Proiect de cercetare-dezvoltare în Tehnologia Vorbirii

- îndrumar de proiect -

Ş.l. Dr. Ing. Horia Cucu

CUPRINS

1. ARHITECTURA GENERALĂ A UNUI SISTEM DE RECUNOAŞTERE AUTOMATĂ A VORBIRII

1.1 Introducere în recunoaşterea automată a vorbirii

Procesul recunoaşterii automate a vorbirii (RAV) vizează transformarea unui semnal audio ce conține vorbire într-o succesiune de cuvinte. Înțelegerea automată a vorbirii extinde scopul RAV (generarea unei secvențe de cuvinte) încercând să producă informații de natură semantică privind înțelesul propoziției generate de sistemul de RAV. În cazul în care semnalul acustic de intrare conține vorbire provenind de la mai mulți vorbitori, procesul de RAV poate fi privit ca având două etape: diarizare (etapă ce încearcă să răspundă la întrebarea: "cine şi când a vorbit?") şi transcriere (etapă ce încearcă să răspundă la întrebarea: "ce a spus?").

Recunoaşterea automată a vorbirii are o gamă largă de aplicabilitate. Domeniul cel mai important pare a fi acela al interfețelor hands-free şi eyes-free. Există multe aplicaţii în care utilizatorii au nevoie să-şi folosească mâinile şi ochii pentru altceva, iar vorbirea rămâne singura lor alternativă de a fi eficient în dialogul cu calculatorul. Mai mult decât atât, vorbirea este cel mai natural mijloc de comunicare pentru fiinţele umane. Alte domenii importante în care RAV se aplică cu succes sunt sistemele de dialog pentru call-centere şi sistemele de traducere speech-to-speech. Traducerea speech-to-speech este în acest moment un subiect foarte „la modă" în mai multe centre de cercetare academice şi industriale. Nu în ultimul rând, RAV se poate utiliza şi pentru dictare: transcrierea unui monolog al unui vorbitor anume. Transcrierea unui astfel de monolog este utilă în diverse domenii, de la cea mai simpla utilizare pentru editarea documentelor de către jurnalişti, scriitori, etc., până la utilizarea pentru transcrierea discuțiilor din timpul proceselor de judecată. Fiecare dintre aceste aplicaţii este de obicei mult mai restrictivă decât problema generală care impune transcrierea automată a vorbirii naturale, continue, provenind de la un vorbitor necunoscut, în orice mediu (zgomotos sau nu). Diferitele surse de variabilitate în vorbire, care vor fi discutate în continuare, fac sarcina generală de RAV una foarte dificilă. Cu toate acestea, în multe situaţii practice, variabilitatea este restricţionată. De exemplu, vorbitorul ar putea fi cunoscut (în loc un număr oarecare de vorbitori necunoscuți sistemului) sau vorbirea ar putea fi dictată (în loc de vorbire spontană) sau mediul de înregistrare ar putea ne-zgomotos şi ne-reverberant. În recunoaşterea automată a vorbirii se face o distincţie între modulul care

abordează variabilitatea acustică (modelul acustică) şi modulul care tratează incertitudinea lingvistică (modelul de limbă).

Unul dintre cei mai importanţi factori de care depinde dificultatea procesului de transcriere este sarcina de recunoaştere. Aceasta include specificitatea limbii, numărul de cuvinte ce pot fi pronunţate şi incertitudinea lingvistică a sarcinii de recunoaştere. Diversele limbi vorbite prezintă provocări diferite pentru un sistem de recunoaştere a vorbirii. Pentru un număr covârşitor de mare de limbi nu există suficiente resurse acustice (baze de date de vorbire) şi lingvistice (corpusuri de text). Aceste aşa-numite limbi slab dotate sunt vorbite de un număr mare de oameni, însă până acum s-au achiziţionat prea puţine resurse acustice şi lingvistice pentru a putea dezvolta un sistem de RAV. În consecinţă, proiectarea şi crearea unui sistem de RAV trebuie să prevadă şi colectarea acestor resurse.

Alte limbi "suferă" de o morfologie foarte complexă. De exemplu limbile cu morfologie complexă, cum ar fi Franceza sau Româna, au vocabulare de dimensiuni mai mari decât limbile cu morfologie simplă, cum ar fi Engleza. Timpul prezent al verbului *a merge* are cinci forme diferite în limba Română: *merg, mergi, merge, mergem* şi *mergeţi*, şi şase forme diferite în Franceză: *vais, vas, va, allons, allez, vont*. În Engleză, acelaşi verb, la acelaşi timp, are doar două forme morfologice diferite: *go* şi *goes*. Limbile Germană şi Turcă sunt două dintre aşa-numitele limbi aglutinative. În aceste limbi se pot forma cuvinte noi prin concatenarea altor cuvinte sau morfeme. Astfel dimensiunea vocabularului de cuvinte pentru o limbă aglutinativă este mult mai mare, acest lucru complicând sarcina de recunoaştere automată a vorbirii.

Dimensiunea vocabularului este de asemenea un factor important care influenţează dificultatea sarcinii de RAV. Este evident faptul că o sarcină de recunoaştere de cifre (cu un vocabular de 10 cuvinte) este mult mai simplă decât, de exemplu, recunoaşterea vorbirii spontane (cu un vocabular de 64.000 de cuvinte). Cu toate acestea, dimensiunea mare a vocabularului nu înseamnă neapărat că sarcina de recunoaştere va fi mai dificilă. Incertitudinea lingvistică a potenţialelor discursuri ce trebuie recunoscute constituie de asemenea un factor important. De exemplu, o sarcină de RAV specifice turismului (cu un vocabular de 64 de mii de cuvinte conţinând în general nume de locuri, hoteluri, restaurante, etc.) nu este atât de dificilă ca o sarcină de recunoaştere a vorbirii spontane (cu un vocabular de de aceeaşi dimensiune). Incertitudinea lingvistică (perplexitatea) scăzută a primeia o face mult mai simplă.

Procentul de cuvinte incorecte obţinut pentru diverse sarcini de recunoaştere este prezentat în Tabelul 1.1. Datele se referă la sisteme state-of-the-art de RAV pentru limba Engleză [Jurafsky, 2009]. Rata de cuvinte eronate (word error rate - WER) este criteriul de performanţă standard utilizat în evaluarea sistemelor de RAV. Pentru mai multe informaţii despre acest criteriu de performanţă consultaţi platforma „Evaluarea unui sistem de recunoastere a vorbirii continue".

Tabelul 1.1. Rezultate WER raportate în anul 2005 pentru diverse sarcini de RAV [Jurafsky, 2009]

Sarcina de RAV	Dimensiunea vocabularului	WER [%]
TI Digits	11 cuvinte	0.55
Wall Street Journal read speech	5000 cuvinte	3.0
Wall Street Journal read speech	20000 cuvinte	<6.6
Broadcast News	64000+ cuvinte	9.9
Conversational Telephone Speech	64000+ cuvinte	20.7

Un alt factor important care influenţează dificultatea procesului de RAV este stilul vorbirii. Stilul vorbirii se referă la cât de fluentă, naturală sau conversaţională este vorbirea ce trebuie recunoscută. În mod evident, recunoaşterea de cuvinte izolate, unde cuvintele sunt separate de pauze de linişte, este mult mai simplă decât recunoaşterea vorbirii continue, unde cuvintele sunt pronunțate legat şi vorbirea trebuie segmentată înainte de recunoaştere. De fapt, primele sisteme de recunoaştere automată a vorbirii puteau rezolva problema localizării granițelor între cuvinte numai dacă vorbitorul făcea pauze generoase în pronunțarea lor: Dragon Dictation [Baker, 1989] este un bun exemplu de un sistem de recunoaştere de cuvinte izolate cu vocabular extins.

Sarcinile de recunoaştere a vorbirii continue pot fi mai departe clasificate în funcție dificultatea lor. De exemplu, sarcina de recunoaştere a vorbirii citite este mult mai simplă decât sarcina de recunoaştere a vorbirii conversaţionale sau spontane. Variabilitatea acustică mai mare face ca sarcina urmă să fie mai dificilă. Această diferenţă în dificultate între sarcinile de vorbire continue este reflectată în ratele crescute de eroare pentru recunoaşterea vorbirii spontane, comparativ cu recunoaşterea vorbirii citite (a se vedea Tabelul 1.1).

Dificultatea şi, în consecinţă, acurateţea procesului de recunoaştere a vorbirii este, de asemenea, influenţată de mediul acustic în care este înregistrată vorbirea şi de canalul de transmisie. În afara studiourilor de înregistrare sau laboratoarelor de cercetare, există, de obicei, mai multe surse acustice, inclusiv alți vorbitori, zgomot de fundal, etc.. În cele mai multe cazuri separarea semnalele acustice diferite este o problemă foarte dificilă. Microfonul folosit pentru înregistrare are, de asemenea, un impact semnificativ asupra acurateţii recunoaşterii vorbirii.

Sistemele comerciale de dictare şi majoritatea experimentelor de cercetare în domeniul recunoaşterii vorbirii sunt realizate cu microfoane de înaltă calitate. Alte tipuri de microfoane vin cu dezavantaje diferite, care contribuie la calitatea sistemului de RAV. Variaţiile în canalul de transmisie apar din cauza mişcărilor capului vorbitorului relativ la microfon şi din cauza transmisiei printr-o reţea de telefonie sau internet. Cea mai mare diferență dintre acurateţea sistemelor de RAV comparativ cu acurateţea cu care omul recunoaştere vorbirea apare în situaţia în care semnalul acustic este poluat cu zgomot aditiv

mare, sau provine de la mai multe surse acustice, sau este înregistrat într-un mediu reverberant. Recunoaşterea semnalului de vorbire zgomotos, cu un raport semnal-zgomot scăzut, poate avea rezultate de 2 până la 4 ori mai slabe, comparativ cu recunoşterea semnalului de vorbire nezgomotos.

În cele din urmă, caracteristicile vorbitorilor au, de asemenea, un impact semnificativ asupra preciziei unui sistem de recunoaştere a vorbirii. Variabilitatea acestor caracteristici include accentul vorbitorului, limba sau dialectul folosit, faptul că vorbitorul este nativ sau nu (dacă limba utilizată limba lui mamă), rapiditatea pronunției, vârsta vorbitorului, şi, desigur, diferenţele anatomice şi fiziologice care influențează producerea vorbirii. Mai mult decât atât, vorbitori diferiți prezintă grade diferite de variabilitate intrinsecă, bazate pe starea emoţională, probleme temporare de sanatate, etc. Variabilitatea inter-vorbitori poate fi soluționată prin proiectarea de sisteme de RAV dependente de vorbitor (specifice fiecărui vorbitor). Dezavantajul aici este faptul că pentru fiecare vorbitor nou treabuie antrenat un nou model acustic. Acest lucru conduce la un sistem mai complex şi ridică mai multe probleme de antrenabilitate (date insuficiente de antrenare pentru fiecare vorbitor şi altele). Pe de altă parte, sistemele de RAV independente de vorbitor sunt mai simple şi mai flexibile (acestea pot fi folosite pentru a recunoaşte vorbirea oricărui vorbitor). Cu toate acestea, un sistem independent de vorbitor este mai puţin performant pentru un anumit vorbitor în comparaţie cu un sistem dependent de vorbitor creat special pentru respectivul vorbitor (în cazul în care există suficiente date de antrenare pentru respectivul vorbitor). Cu toate că algoritmii de adaptare la vorbitor au progresat mult în ultimii 15 de ani, adaptabilitatea şi robusteţea sistemelor de RAV pentru recunoaşterea diverselor voci este încă foarte limitată în comparaţie cu performanţa umană.

Paradigma state-of-the-art în domeniul recunoaşterii vorbirii continue cu vocabular extins este modelul Markov ascuns (Hidden Markov Model - HMM). Infrastructura HMM a fost introdusă ca un candidat viabil pentru partea de modelare acustică a recunoaşterii vorbirii în 1975 [Baker, 1975]. În particular pentru sistemele de RAV cu vocabular extins, modelul acustic bazat pe HMM este utilizat împreună cu un model de tip n-gram, care este responsabil pentru partea de modelare limbă. Modelele lingvistice statistice (n-gram) au devenit soluţia stat-of-the-art pentru modelarea limbajului în primul rând datorită extinderii extraordinare a Internetului, care a furnizat date suficiente pentru antrenarea corespunzătoare a acestor sisteme.

1.2 Formalismul recunoaşterii automate a vorbirii

Procesul recunoaşterii automate a vorbirii (RAV) vizează transformarea unui semnal audio ce conține vorbire într-o succesiune de cuvinte. Recunoaşterea automată a vorbirii este unul dintre primele domenii în care tehnicile de modelare statistică (în care modelele se crează pe baza unor catințăți mari de date) s-au impus ca şi standard. Platforma statistică

pentru recunoaşterea automată a vorbirii a fost creată şi dezvoltată de-a lungul a două decenii de către Baker [Baker, 1975], o echipă de cercetători de la compania IBM [Jelinek, 1976; Bahl, 1983] şi o echipă de cercetare-dezvoltare de la compania AT&T [Levinson, 1983; Rabiner, 1989].

Procesul de transformare a vorbirii în text poate fi formulat într-o manieră statistică astfel: *Care este cea mai probabilă secvență de cuvinte W* în limba L, dat fiind mesajul vorbit X?*

Reprezentarea formală utilizează funcția argmax, care selectează argumentul ce maximizează probabilitatea secvenței de cuvinte:

$$W^* = \arg\max_W p(W \mid X) = \arg\max_W \frac{p(X \mid W)p(W)}{p(X)} = \arg\max_W p(X \mid W)p(W) \qquad (1.1)$$

Dezvoltarea din Ecuația 1.1 are la bază regula Bayes şi s-a făcut ținând cont de faptul că probabilitatea mesajului vorbit *p(X)* este independentă de secvența de cuvinte *W*. Ultimul rezultat evidențiază doi factori care pot fi estimați direct. Problema inițială (găsirea secvenței de cuvinte pe baza mesajului vorbit) a fost împărțită în două sub-probleme mai simple: a) estimarea probabilității apriori a secvenței de cuvinte *p(W)* şi b) estimarea probabilității mesajului vorbit dată fiind secvența de cuvinte pronunțată *p(X|W)*. Primul factor poate fi estimat utilizând exclusiv un *model de limbă*, iar cel de-al doilea poate fi estimat cu ajutorul unui *model acustic*. Cele două modele pot fi construite independent aşa cum se va vedea în secțiunea următoare, dar vor fi folosite împreună pentru a decoda un mesaj vorbit, aşa cum arată Ecuația 1.1.

1.3 Arhitectura unui sistem de recunoaştere automată a vorbirii

Arhitectura generală a unui sistem de recunoaştere automată a vorbirii (RAV) este prezentată în Figura 1.1. Din figură reies două lucruri esențiale în ceea ce priveşte procesul de recunoaştere a vorbirii: a) recunoaşterea se face utilizând o serie de parametri vocali extraşi din semnalul vocal (nu direct, utilizând mesajul vorbit) şi b) recunoaşterea se face pe baza unor modele (acustic, fonetic şi lingvistic) dezvoltate în prealabil.

Modelul acustic are rolul de a estima probabilitatea unui mesaj vorbit, dată fiind o succesiune de cuvinte. În sistemele de RVC actuale modelul acustic nu foloseşte cuvinte ca unități acustice de bază pentru că: a) fiecare sarcină de RAV are un vocabular de cuvinte diferit pentru care nu există modele deja antrenate şi nici date de antrenare disponibile şi b) numărul de cuvinte diferite dintr-o limbă este prea mare. În locul cuvintelor, se utilizează unități acustice de bază sub-lexicale, de exemplu foneme, sau, mai recent, chiar unități sub-fonemice numite senone. Prin urmare, modelul acustic este format dintr-un set de modele

pentru foneme (sau senone) care se conectează în timpul procesului de decodare pentru a forma modele pentru cuvinte şi apoi modele pentru succesiuni de cuvinte. Acestea sunt utilizate în final pentru a estima probabilitatea ca mesajul rostit de vorbitor să fie format dintr-o succesiune sau alta de cuvinte. Experiența a arătat că această abordare generativă pentru modelele acustice poate fi foarte bine implementată utilizând modele Markov ascunse (hidden Markov models - HMM). Pentru mai multe informații despre modul cum funcționează modelele acustice consultați platforma „Principiile de bază ale modelării acustice".

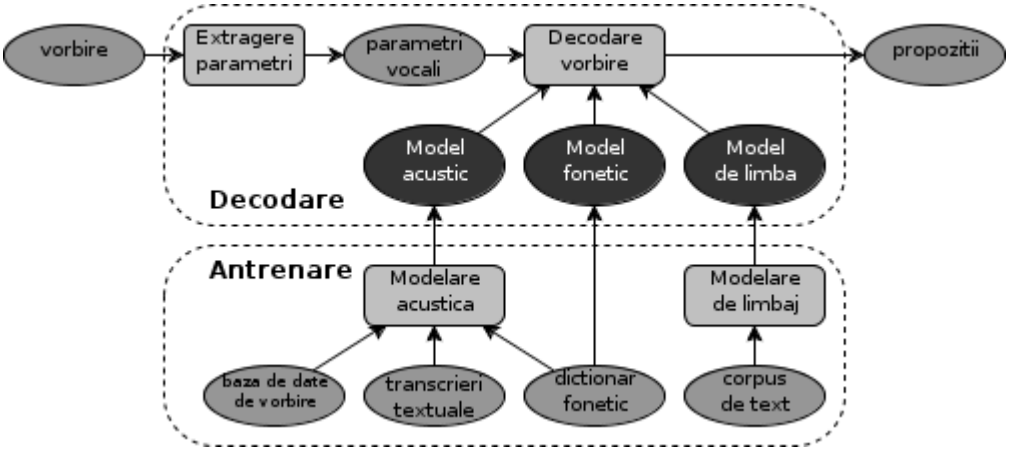


Figura 1.1. Arhitectura generală a unui sistem de recunoaştere a vorbirii

Modelul de limbă este utilizat în timpul decodării pentru a estima probabilitățile tuturor secvențelor de cuvinte din spațiul de căutare. În general, rolul unui model de limbă este de a estima probabilitatea ca o secvență de cuvinte $W = w_1, w_2, ..., w_n$, să fie o propoziție validă a limbii. Probabilitatea acestor secvențe de cuvinte ajută foarte mult modelul acustic în procesul de decizie. De exemplu, aceste două fraze: *ceapa roşie este sănătoasă* şi *ce apar oşti ie este sănătoasă* sunt similare din punct de vedere acustic, însă cea de-a doua nu are niciun sens. Rolul modelului de limbă este acela de a asigna o probabilitate mult mai mare primei secvențe de cuvinte şi, prin urmare, de a ajuta sistemul de RAV să decidă în favoarea acesteia.

Modelul fonetic are rolul de a conecta modelul acustic (care estimează probabilitățile acustice ale *fonemelor*) cu modelul de limbă (care estimează probabilitățile secvențelor de *cuvinte*). De cele mai multe ori modelul fonetic este un dicționar de pronunție care asociază fiecărui cuvânt din vocabular una sau mai multe secvențe de foneme adecvate, reprezentând modul în care se poate pronunța respectivul cuvânt.

Figura 1.1 ilustrează, de asemenea, procesele implicate în dezvoltarea unui sistem de RAV, dar şi resursele necesare creării modelelor acustice, lingvistice şi fonetice. Modelul acustic se construieşte strict pe baza unui set de clipuri audio înregistrate asociate cu transcrierea textuală a mesajelor vorbite şi a unui dicționar fonetic ce cuprinde toate cuvintele din respectiva transcriere textuală. În cazul sistemelor de RVC cu vocabular extins se

folosesc modele de limbă statistice, care se construiesc utilizând corpusuri de text de dimensiuni cât mai mari şi cât mai adaptate domeniului din care fac parte mesajele vorbite ce trebuie decodate. În sistemele de RVC cu vocabular redus se folosesc preponderent modele de limbă de tip gramatică cu reguli, iar pentru construcția acestora nu este nevoie de resurse textuale.

2. RESURSE FONETICE, ACUSTICE ŞI LINGVISTICE NECESARE ÎN CONSTRUCȚIA UNUI SISTEM DE RECUNOAŞTERE A VORBIRII CONTINUE

2.1 Generalități

Sistemele actuale de recunoaştere a vorbirii continue (RVC) transformă semnalul vocal în text pe baza unor modele (acustic, fonetic şi lingvistic) dezvoltate în prealabil. Figura 2.1 ilustrează arhitectura generală a sistemelor de recunoaştere a vorbirii continue punând accent pe resursele necesare antrenării celor trei modele.

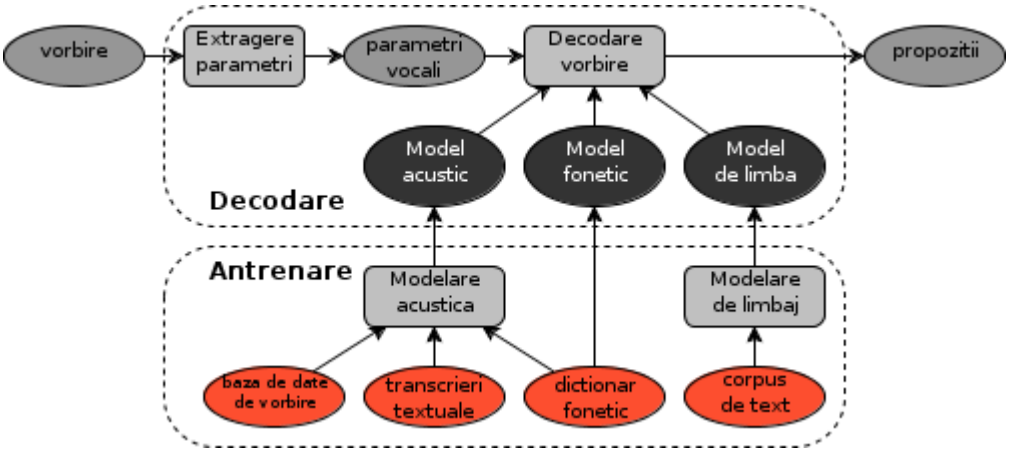


Figura 2.1. Resursele necesare construcției unui sistem de RVC

Aşa cum arată Figura 2.1, modelul acustic, care modelează de obicei fonemele limbii, se contruieşte pe baza unui set de clipuri audio înregistrate şi a transcrierilor textuale aferente. De asemenea, pentru antrenarea acestui model este necesară şi existența unui dicționar fonetic care să specifice modul de pronunție al cuvintelor din transcrierile textuale.

Dicționarul fonetic stă şi la baza dezvoltării modelului fonetic necesar în procesul de decodare. Dacă dicționarul fonetic cuprinde mai multe pronunții pentru acelaşi cuvânt, în cadrul modelului fonetic aceste pronunții sunt comasate formând un model de pronunție pentru respectivul cuvânt.

În cele din urmă, modelul de limbă necesar unui sistem de recunoaştere a vorbirii continue se construieşte utilizând un corpus de text de mari dimensiuni. Din această resursă se extrag statisticile caracteristice limbii, iar acestea se vor utiliza în procesul de decodare pentru a atribui probabilități diverselor cuvinte şi secvențe de cuvinte propuse de modelul acustic.

2.2 Elemente de fonetică a limbii române

În sistemele actuale de recunoaştere a vorbirii continue modelul acustic nu modelează acustica cuvintelor limbii în mod direct, ci indirect, folosind unități acustice sub-lexicale, numite foneme sau chiar unități acustice sub-fonemice.

Fonemul este unitatea de sunet fundamentală din limbile vorbite care ajută la diferenţierea cuvintelor şi morfemelor. Prin modificarea unui fonem al unui cuvânt, se generează fie un cuvânt inexistent, dar perceput ca diferit de către vorbitorii limbii, fie un cuvînt cu alt sens.

În limba română se folosesc 7 vocale de bază şi două vocale împrumutate, 4 semi-vocale şi 22 de consoane, conform reprezentării din Tabelul 2.1.

2.3 Dicționarul fonetic

Importanța dicționarului fonetic într-un sistem de recunoaştere automată a vorbirii este evidențiată în Figura 2.1. Se poate observa că dicționarul fonetic este utilizat în partea de dezvoltare a unui sistem de recunoaştere, mai precis la antrenarea modelului acustic şi modelului fonetic.

Un dicționar fonetic este un instrument lingvistic care specifică modul în care se pronunță cuvintele unei limbi. Altfel spus, dicționarul fonetic face corespondența între forma scrisă şi forma fonetică a cuvintelor unei limbi. Forma fonetică a unui cuvânt este o succesiune de foneme. Într-un sistem de recunoaştere a vorbirii continue, dicționarul fonetic are rolul de a face legătura între modelul acustic (care modelează modul de producere a *sunetelor specifice limbii*) şi modelul de limbă (care modelează modul în care se succed *cuvintele limbii*). În consecință, dicționarul fonetic trebuie să conțină toate cuvintele posibile pentru o anumită sarcină de recunoaştere a vorbirii alături, bineînțeles, de transcrierile fonetice ale acestor cuvinte.

Tabelul 2.1. Fonemele limbii române

Fonemul			**Exemple de cuvinte**	
Tip	**Simbol IPA**	**Simbol intern**	**Forma scrisă**	**Forma fonetică**
vocale	a	a	s**a**t	s a t
	e	e	mar**e**	m a r e
	i	i	l**i**ft	l i f t
	o	o	l**o**c	l o c
	u	u	ş**u**t	s1 u t
	ə	a1	gur**ă**	g u r a1
	ɨ	i2	**î**ntre	i2 n t r e
vocale împrumutate	y	y	ecr**u**	e c r y
	ø	o2	bl**eu**	b l o2
semivocale	e̯	e1	d**e**al	d e1 a l
	j	i3	f**i**ară	f i3 a r a1
	o̯	o1	**o**ase	o1 a s e
	w	w	sa**u**	s a w
consoane	c	k2	**ch**em	k2 e m
	b	b	**b**ar	b a r
	p	p	**p**ar	p a r
	k	k	a**c**um	a k u m
	ʧ	k1	**c**enuşă	k1 e n u s1 a1
	g	g	**g**alben	g a l b e n
	ʤ	g1	**g**irafă	g1 i r a f a1
	ɟ	g2	un**gh**i	u n g2
	d	d	**d**ar	d a r
	t	t	**t**ot	t o t
	f	f	**f**aţa	f a t1 a
	v	v	**v**apor	v a p o r
	h	h	**h**arta	h a r t a
	ʒ	j	a**j**utor	a j u t o r
	ʃ	s1	co**ş**	k o s1
	l	l	**l**ac	l a c
	m	m	**m**ăr	m a1 r
	n	n	**n**as	n a s
	s	s	**s**are	s a r e
	z	z	**z**ar	z a r
	r	r	**r**isc	r i s k
	ʦ	t1	**ţ**ăran	t1 a1 r a n
consoană palatalizată	ʲ	i1	tar**i**	t a r i1

Procesul de construcție al unui dicționar fonetic presupune fonetizarea (crearea formei fonetice) cuvintelor vocabularului sarcinii de recunoaştere a vorbirii. Regulile de fonetizare ale cuvintelor limbii române au fost studiate şi explicate destul de puțin în literatura de specialitate. De aceea, orice proces de contrucție al unui dicționar fonetic trebuie să înceapă cu stabilirea unor reguli de fonetizare şi excepții de la aceste reguli.

Construcția dicționarului fonetic se poate realiza manual, semi-automat sau chiar automat. În cazul în care vocabularul sarcinii de RVC este redus (câteva zeci-sute de cuvinte), crearea dicționarului fonetic se poate realiza manual într-un timp suficient de scurt. Dacă însă sarcina de RVC necesită utilizarea unui vocabular extins (mii sau chiar zeci de mii de cuvinte), atunci fonetizarea manuală a acestui vocabular nu mai este o opțiune.

Din fericire pentru limba română corespondența între forma scrisă şi pronunția cuvintelor este de cele mai multe ori bijectivă. În general fiecărei litere îi corespunde întotdeauna acelaşi fonem. Există doar câteva clase de ambiguitate (vezi Tabelul 2.2) în care aceeaşi literă se pronunță diferit în funcție de contextul în care apare.

