluția spectrului dat de DFT este direct proporțional cu N, o valoare mai mare a acestuia rezultă într-o estimare mai bună. Enunțul anterior este valabil pentru componente ale semnalului care variază lent în timp, dar

dată fiind carecteristica nestaționară a semnalelor audio, lungimea ferestrei nu trebuie să fie pre mare, astfel încât informația despre evenimentele de scurtă durată să nu fie pierdută în proces.

Scopul principal al unei ferestre, este acela de a minimiza discontinuitățile apărute la finalul fiecărui set de date(în contextul acestei metode, soluția uzuală este fereastra Hanning). Acest rezultat se obține prin suprapunerea ferestrelor. Algoritmul de post-procesare se folosește de această informație pentru a reduce distorsiunile introduse de SS.

2.2. Tehnici de extragere a parametrilor

O parte crucială a procesului de recunoaștere vocală este reprezentată de extragerea parametrilor, mai ales când aplicația este proiectată să funcționeze în medii zgomotoase. Procesul vorbirii poate fi parametrizat prin codare liniar predictivă(LPC), predicție perceptual liniară(PLP), coeficienți mel cepstrali in frecvență(MFCC), PLP-RASTA etc.O parte dintre aceștia iau în considerare natura vorbirii în timp ce extrag parametrii(MFCC și PLP) în timp ce LPC face o estimare a parametrilor viitori bazându-se pe parametrii anteriori[8]

Extragerea parametrilor în sistemele de recunoaștere vocală reprezintă calculul unui vector de parametri care generează o reprezentare compactă a unui semnal dat. Acest proces impică de obicei un număr de trei pași: analiza spectral-temporală a semnalului vorbit, care generează parametri inițial care descriu anvelopa spectrului de puteri pentru intervale scurte de vorbire. Al doilea pas face compilarea unui vector extins de parametri care conține parametrii statici și dinamici. Într-un final, ultima etapă transformă acești vectori de parametri într-o reprezentare mai compactă, care va fi furnizată sistemului de recunoaștere.

2.2.1. MFCC(Mel-Frequency Cepstral Coefficients)

Fiind una dintre cele mai populare metode de extragere a parametrilor, MFCC-ul se bazează pe analiza în domeniul frecvență folosind scara Mel, bazată pe scara de sensibilitate a urechii umane. MFCC-ul denumește reprezentarea cepstrului semnalului ferestruit și extras folosind Transformarea Fourier Discretă. Diferența față de cepstrul real fiind folosirea unei scări neliniare în frecvență pentru aproximarea comportamentului receptorilor acustici umani.

Scara *mel* reprezintă o scară care împarte tonurile după modul în care un ascultător poate distinge distanța între ele, considerând pași egali între cele două tonuri, pornind de la presupunerea că în intervalul 0 – 1000 Hz, scara *mel* corespunde reprezentării în *Hz*.[8]

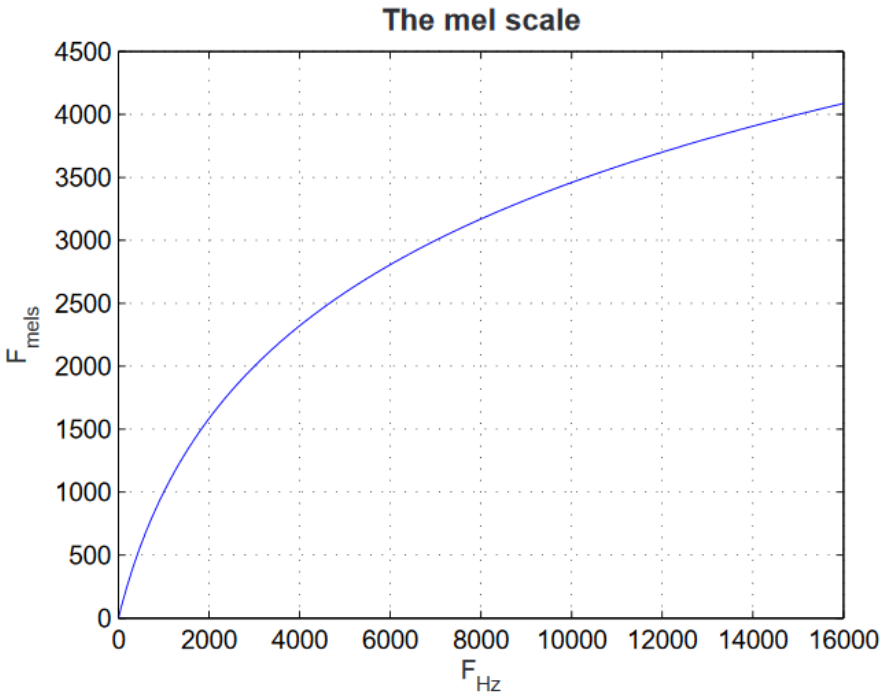

Figura 2.3 : relatia intre scara Hz si scara mel

Relația care leagă cele două scări poate fi aproximată prin următoarea ecuație:

$$F_{mel} = \frac{1000}{\log(2)}[\,1 + \frac{F_{Hz}}{1000}]$$

Cepstrul(nume provenit din inversarea primei silabe a cuvântului spectru) unui semnal este definit ca fiind "spectrul de puteri al logaritmului spectrului de puteri"[9], printre aplicațiile sale originale numarându-se și detecția ecoului în semnalele seismice, unde s-a dovedit și superioritatea acestuia față de funcția de autocorelație[10].

Cepstrul real al unui semnal poate fi definit după următoarea relație:

$$c_x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} \log|X(e^{j\omega})|e^{j\omega n}d\omega$$

Iar cepstrul complex este dat de:

$$c_x[n] = \frac{1}{2\pi}\int_{-\pi}^{\pi} [\log|X(e^{j\omega})| + jarg(X(e^{j\omega}))]e^{j\omega n}d\omega$$

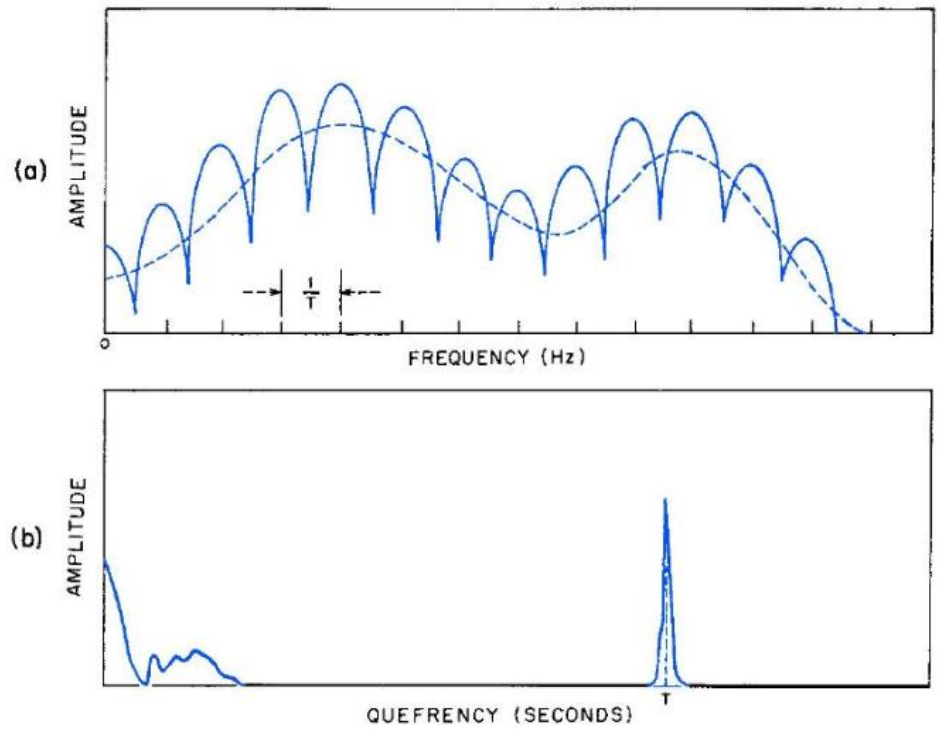

Figura 2.4[11] Diagramă în care se face reprezentarea unui semnal vocal generic în frecvență și în quefrecvență

De notat faptul că, contribuțiile datorate excitațiilor periodice apar la multpli întregi de perioada fundamentală, ceea ce ar putea cauza probleme în sistemul de recunoaștere pentru copii și femei cu voci înalte, în timp ce contribuțiile parametrilor care trebuiesc modelați se vor concentra în zona de „cerfvrență" joasă.

Tehnica MFCC se folosește de aceste două concepte pentru a extrage parametrii din vorbire într-un mod similar cu cel folosit de corpul uman pentru a auzi vorbirea. În primul rând, semnalul vorbit trebuie împărțit în cadre având un număr suficient de eșantioane. Cele mai multe sisteme folosesc cadre suprapuse pentru a atenua tranziția de la un cadru la altul, apoi acestor cadre li se aplică o fereastră Hamming pentru a elimina discontinuitățile care există la marginile cadrelor.