Tabelul 2.2. Clasele de ambiguitate în fonetizarea cuvintelor limbii române

Litera	Fonemul	Exemple
e	e	c**e**r, r**e**st, r**ec**e
	e1	d**e**al, t**e**-am, c**e**ainic, gh**e**orghe
	i3 e	**e**u, **e**l, **e**ram, **e**rai
i	i	f**i**r, c**i**t**i**t, pr**i**ns
	i1	tar**i**, rup**i**, dec**i**
	i3	**i**arbă, do**i**, f**i**er
o	o	g**o**l, p**o**trivit
	o1	v**o**al, s**o**are
u	u	s**u**d, f**u**rt**u**n
	w	băcă**u**an, no**u**ă
c	k	**c**asă, sa**c**
	k1	**c**er, **c**irc
	k2	**ch**iar, o**ch**elari
g	g	**g**ară, olo**g**
	g1	**g**irafă, **g**eor**g**e
	g2	un**gh**i, **gh**em
h	h	**h**artă, ra**h**at
	mut	oc**h**i, ung**h**i
x	k s	pi**x**, fi**x**
	g z	e**x**emplu, e**x**amen

Odată ce setul de reguli a fost stabilit, fonetizarea se poate realiza şi automat, utilizând aplicații software care implementează setul de reguli de fonetizare stabilit a priori. Chiar dacă la prima vedere ar părea simplă, implementarea acestor reguli nu este deloc trivială. Pentru rezolvarea celor mai des întâlnite clase de ambiguități (de exemplu ambiguitatea vocală/semivocală) este nevoie de o informație suplimentară: modul de despărțire al

cuvintelor în silabe. Fonetizarea manuală se bazează pe faptul că expertul uman cunoaşte aceste reguli de despărțire în silabe, însă, în cazul în care se doreşte automatizarea acestui proces, regulile de despărțire trebuie şi ele formalizate.

O alternativă la aplicațiile de fonetizare automată pe bază de reguli sunt aplicațiile de învățare automată statistică. Utilizând un dicționar fonetic creat manual se poate antrena un model statistic de fonetizare, care poate fi utilizat în continuare pentru fonetizarea unor vocabulare mult mai extinse.

2.4 Baza de date de vorbire

Importanța bazei de date de vorbire într-un sistem de recunoaştere automată a vorbirii (RAV) este evidențiată în Figura 2.1. Se poate observa că această bază de date este utilizată în partea de dezvoltare a unui sistem de recunoaştere, mai precis la antrenarea modelului acustic. Baza de date de vorbire este utilizată şi în procesul de evaluare a unui sistem de recunoaştere de vorbire, proces ce nu este ilustrat în Figura 2.1.

O bază de date de vorbire completă cuprinde trei componente:

- un set de clipuri audio ce conțin vorbire;
- un set de fişiere text ce conțin textul ce a fost pronunțat în clipurile audio şi eventual marcaje temporale pentru fiecare cuvânt/fonem;
- informații suplimentare privind stilul şi domeniul vorbirii, identitatea vorbitorilui, etc.

În majoritatea cazurilor clipurile audio ce conțin vorbirea sunt eşantionate cu frecvența de 16kHz, dimensiunea eşantioanelor fiind de 16 biți, iar formatul fişierelor mswav.

Baza de date de vorbire utilizată în antrenarea unui sistem de RAV este un element cheie care afectează în mod direct performanța sistemului: rata de eroare şi viteza de recunoaştere. Cele mai importante caracteristici ale bazei de date de vorbire (cele care determină în mod direct calitatea acesteia) sunt:

- dimensiunea bazei de date (numărul de ore de vorbire, numărul de vorbitori),
- stilul vorbirii (cuvinte izolate, vorbire continuă citită, vorbire conversațională),
- variabilitatea (calitatea înregistrărilor, zgomotul de fundal, variabilitatea vorbitorilor)

Datorită faptului că modelarea acustică este abordată statistic în sistemele RAV moderne, dimensiunea bazei de date de antrenare este foarte importantă. Tutorialul de antrenare acustică CMU Sphinx sugerează nişte valori numerice concrete în ceea ce priveşte dimensiunea unei bazei de date de vorbire (Tabelul 2.3). Sarcina de comandă şi control este

o sarcină tipică de recunoaştere a vorbirii pseudo-continuă cu vocabular redus (RVC-VR), în timp ce sarcina de dictare este una tipică de recunoaştere a vorbirii continue cu vocabular extins (RVC-VE). De reținut este faptul că dezideratul independenței de vorbitor pare foarte dificil de atins, deoarece pentru aceasta este nevoie de material vorbit de la aproximativ 200 de vorbitori. Acest număr ar putea părea exagerat, dar variabilitatea vorbitorului este într-adevăr un factor important şi poate fi depăşită doar prin modelarea diferitelor pronunții posibile ale fonemelor. Acest lucru poate să fie realizat prin înregistrarea mai multor vorbitori.

Tabelul 2.3. Dimensiuni ale bazelor de date de vorbire sugerate de CMU Sphinx

Sarcina RAV	Sistem dependent de vobitor	Sistem independent de vorbitor
comandă şi control (RVC-VR)	1 oră de înregistrări, 1 vorbitor	5 ore de înregistrări, 200 de vorbitori
dictare (RVC-VE)	10 ore de înregistrări, 1 vorbitor	50 ore de înregistrări, 200 de vorbitori

Achiziționarea unei baze de date de vorbire se poate face în două moduri: a) înregistrarea unor texte predefinite sau b) etichetarea unor materiale audio ce contin vorbire. În funcție de aplicația ce urmează a fi dezvoltată, vor prima avantajele sau dezavantajele uneia dintre cele doua metode de achiziție.s

2.4.1 Înregistrarea unor texte predefinite

Înregistrarea unor texte predefinite implică alegerea locului de înregistrare (studio, laborator, etc.), alegerea microfonului, alegerea stației de înregistrare (placa audio folosită pentru achiziție, amplitudinea semnalului, etc.) şi, în sfârşit alegerea textelor ce urmează a fi înregistrate. Alegerea textelor este un proces foarte important de care depinde timpul şi costul achiziției bazei de date, dar şi corectitudinea ei (corespondența între texte şi materialul înregistrat trebuie să fie perfectă). De aceea, în alegerea textelor trebuie să se țină cont de următoarele aspecte:

- frazele ar trebui să fie corecte din punct de vedere lexical şi gramatical. În afara greşelilor gramaticale trebuie evitate şi schimbările de topică a propoziției, schimbări ce pot pune vorbitorul în dificultate;
- frazele ar trebui să fie scurte (maxim 15-20 de cuvinte, în funcție de lungimea cuvintelor) pentru a putea fi pronunțate fără ezitări sau respirații;
- frazele ar trebui să conțină numai cuvinte uzuale în limbajul de zi cu zi, pentru a putea fi înțelese şi pronunțate uşor şi fără ezitări;
- frazele ar trebui să conectate semantic pentru ca vorbitorul să înțeleagă cât mai repede sensul frazei şi astfel să poată obține o fluență în pronunție (trebuie evitate schimbările de subiect);

- frazele nu ar trebui să conţină cuvinte împrumutate din limbi străine (de obicei ambigue din punct de vedere al pronunţiei) sau interjecţii (de obicei pronunţate foarte diferit de fiecare vorbitor în parte);
- frazele ar trebui să conţină toate semnele de punctuatie (pe care vorbitorul le va utiliza pentru a citi cu intonatie);
- frazele nu ar trebui să conţină numere scrise cu cifre (care pot fi ambigue din punct de vedere al pronunţiei).

În cazul acestei prime metode de achiziţie cele mai importante avantaje sunt următoarele:

- proiectanţii deţin controlul asupra materialelor înregistrate, asupra calităţii înregistrărilor şi asupra variabilităţii vorbitorilor;
- achiziţia se poate face într-un timp relativ scurt, fără ca cei implicaţi în achiziţie să fie antrenaţi special în acest sens;
- erorile de pronunţie sunt de regulă rare, asftfel că baza de date achiziţionată în acest mod poate fi presupusă ca fiind corectă.

Totuşi, un dezavantaj important este faptul că, utilizând această metodă de achiziţie, nu se pot obţine baze de date de vorbire spontană, conversaţională. Vorbitorul ştie a priori textul pe care îl va pronunţa şi este mult mai concentrat să îl pronunţe clar şi corect, fără ezitările, bâlbele sau lipsa de fluenţă caracteristice vorbirii conversaţionale.

2.4.2 Etichetarea unor materiale audio ce conţin vorbire

Etichetarea unor materiale audio ce conţin vorbire implică obţinerea materialelor audio, aducerea lor la un format comun (aceeaşi frecvenţă de eşantionare, aceeaşi dimensiune a eşantioanelor, etc.), selectarea părţilor ce conţin vorbire, fragmentarea materialelor audio şi, în final, etichetarea lor.

Avantajul cel mai important al acestei metode de achiziţie este posibilitatea obţinerii unei baze de date de vorbire spontană, în care vorbesc diverşi vorbitori, în diverse contexte, având diverse stări de spirit, emoţii, etc. Preţul plătit pentru a obţine o astfel de bază de date de vorbire este unul destul de mare, el fiind direct influenţat de dificultatea procesului de achiziţie, procesare şi etichetare pentru aceste clipuri audio. Timpul necesar etichetării unei ore de vorbire spontană este de cel putin 3-4 ori mai mare decât timpul necesar înregistrării unei ore de vorbire. Mai mult, etichetarea nu poate fi făcută fără un antrenament prealabil, iar erorile de etichetare sunt mult mai frecvente decât erorile de pronunţie.

În concluzie, în cazul în care se doreşte recunoaşterea vorbirii continue citite cea mai potrivită metodă de achiziţie este înregistrarea textelor predefinite. În cazul în care se doreşte recunoaşterea vorbirii continue spontane cel mai potrivit ar fi un proces de antrenare

ierarhizat în care să se utilizeze inițial o bază de date achiziționată folosind prima metodă, iar ulterior o bază de date de vorbire spontană.

2.5 Corpusul de text

Rolul şi modul de utilizare al unui corpus de text într-un sistem de recunoaştere automată a vorbirii este evidențiat în Figura 2.1. Se poate observa că această resursă lingvistică este necesară în partea de dezvoltare a unui sistem de recunoaştere, mai precis la antrenarea modelului lingvistic.

Sistemele de RAV de comandă şi control nu necesită un corpus de text pentru antrenarea modelului lingvistic pentru că în aceste sisteme se folosesc de obicei gramatici cu reguli. În cazul sistemelor de RVC cu vocabular extins se folosesc modele de limbă statistice, care se construiesc utilizând corpusuri de text de dimensiuni cât mai mari şi cât mai adaptate domeniului din care fac parte mesajele vorbite ce trebuie decodate. Capacitatea de predicție a modelului lingvistic depinde de dimensiunea corpusului de text utilizat la antrenare, pentru că cu cât numărul cuvintelor şi secvențelor de cuvinte dintr-o limbă este mai mare, cu atât probabilitățile de apariție ale acestora sunt mai dificil de estimat cu acuratațe. De aceea, pentru un sistem de RAV de bună calitate este absolut necesară achiziția unui corpus de text de dimensiuni cât mai mari (milioane-miliarde de cuvinte).

Achiziția unui corpus de text de asemenea dimensiuni nu poate fi făcută decât în mod automat, iar o cantitate atât de mare de text nu poate fi găsită decât pe Internet. În consecință, cele mai populare metode de dezvoltare a corpusurilor de text folosesc principiul Web-as-Corpus şi implică identificarea, descărcarea şi procesarea materialelor text de pe Internet.

Identificarea surselor de text disponibile pe Internet este relativ simplă (site-uri de ştiri online, blog-uri, subtitrări, etc.), însă dificultatea proceselor de descărcare, normalizare şi restaurare a textelor depinde mult de sursa de text. De exemplu, descărcarea unui număr atât de mare de articole de ştiri trebuie făcută automat, însă nu toate ziarele online pun la dispoziție link-uri uşor de procesat în mod automat pentru articolele lor.

2.5.1 Normalizarea şi restaurarea corpusului de text

Normalizarea şi restaurarea corpusului de text presupune procesarea acestuia în vederea obținerii unui corpus de text util în antrenarea modelului lingvistic al unui sistem de RVC. Un sistem de RVC are rolul de a transcrie vorbirea, deci va scoate la ieşire text simplu ce conține numai cuvinte şi simboluri care au un corespondent acustic (fără litere mari, fără simboluri speciale, fără cifre, fără semne de punctuație, etc.). De aceea modelul de limbă trebuie să modeleze numai probabilitățile de apariție ale *cuvintelor şi succesiunilor de cuvinte pronunțabile* (nu ale altor caractere sau simboluri).

În funcție de sursa de achiziție a textului, normalizarea şi restaurarea lor poate presupune mai multe sau mai puține operații, însă, de regulă, următoarele sunt indispensabile:

- eliminarea tututor etichetelor HTML (în cazul în care textele au fost achiziționate de pe Internet);
- normalizarea caracterelor diacritice cu care este scris textul (înlocuind toate versiunile de diacritice cu un singur simbol pentru fiecare caracter diacritic în parte). Această operație se referă la textele scrise cu ş cu sedilă, respectiv ţ cu sedilă;
- expandarea abrevierilor (cuvintele abreviate nu se pronunță aşa cum se scriu, ci în forma lor neabreviată);
- transcrierea numerelor scrise cu cifre în numere scrise cu litere (în cazul numărului 1984, semnalul acustic nu are nimic de-a face cu cifrele propriu zise 1, 9, 8, 4, ci este pur şi simplu pronunția secvenței de cuvinte o mie nouă sute opt zeci şi patru);
- eliminarea sau înlocuirea semnelor de punctuație şi a caracterelor speciale (de exemplu a) virgulele trebuie şterse, b) punctele, semnele de întrebare şi semnele de exclamare trebuie înlocuite cu caracterul de linie nouă, c) parantezele trebuie şterse, iar textul din interiorul lor poate fi plasat pe o linie separată, d) liniuțele, cu excepția celor din interiorul cuvintelor trebuie şterse, e) caracterele procent pot fi înlocuite cu expresia la sută, etc.);
- transformarea tuturor literelor mari în litere mici.

O operație ceva mai dificilă, dar care trebuie abordată pentru multe dintre textele achiziționate de pe Internet este restaurarea diacriticelor. Cuvintele formate din aceleaşi litere, dar care au semne diacritice diferite (deci cuvintele ambigue din punctul de vedere al diacriticelor) au de cele mai multe ori sensuri diferite, se pronunță diferit şi apar în contexte diferite. De aceea, ele nu pot fi modelate, nici din punct de vedere lingvistic, nici din punct de vedere acustic, neținând cont de diacritice. În concluzie, dacă textul achiziționat şi normalizat nu conține diacritice ele trebuie neapărat restaurate pentru ca textul să devină util în antrenarea sistemului de RVC.

2.6 Concluzii

Sistemele moderne de recunoaştere a vorbirii continue utilizează modele statistice pentru a modela probabilitățile de apariție ale cuvintelor şi succesiunilor de cuvinte dintr-o limbă şi pronunția fonemelor. Pentru antrenarea modelelor statistice este nevoie de cantități mari de date reprezentative pentru fenomenul ce trebuie modelat. În cazul sistemelor de RVC este sunt necesare baze de date de vorbire etichetată pentru modelarea pronunției fonemelor şi corpusuri de text pentru modelarea statisticii apariției cuvintelor. Pe lângă aceste două resurse este nevoie şi de un dicționar fonetic care să specifice modul de fonetizare al cuvintelor limbii.

3. PRINCIPIILE DE BAZĂ ALE MODELĂRII ACUSTICE

În platforma intitulată "Arhitectura generală a unui sistem de recunoaştere a vorbirii" am menționat faptul că sistemele de recunoaştere a vorbirii continue (RVC) state-of-the-art nu estimează direct probabilitatea mesajului vorbit, dată fiind o succesiune de cuvinte *p(X|W)*, ci estimează probabilitatea unor unități de vorbire mai mici, de obicei foneme. Prin urmare, modelul acustic (MA) este format dintr-un set de modele pentru foneme care sunt conectate, în timpul procesului de decodare, pentru a forma modele pentru cuvinte şi apoi modele pentru succesiuni de cuvinte. Acestea sunt utilizate în final pentru a estima *p(X|W)*. S-a dovedit că această abordare generativă poate fi foarte bine implementată utilizând modele Markov ascunse (hidden Markov models - HMM) [Baker, 1975; Poritz, 1988; Rabiner, 1989; Jelinek, 1998].

HMM-urile sunt automate probabiliste cu număr finit de stări care pot fi combinate în mod ierarhic pentru a construi modele pentru secvențe de cuvinte din modele pentru unități de vorbire mai scurte. În sistemele de RVC cu vocabular extins modelele pentru secvențe de cuvinte sunt construite din modele pentru cuvinte, care, la rândul lor, sunt construite din modele pentru unități sub-lexicale (de obicei foneme dependente de context) utilizând un dicționar fonetic.

3.1 Parametri acustici

HMM-urile nu modelează semnalul acustic be baza formei de undă a acestuia. Aşa cum ilustrează Figura 1.1, un bloc de extragere de parametri este folosit pentru a calcula nişte vectori de coeficienți care sunt apoi modelați de către modelul acustic.

Semnalul de vorbire este un semnal nestaționar. În consecință analiza spectrală nu poate fi făcută pe întreg semnalul, ci numai pe cadre scurte (20ms – 30ms), cvasi-staționare. Semnalul inițial în domeniul timp este transformat prin ferestruire într-o secvență de ferestre în domeniul timp. Din fiecare dintre aceste ferestre sunt extraşi în continuare parametri de tip spectral sau cesptral. Mai multe tipuri de parametri, ce pot fi utilizați pentru modelarea vorbirii, au fost propuşi încă din anii '80. Cei mai folosiți parametri sunt cei de tip cepstral-perceptual, dintre care aminitim *parametri Mel-cepstrali* (*Mel-Frequency Cepstrum*

Coefficients - MFCCs) şi *parametri perceptuali obținuți prin predicție liniară* (*Perceptual Linear Prediction - PLP*). Un avantaj particular al reprezentării cepstrale comparativ cu reprezentarea spectrală este decorelația coeficienților, comparativ cu gradul mare de corelație observat între coeficienții spectrali vecini. Extragerea parametrilor MFCC, respectiv PLP este detaliată în Figura 3.1 şi Figura 3.2.

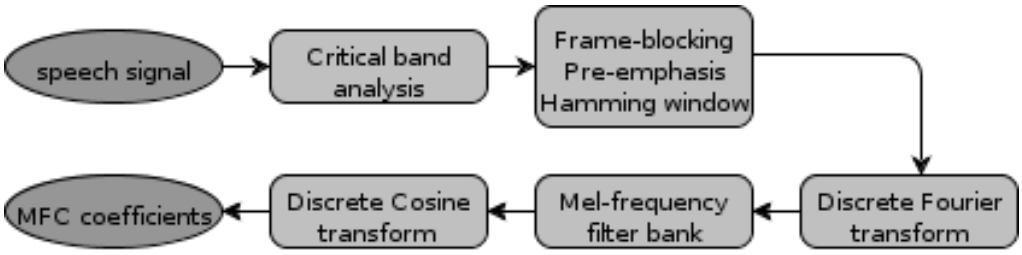


Figura 3.1. Extragerea parametrilor MFCC

Cu toate că fiecare set de coeficienți, fie ei MFCC sau PLP, este calculat pe o fereastră scurtă din semnalul de vorbire, informația conținută de dinamica temporală a acestor parametri este şi ea utilă în RAV. Mai mult, energia totală a parametrilor şi derivatele ei temporale par să fie utile în RAV. De aceea, cel mai utilizat set de parametri în RAV este format din 39 de coeficienți: 12 MFCC + energie, împreună cu derivatele temporale de ordin unu şi doi.

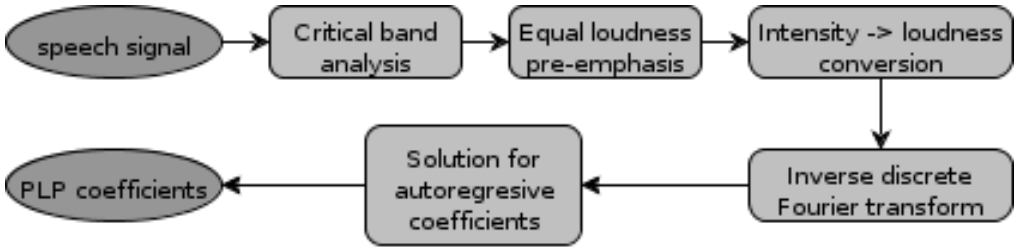


Figura 3.2. Extragerea parametrilor PLP

3.2 Platforma HMM/GMM

Un HMM este un automat cu stări finite format dintr-un set de stări conectate cu arce reprezentând tranziții. Într-un HMM secvența de stări este ascunsă, de aceea întregul proces Markov este considerat ascuns: secvența de stări nu este direct disponibilă observatorului. Observatorul are acces numai la secvența de vectori de parametri acustici generată de o funcție de densitate de probabilitate ataşată fiecărei stări. Acest tip de proces Markov s-a dovedit a fi un foarte bun model al producerii vorbirii. O reprezentare detaliată a unui HMM este ilustrată în Figura 3.3. Aşa cum se vede în figură, un HMM este caracterizat de urmatorii parametri:

- un set de stări $Q=q_1q_2...q_N$;

- un set de probabilități $A=a_{11}a_{12}...a_{NN}$ reprezentând probabilitățile de tranziție între stări. Fiecare $a_{ij}=p(q_j|q_i)$ reprezintă probabilitatea de tranziție din starea *i* în starea *j*;
- un set de observații probabilistice $B=b_i(x_t)=p(x_t|q_i)$, fiecare exprimând probabilitatea ca observația x_t să fi fost generată de starea *i*.

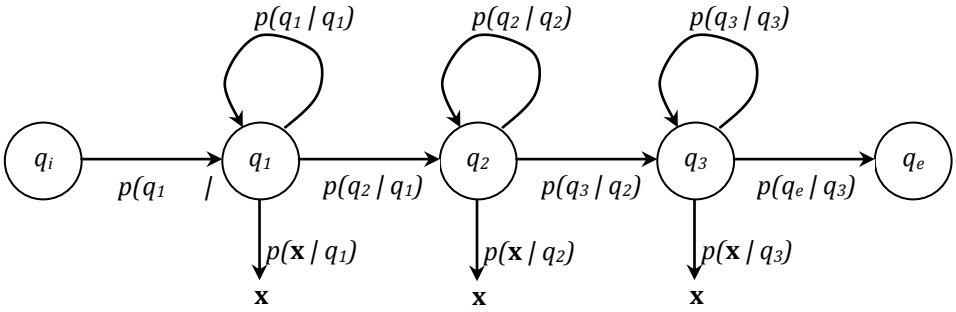


Figura 3.3. Reprezentarea unui HMM ca un automat probabilist cu număr finit de stări

Cu toate că definiția unui HMM permite tranziția din orice stare în orice altă stare, în RAV modelele sunt create astfel încât să interzică tranzițiile arbitrare. Asta pentru că este foarte important şi util ca modelul să țină cont de caracterul secvențial ar producerii vorbirii. De aceea modelul impune constrângeri suplimentare privind tranzițiile înapoi sau salturile peste stări.

Probabilitățile parametrilor generați de o stare q_i sunt de obicei modelate de o funcție densitate de probabilitate. De cele mai ulte ori aceasta ia forma unei sume de *K* funcții Gaussiene (Gaussian mixture model - GMM) în *d* dimensiuni:

$$b_i(x) = p(x \mid q_i) = \sum_{k=1}^{K} c_{ik} \mathrm{N}(x; \mu_{ik}, \Sigma_{ik}) \qquad (3.1)$$

Modelarea vorbirii cu ajutorul HMM-urilor face două presupuneri importante [Renalds, 2010]:

- proces Markov: se presupune că tranziția între stările unui HMM este un proces Markov de ordin întâi în care probabilitatea traziției următoare depinde numai de starea curentă.
- independența observațiilor: probabilitatea ca o stare să genereze un anumit vector de parametri este independentă de vectorii de parametri generați în prealabil.

Aceste două presupuneri pot conduce la un model al producerii vorbirii nerealist, însă ele sunt necesare pentru simplificările matematice şi computaționale pe care le aduc. Problema estimării şi decodării unui HMM nu pot fi abordate sau pot fi abordate într-un mod foarte complicat dacă se renunță la aceste două condiții simplificatoare.

Pentru rezovalrea celor trei probleme ale platformei HMM/GMM (evaluarea, decodarea şi estimarea) există algoritmi standard ce utilizează cele două simplificări

menționate mai sus. Evaluarea (estimarea porbabilității ca o secvență de observații să fi fost generată de un HMM) este rezolvată recursiv de *algoritmul Forward*. Problema decodării HMM-urilor, adică găsirea celei mai probabile secvențe stări care să fi generat o secvență de observații, este rezolvată de o variantă a *algoritmului Viterbi*. Diverşii parametri ai unui sistem HMM/GMM sunt estimați folosind *algoritmul Forward-Backword* sau *Baum-Welch* [Baum, 1972], o particularizare a algoritmului *expectation-maximization* (EM) [Dempster, 1977].

3.3 Selecția, dependența de context şi clusterizarea unităților de vorbire

Am arătat deja că sistemele de RVC modelează unități de vorbire sub-lexicale folosind HMM-uri. Totuşi, mai sunt nişte aspecte ce privesc alegerea unităților de vorbire şi care trebuie discutate. Autorii [Huang, 2001] văd trei caracteristici pe care o unitate de vorbire adecvată trebuie să le aibă:

- *precizie*; unitatea trebuie să reprezinte realizarea acustică care apare în diferite contexte.
- *antrenabilitate*; trebuie să existe destule date pentru a estima parametri respectivei unități.
- *generalizabilitate*; orice nou cuvânt trebuie să poată fi construit având la dispoziție un set de unități de bază (aceasta ar asigura independența de sarcină a sistemului de RAV).

Date fiind acestea, este acum foarte evident de ce *cuvintele* nu pot fi folosite ca unități de vorbire în sistemele de RVC: ele nu sunt nici antrenabile, nici generalizabile.

Ca o alternativă viabilă s-ar putea utiliza *fonemele* ca unități de vorbire de bază. Spre diferență de modelele pentru cuvinte, modelele pentru foneme nu prezintă probleme de antrenare pentru că există suficiente ocurențe pentru fiecare fonem chiar şi într-o bază de date de câteva mii de fraze. Mai mult, fonemele sunt, prin natura lor, indepedente de vocabularul sarcinii de recunoaştere. În consecință fonemele sunt preferate în detrimentul cuvintelor pentru că ele sunt mai antrenabile şi generalizabile. Totuşi, modelul fonetic nu este tocmai adecvat pentru că el presupune că un fonem este pronunțat identic în orice context. Chiar dacă am încerca să pronunțăm fiecare cuvânt ca o succesiune de foneme independente, aceste foneme nu sunt produse independent pentru că sistemul articulator nu se poate mişca instantaneu dintr-o poziție în alta. Deci producerea unui fonem este puternic influențată de producerea fonemelor din imediata sa vecinătate. Dacă modelele pentru cuvinte nu sunt carcacterizate de generalizabilitate, modelele pentru foneme generalizează prea mult şi asta conduce la o scădere semnificativă de precizie.

Utilizarea modelelor fonetice dependente de context ar creşte foarte mult precizia recunoaşterii, în condițiile în care ar exista suficiente date de antrenare pentru a estima corect parametri acestor modele pentru unitațile de vorbire de bază (dependente de context). Fonemele dependente de context au fost utilizate pe scară largă în sistemele de RAV cu vocabular extins în primul rând datorită preciziei şi antrenabilității crescute. Contextul se referă de obicei la fonemele din imediata vecinătate (stânga şi/sau dreapta) a fonemului curent.

Un model de tip *trifonem* este un model fonetic care ia în considerare ambele foneme vecine (stânga/dreapta). Dacă două foneme identice se regăsesc în mesajul vorbit în contexte diferite, ele sunt considerate trifoneme diferite. Diferitele realizări particulare ale unui fonem sunt denumite alofone. Trifonemele sunt exemple de alofone.

Modelele pentru trifoneme sunt mai puternice pentru că ele capturează cele mai importante efecte de co-articulare. Cu toate acestea, din cauza numărului ridicat de modele diferite, ce conduce la un număr mare de parametri ce trebuie estimați, antrenabilitatea lor devine o problemă delicată. Mai multe tehnici de partajare a parametrilor (legarea stărilor, clusterizarea modelelor, etc.) au fost utilizate pentru a achilibra balanța între precizia şi antrenabilitatea modelelor. Hwang şi Huang au introdus clusterizarea la nivel de stare [Hwang, 1991]. Fiecare cluster reprezintă un set de stări Markov similare denumit *senone* [Hwang, 1993]. În momentul de față senonele sunt cele mai utilizate unități de vorbire în sistemele de RAV.

4. PRINCIPIILE DE BAZĂ ALE MODELĂRII LIMBAJULUI NATURAL

În Capitolul 0 am menționat faptul că sistemele de recunoaştere a vorbirii continue cu vocabular extins (RVC-VE) utilizează în timpul decodării un model de limbă (ML) pentru a estima probabilitățile tuturor secvențelor de cuvinte din spațiul de căutare. În general, rolul unui model de limbă este de a estima probabilitatea ca o secvență de cuvinte $W = w_1, w_2, ..., w_n$, să fie o secvență validă pentru sarcina de recunoaştere respectivă. Probabilitățile acestor secvențe de cuvinte ajută foarte mult modelul acustic în procesul de decizie. De exemplu, în cazul unei sarcini de recunoaştere de vorbire continuă (fără restricții) aceste două fraze: *casa ta e mare?* şi *casat a Ema re?* sunt similare din punct de vedere acustic, însă cea de-a doua nu are niciun sens. Rolul modelului de limbă este acela de a asigna o probabilitate mult mai mare primei secvențe de cuvinte şi, prin urmare, de a ajuta sistemul de RAV să decidă în favoarea acesteia.

Sistemele de recunoaştere de vorbire continuă cu vocabular mare de ultimă generație utilizează modele de limbă statistice de tip n-gram. Aceste modele de limbă se construiesc pe baza unor corpusuri mari de text specific sarcinii de recunoaştere, estimând probabilitățile de apariție ale cuvintelor şi secvențelor de cuvinte specifice respectivei sarcini. Modelele de limbă de tip n-gram se utilizează apoi în procesul de decodare (recunoaştere a vorbirii) pentru a calcula probabilitățile secvențelor de cuvinte propuse de modelul acustic. Modelele de limbă statistice vor fi detaliate în următoarea secțiune.

Pentru sarcinile de recunoaştere de tip comandă şi control modelele de limbă statistice nu sunt foarte adecvate. În cazul în care vocabularul sarcinii de recunoaştere conține doar câteva zeci sau sute de cuvinte, iar modul în care acestea se pot succede este bine precizat, modelele de limbă de tip gramatică cu stări finite (FSG – finite state grammar) sunt mult mai potrivite. O gramatică cu stări finite este un model de tip graf în care nodurile reprezintă cuvinte ale vocabularului sarcinii de recunoaştere, iar tranzițiile între cuvinte sunt reprezentate de arcele grafului. Un astfel de model de limbă specifică în mod explicit toate secvențele de cuvinte permise de gramatica sarcinii de recunoaştere. Mai mult, fiecărui arc în parte îi poate fi asignat un cost, specificând astfel probabilitatea ca un cuvânt să fie precedat de un altul (sau altfel spus probabilitatea respectivei secvențe de două cuvinte).

4.1 Modele de limbă statistice

Aşa cum spuneam, rolul unui model de limbă este de a estima probabilitatea ca o secvență de cuvinte $W = w_1, w_2, ..., w_n$, să fie o secvență validă pentru sarcina de recunoaştere respectivă. Probabilitatea secvenței de cuvinte $W = w_1, w_2, ..., w_n$ poate fi descompusă astfel:

$$p(W) = p(w_1, w_2, ..., w_n) = p(w_1)p(w_2 \mid w_1)....p(w_n \mid w_1, w_2, ...w_{n-1}) \qquad (4.1)$$

Aceasta înseamnă că problema estimării probabilității secvenței de cuvinte *W* este împărțită în mai multe probleme de estimare a probabilității unui singur cuvânt, dată fiind secvența de cuvinte anterioare lui (*istoria cuvântului*). Din motive de eficiență computațională, istoriile de cuvinte precendente nu pot include un număr indefinit de cuvinte, ci sunt limitate la ultimile *m* cuvinte. Altfel spus, se face presupunerea că doar un număr limitat de cuvinte afectează probabilitatea următorului cuvânt. Aceasta conduce la modelul de limbă convențional de tip *n-gram*, model ce este state-of-the-art în RVC-VE de peste 20 de ani [Renalds, 2010]. În mod tipic *m* este ales în funcție de cantitatea de text de antrenare disponibil (este nevoie de mai mult text pentru a putea estima cu o precizie bună probabilitățile pentru istorii mai lungi). De obicei se folosesc modele de limbă de tip trigram. Acestea iau în considerare numai ultimile două cuvinte pentru a-l prezice pe al treilea. Pentru aceasta este nevoie de colectarea unor statistici de apariție pentru secvențe de trei cuvinte, aşa numitele 3-grame (trigrame). Se pot construi modele de limbă şi pentru secvențe de două cuvinte (bigrame), sau pentru secvențe formate din mai multe cuvinte (de ordin mai mare).

4.1.1 Construcția modelelor de tip n-gram şi alte probleme specifice

Un model de limbă de tip n-gram se construieşte estimând probabilitățile discutate mai sus pe baza unor corpusuri de text de mari dimensiuni. De exemplu, în cazul unui ML bigram trebuie estimate probabilitățile $p(w_j|w_i)$ pentru fiecare pereche de cuvinte (w_i, w_j). Pentru a calcula aceste probabilități se utilizează principiul *maximum likelihood*. Se numără de câte ori cuvântul w_i este urmat de cuvântul w_j, comparativ cu alte cuvinte:

$$p(w_j \mid w_i) = \frac{count(w_i, w_j)}{\sum_w count(w_i, w)} \qquad (4.2)$$

Pentru a estima corect aceste probabilități este nevoie de o cantitate mare de date de antrenare (tipic de ordinul a câteva zeci de milioane de cuvinte). Pentru construcția modelelor n-gram de ordin mai mare este de nevoie de şi mai multe date.

O problemă cheie în construcția ML de tip n-gram, chiar şi atunci când se dispune de corpusuri mari, este *data sparseness*. Indiferent cât de mare este corpusul de antrenare, vor fi n-grame care nu vor apărea în acest text, însă care ar putea să apară în textul de evaluare. Conform Ecuației 4.2, în acest caz limită, probabilitatea asignată n-gramelor necunoscute este

0. În afară de acest caz există alte n-grame care apar de foarte puține ori (mai puțin de zece ori) în corpusul de antrenare. Această problemă devine mai importantă în cazul ML de tip n-gram de ordin mai mare. În toate aceste cazuri, probabilitățile care au fost estimate pe baza numărului de apariții ale n-gramelor în corpusul de antrenare, trebuie ajustate.

Metodele care se ocupă de acest proces de ajustare se numesc *metode de netezire*. Ele extrag o parte din probabilitatea alocată pentru n-gramele întâlnite la antrenare şi o redistribuie n-gramelor necunoscute. Există mai multe metode de netezire care particularizează modul de redistribuție a probabilității. Dintre acestea, cea mai eficientă metodă este *Good-Turing*, numită şi *netezirea Katz*.

Problema data sparseness este abordată într-o manieră oarecum diferită de către aşa numitele *metode de back-off*. Aceste metode utilizează mai multe modele de limbă, care au avantaje diferite, pentru a crea un model de limbă interpolat, care să poată beneficia de toate părțile constituente. De exemplu, modelele de tip n-gram de ordin mai mare oferă un context de predictibilitate mai mare, însă modelele de ordin mai mic sunt mai robuste. Mai multe metode de back-off, ce încearcă să estimeze probabilitățile n-gramelor necunoscute pe baza probabilităților asignate acestor n-grame de către modele de ordin mai mic, au fost propuse în ultimile două decenii. Dintre acestea, metoda *Kneser-Ney modificată* [Chen, 1998] pare să fie cea mai eficientă la ora actuală.

4.1.2 Criterii de performanță utilizate în evaluarea modelelor de limbă

În cazul în care un sistem de RAV este disponibil, cel mai utilizat criteriu de performanță pentru ML este *rata de eroare la nivel de cuvînt* (*word error rate – WER*). Alternativ, fără a implica un sistem de RAV, se poate evalua puterea de predicție a unui ML măsurând probabilitatea pe care el o asignează unor secvențe de cuvinte de evaluare. Un model de limbă bun ar trebui să atribuie o probabilitate mare unui text corect şi o probabilitate mică unui text prost. În acest caz, cel mai comun criteriu de performanță este *perplexitatea*. O perplexitate mai mare pe o secvență de cuvinte particulară denotă o capacitate de predicție a secvenței mai mică pentru respectivul ML. În general, în contextul RAV, perplexitatea se corelează foarte bine cu WER-ul [Huang, 2001].

Este posibil ca unele dintre cuvintele din textul de evaluare să nu fi fost întâlnite deloc în corpusul de antrenare. Aceste cuvinte sunt numite *cuvinte în afara vocabularului* (*out of vocabulary - OOV*) şi nu pot fi prezise de modelul de limbă. Cuvintele OOV fac evaluarea unui ML mai dificilă: din cauză că perplexitatea lor este infinită, aceasta nu poate fi adunată la perplexitățile celorlalte n-grame pentru a obține perplexitatea întregii secvențe de cuvinte. Astfel, pe lângă perplexitate, procentul de cuvinte OOV trebuie să fie şi el specificat pentru ca evaluarea ML să fie completă.

4.2 Gramatici cu stări finite (FSG – finite state grammar)

Recunoaşterea vorbirii continue este una dintre cele mai dificile sarcini de recunoaştere pentru că vorbitorul poate pronunța orice succesiune de cuvinte care are un înțeles în limba respectivă. De aceea, vocabularul acestei sarcini de recunoaştere ar trebui să conțină practic toate cuvintele limbii, iar modelul de limbă ar trebui să fie capabil să estimeze corect probabilitățile de apariție pentru fiecare succesiune de cuvinte posibilă. În aceste condiții modelele de limbă statistice de tip n-gram sunt foarte potrivite. În condițiile în care sarcina de recunoaştere prezintă restricții cu privire la vocabularul de cuvinte şi la setul de succesiuni valide de cuvinte, atunci modelul n-gram nu mai este atât de potrivit, întrucât efortul de antrenare nu se justifică. Succesiunile valide de cuvinte şi probabilitățile lor de apariție pot fi specificate direct folosind un model de limbă de tip gramatică cu stări finite (FSG). O gramatică cu stări finite este un model de tip graf în care nodurile reprezintă cuvinte ale vocabularului sarcinii de recunoaştere, iar tranzițiile între cuvinte sunt reprezentate de arcele grafului. Un astfel de model de limbă specifică în mod explicit toate secvențele de cuvinte permise de gramatica sarcinii de recunoaştere. Mai mult, fiecărui arc în parte îi poate fi asignat un cost, specificând astfel probabilitatea ca un cuvânt să fie precedat de un altul (sau altfel spus probabilitatea respectivei secvențe de două cuvinte).

4.2.1 Proiectarea unei gramatici cu stări finite

Pentru a ilustra mai bine conceptele descrise mai sus vom porni de la un exemplu simplu de construcție a unei gramatici cu stări finite, specifică unei anumite sarcini de recunoaştere: transcrierea secvențelor audio ce conțin cifre. Această sarcină de recunoaştere presupune că vorbitorul poate rosti numai cuvintele *zero, unu, doi, trei, patru, cinci, şase, şapte, opt* şi *nouă*, în orice ordine şi de oricâte ori. Definirea grafului ce va reprezenta gramatica cu stări finite începe cu definirea nodurilor de intrare şi ieşire din gramatică. Aceste noduri vor fi notate cu *N* (de la *null*) pentru că trecerea prin aceste noduri nu corespunde afişării vreunui cuvânt. Aceste noduri fac parte din infrastructura gramaticii cu stări finite. Urmează crearea nodurilor specifice fiecărui cuvânt în parte şi legarea lor de nodurile de intrare, respectiv ieşire utilizând arce direcționate (dinspre nodul de intrare către fiecare cuvânt şi dinspre fiecare cuvânt către nodul de ieşire). În acest moment am definit o gramatică a cărei parcurgere poate fi făcută dacă se pronunță o singură cifră. Pentru sarcina de recunoaştere propusă (recunoaşterea unei secvențe de cifre) mai avem de făcut un singur lucru: adăugarea unei tranziții înapoi (dinspre nodul de ieşire spre nodul de intrare). În final trebuie sa mai adăugăm încă două noduri *null* pentru că cele precendente nu mai sunt noduri de intrare/ieşire propriu-zise (au şi arce care intră în ele şi arce care ies din ele).

În Figura 4.1 este reprezentată grafic gramatica cu stări finite a sarcinii noastre de recunoaştere. Se poate observa că modelul este format din 14 noduri, dintre care numai zece reprezintă cuvinte ale limbii (cifrele de la zero la nouă), iar celelalte patru sunt noduri de infrastructură, utilizate pentru a realiza „intrarea”, respectiv „ieşirea” din graf şi pentru

tranziția înapoi. Tranzițiile arată modul în care se poate parcurge această gramatică şi implicit secvențele de cuvinte permise: în fiecare clip audio se pot pronunța una sau mai multe cifre.

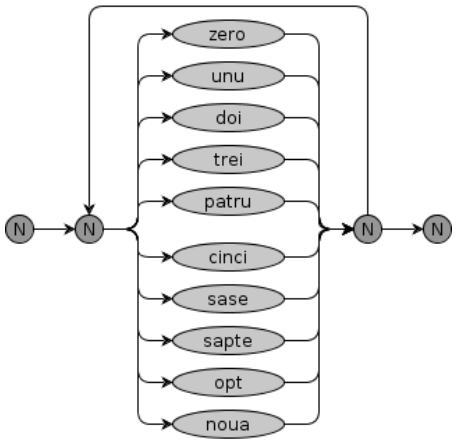


Figura 4.1. Gramatica cu stări finite a sarcinii de recunoaştere de cifre

Implementarea unei astfel de gramatici FSG poate fi făcută foarte simplu utilizând formatul Java Speech Gramar (JSGF) aşa cum vom vedea în continuare.

4.3 JavaSpeech Grammar Format (JSGF)

Formatul de gramatică JavaSpeech defineşte o modalitate de descriere a gramaticilor de reguli (gramatici de comandă şi control) independentă de platformă şi de furnizor. Acest format foloseşte o reprezentare textuală a regulilor ce poate fi uşor citită şi modificată atât de manual, cât şi programatic.

O gramatică cu reguli specifică tipurile de fraze vorbite pe care un vorbitor le-ar putea rosti (o frază vorbită, în această accepțiune, este similară cu o frază scrisă). De exemplu, o gramatică simplă de control al unei ferestre dintr-un program ar putea să "asculte" comenzi precum: *deschide un fişier, închide fereastra*, etc.

Tipologia frazelor pe care utilizatorul le-ar putea rosti este foarte puternic dependentă de context: utilizatorul foloseşte o aplicație de email, formează un număr de telefon sau selectează un font? Aplicațiile cunosc contextul şi, în consecință, pot furniza gramatica cea mai potrivită sistemului de recunoaştere a vorbirii.

În continuare vor fi prezentate câteva convenții utilizate pentru denumiri şi apoi componentele de bază ale gramaticii (antetul şi corpul gramaticii). În antetul gramaticii se specifică numele gramaticii şi se listează alte reguli şi gramatici importate. În corpul gramaticii se declară regulile ce formează gramatica. Aceste reguli sunt o combinație de text ce poate fi pronunțat şi referințe către alte reguli. În final vom prezenta câteva exemple de definiții de gramatici.

4.3.1 Definiții

Fiecare gramatică definită prin JSGF are un nume unic declarat în antetul gramaticii. Numele gramaticii poate fi un nume complet (ce conține şi denumirea pachetului din care face parte gramatica) sau un nume simplu. Denumirea pachetelor de gramatici şi a gramaticilor se face similar cu denumirea pachetelor si claselor din limbajul de programare Java.

Orice gramatică este compusă dintr-un set de reguli care definesc complet tipologia frazelor ce pot fi pronunțate. Regulile sunt combinații de text (ce poate fi pronunțat) şi referințe către alte reguli. Fiecare regulă are o denumire unică. O referință către o altă regulă se realizează folosind denumirea regulii respective, precedată de caracterul < şi urmată de caracterul >. Numele regulilor sunt similare cu numele identificatorilor din limbajul de programare Java, respectând aceleaşi restricții şi convenții.

Simbolurile sau terminalii definesc cuvinte sau secvențe de cuvinte care pot fi pronunțate de către vorbitor. De cele mai multe ori un terminal este asociat unui singur cuvânt, însă aceasta nu reprezintă o restricție (un terminal poate fi asociat şi unui grup de cuvinte). Un terminal poate apărea izolat, ca în următoarele exemple:

```
salutare
bucureşti
```

sau într-o secvență de terminali ca în exemplele de mai jos:

```
bună ziua
douăzeci şi patru
```

Un terminal este de fapt o secvență de caractere delimitată de spații, ghilimele sau alte caractere care au o semnificație specială în JSGF: ; = | * + <> () [] {} /* */ //. Un terminal este în acelaşi timp o referință către un cuvânt din dicționarul fonetic al sistemului de recunoaştere automată a vorbirii. Utilizând pronunția terminalului din dicționarul fonetic, sistemul de recunoaştere automată a vorbirii poate transcrie vorbirea în text.

Un terminal nu trebuie neapărat să corespundă unui singur cuvânt din vocabular, ci poate fi o secvență de de cuvinte sau un simbol. Pentru a defini un terminal multi-cuvânt sau un terminal de tip simbol trebuie folosite ghilimelele ca în exemplele de mai jos:

```
primăria municipiului “Baia de Aramă”
doi “+” trei
```

Un terminal multi-cuvânt este util când pronunția cuvintelor componente diferă în funcție de context. În acest fel se pot defini intrări separate în vocabularul sistemului de RAV, fiecare cu pronunția ei. Terminalii multi-cuvânt pot fi folosiți şi pentru a simplifica procesarea rezultatelor, de exemplu obținerea unui singur terminal pentru expresii precum "Bucureşti", "Baia Mare", "Cluj-Napoca".

Terminalii multi-cuvânt pot fi incluşi în dicţionarul fonetic al sistemului de RAV, exact ca orice alt terminal. În cazul în care terminalul multi-cuvânt este găsit în dicţionarul fonetic, atunci se va folosi pentru recunoaştere pronunţia respectivă. În caz contrar, se vor folosi pronunţiile pentru toţi sub-terminalii acestuia (toate simbolurile separate de spaţii din terminalul multi-cuvânt).

Majoritatea sistemelor de recunoaştere de vorbire au capacitatea de a folosi şi semne de punctuaţie sau diferite simboluri. De exemplu, sistemele de recunoaştere pentru limba română trebuie să poată gestiona cratima (într-o, dă-mi). Totuşi, există multe situaţii care nu pot fi tratate corect şi neambiguu de un sistem de RAV. În aceste cazuri, persoana care dezvoltă gramatica ar trebui să utilizeze terminali cât mai apropiaţi de modul în care se pronunţă cuvintele respective şi care apar în dicţionarul fonetic. Următoarele exemple prezintă câteva cazuri uzuale:

- Numerele: terminalii scrişi cu cifre “0 1 2” trebuie scrişi cu litere “zero unu doi”. În mod similar o comadă de forma “formează 0740 213 449” ar trebui scrisă “formează zero şapte patru zero doi unu trei patru patru nouă”,
- Datele ce conţin cifre trebuie expandate. De exemplu data “25 aprilie 1984” trebuie scrisă sub forma “douăzeci şi cinci aprilie o mie nouă sute opt zeci şi patru”,
- Abrevierile şi acronimele trebuie de asemenea expandate. De exemplu abrevierile de genul “d-l” sau “dl.” trebuie scrise “domnul”, iar acronimele de forma “SUA” trebuie transformate în “Statele Unite ale Americii”.

4.3.2 Antetul gramaticii

O gramatică de tip JSGF este definită într-un singur fişier. Sau, altfel spus, un fişier poate conţine definiţia unei singure gramatici. Definiţia gramaticii conţine două părţi: antetul şi corpul gramaticii. Antetul începe cu un identificator, defineşte numele gramaticii şi conţine eventuale importuri de reguli din alte gramatici. Corpul gramaticii defineşte regulile componente ale gramaticii, dintre care unele pot fi declarate ca fiind publice.

Orice fişier care defineşte o gramatică JSGF trebuie să înceapă cu un identificator care indică faptul că fişierul este o definiţie JSGF şi specifică versiunea formatului folosit. Opţional, identificatorul poate specifica în continuare setul de caractere utilizat în fişier, limba şi zona geografică (locale-ul) aferente gramaticii. Identificatorul trebuie să se incheie cu caracterul punct şi virgulă urmat de caracterul linie nouă. Formatul identificatorului este următorul:

```
#JSGF version char-encoding locale;
```

Iată mai jos câteva exemple de posibili identificatori:

```
#JSGF V1.0;
#JSGF V1.0 ISO8859-5;
#JSGF V1.0 JIS ja;
```

În primul exemplu nu se specifică setul de caractere sau limba. În consecință se vor folosi valorile implicite pentru aceşti parametrii (setul de caractere şi limba implicită din sistemul de operare). În al doilea exemplu se specifică setul de caractere ISO8859-5 (caractere chirilice), iar limba este cea implicită. În ultimul exemplu se specifică atât setul de caractere (JIS – un set de caractere pentru limba japoneză), cât şi limba (ja – limba japoneză).

Orice fişier care defineşte o gramatică JSGF trebuie să înceapă cu caracterul diez (#), iar toate caracterele din acest identifiator trebuie să fie codate ASCII.

După acest identificator despre care am discutat mai sus este obligatoriu să urmeze numele gramaticii. Numele gramaticii poate fi format din numele pachetului din care face parte gramatica urmat de numele gramaticii sau poate fi numai numele gramaticii (în cazul în care pachetul nu este specificat). Astfel, formatul liniei care defineşte numele gramaticii poate fi:

```
grammar numePachetGramatică.numeGramatică;
```

sau

```
grammar numeGramatică;
```

Iată mai jos câteva exemple de nume de gramaticii:

```
grammar org.etti.speech.apps.names;
grammar ro.pub.speed.dates;
grammar numbers;
```

Opțional, antetul gramaticii poate conține şi declarații de import de alte reguli sau gramatici. Declarațiile de import trebuie să urmeze după declarația numelui gramaticii, dar înainte de corpul gramaticii (înainte de definiția regulilor). O declarație de import permite utilizarea în gramatica curentă a uneia sau tuturor regulilor dintr-o altă gramatică. Formatul unei astfel de declarații de import poate fi:

```
import <numeRegulă>;
```

sau

```
import <numeGramatică.*>;
```

Iată mai jos câteva exemple de declarații de import:

```
import <org.etti.speech.apps.names.feminine>;
import <ro.pub.speed.dates.*>;
```

Primul exemplu arată modul în care se poate importa o singură regulă dintr-o gramatică: regula <feminine> din gramatica org.etti.speech.apps.names. Este obligatoriu ca regula importată să fie declarată publică în gramatica din care face parte.

Utilizarea simbolului asterisc în cel de-al doilea exemplu specifică faptul că toate regulile publice ale gramaticii ro.pub.speed.dates vor fi importate în gramatica curentă. Dacă, de exemplu, gramatica ro.pub.speed.dates defineşte trei reguli publice: <shortDateNames>, <dayNames> şi <fullDateNames>, atunci toate aceste trei reguli vor putea fi referite local în gramatica curentă.

Observați că datorită faptului că atât numele gramaticii, cât şi numele regulii (sau asterisc-ul) sunt necesare, o declarație de import nu va avea niciodată forma:

```
import <numeRegulă>; // declarație ilegală
```

O regulă importată cu o declarație de import poate fi utilizată în trei moduri: specificând numai numele regulii (de exemplu, <shortDateNames>), specificând numele gramaticii şi numele regulii (de exemplu, <dates.shortDateNames>) sau specificând numele pachetului gramaticii, numele gramatacii şi numele regulii (de exemplu, <ro.pub.speed.dates.shortDateNames>).

În cazul în care o regulă este utilizată întotdeauna specificând numele pachetului gramaticii, numele gramatacii şi numele regulii, atunci declarația de import nu mai este necesară:

```
// importarea ro.pub.speed.dates este optională
<regulă> = < ro.pub.speed.dates.shortDateNames>;
```

4.3.3 Corpul gramaticii

Definiții de reguli

Corpul unei gramatici JSGF defineşte reguli. O regulă este definită odată ce definiția ei apare în gramatică, iar ordinea în care apar definițiile regulilor nu este importantă.

O definiție de regulă are formatul:

```
<numeRegulă> = expansiuneDeRegulă ;
```

sau

```
public <numeRegulă> = expansiuneDeRegulă ;
```

Astfel, cele cinci componente ale unei definiții de regulă sunt: 1) cuvântul cheie public (opțional), 2) numele regulii ce urmează a fi definită, 3) semnul egal, 4) forma expandată a regulii şi 5) caracterul punct şi virgulă. Spațiile goale (caracterele space şi tab) au relevanță numai în interiorul formei expandate a regulii şi sunt ignorate în rest.

Forma expandată a unei reguli specifică modul în care o regulă poate fi pronunțată. Ea este o combinație logică de terminali (text ce poate fi pronunțat) şi referințe către alte reguli.

Orice regulă din gramatică poate fi definită ca fiind publică utilizând cuvântul cheie public. O regulă publică poate fi utilizată în trei scopuri:

- poate fi referită într-o altă gramatică,
- poate fi folosită ca regulă activă în procesul recunoaşterii vorbirii. Cu alte cuvinte o regulă publică poate fi utilizată de un sistem de recunoaştere a vorbirii pentru a transcrie vorbirea în text.
- poate fi referită local de către orice altă regulă publică sau non-publică definită în gramatica curentă.

Dacă cuvântul cheie public nu este specificat atunci regula este implicit privată şi poate fi referită numai local, de către o altă regulă a gramticii curente.

Expansiuni de reguli

Cele mai simple expansiuni de reguli sunt referințe către un terminal sau o altă regulă. De exemplu:

```
<a> = elefant;
<b> = <x>;
<c> = <com.acme.grammar.y>;
```

Regula <a> se expandează într-un singur terminal (elefant). Astfel pentru a spune regula <a>, vorbitorul trebuie să pronunțe “elefant”.

Regula <b> se expandează folosind regula <x>. Astfel, pentru a spune regula <b>, vorbitorul trebuie să pronunțe ceva conform regulii <x>. În mod similar, pentru a spune regula <c>, vorbitorul trebuie să pronunțe ceva conform regulii <com.acme.grammar.y>.

Cu alte cuvinte, expansiunile legale de reguli fac parte din următoarele categorii:

- orice terminal
- o referință către orice regulă publică sau non-publică definită în gramatica curentă
- o referință către o regulă publică dintr-o altă gramatică (regulă ce a fost importată în prealabil în gramatica curentă)

Nu sunt permise definițiile goale, precum cea de mai jos:

```
<d> = ; // definiție ilegală
```

În continuare vom explica modurile în care se pot defini reguli mai complexe prin combinații logice de expansiuni de reguli utilizând:

- secvențe de expansiuni de reguli şi seturi de expansiuni alternative,
- gruparea expansiunilor de reguli folosind paranteze rotude şi pătrate,
- operatori unari (pentru repetarea unei expansiuni de regulă),
- ataşarea de etichete specifice aplicațiilor.