Coeficienții *w(n)* ai ferestrei Hamming de lungime *n* calculându-se după formula:

$$W(n) = 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), 0 \leq n \leq N-1$$

N fiind numărul de eșantioane și *n* eșantionul curent. După ce se aplică ferestruirea, se poate calcula transformata Fourier discretă a semnalului(se preferă FFT-ul din considerente de complexitate de calcul), după care se aplică un filtru care trece spectrul in scară mel.

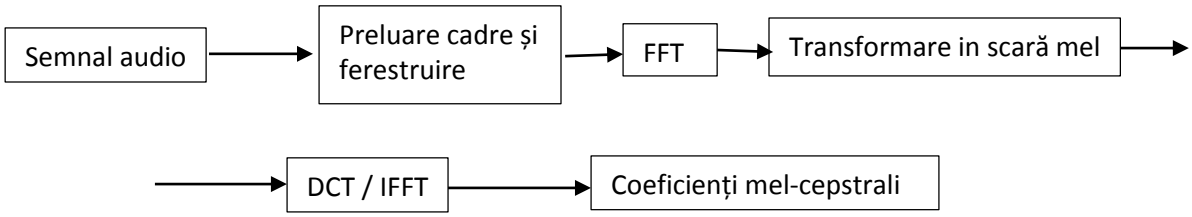

Figura 2.5. Schema bloc de obținere a parametrilor MFCC

Ultimul pas presupune calcularea *Discrete Cosine Transformation(DCT)* din rezultatele filtrului mel. Pentru fiecare cadru de vorbire, se calculează un set de coeficienți MFCC. Mulțimea de coeficienți se mai numește vector acustic deoarece reprezintă caracteristicile importante din punct de vedere fonetic ale vorbirii și este util pentru analiza și procesarea ulterioară.

2.2.2. RASTA-PLP

Modelul PLP dezvoltat de Hynek Hermansky abordează conceptul de psichofizică(relația dintre stimulii fizici și senzațiile/percepțiile pe care aceștia le induc) în auz, acesta având performanțe mai bune fiindcă exclude informația irelevantă din vorbire[12]. Spectrul pe termen scurt al vorbirii analizat prin prisma criteriilor psichofizice este cu toate acestea vulnerabil la răspunsul în frecvență al canalului de comunicație[13]. Prn urmare a fost dezvoltată metodologia *RelAtive SpecTrAl* (RASTA) care transformă sistemul *PLP* în unul mai robust în ceea ce privește distorsiunile liniare în spectru.

Metodologia RASTA înlocuiește spectrul absolut cu o estimare spectrală în care fiecare bandă de frecvență este trecută printr-un filtru trece bandă. Dat fiind că orice componentă statică sau lent variabilă în timp este suprimată de această operațiune, noua estimare asupra spectrului este mai puțin susceptibilă la variații lente ale spectrului pe termen scurt. În momentul in care mediul în care se lucrează cu spectrul logaritmat, componentele spectrale suprimate reflecă efectul factorilor convolutivi în semnalul de intrare, aceștia fiind introduși de răspusul în frecvență al mediului de comunicație.

Pașii prin care trece RASTA-PLP sunt următorii:

1) Calculează spectrul de bandă critică și preia logaritmul acestuia
2) Estimează derivata temporală a spectrului de bandă critică folosind regresia liniară prin 5 valori consecutive
3) Prelucrarea neliniară poate fi executată în acest domeniu
4) Reintegrează derivata log bandă-critică folosind un system IIR de ordinal 1. Poziția polului acestui system poate fi ajustată astfel încât să genereze dimensiunea efectivă a ferestrei.
5) Conform metodei PLP convenționale, adaugă procedura “equal loudness curve” și înmulțește cu 0.33 pentru a simula legea auzului

6) Aplică funcția exponențială log spectrului, obținând un prim spectru auditiv
7) Calculează modelul acestui spectru folosind tehnica convențională a PLP

În cazul prezentat, procesul de integrare-derivare este echivalent unei filtrări trece bandă pentru fiecare canal de frecvență printr-un filtru IIR cu funcția de transfer

$$H(z) = 0.1 * \frac{2 + z^{-1} - z^{-3} + 2z^{-4}}{z^{-4}(1 - 0.98z^{-1})}$$

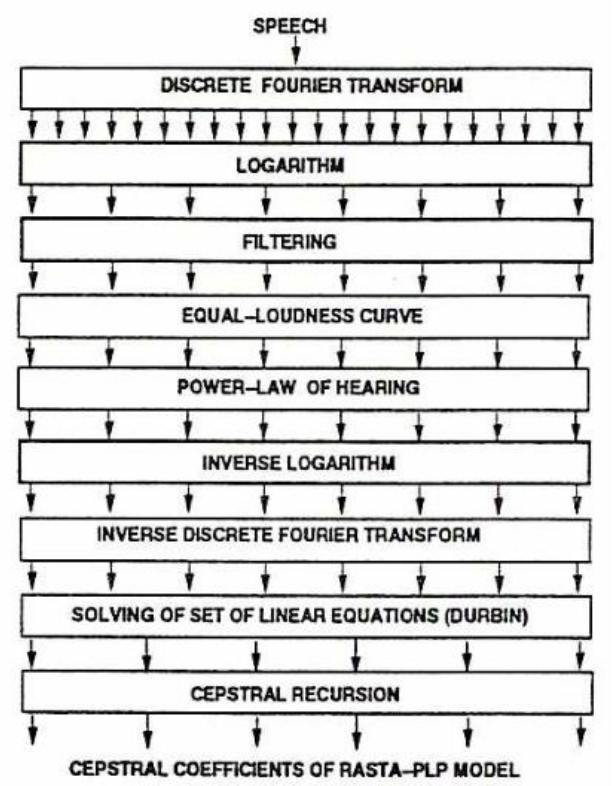

Figura 2.6. Procesul RASTA reprezentat graphic, [13]

Distorsiunile liniare ca cele induse de folosirea unui alt microfon apar ca o constantă în log spectru. Partea trece sus a filtrului trece bandă are ca scop atenuarea efectului zgomotului de convoluție introdus de canal. În timp ce partea trece jos are scopul de a atenua schimbările rapide în spectru datorate artefactelor de analiză.

Capitolul 3.
Arhitectura unui sistem de recunoaștere vocală

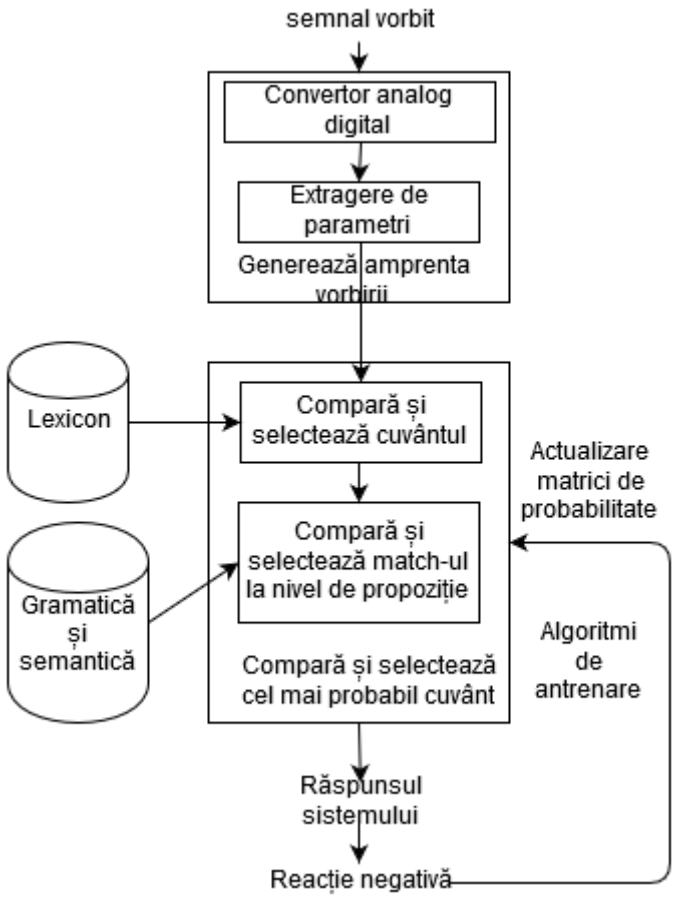

Figura 3.1. Arhitectura unui system de recunoaștere vocală

Sarcina unui sistem de recunoaștere vocală este de a transforma vorbirea într-un șir de cuvinte prin intermediul unui program. Aplicațiile care folosesc recunoașterea vocală dau posibilitatea utilizatorilor de a interacționa cu programul într-un mod accesibil.