Compunerea regulilor

O regulă poate fi definită ca o secvență de expansiuni de reguli. O secvență de expansiuni legale separate prin spațiu este ea însăşi o expansiune legală. De exemplu, datorită faptului că atât terminalii, cât şi referințele la alte reguli sunt expansiuni legale, următoarele sunt de asemenea expansiuni legale:

```
<unde> = Eu locuiesc în România;
<declarație> = acest <obiect> este <OK>;
```

Pentru a pronunța complet o secvență, item-urile din secvență trebuie pronunțate (toate) în ordinea definită de regulă. În primul exemplu, pentru a spune regula <unde>, vorbitorul trebuie să pronunțe cuvintele "Eu locuiesc în România" chiar în această ordine. Al doilea exemplu combină terminali cu referințe către alte reguli (<obiect> şi <OK>). Pentru a spune regula <declarație>, vorbitorul trebuie să spună cuvântul acest, urmat de ceva care să fie se potrivească cu regula <obiect>, apoi cuvântul este şi în final ceva care să fie se potrivească cu regula <OK>.

Într-o secvență putem avea orice expansiune legală, inclusiv structuri mai complexe similare cu cele ce vor fi descrise în continuare pentru alternative, grupuri şi aşa mai departe.

O regulă poate fi definită ca un set de expansiuni alternative separate prin caracterul bară verticală (|) şi spații, ca în exemplele de mai jos:

```
<nume> = Maria | Ion | Daniel | Valentina | <alteNume>;
```

Pentru a spune regula <nume> vorbitorul trebuie să pronunțe unul şi numai unul dintre expansiunile specificate în setul de alternative. De exemplu, vorbitorul ar putea spune cuvântul Maria, sau cuvântul Ion, sau cuvântul Daniel, sau cuvântul Valentina sau orice altceva care să se potrivească cu regula <alteNume>. Vorbitorul nu poate spune însă Ion Daniel.

Secvențele au precedență în fața alternativelor. De exemplu,

```
<oraş>= Cluj Napoca | Baia Mare | Satu Mare;
```

este un set de trei alternative, fiecare dintre ele reprezentând numele unui oraş.

Nu este permisă utilizarea unei alterniative vide ca mai jos:

```
<nume> = Ion | | Maria; // ilegal
<nume> = Ion | Maria | ; // ilegal
```

De obicei nu toate variantele dintr-un set de alternative sunt egal probabile. Fiecărei variante dintr-o alternativă îi poate fi asociată o pondere care să indice probabilitatea ca respectiva alternativă să fie pronunțată. O pondere este un număr aflat între două caractere slash (de exemplu /3.14/). Cu cât numărul este mai mare, cu atât probabilitatea ca acea alternativă să fie pronunțată este mai mare. Ponderea se plasează în fața fiecărei variante din setul de alternative, ca în exemplele de mai jos:

```
<dimensiune> = /10/ mic | /2/ mediu | /1/ mare;
<culoare> = /0.5/ roşu | /0.1/ albastru închis | /0.2/ verde deschis;
<acțiune> = te rog (/20/salvează fişierele |/1/şterge toate fişierele);
<locație> = /20/ <oraş> | /5/ <țară>;
```

Ponderile trebuie să reflecte frecvențele relative de apariție ale variantelor unui set de alternative. În primul exemplu se indică faptul că cuvântul mic are o probabilitate de apariție de zece ori mai mare decât cuvântul mare şi de cinci ori mai mare decât cuvântul mediu.

Atunci când se folosesc ponderi pentru variantele unei alternative, trebuie respectate următoarele condiții:

- dacă se specifică o pondere pentru o variantă a setului de alternative, atunci trebuie să se specifice ponderi pentru toate variantele lui.
- ponderile pot fi numere raționale, următoarele variante fiind toate corecte: 56, 0.056, 3.14e3, 8f.
- între caracterele slash nu poate apărea altceva în afară de pondere şi spații.
- ponderile pot avea şi valoarea zero. O pondere cu valoarea zero îndică faptul că acea variantă nu poaze fi pronunțată. Ponderile cu valoarea zero pot fi utilizate la dezvoltarea gramaticilor.
- într-un set de alternative trebuie să existe cel puțin o variantă cu pondere nenulă.

Valorile potrivite pentru ponderi sunt de obicei dificil de determinat, iar “ghicirea” acestor valori nu duce întotdeauna la îmbunătățirea performațelor sistemului de recunoaştere a vorbirii. Pentru a obține valori potrivite pentru poderile unui set de alternative se realizează studii statistice ale unor situații reale de vorbire sau text.

Gruparea regulilor

Orice expansiune legală de regulă poate fi grupată explicit utilizând paranteze rotunde. Gruparea are precedență în fața altor operații astfel încât poate fi folosită pentru a asigura interpretarea corectă a regulilor. De asemenea, gruparea este utilă şi pentru îmbunătățirea clarității definiției unei reguli. De exemplu, din cauză că secvențele au o

precedență mai mare decât alternativele, este nevoie de paranteze pentru a crea expansiunile te rog închide şi te rog şterge în definiția următoare:

```
<acțiune> = te rog (deschide | închide | şterge);
```

Următorul exemplu prezintă o secvență de trei item-uri, fiecare dintre ele fiind un set de alternative înconjurate de paranteze rotunde, asigurând astfel gruparea lor corectă:

```
<comandă> = (deschide | închide) (geamurile | uşile) (imediat | mai târziu);
```

Pentru a spune ceva care să se potrivească cu regula <comandă>, vorbitorul trebuie să spună câte un cuvânt din fiecare set de alternative. De exemplu: deschide geamurile imediat sau închide uşile mai târziu.

Nu se pot defini grupuri vide. Exemplul de mai jos nu este corect:

```
( ) // grupare ilegală
```

Parantezele pătrate pot fi folosite pentru a indica faptul că o expansiune de regulă (cea din interiorul lor) este opțională. În alte privințe, ele sunt echivalente cu parantezele rotunde şi au aceeaşi precendență. De exemplu, gramatica de mai jos:

```
<formulăDePolitețe> = te rog | te implor;
public <cerință> = ajută-mă [ < formulăDePolitețe > ];
```

îi permite vorbitorului să spună ajută-mă şi să adauge, opțional, una dintre formulele de politețe te rog sau te implor.

Nu se pot defini grupuri opționale vide. Exemplul de mai jos nu este corect:

```
[] // grupare opțională ilegală
```

Operatori unari

Există trei operatori unari în JSGF: operatorul steluță Kleene, operatorul plus şi etichetele. Toți operatorii unari au următoarele proprietăți:

- un operator unar poate fi ataşat unei expansiuni de regulă legale,
- au precedență mare: sunt ataşați expansiunii de regulă pe care o preced,
- unei expansiuni de regulă i se poate asocia un singur operator,
- spațiile dintre expansiunea de regulă şi opratorul unar sunt ignorate.

Datorită faptului că precedența operatorilor unari este mai mare decât cea a secvențelor şi alternativelor, este necesară folosirea parantezelor (gruparea) pentru a ataşa un operator unar unei secvențe sau unei alternative.

O expansiune de regulă urmată de caracterul asterisc indică faptul că acea expansiune poate fi pronunțată de zero sau mai multe ori. Simbolul asterisc este cunoscut sub denumirea de steluța Kleene (după numele lui Stephen Cole Kleene). De exemplu, construcția

```
public <cerință> = ajută-mă <formulăDePolitețe>*;
```

îi permite vorbitorului să rostească ajută-mă te rog sau ajută-mă te rog te implor te rog, etc. sau să nu folosească nicio rugăminte şi să spună scurt ajută-mă.

Precendența operatorului steluță este mai mare decât cea a secvențierii, astfel că în următoarea construcție:

```
public <cerință> = ajută-mă te rog *;
```

operatorul se aplică terminalului rog, nu întregii secvențe. Astfel, pentru a spune ceva care să se potrivească cu regula <cerință>, vorbitorul ar putea spune ajută-mă, ajută-mă te sau ajută-mă te rog rog rog, dar nu ajută-mă te rog te rog. Parantezele pot fi utilizate pentru a modifica acest comportament. De exemplu,

```
public <cerință> = ajută-mă (te rog) *;
```

O expansiune de regulă urmată de caracterul plus indică faptul că acea expansiune poate fi pronunțată de una sau mai multe ori. De exemplu, construcția

```
public <cerință> = ajută-mă <formulăDePolitețe>+;
```

necesită pronunția a cel putin o formulă de politețe.

4.3.4 Câteva exemple de gramatici pentru nişte sarcini uzuale

Gramatica dates pentru limba română

```
#JSGF V1.0;

grammar dates;

<day> = întâi | unu | doi | trei | patru | cinci | şase | şapte | opt | nouă | zece | unsprezece
| doisprezece | treisprezece | paisprezece | cincisprezece | şaisprezece | şaptesprezece |
optsprezece | nouăsprezece | douăzeci | douăzeci şi unu | douăzeci şi doi | douăzeci şi trei |
douăzeci şi patru | douăzeci şi cinci | douăzeci şi şase | douăzeci şi şapte | douăzeci şi opt | douăzeci
şi nouă | treizeci | treizeci şi unu;

<month> = întâia | a doua | a treia | a patra | a cincea | a şasea | a şaptea | a opta | a noua
| a zecea | a unsprezecea | a doisprezecea | a douăsprezecea | ianuarie | februarie | martie |
aprilie | mai | iunie | iulie | august | septembrie | octombrie | noiembrie | decembrie;

<year> = /80/ o mie nouă sute <tens> | /20/ două mii <smallYears>;

public <date> = <day> <month> [<year>];
```

Gramatica numbers pentru limba română

```
#JSGF V1.0;

grammar numbers;

<unitsWO012> = trei | patru | cinci | şase | şapte | opt | nouă;
<unitsWO0> = unu | una | doi | două | <unitsWO012>;
<units> = zero | <unitsWO0>;

<elevens> = zece | unsprezece | doisprezece | douăsprezece | treisprezece | paisprezece |
cincisprezece | şaisprezece | şaptesprezece | optsprezece | nouăsprezece;

<simpleTens> = douăzeci | treizeci | patruzeci | cincizeci | şaizeci | şaptezeci | optzeci |
nouăzeci;

<max2digitNumbers> = <units> | <elevens> | <simpleTens> [şi <unitsWO0>];

<3digitNumbers> = (o sută | (două | <unitsWO012>) sute) [<max2digitNumbers>];

<max3digitNumbers> = <max2digitNumbers> | <3digitNumbers>;

<max6digitNumbers> = <max3digitNumbers> | ((o mie | <max3digitNumbers> [de] mii)
[<max3digitNumbers>]);

<max9digitNumbers> = <max6digitNumbers> | ((un milion | <max3digitNumbers> [de]
milioane) [<max6digitNumbers>]);

public <rationalNumbers> = <max9digitNumbers> [virgulă <max3digitNumbers>];
```

5. EVALUAREA SISTEMELOR DE RECUNOAŞTERE A VORBIRII CONTINUE

5.1 Introducere

Evaluarea sistemului de recunoaştere a vorbirii este o etapă esențială în dezvoltarea şi optimizarea acestuia. Această etapă îşi propune să determine cât de adecvat este sistemul de recunoaştere pentru a transcrie vorbire, în contextul unei sarcini de recunoaştere bine precizate.

Evaluarea unui sistem de recunoaştere a vorbirii necesită existența unei baze de date de înregistrări audio etichetate, numită în continuare bază de date de evaluare. Pentru o evaluare corectă, baza de date de evaluare trebuie să reprezinte cât mai fidel sarcina de recunoaştere la care va fi supus sistemul:

- domeniul vorbirii: general sau particular (de exemplu: vorbire liberă, pe orice subiect sau discuții pe teme bine precizate: ştiințifice/culturale/istorice/etc.);
- tipologia vorbirii: cuvinte pronunțate izolat, vorbire continuă citită, vorbire continuă spontană, etc.;
- caracteristicile vorbitorului/vorbitorilor: un singur vorbitor sau mai mulți vorbitori, vorbitori nativi/non-nativi, persoane tinere/în vârstă, etc.;
- mediul acustic: zgomotos, nezgomotos.

5.2 Evaluarea sistemului de recunoaştere a vorbirii din punctul de vedere al domeniului vobirii

Dintre cele patru caracteristici ale unei sarcinii de recunoaştere de vorbire prezentate mai sus, domeniul vorbirii este singura care nu depinde de elementele acustice ale înregistrărilor. În consecință, sistemul de recunoaştere a vorbirii poate fi evaluat din acest punct de vedere utilizând numai *transcrierile clipurilor audio* din baza de date de evaluare,

aşa cum ilustrează **Figura 5.1**. Bineînțeles că acest tip de evaluare vizează identificarea potențialelor probleme sau deficiențe ale *modelului de limbă* integrat în sistemul de recunoaştere a vorbirii.

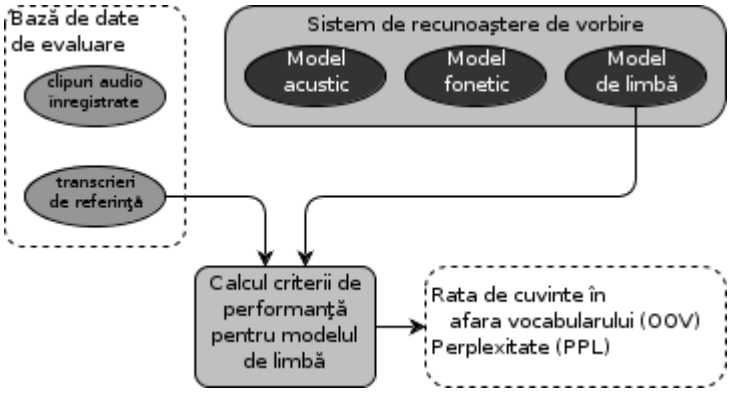


Figura 5.1. Evaluarea modelului de limbă din sistemul de recunoaştere

Într-un sistem de recunoaştere a vorbirii rolul modelului de limbă este de a estima probabilitățile secvențelor de cuvinte din spaţiul de căutare. În consecință, modelul de limbă specifică ce secvențe de cuvinte sunt mai probabile într-un anumit context, pentru o anumită sarcină de recunoaştere, dar şi ce secvențe de cuvinte nu sunt valide.

Capacitatea de predicție a modelului de limbă se poate evalua măsurând probabilitatea pe care el o asignează secvențelor de cuvinte din baza de date de evaluare. Un model de limbă adecvat, adaptat domeniului vorbirii, ar trebui să atribuie o probabilitate mare transcrierilor din baza de date de evaluare. Criteriul de performanță utilizat în acest caz este *perplexitatea*. Peplexitatea (*PPL*) este o mărime derivată din entropie ($H(p_{LM})$ – mărime ce măsoară incertitudinea unei distribuții de probabilitate) şi se defineşte astfel:

$$H(p_{LM}) = -\frac{1}{n}\log p_{LM}(w_1, w_2, ..., w_n) = -\frac{1}{n}\sum_{i=1}^{n}\log p_{LM}(w_i \mid w_1, w_2, ..., w_{i-1})$$

$$PPL = 2^{H(p_{LM})} \qquad (5.1)$$

Probabilitățile p_{LM} $(w_1, w_2, \ldots, w_n)$ şi p_{LM} $(w_i, \mid w_1, w_2, \ldots, w_{i-1})$ sunt probabilitățile atribuite de modelul de limbă secvenței de cuvinte $w_1, w_2, \ldots, w_n$, respectiv cuvântului w_i, în condițiile în care este precedat de secvența de cuvinte $w_1, w_2, \ldots, w_{i-1}$. O perplexitate mai mică pentru o secvență de cuvinte particulară denotă o capacitate de predicție a secvenței mai mare pentru respectivul model de limbă.

Este posibil ca unele dintre cuvintele din textul de evaluare să nu se regăsească în vocabularul sistemului de recunoaştere a vorbirii şi, în consecință, nici în vocabularul modelului de limbă. De aceea aceste cuvinte nu pot apărea în trascrierile rezultate în urma procesului de recunoaştere a vorbirii. Aceste cuvinte sunt numite *cuvinte în afara vocabularului* (*out of vocabulary - OOV*) şi nu pot fi prezise de modelul de limbă

(probabilitatea lor de apariție este nulă). Cuvintele OOV fac evaluarea unui model de limbă mai dificilă: din cauză că perplexitatea lor este infinită, aceasta nu poate fi adunată la perplexitățile celorlalte cuvinte pentru a obține perplexitatea întregii secvențe de cuvinte. Astfel, pe lângă perplexitate, procentul de cuvinte OOV trebuie să fie şi el specificat pentru ca evaluarea modelului de limbă să fie completă. Este evident că un sistem de recunoaştere a vorbirii bun trebuie să aibă un procent de cuvinte OOV cât mai mic.

5.2.1 Evaluarea perplexității şi a ratei de cuvinte OOV: exemplu

Având la dispoziție transcrierile clipurilor audio din baza de date de evaluare şi modelul de limbă (de tip n-gram) integrat în sistemul de recunoaştere a vorbirii, evaluarea perplexității şi a ratei de cuvinte OOV poate fi realizată utilizând *pachetul de programe SRI-LM*, mai precis aplicația *ngram*. Mai jos este prezentată o selecție dintr-un raport de evaluare generat de această aplicație:

```
...
...
iaurturile ajută la prevenirea osteoporozei
	p( <unk> | <s> )	= [OOV] 0 [ -inf ]
	p( ajută | <unk> ...)	= [1gram] 3.07228e-05 [ -4.51254 ]
	p( la | ajută ...)	= [2gram] 0.130606 [ -0.884037 ]
	p( prevenirea | la ...)	= [3gram] 0.0334988 [ -1.47497 ]
	p( <unk> | prevenirea ...)	= [OOV] 0 [ -inf ]
	p( </s> | <unk> ...)	= [1gram] 0.0576355 [ -1.23931 ]
1 sentences, 5 words, 2 OOVs
0 zeroprobs, logprob= -8.11086 ppl= 106.59 ppl1= 505.381
...
...
opoziția e la fel de patetică
	p( opoziția | <s> )	= [2gram] 0.000168717 [ -3.77284 ]
	p( e | opoziția ...)	= [3gram] 0.013834 [ -1.85905 ]
	p( la | e ...)	= [3gram] 0.0211873 [ -1.67393 ]
	p( fel | la ...)	= [3gram] 0.15404 [ -0.812366 ]
	p( de | fel ...)	= [3gram] 0.384352 [ -0.415271 ]
	p( patetică | de ...)	= [2gram] 9.19136e-08 [ -7.03662 ]
	p( </s> | patetică ...)	= [2gram] 0.222222 [ -0.653212 ]
1 sentences, 6 words, 0 OOVs
0 zeroprobs, logprob= -16.2233 ppl= 207.784 ppl1= 505.687
...
...
file transcripts.test: 100 sentences, 1703 words, 34 OOVs
0 zeroprobs, logprob= -4243.58 ppl= 251.315 ppl1= 350.038
```

În raportul de evaluare de mai sus se poate observa modul de calcul al perplexității pentru un fişier text numit transcripts.test ce conține 100 de propoziții de evaluare. În prima parte a exemplului este detaliată evaluarea facută pentru două dintre cele 100 de propoziții. Utilizând modelul de limbă se estimează probabilitatea de apariție a fiecărui cuvânt din propoziție, în contextul respectiv. Se observă că primul cuvânt (“iaurturile”) este un OOV,

probabilitatea lui este 0, iar probabilitatea lui logaritmică este minus infinit. În consecință acest cuvânt nu va fi luat în considerare în calculul perplexității, dar va fi contorizat în cadrul ratei de cuvinte OOV. Al doilea cuvânt (“ajută”) face parte din vocabularul modelului de limbă (ML), iar probabilitatea lui de apariție (în afara contextului) este 3.07e-05. Probabilitatea acestui cuvânt nu poate fi estimată ținând cont de contextul în care apare pentru că cuvântul ce îl precede (“iaurturile”) nu este modelat de ML. Probabilitatea de apariție a celui de-al treilea cuvânt (“la”) este estimată ținând cont de context (“la” precedat de “ajută”) şi este aproximativ 0.13. Probabilitatea de apariție a celui de-al patrulea cuvânt (“prevenirea”) este estimată ținând cont de context (“prevenirea” precedat de “ajută la”) şi este aproximativ 0.03. În fine, ultimul cuvânt din această propoziție (“osteoporozei”) este din nou un cuvânt OOV pentru care probabilitatea estimată de modelul de limbă este nulă. După detalierea acestor probabilități pentru cuvintele propoziției “iaurturile ajută la prevenirea osteoporozei”, raportul de evaluare sumarizează rezultatele obținute: propoziția conține 5 cuvinte, dintre care 2 cuvinte OOV. Probabilitatea logaritmică de apariție a acestei secvențe de cuvinte este -8.11, iar perplexitatea ei este 106.59.

Urmărind al doilea exemplu observăm că propoziția “opoziția e la fel de patetică” conține 6 cuvinte, dar niciun cuvânt OOV. Deşi perplexitatea ei este mai mare decât în primul exemplu (modelul de limbă atribuie o probabilitate mai mică de apariție acestei secvențe de cuvinte), totuşi faptul că numărul de cuvinte OOV este 0 este mai important în contextul recunoaşterii vorbirii. Motivul este simplu: în primul caz sistemul de recunoaştere a vorbirii nu are nicio şansă să recunoască mai mult de 3 cuvinte din 5 (pentru că celelalte 2 nu se află în vocabularul sistemului). Chiar dacă în al doilea caz probabilitățile de apariție ale cuvintelor sunt mai mici, toate aceste cuvinte se află în vocabular şi pot apărea în transcrierea generată de sistem. În concluzie, procentul de cuvinte OOV este cel mai important criteriu de performanță pentru modelul de limbă, iar atunci când această rată este similară pentru mai multe sisteme, perplexitatea este criteriul care îl identifică pe cel mai potrivit.

În finalul raportului de evaluare al modelului de limbă sunt sumarizate valorile pentru întregul text: 100 de propoziții, 1703 cuvinte, dintre care 34 cuvinte OOV, iar perplexitatea globală 251.3.

5.3 Evaluarea completă a sistemului de recunoaştere a vorbirii

În capitolul anterior am arătat cum *transcrierile clipurilor audio* din baza de date de evaluare pot fi utilizate pentru a determina cât de potrivit este un sistem de recunoaştere a vorbirii pentru a transcrie vorbirea dintr-un anumit domeniu. Evaluarea completă a sistemului de recunoaştere a vorbirii se face comparând în mod automat transcrierile clipurilor audio de evaluare (numite în continuare transcrieri de referință) cu transcrierile rezultate în urma

procesului de decodare a clipurilor audio de evaluare (numite în continuare transcrieri ipotetice), aşa cum arată Figura 5.2.

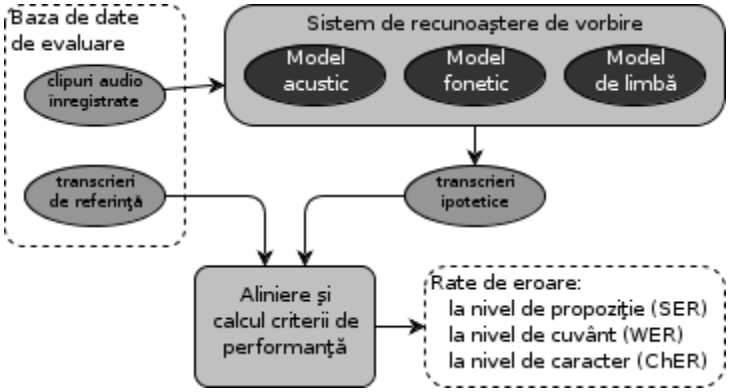


Figura 5.2. Evaluarea completă a sistemului de recunoaştere

Unul dintre criteriile de performaţă utilizate în evaluarea sistemelor de recunoaştere a vorbirii este *rata de eroare la nivel de propoziţie* (SER - sentence error rate). Rata de eroare la nivel de propoziţie se calculează ca raport între numărul de propoziţii eronate (propoziţii ce conţin cel puţin o greşeală) şi numărul total de propoziţii din transcrierea de referinţă:

$$SER[\%] = \frac{\#Propozitii\ eronate}{\#Propozitii\ \hat{i}n\ transcrierea\ de\ referin\check{t}\check{a}} \times 100 \qquad (5.2)$$

Acest criteriu de performanţă este relevant numai în cazul în care transcrierea ipotetică a unei propoziţii este inutilă dacă conţine un cuvânt greşit (de exemplu într-un sistem de recunoaştere de comenzi de 1-2 cuvinte sau într-un sistem de recunoaştere al unui CNP sau IBAN, etc.). În cele mai multe cazuri acest criteriu de performanţă nu este atât de relevant pentru că utilizatorul final poate înţelege mesajul transmis chiar şi dacă transcrierea conţine 10% cuvinte greşite (însă SER penalizează orice greşeală).

Având în vedere cele de mai sus, pentru evaluarea sistemelor de recunoaştere a vorbirii se foloseşte în mod uzual un alt criteriu de performanţă: *rata de eroare la nivel de cuvânt* (WER – word error rate). Pentru calculul WER, analiza celor două texte se face frază cu frază, în două etape: a) fraza ipotetică şi fraza de referinţă sunt aliniate printr-un algoritm care urmăreşte să minimizeze costul de editare al frazei ipotetice în vederea obţinerii frazei de referinţă, după care b) se numără toate erorile de recunoaştere la nivel de cuvânt (cuvinte inserate, cuvinte susbtituite, respectiv cuvinte şterse). După aliniere, WER se calculează pe baza acestor trei tipuri de erori utilizând formula:

$$WER[\%] = \frac{\#Erori\ de\ inserţie + \#Erori\ de\ substituţie + \#Erori\ de\ ştergere}{\#Cuvinte\ \hat{i}n\ transcrierea\ de\ referin\check{t}\check{a}} \times 100 \qquad (5.3)$$

Rata de eroare la nivel de cuvânt este mult mai adecvată pentru măsurarea performanţelor unui sistem de recunoaştere a vorbirii pentru că ea se calculează pe baza costului de editare al frazelor ipotetice. Totuşi aceast criteriu de performanţă nu ţine cont de

faptul că o eroare de substituție, de exemplu, poate fi reparată mult mai ușor dacă procentul de caractere eronate din cuvântul ipotetic este mic (ca în cazul “fotografiez” “fotografieze”), decât dacă procentul de caractere eronate din cuvântul ipotetic este mare (ca în cazul “fotografiez” “gravitez”). Din această cauză, în unele cazuri este mai adecvată folosirea ratei de eroare la nivel de caracter (ChER – character error rate) pentru evaluarea sistemelor de recunoaştere a vorbirii. Această rată de eroare se calculează în mod similar cu WER-ul, singura diferență fiind la nivelul unității de bază utilizate în comparație: caracterul, nu cuvântul:

$$ChER[\%] = \frac{\#Eroride\ inserție\ de\ caract + \#Eroride\ substituție\ de\ caract + \#Eroride\ ștergere\ de\ caract}{\#Caractere\ în\ transcrierea\ de\ referință} \times 100 \quad (5.4)$$

În concluzie, există trei criterii de performanţă utilizate în evaluarea sistemelor de recunoaştere a vorbirii: rata de eroare la nivel de propoziție (SER), rata de eroare la nivel de cuvânt (WER) şi rata de eroare la nivel de caracter (ChER). Dintre acestea, cel mai des folosit (criteriul standard) este WER-ul.

5.3.1 Evaluarea ratei de eroare la nivel de cuvânt: exemplu

Pachetul de programe standard utilizat în evaluarea sistemelor de recunoaştere a vorbirii este *NIST Scoring Toolkit (NIST SCTK)*. Din acest pachet de programe, aplicația *sclite* se poate folosi în mod explicit pentru alinierea transcrierile ipotetice şi transcrierile de referință şi calculul ratei de eroare la nivel de propoziție şi de cuvânt. In continuare este prezentată o selecție dintr-un raport de evaluare generat de această aplicație pentru o sarcină uzuală de recunoaştere de vorbire continuă:

SYSTEM SUMMARY PERCENTAGES by SPEAKER

testProject.cd_cont_4000_16-1-1.match.spk.rec								
SPKR	# Snt	# Wrd	Corr	Sub	Del	Ins	Err	S.Err
0101	100	1703	75.9	20.8	3.3	5.7	29.8	88.0
0201	100	1703	87.3	11.9	0.8	7.6	20.3	82.0
0301	100	1703	86.3	11.7	2.1	6.5	20.3	74.0
0401	100	1703	89.1	9.9	1.0	5.3	16.2	75.0
0601	100	1703	89.8	9.3	0.9	3.2	13.4	68.0
1001	100	1703	88.7	10.4	0.9	4.2	15.5	70.0
1601	100	1703	87.7	10.5	1.8	2.7	15.0	78.0
1701	100	1703	74.4	21.9	3.7	5.5	31.1	88.0
1801	100	1703	87.3	11.2	1.5	3.3	16.0	74.0
1901	100	1703	85.4	12.3	2.3	2.9	17.4	78.0
2001	100	1703	87.1	10.6	2.3	3.2	16.0	71.0
Sum/Avg	1100	18733	82.8	14.7	2.5	4.1	21.2	65.6
Mean	141.7	1522.1	81.7	15.6	2.6	4.1	22.4	69.0
S.D.	55.4	387.4	10.9	8.7	2.4	1.5	11.4	15.3
Median	100.0	1703.0	86.3	11.9	2.0	3.6	19.7	74.0

SENTENCE RECOGNITION PERFORMANCE

sentences		2976
with errors	65.6%	(1951)
with substitions	62.7%	(1865)
with deletions	18.5%	(551)
with insertions	26.5%	(789)

WORD RECOGNITION PERFORMANCE

Percent Total Error	= 21.2%	(6785)
Percent Correct	= 82.8%	(26482)
Percent Substitution	= 14.7%	(4684)
Percent Deletions	= 2.5%	(798)
Percent Insertions	= 4.1%	(1303)
Percent Word Accuracy	= 78.8%	
Ref. words	=	(31964)
Hyp. words	=	(32469)
Aligned words	=	(33267)

CONFUSION PAIRS Total (3142)
With >= 1 occurances (3142)

1: 40 -> da ==> fda
2: 21 -> bucata ==> bucată
3: 18 -> am ==> a

```
   4:   18  ->  da ==> la
   5:   18  ->  fotografiez ==> fotografieze
   6:   18  ->  returnez ==> returneze
   7:   18  ->  rezerv ==> rezerve
   8:   16  ->  da ==> dar
   9:   15  ->  cărămida ==> cărămidă
  10:   14  ->  o ==> rezervă
  ...
  ...
  ...

INSERTIONS                        Total                 (496)
                                  With >=  1 occurances (496)
   1:   67  ->  de
   2:   59  ->  cu
   3:   54  ->  şi
   4:   39  ->  a
   5:   39  ->  o
   6:   37  ->  în
   7:   34  ->  că
   8:   30  ->  nu
   9:   29  ->  să
  10:   24  ->  ce
  ...
  ...
  ...

SUBSTITUTIONS                     Total                 (1022)
                                  With >=  1 occurances (1022)
   1:  195  ->  da
   2:   97  ->  o
   3:   72  ->  de
   4:   72  ->  să
   5:   63  ->  la
   6:   63  ->  rezerv
   7:   51  ->  e
   8:   47  ->  în
   9:   46  ->  a
  10:   45  ->  pe
  ...
  ...
  ...

FALSELY RECOGNIZED                Total                 (1849)
                                  With >=  1 occurances (1849)
   1:  117  ->  în
   2:   99  ->  şi
   3:   98  ->  a
   4:   78  ->  o
   5:   74  ->  de
   6:   56  ->  la
   7:   46  ->  un
   8:   44  ->  să
   9:   41  ->  fda
  10:   38  ->  că
  ...
  ...
  ...
```

```
       id: (0401_0510)
       Scores: (#C #S #D #I) 4 1 0 3
       REF:  iaurturile ajută la prevenirea **** ** ** OSTEOPOROZEI
       HYP:  iaurturile ajută la prevenirea POST OP RO ZEI
       Eval:                                I    I  I  S
       ...
       ...
       ...
       id: (0401_0537)
       Scores: (#C #S #D #I) 8 3 0 1
       REF:  am coborât din autocar ***** VAL VâRTEJ şi am început să FOTOGRAFIEZ
       HYP:  am coborât din autocar BARBU TEI şI     şi am început să FOTOGRAFIEZE
       Eval:                         I     S   S                            S
       ...
       ...
       ...
       id: (0401_0556)
       Scores: (#C #S #D #I) 11 1 1 0
       REF:  aş vrea să afle cumva că GESTUL LUI      disperat nu ne-a fost
indiferent
       HYP:  aş vrea să afle cumva că ****** GESTULUI disperat nu ne-a fost
indiferent
       Eval:                           D      S
       ...
       ...
       ...
       id: (0602_0145)
       Scores: (#C #S #D #I) 4 3 0 2
       REF:  îMI PUTEţI sugera nişte opere interesante **** ** POSTBELICE
       HYP:  îL  PUTEM   sugera nişte opere interesante POST DE ELICE
       Eval: S    S                                      I    I  S
       ...
       ...
       ...
       id: (0603_0033)
       Scores: (#C #S #D #I) 0 1 0 0
       REF:  DA
       HYP:  FDA
       Eval: S
       ...
       ...
       ...
       id: (1601_0534)
       Scores: (#C #S #D #I) 10 2 0 0
       REF:  omul a aruncat CăRăMIDA  într-o fereastră a autocarului şi a STRIGAT
ceva
       HYP:  omul a aruncat CăRăMIDă într-o fereastră a autocarului şi a STRIGA
ceva
       Eval:               S                                                S
id: (1601_0535)
       Scores: (#C #S #D #I) 11 1 0 0
       REF:  apoi a încercat să desfăşoare BUCATA  de pânză pe care scria ceva
       HYP:  apoi a încercat să desfăşoare BUCATă de pânză pe care scria ceva
       Eval:                                S
       ...
       ...
       ...
```

În prima parte a raportului de evaluare este prezentat un tabel ce conține informații sumarizate privind evaluarea sistemului pentru fiecare dintre vorbitorii implicați în procesul de evaluare (câte un vorbitor pe fiecare dintre liniile tabelului). Pentru fiecare vorbitor sunt afişate numărul de clipuri audio de evaluare (coloana #Snt), numărul de cuvinte din aceste clipuri de evaluare (coloana #Wrd), procentul de cuvinte recunoscute corect (coloana Corr),

erorile procentuale de substituție, stergere şi inserție (coloanele Sub, Del şi Ins) şi, în final, ratele de eroare la nivel de cuvânt şi la nivel de propoziție (coloanele Err şi S.Err). Ultima parte a tabelului prezintă valorile medii pentru toate aceste criterii de performanță.

Tabelul de la începutul raportului de evaluare prezintă o multitudine de criterii de performanță care pot fi utilizate pentru caracterizarea sistemului şi optimizarea lui. De exemplu, dacă procentul de inserții este foarte mare comparativ cu procentul de cuvinte şterse, atunci ne putem aştepta să obținem rezultate mai bune dacă modificăm anumiți parametri specifici procesului de decodare: micşorarea wordInsertionProbability în acest caz. Informațiile din acest tabel pot fi utilizate şi pentru a ne face o idee despre "duritatea" SER, comparativ cu WER: pentru vorbitorul 0401, de exemplu, avem doar 16% erori la nivel de cuvânt, însă acestea duc la un SER de 75%!

Chiar dacă toate informațiile prezentate în tabelul de evaluare sunt foarte utile, în momentul în care se cere o informație succintă despre performanța sistemului de recunoaştere de vorbire cel mai relevant criteriu de performanță (criteriul standard, de altfel) este rata de eroare (medie) la nivel de cuvânt. În consecință, utilizând un singur număr, sistemul evaluat în exemplul de mai sus este caracterizat de WER de 21.2%. Astfel de evaluări succinte sunt foarte utile atunci când ne propunem să optimizăm sistemul de recunoaştere de vorbire şi dorim să prezentăm rezultatele obținute în urma procesului de optimizare. De exemplu, dacă variem doi dintre parametrii sistemului (să-i denumim alpha şi beta) şi dorim să punem în evidență dependența performanței sistemului de aceşti doi parametri, vom efectua o serie de

Tabelul 5.1. Exemplu de prezentare a rezultatelor unui proces de optimizare

WER [%]		**alpha**			
		100	**200**	**500**	**1000**
beta	**0.8**	18.3	14.9	11.4	13.2
	1.0	17.9	13.4	10.1	11.0
	1.2	16.2	13.5	8.6	12.0
	1.4	20.4	13.9	9.2	12.5

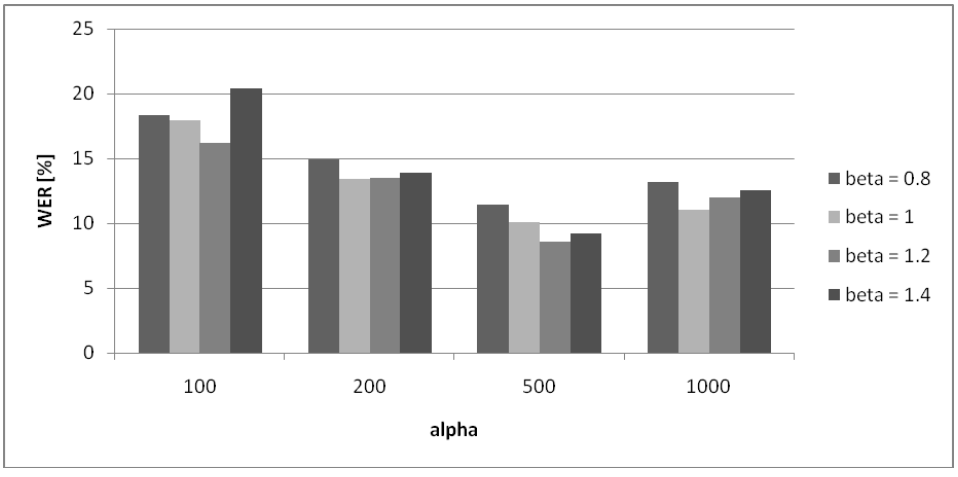


Figura 5.3. Exemplu de prezentare a rezultatelor unui proces de optimizare

experimente şi vom prezenta numai WER-ul obținut pentru fiecare experiment într-un tabel similar cu Tabelul 5.1 şi într-un grafic similar cu cel din Figura 5.3.

Evaluarea per vorbitor este şi ea foarte interesantă întrucât ne arată dacă performanța sistemului de recunoaştere variază în funcție de caracteristicile vorbitorului. În cazul ideal sistemul ar trebui să funcționeze la fel de bine pentru orice vorbitor, însă în exemplul prezentat nu se întâmplă acest lucru. Pentru a facilita analiza acestor rezultate, ele ar trebui sumarizate succint într-un tabel serparat (vezi Tabelul 5.2) şi reprezentate grafic (vezi Figura 5.4). Din grafic se observă foarte uşor că există o serie de vorbitori pentru care sistemul de recunoaştere are performanțe bune (aproximativ 15% WER), doi vorbitori pentru care se obțin valori medii (aproximativ 20% WER) şi doi vorbitori pe care sistemul îi recunoaşte foarte slab (WER de aproximativ 30%).

Tabelul 5.2. Exemplu de evaluare per vorbitor pentru un sistem de recunoaştere de vorbire

WER [%]		**Vorbitorul**											
		01	02	03	04	06	10	16	17	18	19	20	medie
Setul de test	CS_01	29.8	20.3	20.3	16.2	13.4	15.5	15.0	31.1	16.0	17.4	16.0	21.2

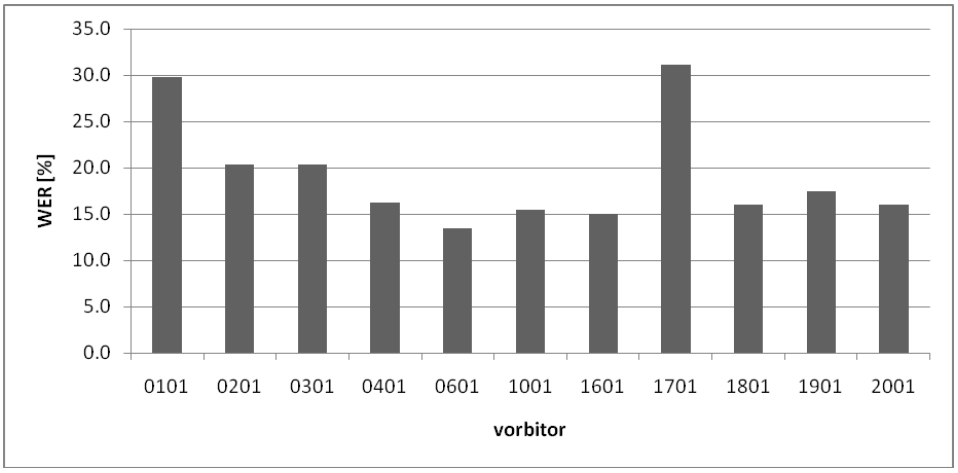


Figura 5.4. Exemplu de evaluare per vorbitor pentru un sistem de recunoaştere de vorbire

Raportul de evaluare continuă cu două secțiuni (*Sentence recognition performance* şi *Word recognition performance*) care sumarizează criteriile de performanță medii prezentate în tabelul anterior.

Urmează alte cinci secțiuni foarte interesante care prezintă statistici despre perechile de cuvinte confundate cel mai frecvent (*Confusion Pairs*), cuvintele inserate cel mai frecvent (*Insetions*), cuvintele şterse cel mai frecvent (*Deletions*), cuvintele substituite cel mai frecvent (*Substitutions*) şi cuvintele recunoscute fals cel mai frecvent (*Falsely Recognized*).

Analiza acestor cinci secțiuni poate furniza informații prețioase despre cele mai frecvente erori ale sistemului şi poate reprezenta baza viitoarelor îmbunătățiri. Secțiunea

Confusion Pairs prezentată în exemplu ne arată faptul că un procent important din erori se datorează apariției cuvântului "fda" în locul cuvântului „da" în transcrierile ipotetice. Acest cuvânt nu este de fapt un cuvânt valid al limbii române. Probabil că el a apărut în vocabularul sistemului şi în modelul de limbă ca urmare a unor erori statistice neidentificate în corpusurile de text de antrenare (pentru modelul de limbă). Exemplul evidențiază de asemenea că sunt confundate foarte frecvent formele de prezent la persoana I singular cu formele de prezent la persoana a doua singular („rezerv" -> „rezerve", „fotografiez" -> „fotografieze", etc.). Această eroare sistematică este tot un efect al utilizării unui model de limbă slab, care permite succesiuni de cuvinte de genul: "am început să fotografieze". O altă confuzie frecventă ilustrată de această secțiune este cea între substantivele feminine articulate hotărât, respectiv nehotărât („bucata" -> „bucată", „cărămida" -> „cărămidă", etc.). Ea este generată tot din cauza modelului de limbă, care permite succesiuni de cuvinte de genul: "să desfăşoare bucată de pânză" sau "a aruncat cu cărămidă". Iată cum secțiunea *Confusion Pairs* poate evidenția unele probleme (erori sistematice) ale sistemului de recunoaştere.

Secțiunea *Insertions* furnizează informații despre cele mai des întâlnite erori de inserție. Se poate observa că cuvintele inserate sistematic sunt cuvinte scurte de 1-2 foneme. Numărul erorilor de inserție poate fi scăzut prin mai multe metode:

- micşorarea parametrului de decodare wordInsertionProbability,
- introducerea unor cuvinte OOV în vocabularul sistemului. Astfel se evită erori de genul: "opere interesante *postbelice*" -> „opere interesante *post de elice*" sau „prevenirea *osteoporozei*" -> „prevenirea *post op ro zei*", apărute din cauza inexistenței cuvintelor "postbelice", respectiv "osteoporozei" în vocabular,
- îmbunătățirea modelului de limbă prin tratarea corectă a tuturor cuvintelor cu cratimă, etc. Astfel se evită erori de genul "roluri care-şi" -> „roluri care şi" apărute datorită probabilității mult mai mici a secvenței "care-şi", comparativ cu secvența „care şi",
- îmbunătățirea modelului acustic prin introducerea unor modele speciale pentru sunetele ce nu reprezintă vorbire (respirații, plescăieli, zgomote scurte), evitând astfel inserții de cuvinte scurte în locul acestora.

Secțiunea *Substitutions* furnizează informații despre cele mai des întâlnite erori de substituție sau, altfel spus, despre cele mai frecvente cuvinte substituite incorect (cu un alt cuvânt). În exemplul dat mai sus în topul listei se află cuvinte precum „da", „o", „de", „să", „la", etc. Aceast tip de eroare apare în principal din cauza probabilității scăzute pe care o atribuie modelul de limbă respectivelor cuvinte sau secvențelor de cuvinte formate cu acestea.

În cele din urmă, secțiunea *Falsely Recognized* listează cele mai frecvente cuvinte introduse în mod eronat (în locul altora). Aceast tip de eroare apare în principal din cauza probabilității mari pe care o atribuie modelul de limbă respectivelor cuvinte sau secvențelor

de cuvinte formate cu aceastea. Remedierea acestui tip de erori poate fi abordată într-un mod similar cu cea pentru erorile de inserție.

Ultima parte a raportului de evaluare conține detalierea tuturor erorilor întâlnite în transcrierile ipotetice. În acestă secțiune analiza erorilor se face pentru fiecare frază în parte astfel:

- trancrierea de referință este aliniată cu transcrierea ipotetică marcându-se toate erorile cu una dintre literele S (substituție), D (ştergere) sau I (inserție),
- numărul de cuvinte recunoscute corect (#C) şi numărul de erori de substituție (#S), ştergere (#D) şi inserție (#I) sunt prezentate sub denumirea *Scores*.

Această ultimă secțiune a raportului de evaluare este foarte utilă pentru identificarea concretă a erorilor sistematice evidențiate în secțiunile anterioare ale raportului. Pe baza alinierii celor două transcrieri se pot trage concluzii importante ce pot fi folosite în continuare pentru a îmbunătăți sistemul de recunoaştere de vorbire.

5.4 Concluzii

Modelul de limbă al unui sistem de recunoaştere de vorbire poate fi evaluat din punctul de vedere al compatibilității lui cu domeniul vorbirii utilizând numai transcrierile de referință ale clipurilor audio din baza de date de evaluare. În acest caz criteriile standard de evaluare sunt rata de cuvinte în afara vocabularului (OOV) şi perplexitatea (PPL).

Evaluarea completă a sistemului de recunoaştere a vorbirii se poate face numai decodând clipurile audio din baza de date de evaluare şi comparând transcrierile ipotetice rezultate cu transcrierile de referință. În acest caz criteriile standard de evaluare sunt rata de eroare la nivel de propoziție (SER), rata de eroare la nivel de cuvânt (WER) şi rata de eroare la nivel de caracter (ChER).

Analiza listei cuvintelor OOV şi a erorilor aparute in procesul de decodare (substituții, ştergeri şi inserții) este utilă în identificarea şi caracterizarea erorilor sistematice şi poate oferi informații prețioase pentru îmbunătățirea ulterioară a sistemului de recunoaştere a vorbirii.

6. CONSTRUCȚIA UNUI SISTEM SIMPLU DE RECUNOAŞTERE A VORBIRII CONTINUE

- recunoaşterea secvențelor audio ce conțin cifre -

6.1 Introducere

Procesul recunoaşterii vorbirii continue (RVC) vizează transformarea unui semnal audio ce conține vorbire într-o succesiune de cuvinte. Un sistem de recunoaştere a secvențelor audio de cifre este un sistem de recunoaştere a vorbirii continue cu vocabular redus, în sensul că singurele cuvinte ce pot fi recunoscute de către sistem sunt cele zece cifre ale sistemului zecimal: zero, unu, ..., nouă.

6.1.1 Arhitectura generală a unui sistem de recunoaştere a vorbirii continue - sumar

Arhitectura generală a unui sistem de recunoaştere automată a vorbirii (RAV) este prezentată în Figura 6.1. Din figură reies două lucruri esențiale în ceea ce priveşte procesul de recunoaştere a vorbirii: a) recunoaşterea se face utilizând o serie de parametri vocali extraşi din semnalul vocal (nu direct, utilizând mesajul vorbit) şi b) recunoaşterea se face pe baza unor modele (acustic, fonetic şi lingvistic) dezvoltate în prealabil.

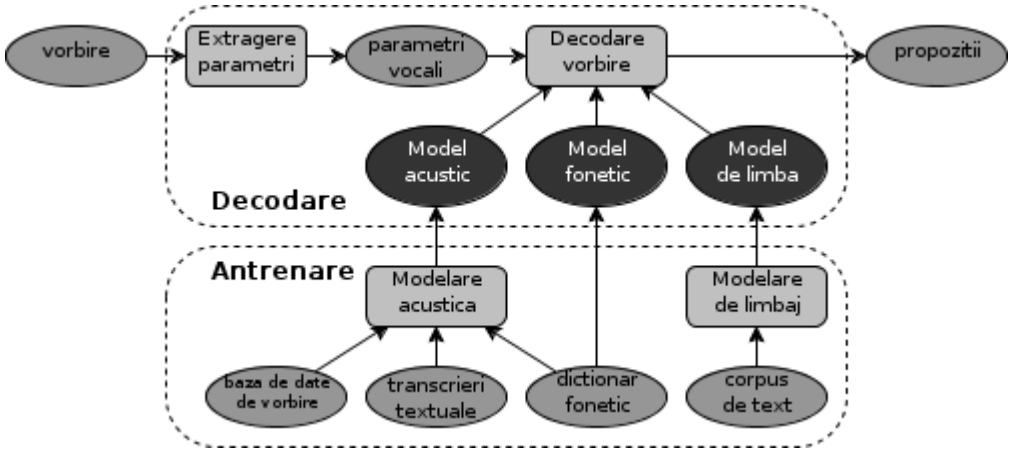


Figura 6.1. Arhitectura generală a unui sistem de recunoaştere a vorbirii

Modelul acustic are rolul de a estima probabilitatea unui mesaj vorbit, dată fiind o succesiune de cuvinte. În sistemele de RVC state-of-the-art modelul acustic nu foloseşte cuvinte ca unităţi acustice de bază pentru că: a) fiecare sarcină de RAV are un vocabular de cuvinte diferit pentru care nu există modele deja antrenate şi nici date de antrenare disponibile şi b) numărul de cuvinte diferite dintr-o limbă este prea mare. În locul cuvintelor, se utilizează unităţi acustice de bază sub-lexicale, de exemplu foneme, sau, mai recent, chiar unităţi sub-fonemice numite senone. Prin urmare, modelul acustic este format dintr-un set de modele pentru foneme (sau senone) care se conectează în timpul procesului de decodare pentru a forma modele pentru cuvinte şi apoi modele pentru succesiuni de cuvinte. Acestea sunt utilizate în final pentru a estima probabilitatea ca mesajul rostit de vorbitor să fie format dintr-o succesiune sau alta de cuvinte. Experienţa a arătat că această abordare generativă pentru modelele acustice poate fi foarte bine implementată utilizând modele Markov ascunse (hidden Markov models - HMM). Pentru mai multe informaţii despre modul cum funcţionează modelele acustice consultaţi bibliografia îndrumarului de proiect.

Modelul de limbă este utilizat în timpul decodării pentru a estima probabilităţile tuturor secvenţelor de cuvinte din spaţiul de căutare. În general, rolul unui model de limbă este de a estima probabilitatea ca o secvenţă de cuvinte $W = w_1, w_2, ..., w_n$, să fie o propoziţie validă a limbii. Probabilitatea acestor secvenţe de cuvinte ajută foarte mult modelul acustic în procesul de decizie. De exemplu, aceste două fraze: *casa ta e mare?* şi *casat a Ema re?* sunt similare din punct de vedere acustic, însă cea de-a doua nu are niciun sens. Rolul modelului de limbă este acela de a asigna o probabilitate mult mai mare primei secvenţe de cuvinte şi, prin urmare, de a ajuta sistemul de RAV să decidă în favoarea acesteia.

Modelul fonetic are rolul de a conecta modelul acustic (care estimează probabilităţile acustice ale *fonemelor*) cu modelul de limbă (care estimează probabilităţile secvenţelor de *cuvinte*). De cele mai multe ori modelul fonetic este un dicţionar de pronunţie care asociază fiecărui cuvânt din vocabular una sau mai multe secvenţe de foneme adecvate, reprezentând modul în care se poate pronunţa respectivul cuvânt.

Figura 6.1 ilustrează, de asemenea, procesele implicate în dezvoltarea unui sistem de RAV, dar şi resursele necesare creării modelelor acustice, lingvistice şi fonetice. Modelul acustic se construieşte strict pe baza unui set de clipuri audio înregistrate asociate cu transcrierea textuală a mesajelor vorbite şi a unui dicţionar fonetic ce cuprinde toate cuvintele din respectiva transcriere textuală. În cazul sistemelor de RVC cu vocabular extins se folosesc modele de limbă statistice, care se construiesc utilizând corpusuri de text de dimensiuni cât mai mari şi cât mai adaptate domeniului din care fac parte mesajele vorbite ce trebuie decodate. În sistemele de RVC cu vocabular redus (cum este şi sistemul pe care îl veţi dezvolta în continuare) se folosesc preponderent modele de limbă de tip gramatică cu stări finite, iar pentru construcţia acestora nu este nevoie de resurse textuale.

6.2 Înregistrarea unei baze de date de clipuri audio

Aşa cum am precizat mai sus, pentru a construi modelul acustic al unui sistem de recunoaştere a vorbirii este nevoie în primul rând de o bază de date de clipuri audio înregistrate. Pentru proiectul de recunoaştere de secvenţe audio ce conţin cifre veţi înregistra un set de 100 de clipuri audio a câte 12 cifre fiecare. Pentru înregistrarea clipurilor audio veţi folosi o aplicaţie de înregistrări accesibilă on-line la adresa http://speed.pub.ro/speech-recorder. Pentru a accesa această aplicaţie aveţi nevoie de numele de utilizator şi de parola care v-au fost comunicate la prima şedinţă de proiect.

Lansaţi aplicaţia de înregistrări

1. Conectaţi un microfon la calculator şi verificaţi ca volumul acestuia să fie la maxim.
   Atenţie: *Nu folosiţi pentru înregistrări microfonul încorporat al laptopului!*

2. Porniţi browser-ul şi încărcaţi pagina http://speed.pub.ro/speech-recorder.
   Notă: *În cazul în care aplicaţia nu se încarcă va trebui să instalaţi plug-in-ul Java pentru browser.*

3. Urmaţi instrucţiunile de pe primul ecran pentru a calibra microfonul şi sistemul audio.
   Notă: *Etapa de calibrare are ca scop configurarea corectă a setărilor microfonului în condiţii de linişte. Este foarte important ca înregistrările să se facă într-o cameră în care nu există surse de zgomot constant (de exemplu un ventilator sau un frigider), dar nici surse de zgomot intermitent (de exemplu un televizor, alte persoane care vorbesc, etc.).*

   În cazul în care aveţi probleme la etapa de calibrare, urmăriţi mesajele de eroare care apar şi încercaţi să modificaţi volumul microfonului, să vorbiţi mai tare/încet sau să utilizaţi setările speciale +10/20/30dB ale driverului audio. Dacă nu reuşiţi să realizaţi calibrarea contactaţi-l prin email pe îndrumătorul de proiect.