Principalul scop al unui sistem de recunoaștere automată a vorbirii este de a intuit cel mai probabil șir de simboluri discrete din toate șirurile posibile în limba L, dat fiind un semnal acustic O. Considerând intrarea un șir de evenimente discrete (O) și secvența de simboluri care trebuiesc recunoscute (W), putem defini următoarele simboluri

$$O = o_1, o_2, o_3, \ldots$$

$$W = w_1, w_2, w_3, \ldots$$

Scopul fundamental al sistemului de recunoaștere automată a vorbirii poate fi exprimat în mod formal în următorul fel:

$$\widehat{W} = argmax\ P(W|O)\ pentru\ W\ \in L$$

Ecuația anterioară determină faptul că, dată fiind o secvență *W* și un set de intrări O, trebuie determinată probabilitatea de apariție a evenimentului *W* condiționat de O. Se poate aplica prin urmare teorema lui Bayes pentru a ne conduce la următoarea relație:

$$P(W|O) = \frac{P(O|W)P(W)}{P(O)}$$

Ceea ce se regăsește în membrul stâng este mai la îndemână decât calculul lui *P(W|O),* în timp ce *P(W)* este definit ca probabilitatea a priori a secvenței. Aceasta este calculată folosindu-ne de informațiile anterioare pe care le avem despre apariția lui W. Dat fiind că *P(O)* este același pentru fiecare *W*, ecuația se poate rescrie sub forma:

$$\widehat{W} = argmax\ P(O|W)P(W)\ pentru\ W\ \in L$$

Probabilitatea *P(O|W),* care înseamnă probabilitatea evenimentului acustic O, condiționat de secvența W, este de obicei determinată folosind un model Markov cu stări ascunse.

În mod tipic, sistemul de recunoaștere vocală este dezvoltat din componente majore cum ar fi *front end-ul acustic, lexicon, model de limbă și decodor.[Chandra]*

Astfel, în mod curent, structura vorbirii este înțeleasă în felul următor: vorbirea reprezintă un flux audio continuu în care se suprapun stări staționare și dinamice. În această succesiune de stări, se pot defini structuri de bază ale vorbirii denumite și foneme. Cuvintele limbii sunt construite pe baza acestor foneme, deși în acest caz trebuie luat în considerare un aspect ulterior, contextul apariției acestor unități fonetice în vorbire.[cmu-sphinx wiki]

Proprietățile formei de undă care corespunde fiecărei foneme în parte sunt puternic dependente și pot varia în sens foarte larg în funcție de un număr de factori, printre care amintim:

contextul apariției fonemei, aceasta poate fi pronunțată diferit în funcție de fonemele care o preced sau succed;

vorbitor, originea geografică sau nativitatea vorbitorului afectează felul în care aceste foneme sunt generate de vorbitor;

stilul vorbirii, de asemenea, o discuție în context colocvial diferă în multe aspecte de o exprimare oficială sau academică, putând chiar să se distanțeze puternic de limba literară atât în ceea ce privește normele impuse în limbaj cât și viteza și introducerea unor structuri lexicale destinate comunicării într-un grup restrâns de vorbitori;

Aceste deplasări de la forma canonică a limbii fac modelarea unui sistem acustic considerând ca unitate fundamentă fonema o sarcină dificilă, astfel apare conceptul de difone, părți ale fonemei care se află între două foneme consecutive, de asemenea se poate discuta de alte unități subfonetice – substări ale fonemei.

O variantă intuitivă a acestei abordări este împărțirea fonemei în trei părți. O parte care depinde de fonema anterioară a cuvântului, o parte mijlocie, stabile care se apropie cel mai mult

de caracteristica fonemei în sine, și o parte finală care depinde de fonema viitoare. Din acest motiv, se preferă folosirea acestui model în recunoașterea automată a vorbirii.

De asemenea, sunt cazuri în care fonemele sunt tratate în contextul cuvântului din care fac parte, apărând astfel noțiunea de trifonemă. Spre deosebire de difone, aceste unități sublexicale sunt tratate ca și cum ar fi foneme, diferența în nume este dată de faptul că cele două denotă sunete oarecum diferite.

Fonemele construiesc unitățile care compun cuvântul, și anume silabele. Aceste forme sunt de interes pentru sistemele de recunoaștere automată a vorbirii deoarece silabele sunt mai de grabă dependente de intonație, astfel, în diferite maniere de vorbire, structura fonemelor poate varia, dar pronunția silabelor rămâne adesea constantă. Silabele formează în cele din urmă cuvinte care sunt de asemenea de interes pentru sistemele de recunoaștere deoarece reduc combinațiile posibile de foneme care se regăsesc în limbă(aspect care contează semnificativ pentru sistemele de recunoaștere a vorbirii continue, dată fiind multitudinea de forme pe care un cuvânt le poate lua în funcție de conjugate, acord și context).

Desigur, pe lângă formele intenționate ale cuvintelor care pot fi rostite, în semnalul acustic pot apărea uterații cum ar fi liniște, tuse, respirație, aceastea neavând un sens semantic, dar trebuiesc tratate în același mod în care sunt tratate cuvintele limbii.

Front-endul acustic se ocupă cu achiziția semnalului și extragerea vectorilor de parametri prin care este surprinsă informația utilă asupra semnalului sonor. Extragerea parametrilor însemnând procesul prin care informația în format audio este transformată prin metode specifice(MFCC; RASTA-PLP) în vectori acustici de dimensiune fixă. Parametrii unui cuvânt/fonemă sunt estimați din vectorii acustici ai datelor provenite de la antrenare.

Algoritmul de căutare sau decodorul verifică toate șirurile de cuvinte admise pentru a 0descoperi secvența de cuvinte pe care observațiile provenite de la semnalul audio le poate genera. Probabilitatea $P(O|W)$ este determinată de modelul acustic, pe când $P(W)$ este generat de modelul de limbă.

Funcționalitatea unui sistem de recunoaștere vocală poate fi descrisă prin extragerea unui set de parametrii din semnalul vorbit pentru fiecare cuvânt sau subunitate lexicală. Acești parametri descriu această unitate lexicală prin variația lor în timp și împreună construiesc un șablon care o caracterizează. După generarea acestor vectori de parametri, intervine etapa de antrenare în care un operator va citi toate cuvintele din vocabularul aplicației. Nu în ultimul rând, modelele de cuvinte sunt stocate și în etapa de evaluare și utilizare, modelul cuvântului de verificat este comparat cu modelul antrenat iar cea mai apropiată pereche este cea pe care sistemul o va considera validă.

3.1 Front-endul acustic

Front end-ul acustic presupune procesare de semnal și extragere de parametri. Principalul scop al extragerii de parametri acustici fiind reprezentarea informației utile din semnalul sonor

într-un mod compact, acest proces se desfășoară de obicei în 3 etape. Prima etapă se numește analiza vorbirii și prin intermediul unei analize cepstrale a semnalului generează parametrii primitivi conținând anvelopa spectrului de puteri a unor intervale scurte de vorbire(de perioade între 5-50 ms). A doua etapă calculează un vector extins de parametri care conține informația necesară referitoare la semnalul acustic. În cele din urmă, ultima etapă prelucrează vectorul provenit din etapa a doua astfel încât să fie cât mai compact și ușor interpretat de către sistemul de recunoaștere.

Este de dorit obținerea unei mulțimi de parametri care să conțină maximul informației utile care se poate extrage din caracteristicile măsurabile ale semnalului. Deși în literatură nu se definește un standard care să se potrivească oricărei aplicații, alegerea parametrilor trebuie să țină cont de un set de criterii:

- Sistemul trebuie să fie capabil de a face diferența între sunetele asemănătoare din vorbire, astfel încât acestea să nu se confunde;
- Este de droit ca modelul acustic să poată fi creat pe baza acestor sunete fără a fi nevoie de o bază de date de antrenare excesiv de mare;
- Statisticile pe care ei le generează trebuie să fie cât mai invariante posibil în ceea ce privește natura vorbitorului și mediul în care se desfășoară vorbirea;

Desigur, fiind vorba de un eveniment probabilistic cu ar fi vorbirea(sistemul poate avea cel mult o capacitate de a intui ceea ce vorbitorul va spune în continuare dată modelul de limbă), sistemele de recunoaștere automată a vorbirii trebuie să dețină mecanisme de reducere a informației aflată în fiecare segment al înregistrării audio într-un număr relativ scăzut de parametric. Acești parametrii ar trebui să descrie fiecare segment în așa fel încât, fără a-și pierde caracterul distinctiv, segmental cu pricina să poată fi grupat cu altele asemănătoare(diferențierea în funcție de context).