Conectaţi-vă la serverul de înregistrări

1. Conectaţi-vă la sistem folosind username-ul şi parola furnizate de îndrumătorul de proiect.

2. Creaţi un vorbitor nou selectând ***New Speaker*** din câmpul ***Select Speaker***. Completaţi apoi informaţiile cerute şi apăsaţi *Save* pentru a salva datele şi *I agree* (pentru a accepta termenii de utilizare ai aplicaţiei).
   Notă: *Această etapă este necesară numai dacă nu aţi creat în prealabil un vorbitor cu numele vostru.*

3. Selectaţi din câmpul ***Select Speaker*** numele vorbitorului nou creat.

Înregistraţi clipurile audio

1. Selectaţi din câmpul ***Phrase Group*** setul de fraze specificat de îndrumătorul de proiect.

Notă: *Aplicația va afişa fraza 1 a acestui grup de fraze şi statusul ei (not recorded yet).*

2. Apăsați butonul ***Record***, aşteptați o secundă şi apoi **pronunțați fraza cât mai natural**.

3. Apăsați butonul ***Stop*** pentru a încheia înregistrarea.
   Notă: *Dacă verificarea semnalului audio se încheie cu succes fraza va fi încărcată pe server, iar aplicația va afişa automat fraza următoare.*

4. Continuați cu înregistrările până când terminați de înregistrat toate frazele din acest set.
   Notă: *Dacă considerați că înregistrarea curentă nu a fost făcută corect (a apărut un zgomot neaşteptat, v-ați bâlbâit, ați respirat zgomotos, ați pronunțat un cuvânt în plus sau în minus, ați ezitat, etc.) ascultați înregistrarea apăsând butonul* ***Play last****. Dacă înregistrarea nu a fost făcută corect reveniți la fraza anterioară apăsând butonul < şi reînregistrați-o.*

5. După ca ați terminat de înregistrat toate frazele utilizați butoanele <, >, <<. >> şi ***Play*** pentru a parcurge şi reasculta toate clipurile audio, reînregistrandu-le pe cele care vi se par incorecte.

6. După ce ați efectuat toate înregistrările şi ați verificat corectitudinea lor contactați-l prin email pe îndrumătorul de proiect.

6.3 Crearea dicționarului fonetic şi a listei de foneme

Un dicționar fonetic este un instrument lingvistic care specifică modul în care se pronunță cuvintele unei limbi. Altfel spus, dicționarul fonetic face corespondența între forma scrisă şi forma fonetică a cuvintelor unei limbi. Forma fonetică a unui cuvânt este o succesiune de foneme (fonemul fiind unitatea de sunet fundamentală a unei limbi). Fonemele limbii române, codificarea internă utilizată în continuare şi câteva exemple de cuvinte transcrise fonetic au fost prezentate în Tabelul 2.1.

Într-un sistem de recunoaştere a vorbirii continue dicționarul fonetic are rolul de a face legătura între modelul acustic (care modelează modul de producere a *sunetelor specifice limbii*) şi modelul de limbă (care modelează modul în care se succed *cuvintele limbii*). În conecință dicționarul fonetic trebuie să conțină toate cuvintele posibile pentru o anumită sarcină de recunoaştere a vorbirii alături, bineînțeles, de transcrierile fonetice ale acestor cuvinte. Pentru mai multe informații despre rolul dicționarului fonetic în cadrul unui sistem de recunoaştere a vorbirii continue consultați bibliografia îndrumarului de proiect.

Pentru sarcina curentă (recunoaşterea secvențelor audio ce conțin cifre) dicționarul fonetic trebuie să conțină numai cele zece cifre ale sistemului zecimal: zero, unu, doi, trei,

patru, cinci, şase, şapte, opt şi nouă. Sistemul de recunoaştere a vorbirii continue CMU Sphinx utilizează dicționare fonetice în format text, ce conțin câte un cuvânt pe fiecare linie a fişierului. Transcrierea fonetică a fiecărui cuvânt trebuie să apară pe aceeaşi linie. Fonemele trebuie separate de câte un spațiu. Crearea dicționarului fonetic în formatul agreeat este descrisă mai jos.

Logați-vă la serverul de dezvoltare, apoi creați un director de lucru temporar:

```
mkdir phonetics
cd phonetics
```

Creați şi editați un nou fişier numit rodigits.dic utilizând editorul vim:

```
vi rodigits.dic
```

Utilizând Tabelul 1 editați acest fişier astfel încât să aveți pe fiecare linie câte o cifră alături de transcrierea ei fonetică. Câteva instrucțiuni despre modul de editare al unui fişier text utilizând editorul vim sunt prezentate aici. În final fişierul ar trebui să arate în felul următor:

```
zero z e r o
unu u n u
...
nouă n o1 w a1
```

Notă: în cazul în care nu puteți tasta caractere cu diacritice utilizând sistemul de operare şi tastatura calculatorului pe care lucrați, puteți copia denumirile cifrelor ce conțin diacritice din lista următoare: şase, şapte, nouă. În consolă se poate da paste utilizând combinația de taste ctrl+alt+V.

În cazul sistemelor de recunoaştere a vorbirii cu vocabular redus (cum este şi cel pe care tocmai îl construiți) manualul CMU Sphinx recomandă utilizarea de foneme specifice fiecărui cuvânt. Altfel spus ni se recomandă să modelăm diferit un acelaşi fonem X în funcție de cuvântul (şi de poziția în cuvânt) în care apare acesta. În consecință veți modifica în continuare fişierul rodigits.dic adaugând în numele fiecărui fonem numele cuvântului din care face parte, cât şi poziția fonemului în cuvânt (numai acolo unde este cazul). După această operație dicționarul fonetic ar trebui să arate astfel:

```
zero z_zero e_zero r_zero o_zero
unu u_unu1 n_unu u_unu2
...
nouă n_nouă o1_nouă w_nouă a1_nouă
```

Notă: în cazul în care nu puteți tasta caractere cu diacritice utilizând sistemul de operare şi tastatura calculatorului pe care lucrați, puteți copia denumirile cifrelor ce conțin

diacritice din lista următoare: şase, şapte, nouă. În consolă se poate da paste utilizând combinația de taste ctrl+alt+V.

În continuare trebuie să creăm lista tuturor fonemelor ce vor fi modelate de modelul acustic. Este necesar sa scriem un fişier text care să conțină câte un fonem pe fiecare linie, iar liniile trebuie să fie sortate în ordine alfabetică. Crearea acestui fişier se poate face manual (parcurgând dicționarul fonetic creat anterior) sau automat (urmând intrucțiunile de mai jos).

Creați un nou fişier (rodigits.phones.temp) care să conțină toate cuvintele şi toate fonemele din dicționarul fonetic (câte un cuvânt/fonem pe linie):

```
sed 's/ /\n/g' rodigits.dic > rodigits.phones.temp
```

Din această listă selectați numai fonemele (profitând de faptul că ele conțin caracterul "_") şi listați-le într-un nou fişier numit rodigits.phones.temp.onlyPhones:

```
grep "_" rodigits.phones.temp > rodigits.phones.temp.onlyPhones
```

Adăugați la sfârşitul acestui fişier "fonemul" SIL (acest "fonem" este utilizat pentru a modela zonele de linişte):

```
echo "SIL" >> rodigits.phones.temp.onlyPhones
```

Sortați fonemele în ordine alfabetică, redenumiți fişierul rezultat şi ştergeți toate celelalte fişiere temporare:

```
sort -u rodigits.phones.temp.onlyPhones > rodigits.phones.temp.onlyPhones.sorted
mv rodigits.phones.temp.onlyPhones.sorted rodigits.phones
rm rodigits.phones.temp*
```

6.4 Antrenarea modelului acustic

Antrenarea modelului acustic se va face în cadrul unui proiect CMU Sphinx şi, aşa cum am precizat în prima secțiune, presupune existența următoarelor resurse:

- un set de clipuri audio ce conțin vorbire (set pe care l-ati înregistrat în prealabil),
- transcrierea textuală corespunzătoare cuvintelor pronunțate în clipurile audio (deja existentă),
- un dicționar fonetic cuprinzând toate cuvintele (dicționar pe care l-ati creat în prealabil),
- un dicționar cu elemente acustice care nu sunt foneme (linişte, tuse, râs, muzică, etc.);

6.4.1 Crearea unui proiect CMU Sphinx numit rodigits

Logați-vă pe serverul de dezvoltare şi creați un director numit projects în care veți crea ulterior toate proiectele:

```
cd ~
mkdir projects
```

Intrați in directorul projects şi creați un nou director numit rodigits pentru proiectul curent. Apoi intrați în directorul rodigits:

```
cd ~/projects
mkdir rodigits
cd ~/projects/rodigits
```

Configurați directorul rodigits pentru a putea fi utilizat ca proiect CMU Sphinx:

```
sphinxtrain_biosinf -t rodigits setup
mkdir logdir
```

6.4.2 Adăugarea resurselor necesare în directorul proiectului

Logați-vă pe serverul de dezvoltare şi intrați în directorul proiectului:

```
cd ~/projects/rodigits
```

Copiați clipurile audio înregistrate în prealabil în directorul wav (important: modificați comanda următoare pentru a include propriul SPEAKER_ID, aşa cum vi l-a specificat îndrumătorul de proiect):

```
mkdir wav
cp -r /home/biosinfShare/resources/wav/SPEAKER_ID/ ~/projects/rodigits/wav/
```

Copiați transcrierea textuală a cuvintelor pronunțate în clipurile audio în directorul etc:

```
cp /home/biosinfShare/resources/transcription/rodigits.data ~/projects/rodigits/etc/
```

Copiați dicționarul fonetic şi lista de foneme create în prealabil în directorul etc:

```
cp ~/phonetics/rodigits.dic ~/projects/rodigits/etc/
cp ~/phonetics/rodigits.phones ~/projects/rodigits/etc/
```

Copiați dicționarul cu elemente acustice care nu sunt foneme în directorul etc:

```
cp /home/biosinfShare/resources/phonetic/rodigits.filler ~/projects/rodigits/etc/
```

Copiați scripturile createFileIds.sh şi createTranscriptions.sh în directorul etc:

```
cp -r /home/biosinfShare/scripts/createFileIds.sh ~/projects/rodigits/etc/
cp -r /home/biosinfShare/scripts/createTranscriptions.sh ~/projects/rodigits/etc/
```

6.4.3 Transformarea fişierelor necesare procesului de antrenare în formatul corespunzător

Înainte de a putea începe antrenarea modelului acustic, trebuie creat un fişier cu lista clipurilor audio ce vor fi folosite la antrenare şi un alt fişier cu transcrierea textuală a respectivelor clipuri audio. Din motive de eficienţă vom crea tot acum şi lista clipurilor audio ce vor fi folosite la evaluare, cât şi fişierul cu transcrierea textuală a acestora.

Logaţi-vă pe serverul de dezvoltare şi intraţi în directorul proiectului, în subdirectorul etc:

```
cd ~/projects/rodigits/etc/
```

Vizualizaţi scriptul createFileIds.sh utilizând editorul vim:

```
vi createFileIds.sh
```

Intraţi în modul de editare (tastând “i”) şi schimbaţi valoarea variabilei SPEAKER_ID pentru a utiliza propriul SPEAKER_ID, aşa cum vi l-a specificat îndrumătorul de proiect. Ieşiţi din modul de editare (tastând Esc), salvaţi modificările (tastând “:w” şi apoi Enter) şi părăsiţi editorul (tastând “:q” şi apoi Enter).

Creaţi lista tuturor clipurilor audio înregistrate:

```
./createFileIds.sh > rodigits.fileids.all
```

Creaţi lista clipurilor audio ce vor fi utilizate pentru antrenare selectând primele 50 şi ultimele 30 de clipuri audio din lista tuturor clipurilor (rodigits.fileids.all):

```
head -50 rodigits.fileids.all > rodigits.fileids.train
tail -30 rodigits.fileids.all >> rodigits.fileids.train
```

Celelalte clipuri audio vor fi utilizate pentru evaluare. Creaţi un fişier cu lista acestora:

```
diff rodigits.fileids.train rodigits.fileids.all | grep '>' | sed 's/> //' > rodigits.fileids.test
```

Vizualizaţi scriptul createTranscriptions.sh utilizând editorul vim:

```
vi createTranscriptions.sh
```

Intraţi în modul de editare (tastând “i”) şi schimbaţi valoarea variabilei SPEAKER_ID pentru a utiliza propriul SPEAKER_ID, aşa cum vi l-a specificat îndrumătorul de proiect. Ieşiţi din modul de editare (tastând Esc), salvaţi modificările (tastând “:w” şi apoi Enter) şi părăsiţi editorul (tastând “:q” şi apoi Enter).

Creaţi transcrierea textuală (în formatul corespunzător cerut de CMU Sphinx) pentru toate clipurile audio:

```
./createTranscriptions.sh > rodigits.transcription.all
```

Creaţi un fişier care să conţină numai transcrierea textuală pentru clipurile audio ce vor utilizate pentru antrenare selectând primele 50 şi ultimele 30 de linii din transcrierea textuală:

```
head -50 rodigits.transcription.all > rodigits.transcription.train
tail -30 rodigits.transcription.all >> rodigits.transcription.train
```

Celelalte linii din transcrierea textuală vor fi utilizate pentru evaluare. Creaţi un fişier cu lista acestora:

```
diff rodigits.transcription.train rodigits.transcription.all | grep '>' | sed 's/> //' >
rodigits.transcription.test
```

6.4.4 Configurarea procesului de antrenare

Orice proiect de recunoaştere de vorbire CMU Sphinx are un fişier de configurare se specifică parametrii modelului acustic ce va fi antrenat şi locaţia resurselor necesare în procesul de antrenare. Pentru primul proiect pe care îl veţi face nu veţi pastra valorile implicite ale parametrilor modelului acustic, specificând numai locaţia resurselor.

Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului, în subdirectorul etc şi vizualizaţi fişierul de configurare al proiectului cu ajutorul editorului vim:

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificaţi în acest fişier liniile care specifică locaţia resurselor utilizate în procesul de antrenare (dicţionarul fonetic, lista de foneme, dicţionarul de sunete care nu sunt foneme, lista de clipuri audio de antrenare şi transcrierea textuală a acestor clipuri audio):

```
$CFG_DICTIONARY    = "$CFG_LIST_DIR/$CFG_DB_NAME.dic";
$CFG_RAWPHONEFILE  = "$CFG_LIST_DIR/$CFG_DB_NAME.phone";
$CFG_FILLERDICT    = "$CFG_LIST_DIR/$CFG_DB_NAME.filler";
$CFG_LISTOFFILES   = "$CFG_LIST_DIR/${CFG_DB_NAME}_train.fileids";
$CFG_TRANSCRIPTFILE = "$CFG_LIST_DIR/${CFG_DB_NAME}_train.transcription";
```

Intraţi în modul de editare (tastând “i”) şi modificaţi aceste linii astfel:

```
$CFG_DICTIONARY    = "$CFG_LIST_DIR/rodigits.dic";
$CFG_RAWPHONEFILE  = "$CFG_LIST_DIR/rodigits.phones";
$CFG_FILLERDICT    = "$CFG_LIST_DIR/rodigits.filler";
$CFG_LISTOFFILES   = "$CFG_LIST_DIR/rodigits.fileids.train";
$CFG_TRANSCRIPTFILE = "$CFG_LIST_DIR/rodigits.transcription.train";
```

Ieşiţi din modul de editare (tastând Esc), salvaţi modificările (tastând “:w” şi apoi Enter) şi apoi închideţi editorul (tastând “:q” şi apoi Enter).

6.4.5 Crearea fişierelor cu coeficienţi MFCC corespunzătoare clipurilor audio de antrenare

Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului şi creaţi fişierele cu coeficienţi cepstrali executând următoarea comandă:

```
cd ~/projects/rodigits/
/usr/local/sphinx/lib/sphinxtrain/scripts/000.comp_feat/slave_feat.pl
```

Notă: este normal ca în urma execuţiei comenzii de mai sus să apară eroarea `Failed to open control … : No such file or directory at …/make_feats.pl line 89`. Ea ne atenţionează asupra faptului că lista de fişiere audio de evaluare nu a fost găsită (acest lucru este normal, pentru că această listă nu a fost încă generată).

Execuţia acestei comenzi are ca efect crearea a 80 de fişiere cu coeficienţi cepstrali corepunzătoare celor 80 de clipuri audio de antrenare. Verificaţi faptul că fişierele cu coeficienţi cepstrali au fost create corect (important: modificaţi comenzile următoare pentru a include propriul `SPEAKER_ID`, aşa cum vi l-a specificat îndrumătorul de proiect):

```
ls feat/SPEAKER_ID/*
ls feat/SPEAKER_ID/* | wc -l
ls wav/SPEAKER_ID/* | wc -l
```

Prima comandă din grupul celor de mai sus va afişa toate fişierele din subdirectorul `feat/SPEAKER_ID`. A doua comandă va afişa numărul de fişiere din respectivul subdirector (rezultatul ar trebui să fie 80 pentru că doar 80 de clipuri audio se folosesc pentru antrenare). A treia comandă va afişa numarul de fişiere din subdirectorul `wav/SPEAKER_ID` (rezultatul ar trebui să fie 100 pentru că în total ar trebui să aveţi 100 de clipuri audio înregistrate).

6.4.6 Antrenarea modelului acustic

Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului şi porniţi antrenarea modelului acustic executând următoarea comandă:

```
cd ~/projects/rodigits/
sphinxtrain_biosinf run
```

Aşteptaţi terminarea procesului de antranare (ar trebui să dureze aproximativ 2 minute). În cazul în care apar erori verificaţi faptul că fişierele cu resurse sunt în formatul corespunzător şi că nu aţi modificat greşit fişierul de configurare. În cazul în care nu reuşiţi să identificaţi şi să rezolvaţi problema discutaţi cu îndrumătorul de proiect.

6.5 Crearea modelului de limbă (gramaticii)

Sistemele de recunoaştere de vorbire cu vocabular mare de ultimă generaţie utilizează modele de limbă statistice de tip n-gram. Aceste modele de limbă se construiesc pe baza unor corpusuri mari de text specific sarcinii de recunoaştere, estimând probabilităţile de apariţie ale cuvintelor şi secvenţelor de cuvinte specifice respectivei sarcini. Modelele de limbă de tip n-gram se utilizează apoi în procesul de decodare (recunoaştere a vorbirii) pentru a calcula probabilităţile secvenţelor de cuvinte propuse de modelul acustic. Pentru mai multe informaţii despre modele de limbă statistice şi despre rolul modelului de limbă în cadrul unui sistem de recunoaştere a vorbirii continue consultaţi bibliografia îndrumarului de proiect.

Sarcina de recunoaştere a secvenţelor audio ce conţin cifre este o sarcină de recunoaştere cu vocabular foarte redus pentru care nu se pretează modelele de limbă statistice. Mai mult, cifrele şi succesiunile de cifre din cadrul clipurilor audio înregistrate apar cu probabilităţi aproximativ egale (nu se poate afirma că o cifră este pronunţată sistematic mai des decât alta). În aceste condiţii, modelele de limbă de tip gramatică cu stări finite (FSG – finite state grammar) sunt mult mai potrivite. O gramatică cu stări finite este un model de tip graf în care nodurile reprezintă cuvinte ale limbii, iar tranziţiile între cuvinte sunt reprezentate de arcele grafului. Un astfel de model de limbă specifică în mod explicit toate secvenţele de cuvinte permise de gramatica sarcinii de recunoaştere. Mai mult, fiecărui arc în parte îi poate fi asignat un cost specificând astfel probabilitatea ca un cuvânt să fie precedat de un altul (sau altfel spus probabilitatea respectivei secvenţe de două cuvinte).

În **Figura 6.2** este reprezentată grafic gramatica cu stări finite a sarcinii noastre de recunoaştere. Se poate observa că modelul este format din 14 noduri, dintre care numai zece reprezintă cuvinte ale limbii (cifrele de la zero la nouă), iar celelalte patru sunt utilizate pentru a realiza „intrarea”, respectiv „ieşirea” din graf şi pentru tranziţia înapoi. Tranziţiile arată modul în care se poate parcurge această gramatică şi implicit secvenţele de cuvinte permise: în fiecare clip audio se pot pronunţa una sau mai multe cifre.

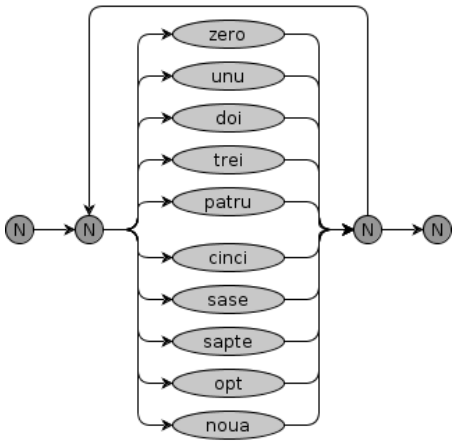


Figura 6.2. Gramatica cu stări finite a sarcinii de recunoaştere rodigits

Implementarea unei astfel de gramatici FSG poate fi făcută foarte simplu utilizând formatul Java Speech Gramar (JSGF) aşa cum vom vedea în continuare.

Logați-vă pe serverul de dezvoltare, intrați în directorul proiectului, în subdirectorul etc şi creați fişierul rodigits.jsgf utilizând editorul vim:

```
cd ~/projects/rodigits/etc/
vi rodigits.jsgf
```

Intrați în modul de editare (tastând "i") şi scrieți următoarele trei linii:

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

Ieşiți din modul de editare (tastând Esc), salvați modificările (tastând ":w" şi apoi Enter) şi apoi închideți editorul (tastând ":q" şi apoi Enter). Prima linie pe care ați scris-o în fişier specifică numele gramaticii, iar cea de-a doua linie specifică cuvintele şi succesiunile de cuvinte permise astfel: numbers este *zero* sau *unu* sau ... *nouă* de oricâte ori. Sau-ul este specificat prin carecterul |, iar faptul că cifrele se pot repeta de oricâte ori este specificat de caracterul *. Pentru mai multe detalii despre formatul Java Speech Grammar şi modul în care puteți crea şi alte tipuri de gramatici cu stări finite consultați site-ul JSGF precizat în bibliografia îndrumarului de proiect.

Transformați (cu ajutorul utilitarului sphinx_jsgf2fsg) gramatica sarcinii de recunoaştere din format JSGF (Java Speech Grammar Format) în formatul intern FSG (Finite State Grammar) utilizat de CMU Sphinx:

```
sphinx_jsgf2fsg -jsgf rodigits.jsgf -fsg rodigits.fsg
```

Vizualizați fişierul rezultat (rodigits.fsg) utilizând editorul vim:

```
vi rodigits.fsg
```

Observați modul în care se specifică nodurile, tranzițiile şi probabilitățile de tranziție ale gramaticii cu stări finite în formatul intern uztilizat CMU Sphinx. Ca exercițiu, desenați pe o bucată de hârtie graful definit in acest fişier şi comparați-l cu graful prezentat în **Figura 6.2**.

6.6 Evaluarea sistemului de recunoaştere şi interpretarea rezultatelor

Cele trei componente fundamentale ale unui sistem de recunoaştere a vorbirii (modelul acustic, modelul de limbă şi modelul fonetic) sunt în acest moment disponibile. În consecință se poate trece la decodarea clipurilor audio de evaluare şi apoi la compararea

transcrierii textuale rezultate cu transcrierea textuală de referinţă în vederea evaluării sistemului de recunoaştere de secvenţe audio ce conţin cifre.

6.6.1 Configurarea procesului de decodare (şi evaluare)

Pentru a configura procesul de decodare vom edita acelaşi fişier de configurare care a fost modificat şi pentru antrenare (sphinx_train.cfg). De această dată nu ne vom concentra asupra parametrilor de antranare, ci asupra parametrilor de decodare.

Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului, în subdirectorul etc şi vizualizaţi fişierul de configurare al proiectului cu ajutorul editorului vim:

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificaţi în acest fişier linia care specifică numele scriptului utilizat pentru decodare:

```
$DEC_CFG_SCRIPT = 'psdecode.pl';
```

Intraţi în modul de editare (tastând “i”) şi modificaţi acestă linie astfel:

```
$DEC_CFG_SCRIPT = 'psdecode_fsg.pl';
```

Identificaţi apoi liniile care specifică numele şi locaţia resurselor utilizate în procesul de decodare (dicţionarul fonetic, dicţionarul de sunete care nu sunt foneme, lista de clipuri audio de evaluare şi transcrierea textuală a acestor clipuri audio) şi numele subdirectorului în care se vor salva rezultatele:

```
$DEC_CFG_DICTIONARY    = "$CFG_BASE_DIR/etc/$CFG_DB_NAME.dic";
$DEC_CFG_FILLERDICT    = "$CFG_BASE_DIR/etc/$CFG_DB_NAME.filler";
$DEC_CFG_LISTOFFILES   = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.fileids";
$DEC_CFG_TRANSCRIPTFILE = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}_test.transcription";
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result";
```

Modificaţi aceste linii astfel:

```
$DEC_CFG_DICTIONARY    = "$CFG_BASE_DIR/etc/rodigits.dic";
$DEC_CFG_FILLERDICT    = "$CFG_BASE_DIR/etc/rodigits.filler";
$DEC_CFG_LISTOFFILES   = "$CFG_BASE_DIR/etc/rodigits.fileids.test";
$DEC_CFG_TRANSCRIPTFILE = "$CFG_BASE_DIR/etc/rodigits.transcription.test";
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_200_8";
```

Identificaţi apoi linia care specifică numele modelului de limbă ce va fi utilizat la decodare:

```
$DEC_CFG_LANGUAGEMODEL  = "$CFG_BASE_DIR/etc/${CFG_DB_NAME}.lm.DMP";
```

Şi modificaţi-o astfel:

```
$DEC_CFG_LANGUAGEMODEL  = "$CFG_BASE_DIR/etc/rodigits.fsg";
```

Identificați apoi linia care specifică aplicația cu care se va face alinierea textului rezultat în urma recunoașterii cu textul de referință (în vederea evaluării ratei de eroare la nivel de cuvânt):

```
$DEC_CFG_ALIGN = "builtin";
```

Și modificați-o astfel:

```
$DEC_CFG_ALIGN = "sclite";
```

Ieșiți din modul de editare (tastând Esc), salvați modificările (tastând ":w" și apoi Enter) și apoi închideți editorul (tastând ":q" și apoi Enter).

6.6.2 Crearea fișierelor cu coeficienți MFCC corespunzătoare clipurilor audio de evaluare

Logați-vă pe serverul de dezvoltare, intrați în directorul proiectului și creați fișierele cu coeficienți cepstrali executând următoarea comandă:

```
cd ~/projects/rodigits/
/usr/local/sphinx/lib/sphinxtrain/scripts/000.comp_feat/slave_feat.pl
```

Execuția acestui script are ca efect crearea a 100 de fișiere cu coeficienți cepstrali corepunzătoare tuturor clipurilor audio înregistrate (atât cele utilizate pentru antrenare, cât și cele utilizate pentru evaluare). Verificați faptul că fișierele cu coeficienți cepstrali au fost create corect (important: modificați comenzile următoare pentru a include propriul SPEAKER_ID, așa cum vi l-a specificat îndrumătorul de proiect):

```
ls feat/SPEAKER_ID/*
ls feat/SPEAKER_ID/* | wc -l
ls wav/SPEAKER_ID/* | wc -l
```

Prima comandă din grupul celor de mai sus va afișa toate fișierele din subdirectorul feat/SPEAKER_ID. A doua comandă va afișa numărul de fișiere din respectivul subdirector (rezultatul ar trebui să fie 100 pentru că în total sunt 100 de clipuri audio înregistrate). A treia comandă va afișa numarul de fișiere din subdirectorul wav/SPEAKER_ID (rezultatul ar trebui să fie tot 100).