Deși există o multitudine de metode posibile de extragere a parametrilor din semnalul acustic, una dintre cele mai folosite este cea care se bazează pe coeficienții mel cepstrali(MFCC). După cum a fost prezentat și într-un capitol anterior, tehnica MFCC constă într-un număr de etape prin care semnalul este prelucrat și transformat în informație utilă pentru sistemul de recunoaștere. O scurtă descriere a acestor pași este după cum urmează:

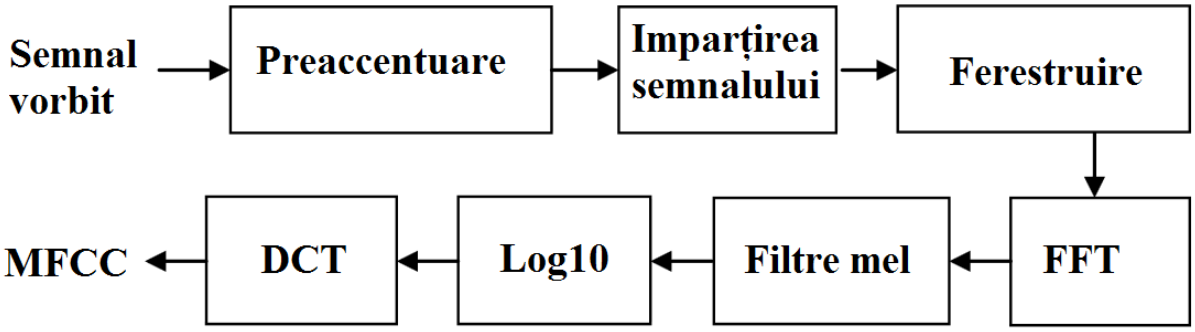

Figura 3.2: schema bloc a procesului de extragere a parametrilor

- Frecvențele înalte ale spectrului sunt amplificare pentru a permite recunoașterea mai ușoară a acestora de către procesul de antrenare și recunoaștere(din cauza neliniarității în scara mel);
- Ferestruirea, process foarte important pentru procesarea ulterioară a semnalului. Această etapă împarte semnalul în segmente de câte N milisecunde, se adoptă de obicei o fereastră Hamming(sau Hann) pentru acest tip de aplicații astfel încât să nu se piardă informație datorită discontinuităților la limitele domeniului de eșantionare
- Transformarea Fourier Discretă este aplicată semnalului trecut prin fereastra Hamming, rezultând astfel o reprezentare în amplitudine și fază pentru semnalul cu pricina. De obicei la acest pas se preferă implementarea unui algoritm de FFT din considerente legate de complexitatea de calcul al acestei transformări, lucru care poate impune probleme majore platformelor hardware care dispun de resurse limitate;
- Bancul de filter mel, are rolul de a trece reprezentarea spectrului din frecvența în Hz, într-o scară mai potrivită caracteristicilor de sensibilitate la ton a urechii umane(se cunoaște faptul că urechea umană este mai puțin sensibilă la frecvențe de peste 1000 Hz. Astfel se preferă utilizarea scării mel, prin introducerea unui banc de filtre triunghiulare distribuite la intervale egale sub frecvența de 1000 Hz și spațiate logaritmic la frecvențe mai mari de 1000 Hz), obținându-se astfel așa-zisul spectru mel, care apoi se logaritmează;
- Ultimul pas al obținerii parametrilor MFCC este aplicarea transformării cosinusoidale discretă asupra coeficienților spectului mel;
- Nu în ultimul rând, este de interes pentru sistemul de recunoaștere vocală și perioada în care segmentele în care a fost împărțit semnalul se suprapun, astfel, ca ultim pas al tehnicii MFCC este specificată calcularea derivatei de ordinul I și II ai coeficienților mel-cepstrali;

Un aspect negative al tehnicii MFCC este faptul că valorile coeficienților nu sunt deosebit de robuste pentru semnale care prezintă zgomot, astfel(în cazul în care nu s-au efectuat filtrări asupra semnalului cu scopul de a reduce efectul zgomotului) se obișnuiește normalizarea valorii acestora,

3.2. Modelul acustic

Modelul acustic este poate cea mai importantă sursă de informație de care dispune sistemul, deoarece acesta conține reprezentările specifice fiecărei unități acustice din vocabularul sistemului, împreună cu parametrii care le caracteriează. În construirea modelului acustic, cel mai important aspect care trebuie luat în considerare este alegerea unităților fundamentale pe care acesta se bazează.

În general, modelarea acustică se referă la stabilirea unei reprezentări statistice a șirului de vectori de parametri calculați din semnalul vorbit. În acest scop se folosesc modele Marov cu stări

ascunse pentru a construi modelul acustic(de asemenea se pot folosi și rețele neurale). În modelul acustic se regăsesc reprezentările statistice pentru fiecare dintre sunetele care compun limba. Fiecărei reprezentări de tipul acesta îi este asociat un model acustic al unei foneme prin preluarea unei baze de date de vorbire numită și corpus de text care este antrenat prin algoritmi speciali. Dacă vorbim de modele independente de context, fiecare fonemă are asociat un HMM. În schimb modelele dependente de context sunt formate din HMM-uri cu 3 stări. Decodorul ascultă semnalul acustic achiziționat și caută HMM-ul corespunzător în modelul acustic.

În concluzie, fiecare cuvânt din înregistrarea audio este descompus într-o secvență de sunete care reprezintă unități fonetice fundamentale. Modelul acustic descrie probabilitatea unei observații a sistemului.

3.2.1. Fonologia limbii române

Pentru a putea realiza un model acustic eficient, trebuie să avem în vedere aspectele fonetice specifice limbii române. Scopul fiind acela de a extrage parametrii specifici vorbirii, caracteristicile limbii sunt un pas important în înțelegerea acestui demers.

3.2.1.1. Fonemele limbii române

Analizând valoarea funcțională a diferitor aspecte fonetice, se ajunge la concluzia că limba română standard folosește următoarele unități segmentale:

- șapte vocale;
- patru semivocale, care în unele analize sunt echivalate cu vocalele corespunzătoare;
- o vocală asilabică și devocalizată o vocală asilabică și devocalizată /i̯/, care uneori este echivalată cu vocala /i/;
- douăzeci de consoane;

Și două unități suprasegmentale:

- accent;
- intonație;

În limba română, distincția între consoane și vocale, este că cele din urmă sunt sunete care pot alcătui singure o silabă și pot purta accentul, pe când consoanele nu au această calitate.

Putem distinge prin urmare următoarele tipuri de vocale:

- Vocale propriu-zise:

Sunet	Exemplu	Notație fonetică
[a]	amar	a
[e]	elev	e
[i]	iris	i

[o]	ocol	o
[u]	uluc	u
[ə]	fără	ă
[ɨ]	vîrî	î
[ø]	bleu	Ö
[ʏ]	Bruxelles	ü

Tabelul 3.1: Vocalele limbii române

De asemenea, distingem și un număr de patru semivocale: /e̯/ (în deal),/i̯/ (în *fier*),/o̯/ (în *coasă*), /u̯/ (în *ziua*). Semivocalele /e̯/ și /o̯/ pot apărea doar înainte de o vocală, în timp ce restul de două pot apărea atât înainte cât și după. [16]

Consoanele care apar în limba română sunt următoarele:

Tabelul 3.2: Consoanele limbii române

Sunet	Exemplu	Notație fonetică
[p]	pace	p
[b]	bun	b
[t]	tare	t
[d]	dor	k
[k]	cal	k
[c]	chel	k'
[g]	gol	g
[ɟ]	ghid	g'
[ʦ]	țap	ț
[ʧ]	cer	č
[ʤ]	ger	ğ
[f]	afară	f
[v]	var	v
[s]	sare	s
[z]	vază	z
[ʃ]	șarpe	ș
[ʒ]	ajutor	j
[h]	ham	h
[ç]	mo**h**ican	
[x]	**h**rib	x
[m]	amic	m
[ɱ]}	conversa	
[n]	nor	n
[ŋ]	pru**n**c	ŋ

[l]	lung	L
[r]	arid	R

În ceea ce privește accentul de intensitate, acesta este considerat în limba română ca fiind fonemic, adică poate determina diferențe de sens. Se poate observa acest efect în cuvinte precum: móbilă - mobílă - mobilắ,. acéle – ácele, véselă – vesélă, copíi – cópii, umblắ - úmblă, desfác ásta - dés fac asta. Foneticizarea accentului de intensitate este o characteristică răspândită în limbile lumii. În limba română, doar o singură silabă poate fi accentuată, celelalte silabe putând primmi unul sau mai multe accente mai slabe, mai ales în cuvintele cu mai multe silabe.[

3.3. Modelul de limbă

Modelul de limbă oferă sistemului o serie de constrăngeri aplicabile secvenței de cuvinte care pot fi recunoscute, în sensul că exclude combinațiile de foneme care nu se regăsesc în limbă. Aceste constrângeri sunt reprezentate fie de regulile gramaticale ale limbii, fie de statistici estimate din corpusul de antrenare(aplicațiile de control prin recunoaștere vocală suferă în acest caz din pricina incertitudinii extinse asociate limbii. Prin analogie, deși există cuvinte care conțin foneme asemănătoare, unui utilator uman nu îi este dificil să distingă aceste diferențe, în principal datorită faptului că are capacitatea de a înțelege contextul, și de asemenea, este obișnuit să audă fraze comune, așteptându-se astfel să apară anumite cuvinte într-un context anume.

Acest context este dat de modelul de limbă în cadrul unui sistem de recunoaștere, modelul precizând cuvintele care pot exista și modul în care acestea pot apărea. Modelul acustic este antrenat prin modele probabilistice de tip N-gram prin verificarea unor corpusuri extinse de text și de asemenea, prin reducerea perplexității datelor de antrenare. Prin urmare, algoritmiii care îmbunătățesc modelul de limbă bazându-se pe efectul pe care cuvintele le au asupra recunoașterii vocale preferă un model de limbă bazat pe distribuția de probabilitate a cuvintelor pe care vorbitorul le poate exprima în continuare, dat un istoric al cuvintelor pe care le-a vorbit. Modelele comune folosite sunt de tip bigram sau trigram. Aceste modele conțin probabilități calculate ale unor grupuri de două sau trei cuvinte dintr-o secvență dată.

3.4. Decodorul

Scopul decodorului este de a descoperi cel mai probabil șir de cuvinte *W* dat de șirul de observații *O*, și de modelele fonetice și acustice. Problema decodării se rezolvă prin utilizarea algoritmilor de programare dinamică, de exemplu algoritmul Viterbi. Acesta din urmă nu evaluează probabilitatea fiecărei posibilități oferie de model, fiind mai de grabă atent la găsirea căii optime care oferă cea mai bună aproximare a lui *O*. Pe scurt, algoritmul Viterbi presupune căcea mai bună cale la momentul *t* este dată de cea mai bună cale identificată la momentul *t-1*. Lucru care nu este neapărat valabil în toate cazurile, dat fiind că o cale care pare nu foarte probabilă la început poate deveni mult mai probabilă în cazul în care luăm în considerare contextul, caz tratat de o extensie a algoritmului Viterbi și de algoritmul Forward-Backward.

3.5. Abordări ale recunoașterii vocale în contextul unui sistem electronic

Deși recunoașterea vocală reprezintă un câmp vast de cercetare, se pot distinge trei abordări principale ale subiectului, fiecare cu particularitățile lui și cu compromisuri specifice. Acestea trei sunt abordarea acustic-fonetică, abordarea prin recunoaștere de caracteristici și abordarea prin inteligență artificială.

Abordarea acustic-fonetică[14]: Se bazează pe presupunerea că în limbă există o mulțime finită și discretă de unități fonetice. Aceste unități fonetice fiind caracterizate de proprietăți acustice manifestate în vorbire sau în spectru. Primul pas în abordarea acustic-fonetică este aplicarea unei analize spectrale a vorbirii combinate cu o metodă de extragerii parametrilor care transformă caracteristicile spectrale într-o mulțime de parametrii care descriu proprietățile acustice ale diferitor unități fonetice. Următorul pas este reprezentat de etapa de ferestruire în care semnalul acustic este segmentat în părți staționare din punct de vedere acustic, urmat de atașarea unei etichete fonetice pentru fiecare regiune, rezultând astfel o latice de foneme. Următorul par ar fi determinarea cuvântului posibil din șirul de etichete fonetice produse de pasul anterior

Recunoașterea de caracteristici: Este compusă din doi pași esențiali, și anume, antrenarea caracteristicilor și compararea lor. În etapa de comparare a caracteristicilor, se face o diferențiere directă între vorbirea necunoscută și fiecare caracteristică învățată în etapa de antrenare astfel încât să se determine natura necunoscutei în funcție de cât de bine se pliază pe ceea ce sistemul cunoaște. Caracteristica principală a acestei abordări este reprezentată de modelul matematic riguros care stă în spatele ei, aceasta construind o reprezentare consistentă a caracteristicilor pentru comparare antrenând eșantioane prin intermediul unui algoritm specific de antrenare(Backward-Forward). Caracteristica prin care este identificată vorbirea se bazează pe un model statistic(model Markov ascuns de exemplu) și poate fi aplicată unei unități care compune un cuvânt, unui cuvânt în sine sau unei fraze. Datorită acestor aspecte, acesă abordare a devenit metoda predominantă în domeniul recunoașterii vocale.

Inteligența artificială: Inteligența artificială este un hibrid între cele două aborări enunțate anterior. Acest lucru se datorează faptului că preia concepte și din abordarea acustic-fonetică și din recunoașterea de caracteristici. Principalele abordări utilizate recunoașterea vocală din domeniul recunoașterii de caracteristici sunt recunoașterea determinită bazată pe procedeul numit *dynamic time warping(DTW)* și recunoașterea prin procese stohastice care fac uz de HMM-uri. În contextul *DTW*-ului, fiecare clasă care trebuie recunoscută este reprezentată de unul sau mai multe șabloane, dat fiind că folosirea mai multor șabloane de referință este preferată pentru creșterea variabilității vorbitorului. În timpul recunoașterii, se calculează o distanțiere între vorbirea observată de system și caracteristica data de clasă. Cuvântul recunoscut corespunde căii prin model care minimizează distanța acumulată. În sistemele comerciale, se preferă recunoașterea caracteristicilor prin intermediul HMM-

urilor în schimb datorită proprietățile de generalizare specific modelelor probabilistice și de asemenea, din considerente de ocupare a resurselor de calcul.

De asemenea, acestor abordări li se adaugă o serie de extensii cu menirea de a optimiza procesul de recunoașterea, problemă care devinde dificilă pentru abordările clasice în cazul în care avem de a face cu un vocabular extins.

Învățarea generativă de tip HMM-GMM: În cazul sistemelor de recunoaștere vocală, abordarea de tip generativ folosește de obicei modele Markov bazate pe mixturi Gaussiene. Sistemele de recunoaștere vocală convenionale folosesc modelul cu mixturi Gaussiene bazate pe modele Markov ascunse pentru a reprezenta structura secvențială a semnalului vorbit. Modelele Markov sunt folosite în sistemele de recunoaștere vocală deoarece semnalul sonor vorbit poate fi văzut ca unul staionar pe perioade scurte de timp. Vorbirea fiind văzută ca un model Markov din considerente stohastice. În mod timipc stările HMM-ului folosesc o mixtură de Gaussiene pentru a modela reprezentarea spectrală a undei audio. Un sistem de tip GMM-HMM este parametrizat de $\lambda = (A; B; \pi)$. π vector care conține probabilitățile a priori de stare; A = (a_{ij}) fiind matricea de tranziție a stării, iar $B = \{(b_1, \ldots, b_n)\}$ o mulțime unde b_j reprezintă modelul mixturii Gaussiene pentru starea j. Starea fiind asociată unui subsegment al fonemei în vorbire(senone).

3.6. Caracteristici ale sistemelor de recunoaștere vocală[17]

Sistemele de recunoaștere automată a vorbirii sunt de obicei proiectate pentru a face față unei anumite sarcini specifice, robustețea fiind una dintre marile probleme cu care aceste sisteme se confruntă. Există un număr de probleme care trebuiesc adresate pentru a defini clasa de funcționare a unui astfel de sistem. Printre acestea se numără modelarea unităților (cuvintele, silabele și fonemele) utilizate pentru recunoaștere, dimensiunea vocabularului(mic, mediu, extins), complexitatea sintaxei, care poate varia de la simplă la complexă(pentru care se impune folosirea modelelor de limbă de tip N-gram, perplexitatea sarcinii, modul în care se recunoașterea vorbirii(izolată, continuă, spontană), raportarea la vorbitor(sistem antrenat pe un anume vorbitor, adaptiv, independent de vorbitor, dependent), mediul vorbirii(liniște, mediu zgomotos) și felul în care se face achiziția semnalului sonor(dependent de calitatea microfonului, telefon, matrice de microfoane, sau canalul de transmisie).

Calitatea unui sistem de recunoaștere a vorbirii este direct proporțională cu robustețea sa, aceasta fiind înțeleasă drept calitatea sistemului de a conduce la rezultate satisfăcătoare în dată fiind variabilitatea semnalului sonor. Există un număr de factori care determină acuratețea unui astfel de sistem. Aceștia din urmă fiind variabilitatea vorbitorului, variabilitatea pronunției, variabilitatea regională(diferite accente care pot apărea), variabilitatea ratei vorbirii(viteza de pronunție), variabilitatea contextului, variabilitatea canalului și variabilitatea mediullui în care se face vorbirea. În proiectarea sistemelor de recunoaștere vocală, trebuie luați în considerare acești factori astfel încât să fie posibilă construirea unui model care să ducă la o bună rată a recunoașterii,

fiind în același timp independentă de acești factori[15]. În ceea ce privește sistemele avansate, trebuie luat în considerare aspectul automatizării generării lexiconului și construirea unor algoritmi avansați se segmentarea a vorbirii și algoritmi de rejecție a uteraților, astfel încât aceste sisteme să poată atinge sau chiar surclasa performanțele umane în sarcini de recunoaștere vocală.

3.7. Evaluarea performanței sistemelor de recunoaștere vocală

Performanța sistemelor de recunoaștere automată a vorbirii este de obicei specificată în strânsă legătură cu acuratețea și viteza cu care se realizează sarcina. Acuratețea unui astfel de sistem, poate, prin urmare să fie măsurată în termeni de erori raportate la cuvânt(WER), pe când viteza se măsoară folosind timpul real.
Alte unități de măsură pentru acuratețe includ:

- rata de eroare raportată la un singur cuvânt(SWER);
- rata de success a comenzii(CSR)

Performanța unui sistem de recunoaștere este prin urmare măsurată folosindu-ne de termeni precum Word Error Rate și Word Recognition Rate(WRR). Erorile de cuvânt sunt categorizate după numărul de inserții, substituții și ștergeri pe care sistemul le efectuează în timpul decodării. Cele din urmă pot fi definite în următorul mod:

- inserție: cuvânt adăugat de sistem astfel încât ipoteza să se regăsească în referința sistemului(baza de date)
- ștergere: cuvânt eliminat de sistem și perceput ca zgomot, astfel încât ipoteza generată de sistem să se plieze pe modelul de limbă
- substituție: cuvânt ambiguu, înlocuit de sistem cu cea mai bună corespondență pe care o poate găsi.