6.6.3 Decodarea clipurilor audio de evaluare

Logați-vă pe serverul de dezvoltare, intrați în directorul proiectului și porniți procesul de evaluare executând următoarea comandă:

```
cd ~/projects/rodigits/
/usr/local/sphinx/lib/sphinxtrain/scripts/decode/slave.pl
```

Aşteptaţi terminarea procesului de decodare (ar trebui să dureze aproximativ 1 minut). În cazul în care apar erori încercaţi să identificaţi problema vizualizand fişierul de log al procesului de decodare:

```
vi logdir/decode/rodigits-1-1.log
```

În cazul în care nu reuşiţi să identificaţi şi să rezolvaţi problema discutaţi cu îndrumătorul de proiect. În cazul în care procesul de decodare se încheie cu succes ar trebui ca pe ecran să vă apară un mesaj similar cu acesta:

```
MODULE: DECODE Decoding using models previously trained (decode script:
psdecode_fsg.pl)
    Decoding 20 segments starting at 0 (part 1 of 1)
    0%
    Aligning results to find error rate
    SENTENCE ERROR: 50.0% (10/20)   WORD ERROR RATE: 12.9% (30/240)
```

6.6.4 Vizualizarea şi interpretarea rezultatelor evaluării

Evaluarea unui sistem de recunoaştere a vorbirii se face comparând în mod automat cele două transcrieri textuale ale clipurilor audio de evaluare: transcrierea textuală de referinţă (cea despre care se ştie a priori că este corectă) şi transcrierea textuală ipotetică (cea rezultată în urma procesului de decodare). Analiza se face frază cu frază, în două etape: a) fraza ipotetică şi fraza de referinţă sunt aliniate printr-un algoritm care urmăreşte să minimizeze numărul de erori de transcriere, după care b) se numără toate erorile de recunoaştere la nivel de cuvânt (cuvinte inserate, cuvinte susbtituite, respectiv cuvinte şterse). Se disting astfel două criterii de performanţă standard folosite în evaluarea sistemelor de recunoaştere a vorbirii continue: rata de eroare la nivel de propoziţie (SER - sentence error rate) şi rata de eroare la nivel de cuvânt (WER – word error rate). Rata de eroare la nivel de propoziţie se calculează ca raport între numărul de propoziţii corecte (propoziţii ce nu conţin nicio greşeală) şi numărul total de propoziţii. Rata de eroare la nvel de cuvânt se calculează cu formula:

$$WER[\%] = \frac{\#Erori\ de\ insertie + \#Erori\ de\ substitutie + \#Erori\ de\ ştergere}{\#Cuvinte\ în\ transcrierea\ de\ referinţă} \times 100 \qquad (6.1)$$

Aşa cum se observă din mesajul afişat în urma execuţiei procesului de decodare (dat ca şi exemplu mai sus), în urma evaluării transcrierii ipotetice s-a obţinut o rată de eroare la nivel de propoziţie de 50% şi o rată de eroare la nivel de cuvânt de 12.9%. Mai multe informaţii despre rezultatele evaluării sistemului de recunoaştere puteţi afla vizualizând întreg raportul de evaluare:

```
cd ~/projects/rodigits/
vi result.cd_cont_200_8/rodigits.align
```

Primul tabel din raportul de evaluare afişează pe câte o linie sumarul rezultatelor pentru diverşii vorbitori din baza de date de evaluare. În cazul de faţă aveţi un singur vorbitor în baza de date, deci o singură linie în tabel. Pe coloane aveţi mai multe informaţii: id-ul vorbitorului, numărul de propoziţii de evaluare, numărul de cuvinte de evaluare, procentul de cuvinte recunoscute corecte, procentul de cuvinte substituite/şterse/inserate, rata de eroare la nivel de cuvânt şi rata de eroare la nivel de propoziţie.

La sectiunea *Sentence Recognition Performance* se oferă mai multe detalii cu privire la erorile la nivel de propoziţie, iar la secţiunea *Word Recognition Performance* se oferă mai multe detalii cu privire la erorile la nivel de cuvânt.

Urmează alte cinci secţiuni foarte interesante care prezintă statistici despre perechile de cuvinte confundate cel mai frecvent (Confusion Pairs), cuvintele inserate cel mai frecvent (*Insetions*), cuvintele şterse cel mai frecvent (*Deletions*), cuvintele substituite cel mai frecvent (*Substitutions*) şi cuvintele recunoscute fals cel mai frecvent (*Falsely Recognized*).

În final, raportul de evaluare prezintă toate frazele ipotetice aliniate la frazele de referinţă corespunzătoare specificând, de asemenea, şi tipurile de erori apărute la fiecare frază în parte.

Toate aceste informaţii şi rezultate din raportul de evaluare al sistemului de recunoaştere vor fi comparate cu rezultatele ce vor fi obţinute în secţiunea următoare şi vor fi în final prezentate şi discutate în raportul final al proiectului de cercetare-dezvoltare.

6.7 Optimizarea modelului acustic

Înainte de a putea face câteva optimizări simple ale modelului acustic trebuie să studiaţi şi să înţelegeţi câteva aspecte importante privind modul de construcţie al unui astfel de model:

- arhitectura generala a modelului acustic (platforma HMM-GMM). Informaţii despre acest subiect găsiţi în bibliografia îndrumarului de proiect: [3] pag. 31, [4] pag. 23, [5] pag. 8, [6] pag. 306.
- alegerea unităţilor de vorbire (foneme, trifoneme, senone) şi dependenţa/independenţa de context. Informaţii despre aceste subiecte găsiţi în bibliografia îndrumarului de proiect: [3] pag. 33, [5] pag. 52, [6] pag. 310.

Utilizând deja noţiunile amintite mai sus (şi explicate pe larg în bibliografie), vom sumariza în continuare modul particular în care toolkitul de recunoaştere a vobirii pe care îl folosim (CMU Sphinx) realizează antrenarea modelului acustic. În mod implicit modelul acustic creat de CMU Sphinx este construit cu unităţi de vorbire dependente de context de tip trifonem (fonem căruia i se precizează vecinii stânga-dreapta). Tot în mod implicit aceste trifoneme sunt implementate ca HMM-uri cu trei stări emisive, fiecare având o distribuţie de

probabilitate de ieşire impementată cu un GMM. Fiecare astfel de stare-model se numeste *senone* şi ea poate fi comună mai multor trifoneme (în funcție de similaritățile dintre ele: de exemplu trifonemele p-a-r şi t-a-v pot avea senone-ul inițial comun). În mod implicit numărul de senone este configurat la 200, iar numărul de densități de probabilitate pentru fiecare GMM este configurat la 8.

În consecință modelul acustic antrenat în prealabil modelează cele câteva foneme specificate în dicționarul fonetic utilizând modele fonetice dependente de context având în total 200 de senone, fiecare senone folosind câte 8 densități Gaussiene de probabilitate pentru a modela parametri acustici de ieşire. Procesul de antrenare implementat de CMU Sphinx conduce la obținerea acestui model acustic astfel:

1. inițializarea modelelor fonetice independente de context (câte un model cu câte trei senone folosind câte o singură densitate Gaussiană pentru fiecare *fonem* în parte)
2. antrenarea modelelor fonetice (independente de context) rezultate la pasul anterior
3. transformarea modelelor fonetice independente de context în modele fonetice dependente de context (se obține câte un model cu câte trei senone folosind câte o singură densitate Gaussiană pentru fiecare *trifonem* în parte; în total modelul acustic poate avea *oricâte senone*!)
4. antrenarea modelelor fonetice (dependente de context) rezultate la pasul anterior
5. legarea stărilor (senonelor) modelelor fonetice dependente de context pe baza similarităților dintre ele (se obține câte un model cu câte trei senone folosind câte o singură densitate Gaussiană pentru fiecare trifonem în parte; în total modelul acustic va avea *200 senone*!)
6. antrenarea modelelor fonetice (dependente de context) rezultate la pasul anterior
7. dublarea numărului de densități Gaussiene în cadrul fiecărei senone (se obține câte un model cu câte trei senone folosind câte *două densități Gaussiane* pentru fiecare trifonem în parte; în total modelul acustic va avea 200 senone!)
8. antrenarea modelelor fonetice (dependente de context) rezultate la pasul anterior
9. dublarea numărului de densități Gaussiene în cadrul fiecărei senone (se obține câte un model cu câte trei senone folosind câte *patru densități Gaussiane* pentru fiecare trifonem în parte; în total modelul acustic va avea 200 senone!)
10. antrenarea modelelor fonetice (dependente de context) rezultate la pasul anterior
11. dublarea numărului de densități Gaussiene în cadrul fiecărei senone (se obține câte un model cu câte trei senone folosind câte *opt densități Gaussiane* pentru fiecare trifonem în parte; în total modelul acustic va avea 200 senone!)

În consecință, pentru a obține modelul acustic cu modele fonetice dependente de context având 200 de senone şi 8 densități Gaussiene per senone s-a trecut şi prin etapele în care modelul acustic era format din:

- modele fonetice independente de context (după pasul 2)
- modele fonetice dependente de context cu o singură densitate Gaussiană per senone (după pasul 6)

- modele fonetice dependente de context cu 2 densități Gaussiane per senone (după pasul 8)
- modele fonetice dependente de context cu 4 densități Gaussiane per senone (după pasul 10)

Toate aceste modele se află în directorul proiectului, în subdirectorul model_parameters. Puteți verifica acest lucru astfel:

```
cd ~/projects/rodigits
ls -all model_parameters
```

Lista subdirectoarelor afişate ar trebui să fie următoarea:

```
rodigits.cd_cont_200
rodigits.cd_cont_200_1
rodigits.cd_cont_200_2
rodigits.cd_cont_200_4
rodigits.cd_cont_initial
rodigits.cd_cont_untied
rodigits.ci_cont
rodigits.ci_cont_flatinitial
```

În mod implicit decodarea realizată în capitolul 6.6 a utilizat modelele din subdirectorul rodigits.cd_cont_200 (modelele fonetice dependente de context cu 8 densități Gaussiane per senone). În continuare ne propunem să configurăm procesul de decodare pentru a utiliza pe rând modelele din subdirectoarele:

- rodigits.ci_cont (modelele fonetice independente de context)
- rodigits.cd_cont_200_1 (modelele fonetice dependente de context cu o densitate Gaussiană per senone)
- rodigits.cd_cont_200_2 (modelele fonetice dependente de context cu 2 densități Gaussiene per senone)
- rodigits.cd_cont_200_4 (modelele fonetice dependente de context cu 4 densități Gaussiene per senone)

6.7.1 Decodarea clipurilor audio de evaluare folosind modele fonetice independente de context

Logați-vă pe serverul de dezvoltare, intrați în directorul proiectului, în subdirectorul etc şi vizualizați fişierul de configurare al proiectului cu ajutorul editorului vim:

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificați în acest fişier linia care specifică modelele acustice utilizate pentru decodare:

```
# Models to use.
$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
```

Intraţi în modul de editare (tastând “i”), copiaţi linia respectivă şi comentaţi una dintre copii (pentru a păstra un back-up) şi modificaţi cealaltă copie astfel încât să specificaţi locaţia modelelor fonetice independente de context. Această zonă a fişierului ar trebui să arate în final ca mai jos:

```
# Models to use.
#$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
$DEC_CFG_MODEL_NAME = "$CFG_EXPTNAME.ci_${CFG_DIRLABEL}";
```

Identificaţi apoi linia care specifică numele subdirectorului în care se vor salva rezultatele:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_200_8";
```

Şi modificaţi-o astfel încât numele subdirectorului în care se vor salva rezultatele să fie sugestiv:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.ci_cont";
```

Ieşiţi din modul de editare (tastând Esc), salvaţi modificările (tastând “:w” şi apoi Enter) şi apoi închideţi editorul (tastând “:q” şi apoi Enter).

Intraţi apoi în directorul proiectului şi porniţi procesul de evaluare executând următoarea comandă:

```
cd ~/projects/rodigits
/usr/local/sphinx/lib/sphinxtrain/scripts/decode/slave.pl
```

Aşteptaţi terminarea procesului de decodare (ar trebui să dureze aproximativ 1 minut). În cazul în care apar erori încercaţi să identificaţi problema vizualizand fişierul de log al procesului de decodare:

```
vi logdir/decode/rodigits-1-1.log
```

În cazul în care nu reuşiţi să identificaţi şi să rezolvaţi problema discutaţi cu îndrumătorul de proiect. În cazul în care procesul de decodare se încheie cu succes ar trebui ca pe ecran să vă apară un mesaj similar cu acesta:

```
MODULE: DECODE Decoding using models previously trained (decode script:
psdecode_fsg.pl)
    Decoding 20 segments starting at 0 (part 1 of 1)
    0%
    Aligning results to find error rate
    SENTENCE ERROR: 55.0% (11/20)   WORD ERROR RATE: 6.3% (15/240)
```

Aceste rezultate, precum şi informaţiile şi rezultatele detaliate din raportul de evaluare generat la acest pas (result.ci_cont/rodigits.align) vor fi comparate cu rezultatele obţinute în prealabil şi cu rezultatele ce vor fi obţinute în continuare şi vor fi în final prezentate şi discutate în raportul final al proiectului de cercetare-dezvoltare.

6.7.2 Decodarea clipurilor audio de evaluare folosind modele fonetice dependente de context cu mai puţine densităţi Gaussiene per senone

Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului, în subdirectorul etc şi vizualizaţi fişierul de configurare al proiectului cu ajutorul editorului vim:

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificaţi în acest fişier linia care specifică modelele acustice utilizate pentru decodare:

```
# Models to use.
#$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
$DEC_CFG_MODEL_NAME = "$CFG_EXPTNAME.ci_${CFG_DIRLABEL}";
```

Intraţi în modul de editare (tastând “i”), ştergeţi linia decomentată (cea care specifica locaţia modelelor fonetice independente de context), apoi copiaţi linia comentată (pentru a păstra un back-up) şi decomentaţi una dintre copii. Modificaţi copia decomentată astfel încât să specificaţi locaţia modelelor fonetice dependente de context ce folosesc o singură densitate Gaussiană per senone. Această zonă a fişierului ar trebui să arate în final ca mai jos:

```
# Models to use.
#$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}_1";
```

Identificaţi apoi linia care specifică numele subdirectorului în care se vor salva rezultatele:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_200_8";
```

Şi modificaţi-o astfel încât numele subdirectorului în care se vor salva rezultatele să fie sugestiv:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_200_1";
```

Ieşiţi din modul de editare (tastând Esc), salvaţi modificările (tastând “:w” şi apoi Enter) şi apoi închideţi editorul (tastând “:q” şi apoi Enter).

Copiați definiția modelului acustic din subdirectorul ce conține modelele fonetice dependente de context ce folosesc 8 densități Gaussiane per senone (rodigits.cd_cont_200) în subdirectorul ce conține modelele fonetice dependente de context ce folosesc o singură densitate Gaussiană per senone (rodigits.cd_cont_200_1):

```
cd ~/projects/rodigits
cp model_parameters/rodigits.cd_cont_200/mdef
model_parameters/rodigits.cd_cont_200_1/
```

Porniți apoi procesul de evaluare executând următoarea comandă:

```
/usr/local/sphinx/lib/sphinxtrain/scripts/decode/slave.pl
```

Aşteptați terminarea procesului de decodare (ar trebui să dureze aproximativ 1 minut). În cazul în care apar erori încercați să identificați problema vizualizand fişierul de log al procesului de decodare:

```
vi logdir/decode/rodigits-1-1.log
```

În cazul în care nu reuşiți să identificați şi să rezolvați problema discutați cu îndrumătorul de proiect. În cazul în care procesul de decodare se încheie cu succes ar trebui ca pe ecran să vă apară un mesaj similar cu acesta:

```
MODULE: DECODE Decoding using models previously trained (decode script:
psdecode_fsg.pl)
    Decoding 20 segments starting at 0 (part 1 of 1)
    0%
    Aligning results to find error rate
    SENTENCE ERROR: 40.0% (8/20)   WORD ERROR RATE: 9.6% (23/240)
```

Aceste rezultate, precum şi informațiile şi rezultatele detaliate din raportul de evaluare generat la acest pas (result.cd_cont_200_1/rodigits.align) vor fi comparate cu rezultatele obținute în prealabil şi cu rezultatele ce vor fi obținute în continuare şi vor fi în final prezentate şi discutate în raportul final al proiectului de cercetare-dezvoltare.

Repetați paşii de mai sus realizând decodarea clipurilor audio de evaluare cu modele fonetice dependente de context cu 2 şi respectiv 4 densități Gaussiene per senone.

6.7.3 Antrenarea unui model acustic având în total 100 de senone

În continuare ne propunem să verificăm dacă numărul de 200 de senone (configurat implicit într-un proiect CMU Sphinx) este optim sau nu pentru sarcina de recunoaştere a vorbirii propusă în acest îndrumar (recunoaşterea secvențelor audio ce conțin cifre).

Vom începe investigația antrenând un nou model acustic având în total 100 de senone disponibile pentru modelarea fonemelor ce formează cuvintele specifice sarcinii de recunoaştere. Pentru a realiza acest lucru ar trebui urmăriți în principiu paşii descrişi în

capitolul 6.4. Însă, pentru această sarcină nu vom crea un nou proiect CMU Sphinx, în consecință nu va trebui nici să adăugăm alte fișiere de resurse și nici să transformăm fișierele cu resurse în formatul cerut de CMU Sphinx. Vom modifica numai fișierul de configurare al proiectului rodigits astfel încât numărul de senone să fie 100 (nu 200, așa cum era în mod implicit):

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificați în acest fișier linia care specifică numărul de senone:

```
# Number of tied states (senones) to create in decision-tree clustering
$CFG_N_TIED_STATES = 200;
```

Intrați în modul de editare (tastând "i") și modificați-o astfel:

```
# Number of tied states (senones) to create in decision-tree clustering
$CFG_N_TIED_STATES = 100;
```

Ieșiți din modul de editare (tastând Esc), salvați modificările (tastând ":w" și apoi Enter) și apoi închideți editorul (tastând ":q" și apoi Enter).

Intrați în directorul proiectului și porniți antrenarea modelului acustic executând următoarea comandă:

```
cd ~/projects/rodigits
sphinxtrain_biosinf run
```

Așteptați terminarea procesului de antranare (ar trebui să dureze aproximativ 2 minute). În cazul în care apar erori verificați faptul că fișierele cu resurse sunt în formatul corespunzător și că nu ați modificat greșit fișierul de configurare. În cazul în care nu reușiți să identificați și să rezolvați problema discutați cu îndrumătorul de proiect.

După încheierea cu succes a procesului de antrenare verificați faptul că au fost generate modelele acustice corespunzătoare listând subdirectoarele din directorul proiectului, subdirectorul model_parameters:

```
cd ~/projects/rodigits
ls -all model_parameters
```

Lista subdirectoarelor afișate ar trebui să fie următoarea:

```
rodigits.cd_cont_100
rodigits.cd_cont_100_1
rodigits.cd_cont_100_2
rodigits.cd_cont_100_4
rodigits.cd_cont_200
rodigits.cd_cont_200_1
rodigits.cd_cont_200_2
rodigits.cd_cont_200_4
rodigits.cd_cont_initial
rodigits.cd_cont_untied
rodigits.ci_cont
rodigits.ci_cont_flatinitial
```

Observaţi că, pe lângă subdirectoarele existenete anterior (cele care conţin modele acustice cu 200 de senone şi număr variabil de densităţi Gaussiene), în această listă apar acum şi subdirectoarele rodigits.cd_cont_100* ce conţin modele acustice cu 100 de senone şi număr variabil de densităţi Gaussiene.

6.7.4 Decodarea clipurilor audio de evaluare folosind modele acustice cu 100 de senone

Urmează ca acum să evaluăm aceste 4 noi modele acustice. Vom începe cu modelul acustic cu 100 de senone şi 8 densităţi Gaussiene per senone. Logaţi-vă pe serverul de dezvoltare, intraţi în directorul proiectului, în subdirectorul etc şi vizualizaţi fişierul de configurare al proiectului cu ajutorul editorului vim:

```
cd ~/projects/rodigits/etc/
vi sphinx_train.cfg
```

Identificaţi în acest fişier linia care specifică modelele acustice utilizate pentru decodare:

```
# Models to use.
#$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}_4";
```

Intraţi în modul de editare (tastând “i”), ştergeţi linia decomentată (cea care specifica locaţia modelelor fonetice dependente de context cu 4 densităţi Gaussiene), apoi decomentaţi linia comentată (ce specifică locaţia modelelor fonetice implicite). Această zonă a fişierului ar trebui să arate în final ca mai jos:

```
# Models to use.
$DEC_CFG_MODEL_NAME =
"$CFG_EXPTNAME.cd_${CFG_DIRLABEL}_${CFG_N_TIED_STATES}";
```

Identificați apoi linia care specifică numele subdirectorului în care se vor salva rezultatele:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_200_4";
```

Şi modificați-o astfel încât numele subdirectorului în care se vor salva rezultatele să fie sugestiv:

```
$DEC_CFG_RESULT_DIR    = "$CFG_BASE_DIR/result.cd_cont_100_8";
```

Ieşiți din modul de editare (tastând Esc), salvați modificările (tastând ":w" şi apoi Enter) şi apoi închideți editorul (tastând ":q" şi apoi Enter).

Intrați apoi în directorul proiectului şi porniți procesul de evaluare executând următoarea comandă:

```
cd ~/projects/rodigits
/usr/local/sphinx/lib/sphinxtrain/scripts/decode/slave.pl
```

Aşteptați terminarea procesului de decodare (ar trebui să dureze aproximativ 1 minut). În cazul în care apar erori încercați să identificați problema vizualizand fişierul de log al procesului de decodare:

```
vi logdir/decode/rodigits-1-1.log
```

În cazul în care nu reuşiți să identificați şi să rezolvați problema discutați cu îndrumătorul de proiect. În cazul în care procesul de decodare se încheie cu succes ar trebui ca pe ecran să vă apară un mesaj similar cu acesta:

```
MODULE: DECODE Decoding using models previously trained (decode script:
psdecode_fsg.pl)
    Decoding 20 segments starting at 0 (part 1 of 1)
    0%
    Aligning results to find error rate
    SENTENCE ERROR: 5.0% (1/20)   WORD ERROR RATE: 0.4% (0/240)
```

Aceste rezultate, precum şi informațiile şi rezultatele detaliate din raportul de evaluare generat la acest pas (result.cd_cont_100_8/rodigits.align) vor fi comparate cu rezultatele obținute în prealabil şi cu rezultatele ce vor fi obținute în continuare şi vor fi în final prezentate şi discutate în raportul final al proiectului de cercetare-dezvoltare.

Repetați paşii de mai sus (ținând cont şi de capitolul care prezintă decodarea folosind modele cu mai puține densități Gaussiene per senone) realizând decodarea clipurilor audio de evaluare cu modele fonetice dependente de context cu 100 de senone şi 1, 2 şi respectiv 4 densități Gaussiene per senone.

6.8 Construcţia unui sistem de recunoaştere independent de vorbitor

Acest capitol vă propune realizarea unui al doilea proiect de recunoaştere a secvenţelor audio ce conţin cifre, de această dată *independent de vorbitor* (îl vom numi rodigitsIndep). Pentru a justifica necesitatea unui sistem independent de vorbitor ar trebui să evaluaţi sistemul rodigits creat anterior pe fişierele de evaluare ale celorlalţi colegi de grupă (fişiere ce se află pe serverul de dezvoltare în directorul /home/biosinfShare/resources/wav/). În urma evaluării veţi constata că rezultatele de recunoaştere a vorbirii sunt mai slabe pentru ceilalţi vorbitori decât pentru voi înşivă. Explicaţia este simplă: modelul acustic din proiectul rodigits a fost antrenat doar cu vocea voastră şi, în consecinţă, vă recunoaşte bine numai pe voi!

Crearea unui proiect independent de vorbitor presupune reluarea tuturor etapelor prezentate în acest îndrumar, începând cu capitolul 6.4, cu o singură modificare: clipurile audio de antrenare şi evaluare nu vor mai fi doar clipurile înregistrate de voi, ci toate clipurile înregistrate de toţi colegii de grupă. Bineînţeles că această modificare poate duce şi la alte modificări (de exemplu adaptarea scripturilor createTranscriptions.sh şi createFileIds.sh), însă pe toate acestea le veţi aborda şi eventual discuta cu îndrumătorul de proiect la momentul respectiv.

6.9 Redactarea raportului final al proiectului de cercetare-dezvoltare

După efectuarea experimentelor prezentate în acest îndrumar veţi sumariza toate rezultatele obţinute şi veţi face o discuţie asupra lor într-un raport final aferent proiectului de cercetare-dezvoltare.

Raportul de cercetare-dezvoltare trebuie să înceapă cu o scurtă secţiune care să cuprindă noţiuni introductive cu privire la arhitectura sistemelor de recunoaştere a vorbirii, sub-sistemele implicate, etc. Această primă secţiune are rolul de a evidenţia nivelul de înţelegere (asupra sistemelor de recunoaştere a vorbirii) pe care l-aţi dobândit pe parcursul realizării proiectului.

O a doua secţiune a raportului de cercetare-dezvoltare va prezenta pe scurt cerinţa (tema) proiectului şi va enumera paşii ce au fost urmaţi pentru a dezvolta sistemul de recunoaştere a secvenţelor audio ce conţin cifre.

În final, partea cea mai consistentă a raportului trebuie să prezinte **toate** rezultatele obţinute şi să le discute în mod comparativ. În funcţie de stadiul în care aţi ajuns cu experimentele veţi avea rezultate de recunoaştere pentru:

- modelul acustic dependent de vorbitor:
  - cu 200 de senone:
    - independente de context
    - dependente de context, cu 1/2/4/8 densități Gaussiene per senone
  - cu 100 de senone:
    - independente de context
    - dependente de context, cu 1/2/4/8 densități Gaussiene per senone
- modelul acustic independent de vorbitor (cu acealaşi variații ale parametrilor)

Prezentarea individuală a rezultatelor (pentru fiecare experiment în parte)va trebui să se facă punând accentul pe rata de eroare la nivel de cuvânt (Percent Total Error = Word Error Rate), dar se pot discuta şi alte metrici adiacente cum sunt rata de cuvinte inserate (Percent Insertions), rata de cuvinte substituite (Percent Substitutions), rata de cuvinte şterse (Percent Deletions), rata de eroare la nivel de propoziție (Sentences with errors), etc. De asemenea, în funcție de rezultatele obținute, pot fi prezentate perechile de cuvinte confundate foarte frecvent, cuvintele inserate/substituite/şterse cel mai frecvent şi eventual 1-2 exemple de fraze (evidențiind varianta corectă a frazei şi varianta obținută la ieşirea sistemului).

Compararea rezultatelor obținute în diversele experimente se va face a) numai în funcție de Word Error Rate (WER), dacă ați efectuat multe experiemente sau b) în funcție de mai multe metrici, dacă aveți puține rezultate. În primul caz rezultatele vor fi prezentate folosind tabele şi grafice de variație a ratei de eroare la nivel de cuvânt (WER). Rezultatele se vor discuta pe baza acestor tabele şi grafice şi, în final, se vor trage concluzii cu privire la dependența/independența de context, numărul optim de senone, numărul optim de densități Gaussiene, etc. pentru modelul acustic dependent de vorbitor, respectiv independent de vorbitor.

Notă: redactarea raportului de cercetare-dezvoltare este obligatorie, iar transmiterea lui (prin e-mail) către îndrumătorul de proiect trebuie să se facă până în săptămâna a 12-a a semestrului.

Notă: raporul final trebuie să fie redactat şi transmis îndrumătorului de proiect indiferent de stadiul în care ați ajuns cu experimentele.

Notă: toate rezultatele prezentate în raportul de cercetare-dezvoltare vor fi însoțite de referințe către rapoartele de evaluare de pe serverul de dezvoltare (calea către fişierele corespunzătoare).

6.10 Desfăşurarea activităţii pe server-ul de dezvoltare

Proiectul de recunoaştere a secvenţelor audio ce conţin cifre va fi realizat în întregime pe un server de dezvoltare aflat în cadrul laboratorului de cercetare SpeeD (http://speed.pub.ro). Studentul va trebui să dispună de un calculator conectat la Internet care va fi folosit pentru conectarea de la distanţă (prin intermediul Internetului) la serverul de dezvoltare.

Serverul de dezvoltare utilizează un sistem de operare de tip unix (Ubuntu 12.04). În consecinţă studentul trebuie să fie familiarizat sau să se familiarizeze cu modul de operare pe un astfel de sistem. În continuare sunt date câteva informaţii despre modul de conectare la server şi despre utilizarea editorului vim.

Conectarea la serverul de dezvoltare

Conectarea la serverul de dezvoltare se face extrem de simplu în cazul în care calculatorul de care dispuneţi utilizează un sistem de operare de tip unix/linux. În acest caz conectarea la server se face utilizând un terminal în care executaţi următoarea comandă (important: modificaţi comanda următoare pentru a utiliza:

- propriul nume de utilizator USER_NAME
- portul PORT şi numele serverul SERVER_NAME, comunicate de îndrumătorul de proiect.