Într-un final, WER-ul și WRR-ul pot fi exprimate în felul următor:

$$Word\ Error\ rate(\%) = \frac{Inserţii + Substituţii + Ştergeri}{Numărul\ de\ cuvinte\ de\ referinţă} * 100$$

$$Word\ Recognition\ Rate\ (WRR) = 1 - WER = \frac{N - S - D - I}{N}$$

3.7. Criterii de performanță pentru sisteme de recunoaștere automată a vorbirii

În construcția unui astfel de sistem, având predeterminată o serie de standarde de performanță, astfel încât să fie posibilă evaluare obiectivă a produsului rezultat și, de asemenea, ierarhizarea acestor sisteme în funcție de calitatea lor. Inspectând structura sistemului, putem defini câteva aspecte care să ne ajute în acest demers:

- resursele necesare construirii sistemului:
  - hardware(sistemul de calcul necesar rulării aplicației de recunoaștere, microfonul prin care se face achiziția, interfața cu un alt sistem pentru care sistemul de recunoaștere furnizează date)
  - software(complexitatea de calcul pe care o presupune metoda aleasă de recunoaștere, dificultatea implementării ei, expertiza proiectantului)
  - resurse monetare(necesare finanțării proiectului, de la cercetare la echipa care trebuie să îl pună în funcțiune
- popularitatea pe piață(oportunitățile de monetizare)
- ușurința implementării(de exemplu extragerea parametrilor prin MFCC necesită mai puține resurse de calcul și manoperă decât folositea tehnicii RASTA-PLP
- sistemul în care aplicația se poate folosi, acesta fiind dependent de următoarele caracteristici:
  - numărul de vorbitori care pot fi identificați de sistem
  - numărul de limbi recunoscute
  - dimensiunea vocabularului

Acestea fiind spuse, putem afirma că în construirea unui sistem de recunoaștere vocală trebuie să luăm în considerare nu numai aspectele tehnice ale implementării, dar și caracteristica funcțională a acestuia.

Capitolul 4

Implementare practică

Partea practică a proiectului constă în implementarea unui sistem de recunoaștere vocală pe platforma Raspberry Pi 3 cu ajutorul toolkitului de recunoaștere vocală CMUSphinx.

Astfel, sistemul construit este capabil de a recunoaște o secvență oarecare de cifre legate. Necesitatea unui astfel de sistem este dată de utilitatea pe care o pot avea interfețele wireless în ajutorarea persoanelor bolnave de afecțiuni ale vederii sau care au probleme locomotorii. De asemenea, sistemul poate fi folosit pentru recunoașterea CNP-ului unei anumite persoane sau, prin intermediul unor codări de tip Huffman, se poate construi un sistem de recunoaștere a comenzilor(care necesită în această implementare nu necesită un model complex de limbă sau acustic).

4.1 Resurse hardware

4.1.1 Placa Raspberry Pi

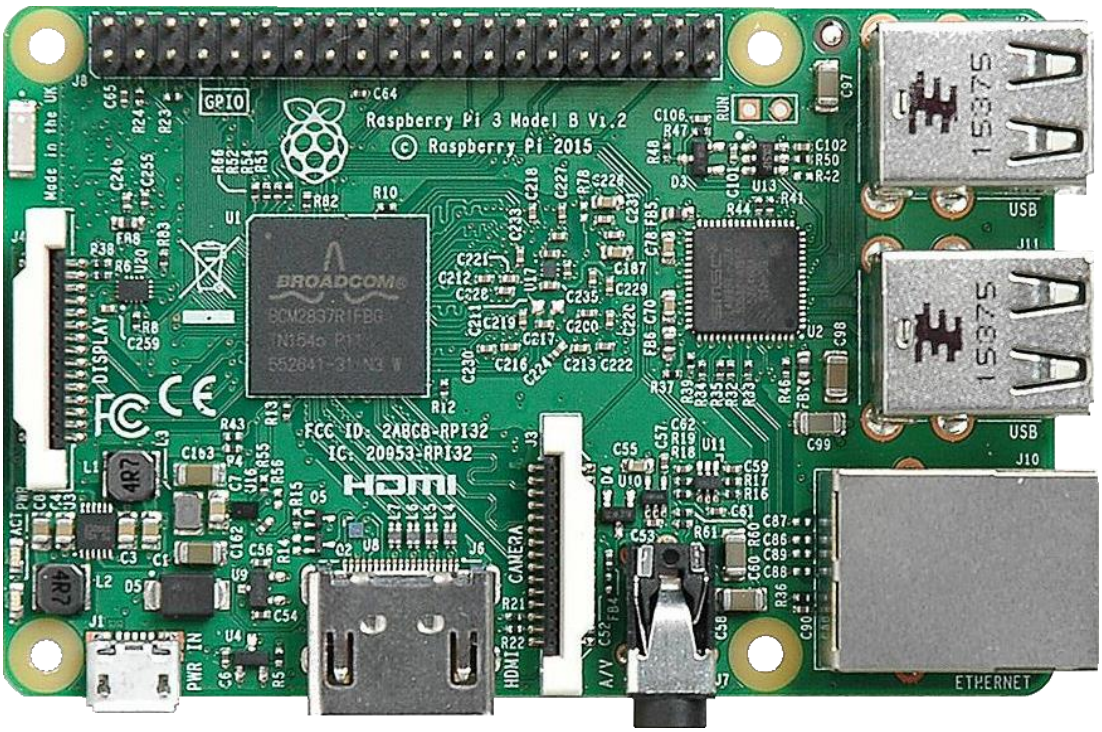

Figura 4.1: Placa Raspberry Pi 3 Model B

Specificații tehnice Raspberry Pi 3	
Procesor	Broadcom BCM2837 64bit Quad Core Processor 1.2GHz
Memorie RAM	1 GB
Conectivitate – wifi	BCM43438 WiFi
Conectivitate – bluetooth	Bluetooth Low Energy (BLE)
Conectivitate – USB	4 porturi USB 2
Audio – iesire	Iesire Stereo
Conectivitate cu camera	Port HDMI pentru camera
Conectivitate – ecran tactil	DSI display port
MicroSD	Port pentru conectare MicroSD cu sistem de operare
Sursa de alimentare	MicroUSB cu suport de pana la 2.4 A

Tabelul 4.1 Specificațiile tehnice ale plăcii

Raspberry Pi este o serie de plăci de dezvoltare de dimensiuni reduse, proiectate în Regatul Unit de către Raspberry Pi Foundation pentru a promova predarea conceptelor de știința calculatoarelor în școli și în țările în curs de dezvoltare, fiind oferită la un preț accesibil.

4.2. Resurse software

4.2.1. Toolkit-ul CMU Sphinx

CMU Sphinx reprezintă un toolkit intens utilizat în aplicațiile de recunoaștere automată a vorbirii și include o serie de unelte care pot fi utilizate pentru a construi aplicații care au legătură cu procesarea semnalului vorbit.

Toolkit-ul include patru utilitare care se ocupă de diferite aspecte ale procesului de construire a unei aplicații de recunoaștere de vorbire, aceste fiind:

- Pocketsphinx – bibliotecă scrisă în C prin intermediul căreia se face decodarea semnalului vobit;
- Sphinxbase – bibliotecă support care este necesară pentru Pocketsphinx;
- Sphinx4 – decodor de voce adjustabil, scris în Java;
- Sphinxtrain – utilitare pentru antrenarea modelului acustic;

În cadrul proiectului, am portat utilitarul Pocketsphinx pe sistemul Raspberry Pi și am folosit un model acustic preantrenat pentru a face recunoașterea.

Procesul prin care a fost construit sistemul de recunoaștere a vorbirii urmează procesul tipic utilizat în realizarea oricărui sistem de recunoaștere a vorbirii automate. Astfel

- s-a folosit o bază de date de vorbire de 100 de vorbitori diferiți pentru antrenare, de la fiecare vorbitor fiind preluat un număr de 80 de fișiere;

- din aceste fișiere de antrenare s-au extras parametrii MFCC(39, 13 parametrii diferiți și derivatele de ordinul I și II);
- pe baza acestor parametrii, s-a antrenat modelul de acustic pentru sistemul de recunoaștere;

Toolkitul, având la îndemână parametrii MFCC, după antrenare va crea un număr de fișiere care conțin definițiile modelului acustic, acestea se numesc:

mdef – corespondențeler

means – mediile gaussiene

mixture_weights – mixturile de gausiene

noiseedict – dicționarul care conține filler-ele

transition_matrices – matricile de tranziție ale HMM-ului

variances – variațiile gaussienelor

feat.params – parametrii extrași

Dintre toate acestea, mdef și feat.params reprezintă fișiere de tip plain text, celelalte fiind în format binar. Acestea conțin un header care explicitează numărul de fluxuri și dimensiunea datelor, fiind urmate de datele în format raw.

Pe de altă parte, fișierul mdf conține corespondențele dintre trifonemele dependente de context și id-urile senonelor de stare.

4.2.2 Construcția dicționarului fonetic

Modelul fonetic al limbii este foarte important pentru dezvoltarea proiectului deoarece acesta execută transcrierea fonetică a transcrierilor valabile pentru baza de date de antrenare. Acesta are rol și la decodare, fiind utilizat de algoritmul Viterbi pentru a traduce stările HMM-ului în cuvinte. Prin urmare, dicționarul fonetic include toate cuvintele posibile pentru o anumită sarcină de recunoaștere a vorbirii alături de transcrieriler fonetice

Pentru limba curentă dat fiind că este posibil să avem numai succesiuni de cifre, dicționarul trebuie să conțină numai fonemele care apar în cuvintele:

- zero
- unu
- doi
- trei
- patru
- cinci
- șase

- șapte
- opt
- nouă

Prin urmare, dicționarul fonetic are următoarea formă:

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_cinci n_cinci k1_cinci2 i1_cinci

șase s1_șase a_șase s_șase e_șase

șapte s1_șapte a_șapte p_șapte t_șapte e_șapte

opt o_opt p_opt t_opt

nouă n_nouă o1_nouă w_nouă a1_nouă

Se observă că dicționarul fonetic transcrie cuvintele sub forma *[fonem]_[cuvânt]*, acest lucru este propus de către toolkitul CMUSphinx pentru a reduce perplexitatea limbii.

Dicționarul fonetic este un instrument lingvistic prin care este specificat modul în care se pronunță cuvintele limbii. Prin urmare, dicționarul fonetic face corespondența între forma scrisă și cea fonetică a componentelor limbii[18]. Forma fonetică a unui cuvânt este o succesiune de foneme.

Astfel, în sistemul nostru, dicționarul fonetic va face legătura între modelul acustic(care modelează modul de producere a sunetelor limbii) și modelul de limbă(care modelează felul în care o poate apărea un sir de cuvinte).

4.2.3. Creearea gramaticii cu stări finite

Deși sistemele de recunoaștere a vorbirii de ultimă generație utilizează modele de limbă statistice de tip n-gram(modele care se construiesc pe baza unei baze de date de text foarte mare, adaptată sarcinii de recunoaștere, care face estimarea a priorii a probabilității de apariție ale cuvintelor și secvențelor de cuvinte, aceste date fiind mai apoi utilizate pentru a calcula probabilitatea apariției unui cuvânt, dat fiind cuvântul anterior), sarcina noastră, din pricina faptului că deține un vocabular foarte redus, nu se pretează modelelor de limbă statistice, una dintre marile probleme care se ivesc în acest caz fiind chiar ambiguitatea probabilității de estimare a următoarei cifre, dat fiind că aceasta din urmă este egală pentru toți membrii limbii.

Astfel, pentru construirea modelului de limbă, s-a ales folosirea unei gramatici cu stări finite cu ajutorul formatului *Java Speech Grammar(JSGF).*

```
FSG_BEGIN <rodigits.numbers>
NUM_STATES 17
START_STATE 0
FINAL_STATE 1
TRANSITION 0 2 1.000000
TRANSITION 2 4 0.500041
TRANSITION 2 5 0.500041
TRANSITION 3 1 1.000000
TRANSITION 4 3 1.000000
TRANSITION 5 7 0.100000 nouă
TRANSITION 5 8 0.100000 opt
TRANSITION 5 9 0.100000 şapte
TRANSITION 5 10 0.100000 şase
TRANSITION 5 11 0.100000 cinci
TRANSITION 5 12 0.100000 patru
TRANSITION 5 13 0.100000 trei
TRANSITION 5 14 0.100000 doi
TRANSITION 5 15 0.100000 unu
TRANSITION 5 16 0.100000 zero
TRANSITION 6 2 1.000000
TRANSITION 6 3 1.000000
TRANSITION 7 6 1.000000
TRANSITION 8 6 1.000000
TRANSITION 9 6 1.000000
TRANSITION 10 6 1.000000
TRANSITION 11 6 1.000000
TRANSITION 12 6 1.000000
TRANSITION 13 6 1.000000
TRANSITION 14 6 1.000000
TRANSITION 15 6 1.000000
TRANSITION 16 6 1.000000
FSG_END
```

În cazul de față, gramatica cu stări finite este mult mai potrivită pentru sarcina curentă, descrierea expusă mai sus generează un graf orientat, iar fiecare coardă reprezintă "costul probabilistic" al posibilității de a fi precedat de un altul. Deși în urma procesului automat de generare a gramaticii, au apărut un număr de 16 noduri de tranziție, există efectiv în limbă doar 10 care pot reprezintă cuvinte, excluzând de asemenea nodurile care reprezintă starea inițială și finală a automatului.

4.2.4 Portarea CMU Sphinx pe Raspberry Pi3

Portarea suitei CMU Sphinx pe Raspberry PI 3 a fost facut sub denumirea de pocketsphinx, numele pocket – buzunar sugerand transformarea suitei intr-una de buzunar – compatibilitate pe sisteme integrate. Instalarea completa pe Raspberry Pi 3 contine 3 etape si anume: instalarea sistemului de operare, configurarea Arhitecturii Avansate de Sunet de Linux (**A**dvanced **L**inux **S**ound **A**rchitecture) si instalarea pocketsphinx.

Sistemul de operare ce ruleaza pe Raspberry Pi 3 se numeste Raspbian fiind o distributie de Linux Debian portata special pentru Raspberry Pi. Aceasta poate fi descarcata de pe site-ul oficial (https://www.raspberrypi.org/) si instalata cu ajutorul programului NOOBS. NOOBS este un utilitar ce boot-eaza la deschiderea sistemului integrat si pune la dispozitie mai multe sisteme de operare special concepute pentru Raspberry Pi. Instalarea se face cu doar cateva click-uri necesare pentru a selecta distributia de sistem de operare dorita a se instala. Intreg procesul dureaza in jur de 30 de minute, dupa care sistemului integrat este utilizabil.

Pasul 2 presupune instalarea si configurarea arhitecturii de sunet pentru a putea capta informatia de la microfonul atasat de noi. In primul rand, trebuie sa ne asiguram ca sistemul nostru este actualizat la ultima versiune si de asemenea toate programele instalate sunt updatate. Acest lucru il realizam cu urmatoarele comenzi:

sudo apt-get update

sudo apt-get upgrade

Intregul proces poate dura pana la aproximativ 20 de minute. Dupa ce ne-am asigurat ca suntem la zi cu software-ul instalat pe Raspberry PI va trebui sa vedem daca microfonul nostru este detectate de sistem si sa aflam cateva informatii despre el. Acest lucru il vom face cu urmatoarea comanda:

cat /proc/asound/cards

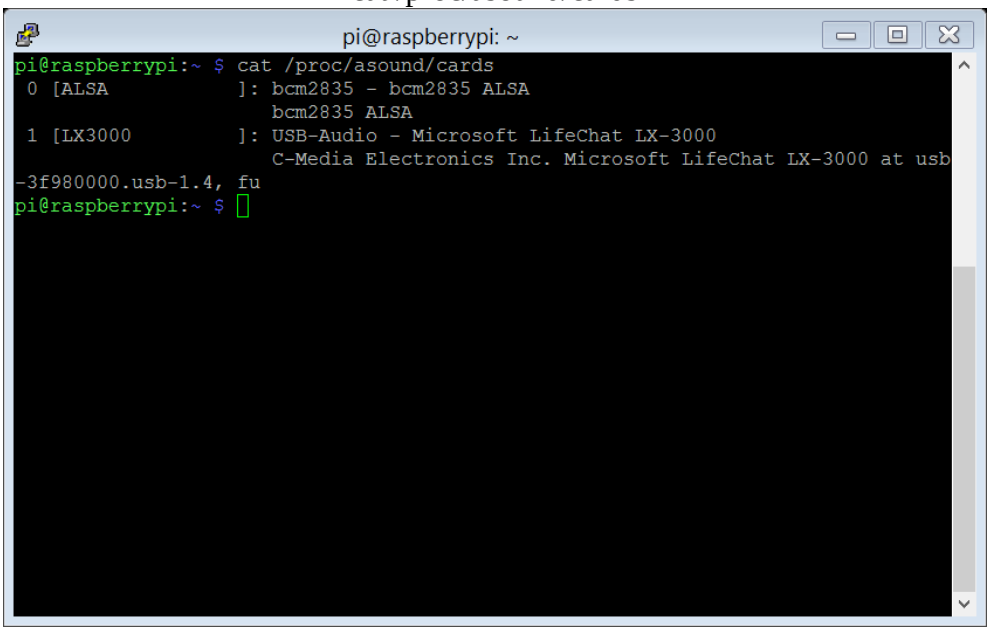

Fig. 4.2. Dispozitive Intrare/Iesire conectate la placa Raspberry PI 3

In figura de mai sus observam cum microfonul nostru este detectat si ii este atribuit id-ul 1, dupa modului ALSA integrat. Acest ID il vom folosi mai tarziu cand vom apela pocketsphinx-ul. Mai mult, trebuie sa cream un fisier cu calea ~/.asoundrc si vom adauga urmatoarele linii pentru a seta in mod implicit detectia microfonului.

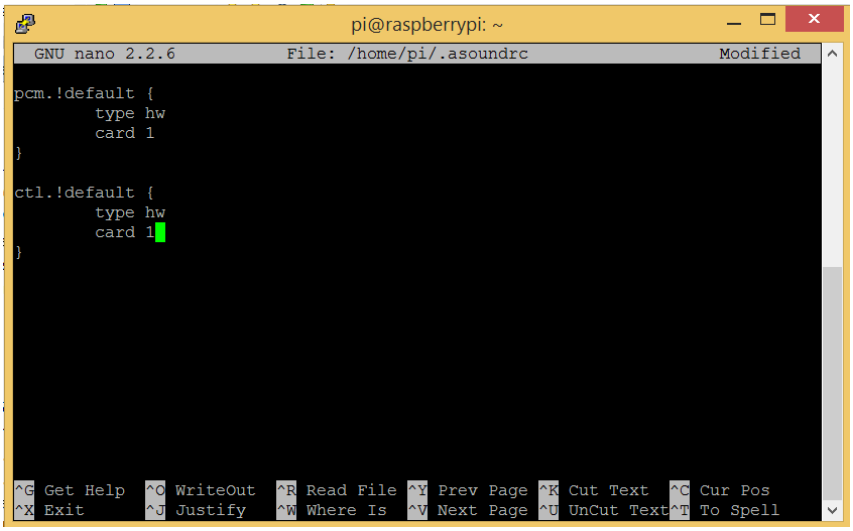

Fig. 4.3. Continutul fisierului ~/.asoundrc

Dupa ce ne-am asigurat ca lucrurile sunt in ordine, trebuie sa trecem la pasul final si anume instalarea pocketsphinx-ului. Inainte de a instala pocketsphinx-ul propriu-zis, trebuie sa ne asiguram ca dependintele acestuia sunt satisfacute. Acestea le vom realiza cu urmatoarele comenzi:

Sudo apt-get install bison

Sudo apt-get install libasound2-dev

Din dependintele necesare mai face parte si sphinxbase(http://sourceforge.net/projects/cmusphinx/files/sphinxbase/5prealpha) pe care il vom instala si configura cu urmatoarele comenzi:

```
./configure --enable-fixed

  make

  sudo make install
```

Asadar, in ultimul subpunct de la ultimul pas, va trebui sa descarcam, sa instalam si sa configuram pocketsphinx-ul(http://sourceforge.net/projects/cmusphinx/files/pocketsphinx/5prealpha). Dupa descarcare, vom executa urmatoarele comenzi necesare:

```
./configure
```

```
make

sudo make install

export LD_LIBRARY_PATH=/usr/local/lib

export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig
```

In acest moment, avem instalate toate resursele necesare pentru a rula pocketsphinx-ul. Vom verifica functionalitatea acestuia cu urmatoarea comanda:

Pocketsphinx_continuous –inmic yes –adcdev plughw:1

Fig. 4.4. Fisier de log pentru pocketsphinx_continuous

In imaginea de mai sus vedem ultimele doua linii din log-ul pocketsphinx-ului instalat corect.

Pentru a creste acuratetea programului, va trebui sa-i adaugam ca parametri un model de limba, model acustic si model fonetic pentru detectia de cifre din limba romana. Astfel, comanda finala va arata asa:

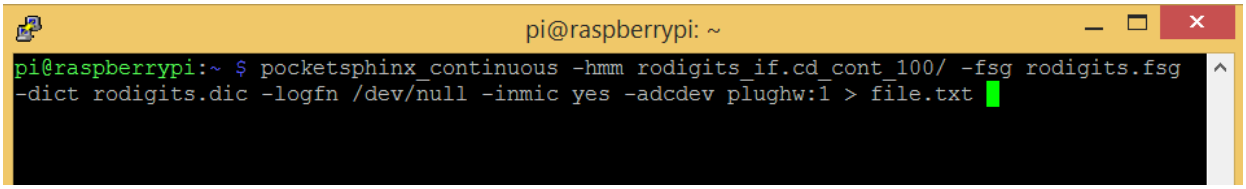

Fig. 4.5. Comanda pocketsphinx_continuous pentru detectie de cifre in limba romana

In plus fata de ceea ce am amintit mai sus, am adaugat si parametrul –logfn /dev/null care va exclude din log-ul afisat orice altceva in afara de cifrele detectate. De asemenea, log-ul de iesire va fi redirectat catre fisierul file.txt, fisier ce va fi folosit ca intrare pentru programul principal realizat in python de detectie si afisare a unui numar de telefon mobil din România.

4.3 Aplicația de detecție

Scopul aplicației este de a recepționa un semnal vorbit, verifică dacă acesta reprezintă un număr de telefon și în acest caz, să îl afișeze pe un ecran LCD de tip 16x2. În timpul rulării acestui program, utilitarul pocketsphinx va rula în background, scriind într-un fișier cuvintele detectate.

Aplicația parcurge acel fișier și la fiecare modificarea a sa, preia informația și verifică veridicitatea sa. Conexiunea low-level cu perifericele este asigurată de biblioteca Rpi.GPIO, astfel

încât să putem folosi displayul LCD cu placa Raspberry Pi. Acest lucru se realizează prin comanda Python:

```
import RPi.GPIO
```

Interfațarea cu informația extrasă de pocketsphinx se face prin intermediul unui dicționar python care realizează corespondența dintra cifrele reprezentate sub formă fonetică și cifrele reprezentate ca simboluri.

Fișierul în care scrie utilitarul pocketsphinx și din care citește aplicația curentă este specificat sub forma unui command line argument. Acesta este în mod constant parcurs de parser.

Pentru formatarea informației achiziționate de la pocketsphinx, se înlătură *newline*-ul și se face despărțirea liniilor de cod după *whitespace*.

```
line=f.readline()
line=line.rstrip('\n')
split_line=line.split(' ')
```

Validitatea intrării este verificată prin compararea dimensiunii vectorului *split_line* cu valoarea 10 (dimensiunea unui număr de telefon). De asemenea trebuie verificată condiția ca primul număr să fie 0 iar cel de-al doilea să fie 7, iar dacă această condiție este satisfăcută, se realizează un lookup în dicționarul dict{} pentru a se realiza transcrierea în simboluri a cifrei cu pricina.

```
if length == 10:
                if (split_line[0] == 'zero' and
split_line[1]=='\xc5\x9fapte'):
            nr_tel = ''
            for i in range(0,10):
                nr_tel += str(dict[split_line[i]])
                lcd_string(nr_tel,LCD_LINE_1,2)
```

De asemenea, trebuie tratat cazul în care numărul de telefon nu corespunde specificațiilor, și anume atunci când dimensiunea acestuia este diferită de 10. În acest caz, pe display-ul LCD-ului va fi afișat textul “Număr de telefon // Invalid”

Bibliografie

[1] Rabiner, L., Juang B., Fundamentals of speech recognition, Prentice-Hall International, 1993

[2] https://web.stanford.edu/~jurafsky/slp3/8.pdf

[3] Rabiner, L, A tutorial on hidden Markov models and selected applications in speech recognition
(1989)

[4] Khe Chai SIM, APPROXIMATED PARALLEL MODEL COMBINATION FOR EFFICIENT NOISE-ROBUST, SPEECH RECOGNITION

[5] NOISY SPEECH RECOGNITION USING NOISE REDUCTION METHOD BASED ON KALMAN FILTER M. Fujimoto and Y. Ariki

[6] Advanced Digital Signal Processing and Noise Reduction, Second Edition.Saeed V. Vasegh Copyright © 2000 John Wiley & Sons Ltd, capitolul XI

[7]http://kom.aau.dk/group/04gr742/pdf/MFCC_worksheet.pdf

[8]https://surveillance7.sciencesconf.org/conference/surveillance7/01_a_history_of_cepstrum_analysis_and_its_application_to_mechanical_problems.pdf

[9] Feature Extraction Methods LPC, PLP and MFCC In Speech Recognition Namrata Dave1

[10]B.P. Bogert, M.J.R. Healy, and J.W. Tukey, The Quefrency Alanysis ofTime series for Echoes: Cepstrum, Pseudo-Autocovariance, Cross-Cepstrum, and Saphe Cracking,in Proc.of the Symp. on Time Series Analysis, by M. Rosenblatt (Ed.),. Wiley, NY, (1963),pp. 209-243

[11] http://www.xavieranguera.com/tdp_2011/8-Cepstral-Analysis.pdf

[12] Rasta-PLP speech analysis, Hynek Hermansky, Nelson Morgan, Aruna Bayya, Phil Kohn, Decembrie 1991

[13] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," Acoustical Society of America Journal, vol. 87, pp.1738–1752, Apr. 1990.

[14]http://www.fceia.unr.edu.ar/prodivoz/Oppenheim_Schafer_2004.pdf

[15]M.Forsberg, ―Why Is Speech Recognition Difficult?‖, Chalmers University of Technology, Citeseer,(2003)

[16]Andrei Avram, „Cercetări asupra sonorității în limba română", 1961, p. 25
A Review on Automatic Speech Recognition Architecture and Approaches Karpagavalli

[17]M.A.Anusuya And S.K.Katti, ―Speech Recognition by Machine:A Review‖, International Journal of Computer Science and Information Security, vol. 6, no. 3,(2009),pp. 181 -205.

[18] S.L. Dr. Ing. H. Cucu http://speed.pub.ro/speed3/wp-content/uploads/2014/02/Indrumar-de-proiect-PCDTV-v11.pdf