```
ssh -p PORT USER_NAME@SERVER_NAME
```

Serverul va raspunde cerându-vă să tastaţi parola contului vostru. Pentru a continua va trebui să tastaţi parola contului vostru, aşa cum v-a fost specificată îndrumătorul de proiect. În cazul conectării cu succes pe ecran vă va apărea următorul promt (unde veţi executa comenzile precizate în acest îndrumar):

```
USER_NAME@alpha:~$
```

În cazul în care calculatorul de care dispuneţi utilizează un sistemul de operare Windows atunci conectarea la serverul de dezvoltare se va face cu ajutorul unei aplicaţii intermediare numită *putty*. Această aplicaţie poate fi descărcată gratuit de pe siteul: http://putty.en.softonic.com/. După descărcare, porniţi aplicaţia şi completaţi în primul ecran *Host Name:* SERVER_NAME şi *Port:* PORT. Atenţie: SERVER_NAME şi PORT sunt date ce v-au fost communicate de îndrumătorul de proiect. Pentru a salva aceste informaţii şi a nu fi nevoiţi să le completaţi de fiecare dată completaţi *Saved Sessions:* biosinfServer şi apăsaţi butonul *Save*. Pentru ca această aplicaţie să poată afişa corect caracterele diacritice, intraţi în meniul *Window*, submeniul *Translation* şi selectaţi *Remote Character Set*: UTF-8. Apoi reveniţi la ecranul iniţial, selectând meniul *Session* şi apăsaţi butonul *Open* pentru a stabili conexiunea. Serverul vă va cere numele de utilizator şi apoi parola (informaţii pe care vi le-a

specificat îndrumătorul de proiect). În cazul conectării cu succes pe ecran vă va apărea următorul promt (unde veți executa comenzile precizate în acest îndrumar):

```
USER_NAME@alpha:~$
```

Recomandare: primul lucru pe care ar trebui să îl faceți după ce vă conectați pe server pentru prima dată este să vă schimbați parola. Pentru aceasta executați următoarea comandă şi introduceți, pe rând, parola actuală, noua parola şi încă o dată noua parolă:

```
passwd
```

Notă: indiferent de modul de conectare la serverul de dezvoltare (terminal linux sau putty) aveți posibilitatea şi de multe ori este chiar foarte util să deschideți mai multe conexiuni simultane către server.

Utilizarea editorului în mod text vim

În cadrul acestui proiect veți lucra extensiv (creare, vizualizare, editare, modificare, etc.) cu fişiere text. Cel mai simplu mod de a manipula fişierele text aflate pe un server linux este cu ajutorul editorului vim.

Pentru a vizualiza un fişier deja existent sau a crea un fişier nou (cu numele filename) tastați comanda:

```
vi filename
```

În acest moment vizualizați prima parte a fişierului, iar cu ajutorul tastelor cu săgeți şi a tastelor PgUp/PgDn puteți vizualiza şi alte zone ale fişierului. Pentru a căuta un cuvânt (sau orice alt şir de caractere) tastați "/", apoi cuvântul respectiv şi în final Enter. Pentru a căuta acelaşi cuvânt din nou tastați "/" şi apoi Enter. Pentru a intra în modul de editare (în vederea modificării fişierului) tastați "i". Acum puteți şterge şi scrie în acest fişier ca în oricare alt editor de text. Pentru a ieşi din modul de editare tastați Esc. Pentru a salva modificările făcute tastați ":w" şi apoi Enter. Pentru a părăsi editorul tastați ":q" şi apoi Enter. Pentru mai multe informații de utilizare şi alte comenzi de editare, copiere, etc. consultați site-ul vim precizat în bibliografia îndrumarului de proiect [Vi].

BIBLIOGRAFIE

[Bahl, 1991] Bahl, L.R., et al., "Multonic Markov Word Models for Large Vocabulary Continuous Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, 1993, vol. 1, no. 3, pp. 334-344, 1991.

[Baker, 1975] Baker, J., "The DRAGON system – an overview," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 23, no. 1, pp. 24-29, 1975.

[Baum, 1972] Baum, L.E., "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," *Inequalities*, vol. 3, no. 1, pp. 1-8, 1972.

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

[Cucu, 2011] Cucu, H., "Towards a speaker-independent, large-vocabulary continuous speech recognition system for Romanian," Teză de doctorat, Universitatea Politehnica din Bucureşti, România, 2011, http://speed.pub.ro/academic/research-and-development-project-in-spoken-language-technology.

[Dempster, 1977] Dempster, A., Laird, N., Rubin, D., "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society*, Series B, vol. 39, pp. 1-38, 1977.

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

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

[Hwang, 1993] Hwang, M.Y., Huang, X., Alleva, F., "Predicting Unseen Triphones with Senones," *ICASSP 1993*, Minneapolis, pp. 311-314, 1993.

[JSGF] Java Speech Grammar Format, http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html.

[Jelinek, 1998] Jelinek, F., *Statistical Methods for Speech Recognition*, The MIT Press, Cambridge, MA, 1998.

[Jurafsky, 2009] Jurafsky, D., Martin, J., "Automatic Speech Recognition," Chapter 9 in *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition (2nd Ed.)*, Pearson Education, 2009.

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

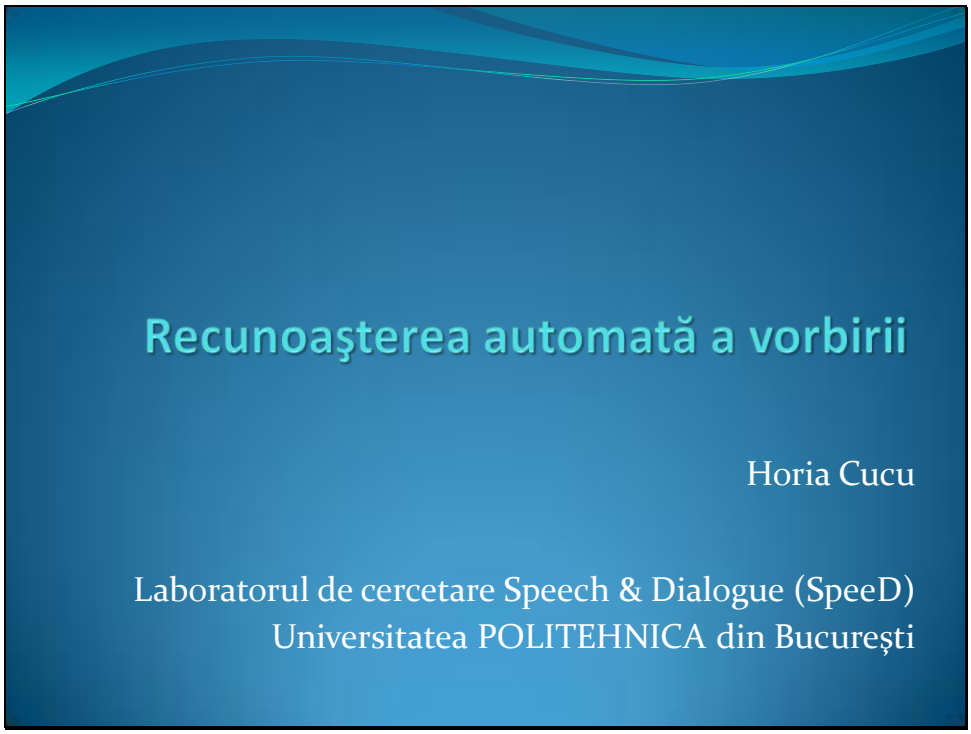
[Rabiner, 1989] Rabiner, L.R., "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

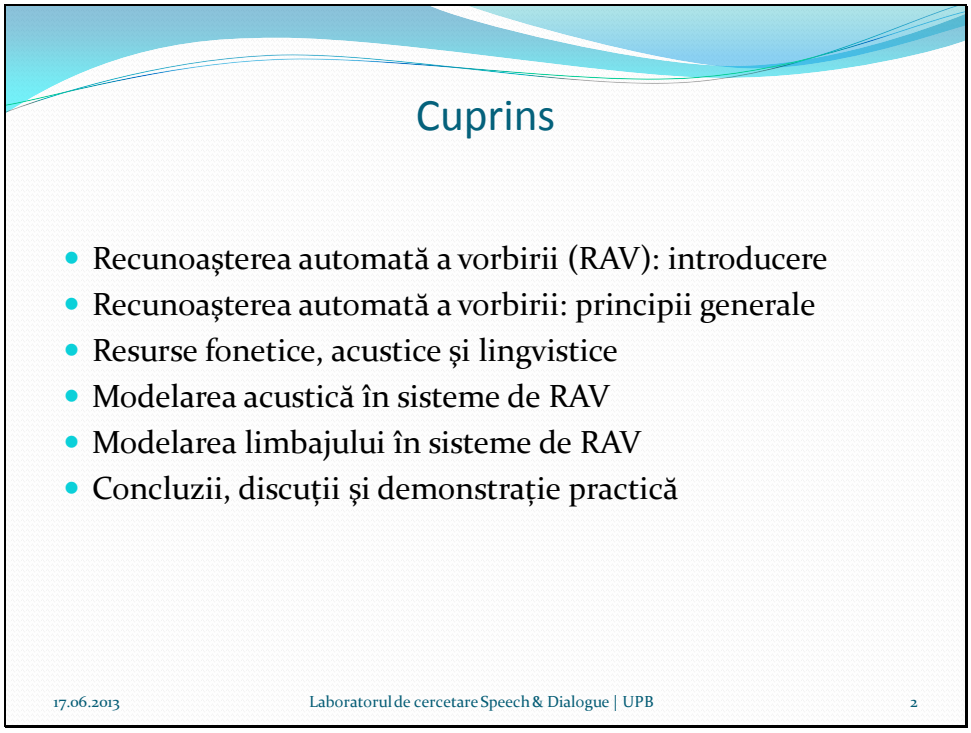
[Renalds, 2010] Renals S., Hain, T., "Speech Recognition," *Handbook of Computational Linguistics and Natural Language Processing*, 2010.

[Sphinx] Toolkitul CMU Sphinx şi tutorialele asociate, http://cmusphinx.sourceforge.net/wiki/

[Vi] Manualul editorului de text vi, http://www.cs.fsu.edu/general/vimanual.html.

ANEXA 1. DIAPOZITIVE DE PREZENTARE





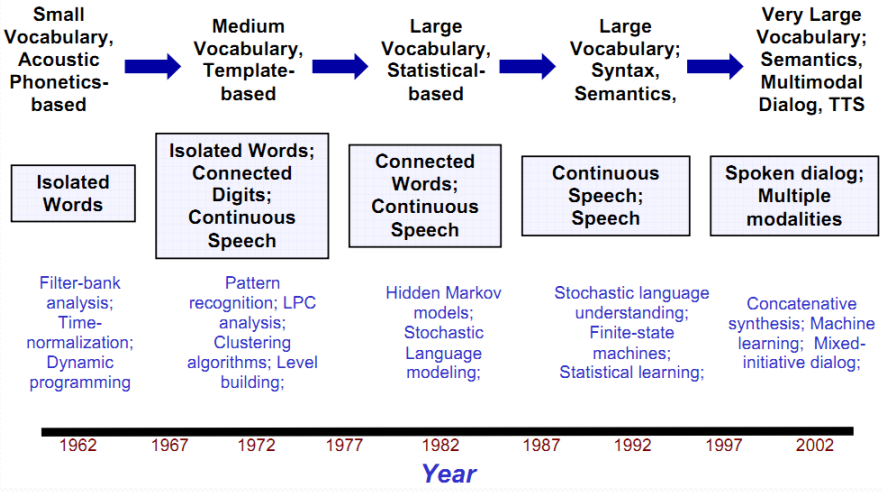
Cuprins

- Recunoaşterea automată a vorbirii (RAV): introducere
  - Introducere în recunoaşterea automată a vorbirii
  - Aplicaţii ale recunoaşterii automate a vorbirii
- Recunoaşterea automată a vorbirii: principii generale
- Resurse fonetice, acustice şi lingvistice
- Modelarea acustică în sisteme de RAV
- Modelarea limbajului în sisteme de RAV
- Concluzii, discuţii şi demonstraţie practică

Introducere în recunoaşterea automată a vorbirii

- Recunoaşterea automată a vorbirii (RAV) -> procesul de transformare în text a unui semnal audio ce conţine de vorbire
- Face parte din domeniul mai larg de recunoaşterea formelor
- Abordarea din starea artei este în mod principal statistică

Istoria recunoaşterii automate a vorbirii



Aplicații de transcriere speech-to-text (I)

- Completarea documentelor structurate (rapoarte medicale)
  - Sarcină de transcriere simplă (cuvinte izolate și vorbire continuă)
  - Un singur utilizator per aplicație (se poate face adaptare la vorbitor)
  - Utilizatorul își dorește să fie recunoscut de aplicație!
  - Limbajul este restricționat (depinde puternic de domeniul aplicației)
- Transcrierea vorbirii continue citite (știri, conferințe de presă)
  - Recunoaşterea vorbirii continue citite
  - Mai mulți vorbitori per aplicație
  - Limbajul nu este restricționat
  - Vocabularul de cuvinte conține practic toate cuvintele limbii

Aplicații de transcriere speech-to-text (II)

- Dictarea documentelor (SMS-uri, email-uri, documente word)
  - Recunoaștere de vorbire continuă spontană
  - Un singur utilizator per aplicație (se poate face adaptare la vorbitor)
  - Utilizatorul își dorește să fie recunoscut!
  - Limbajul și vocabularul nu sunt restricționate (toate cuvintele limbii)
- Transcrierea vorbirii continue spontane (interviuri, talk-shows)
  - Cea mai dificilă sarcină de transcriere (vorbire continuă spontană)
  - Mai mulți vorbitori per aplicație
  - Limbajul și vocabularul nu sunt restricționate (toate cuvintele limbii)
  - Dificultăți suplimentare: calitatea înregistrărilor, zgomotul de fundal

Aplicații de căutare şi indexare

- Indexarea materialelor audio ce conțin vorbire
  - Nu este necesar ca recunoașterea să fie perfectă
  - Dificultăți: procesarea unei cantități mari de date
- Identificarea cuvintelor cheie (Keyword spotting)
  - Query-ul (cuvântul cheie) este introdus sub formă de text
  - Căutarea se face în materiale audio indexate sau direct în transcriere
- Detectarea termenilor vorbiți (Spoken term detection)
  - Query-ul este vorbit! Contentul este material vorbit!
  - Interpretarea termenului (transcriere sau clasificare) este esențială

Aplicații de comandă şi control

- Controlul telefonului/calculatorului/televizorului/casei inteligente
  - Sarcină foarte simplă: recunoașterea câtorva comenzi fixe
  - Singura dificultate: separarea comenzilor de vorbirea uzuală
  - Aplicație adaptată vorbitorului/vorbitorilor (3-4)
  - Utilizatorul își dorește să fie recunoscut de aplicație!
- Controlul avioanelor și al elicopterelor
  - Condiții extreme de zgomot!
  - Respirațiile vorbitorului trebuie tratate special
  - Precizia recunoașterii este esențială

Aplicații de dialog om-maşină

- Presupun trei blocuri de procesare:
  - Recunoașterea vorbirii continue
  - Înțelegerea vorbirii și generarea răspunsurilor adecvate
  - Sinteza vorbirii
- Dialog în call-centers (rezervări de bilete, suport tehnic)
  - Ghidarea inteligentă a utilizatorului prin meniuri ajută procesele de recunoaștere și înțelegere a vorbirii
- Dialog cu smart-phones (informații despre vreme, locații, etc.)
  - Puterea de procesare determină viteza și performanța aplicației
  - De cele mai multe ori procesarea se face în cloud

Aplicații speech-to-speech

- Presupun trei blocuri de procesare:
  - Recunoașterea vorbirii continue în limba sursă
  - Traducerea automată din limba sursă în limba destinație
  - Sinteza vorbirii în limba destinație
- Toate cele trei blocuri introduc erori
- Status:
  - Demostrații în cadrul manifestărilor științifice
  - Nu există încă aplicații comerciale

Cuprins

- Recunoașterea automată a vorbirii (RAV): introducere
- Recunoașterea automată a vorbirii: principii generale
  - Surse de variabilitate în recunoașterea automată a vorbirii
  - Arhitectura unui sistem de RAV
  - Performanțe uzuale ale sistemelor de RAV
- Resurse fonetice, acustice și lingvistice
- Modelarea acustică în sisteme de RAV
- Modelarea limbajului în sisteme de RAV
- Concluzii, discuții și demonstrație practică

Surse de variabilitate în RAV

- Domeniul/Sarcina de recunoaştere
  - Limba şi dimensiunea vocabularului
  - Incertitudinea lingvistica a discursului
- Caracteristicile vorbitorului
  - Accent, dialect, nativ/non-nativ, rapiditatea pronunției, vârsta
  - Sisteme de recunoaştere depedente/independente de vorbitor
- Stilul vorbirii
  - Cuvinte izolate, vorbire continuă, vorbire spontană (conversațională)
- Mediul acustic
  - Alți vorbitori, zgomot de fundal, caracteristicile microfonului

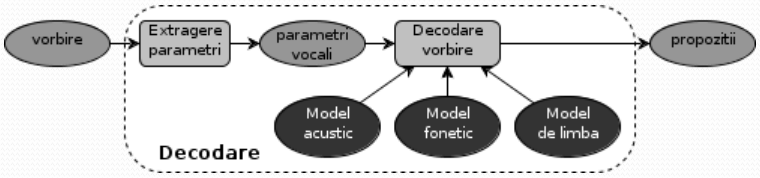
Formalismul actual al RAV

- Care este cea mai probabilă secvență de cuvinte W^* dat fiind mesajul vorbit X?

$$W^* = \arg\max_W p(W \mid X) = \arg\max_W \frac{p(X \mid W)p(W)}{p(X)} = \arg\max_W p(X \mid W)p(W)$$

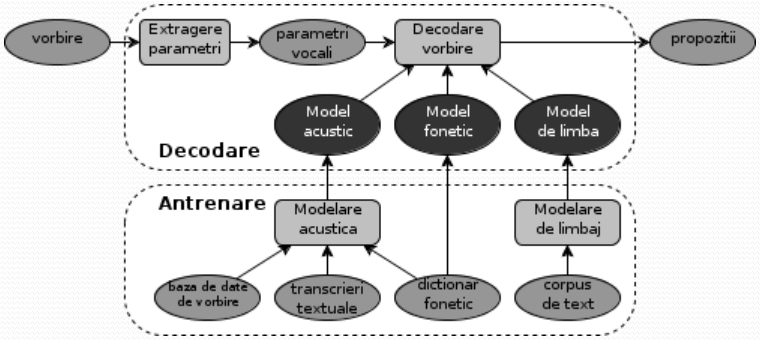
- W – toate secvențele posibile de cuvinte
- $p(X|W)$ – probabilitatea measajului vorbit, dată fiind secvența de cuvinte W; este estimată utilizând un model acustic!
- $p(W)$ – probabilitatea de apariție a secvenței de cuvinte W; este estimată strict pe baza unui model de limbă!

Arhitectura unui sistem de recunoaştere a vorbirii

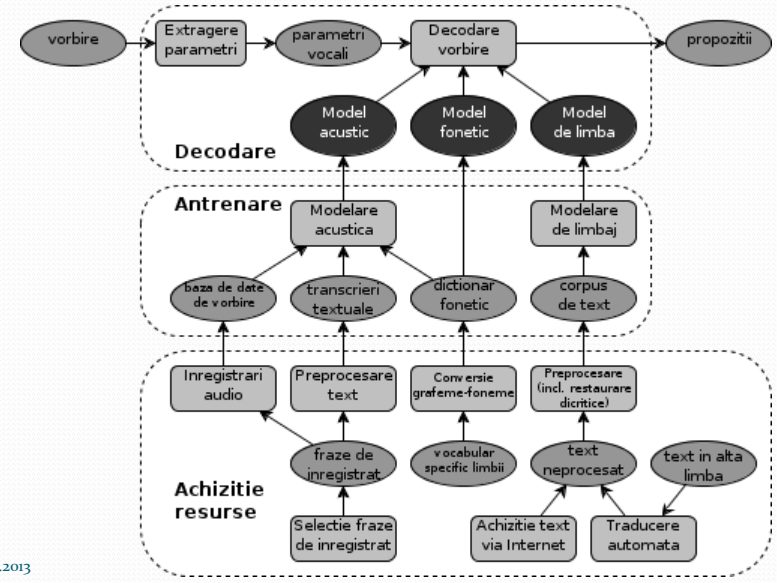


- Modelarea acustică (state-of-the-art):
  - Sisteme statistice de tip HMM-GMM
  - Unitățile de vorbire de bază: foneme şi unități sub-fonetice (senone) dependente de context
  - Parametri acustici perceptuali (MFCC, PLP, etc.)
  - Toolkit-uri: HTK, CMU Sphinx, etc.
- Modelarea lingvistică (state-of-the-art):
  - Sisteme statistice de tip n-gram
  - Metode de netezire: Good-Turing, Witten-Bell, Kneser-Ney, etc.
  - Rolul modelului de limbă: “casa ta e mare” sau “casat a Ema re”?
  - Toolkit-uri: SRI-LM, CMU-SLM, etc.

Arhitectura unui sistem de recunoaştere a vorbirii



Arhitectura unui sistem de recunoaştere a vorbirii



Evaluarea sistemelor de recunoaştere a vorbirii

- Compararea transcrierii de referință cu transcrierea ipotetică produsă de sistemul de RAV
- Rata de eroare la nivel de cuvânt (WER) – echivalentul unui cost de editare;
  - Alinierea celor două secvențe de cuvinte (DTW)
  - Calcularea costului pe baza numărului de erori de inserție, ștergere, substituție:

$$WER[\%] = \frac{\#Erori\ de\ inserție + \#Erori\ de\ substituție + \#Erori\ de\ ștergere}{\#Cuvinte\ în\ transcrierea\ de\ referință} \times 100$$

- Rata de eroare la nivel de propoziție (SER)

$$SER[\%] = \frac{\#Propoziții\ eronate}{\#Propoziții\ în\ transcrierea\ de\ referință} \times 100$$

Exemple de evaluare WER pentru RAV

```
REF:  iaurturile ajută la prevenirea **** ** ** OSTEOPOROZEI
HYP:  iaurturile ajută la prevenirea POST OP RO ZEI
Eval:                               I    I  I  S
Counts: (#C #S #D #I) 4 1 0 3
WER: 4/5 = 80.0%


REF:  am coborât din autocar ***** VAL VÂRTEJ și am început să FOTOGRAFIEZ
HYP:  am coborât din autocar BARBU TEI ȘI     și am început să FOTOGRAFIEZE
Eval:                         I     S   S                           S
Counts: (#C #S #D #I) 8 3 0 1
WER: 4/11 = 36.3%


REF:  aș vrea să afle cumva că GESTUL LUI      disperat nu ne-a fost indiferent
HYP:  aș vrea să afle cumva că ****** GESTULUI disperat nu ne-a fost indiferent
Eval:                           D      S
Counts: (#C #S #D #I) 11 1 1 0
WER: 2/13 = 15.3%
```


Performanțele sistemelor de RAV actuale

- Sarcina de recunoaștere și dimensiunea vocabularului influențează semnificativ performanțele sistemului de RAV

Sarcina de RAV	Limba	Vocabular	WER
TI Digits	Engleză	11 words	0.55%
Wall Street Journal read speech		5k words	3.0%
Wall Street Journal read speech		20k words	<6.6%
Broadcast News		64k+ words	9.9%
Conversational Telephone Speech		64k+ words	20.7%

* rezultate raportate în 2005 pentru diverse sarcini de RAV în limba engleză

- Cea mai dificilă sarcină de RAV: large vocabulary conversational speech

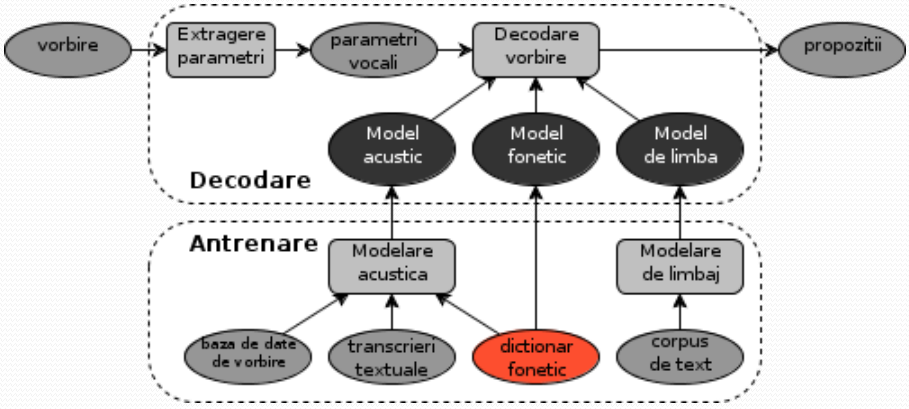
Sistem dezvoltat de	Limba	Vocabular	WER
RWTH Aachen University	Engleză	150k words	17.9%
	Germană	300k words	17.3%
	Franceză	200k words	20.9%
Karlsruhe Institute of Technology	Rusă	500k words	19.3%

* rezultate raportate în 2011 pentru broadcast conversational speech în diverse limbi

Cuprins

- Recunoaşterea automată a vorbirii (RAV): introducere
- Recunoaşterea automată a vorbirii: principii generale
- Resurse fonetice, acustice şi lingvistice
  - Dicţionare fonetice şi fonetizarea automată
  - Baze de date de vorbire: metode de achiziţie
  - Corpusuri de text: achiziţie, normalizare şi restaurare
- Modelarea acustică în sisteme de RAV
- Modelarea limbajului în sisteme de RAV
- Concluzii, discuţii şi demonstraţie practică

Dicţionarul fonetic într-un sistem de RAV



Noțiuni generale de fonetică

- Fonemul – unitatea de sunet fundamentală din limba vorbită
  - Succesiune de foneme -> cuvinte
- Dicționarul fonetic: corespondență între forma scrisă și forma fonetică a cuvintelor
- Limba română folosește
  - 7 vocale de bază + 2 vocale împrumutate
  - 4 semivocale
  - 22 de consoane
  - Un simbol special pentru a arăta palatalizarea consoanei anterioare

Vocalele şi semivocalele limbii române

Fonemul				Exemple de cuvinte	
Tip	**Nr.**	**Simbol IPA**	**Simbol intern**	**Forma scrisă**	**Forma fonetică**
vocale	**1**	a	a	s**a**t	s a t
	2	e	e	mar**e**	m a r e
	3	i	i	l**i**ft	l i f t
	4	o	o	l**o**c	l o c
	5	u	u	ş**u**t	s1 u t
	6	ə	a1	gur**ă**	g u r a1
	7	ɨ	i2	**î**ntre	i2 n t r e
vocale împrumutate	**8**	y	y	ecr**u**	e c r y
	9	ø	o2	bl**eu**	b l o2
semivocale	**10**	e̯	e1	d**e**al	d e1 a l
	11	j	i3	f**i**ară	f i3 a r a1
	12	o̯	o1	**o**ase	o1 a s e
	13	w	w	s**au**	s a w

Consoanele limbii române (I)

Fonemul			Exemple de cuvinte	
Nr.	**Simbol IPA**	**Simbol intern**	**Forma scrisă**	**Forma fonetică**
14	ʲ	i1	tar**i**	t a r i1
15	c	k2	**ch**em	k2 e m
16	b	b	**b**ar	b a r
17	p	p	**p**ar	p a r
18	k	k	**a**cum	a k u m
19	ʧ	k1	**c**enuşă	k1 e n u s1 a1
20	g	g	**g**alben	g a l b e n
21	ʤ	g1	**g**irafă	g1 i r a f a1
22	ɟ	g2	un**gh**i	u n g2
23	d	d	**d**ar	d a r
24	t	t	**t**ot	t o t
25	f	f	**f**aţa	f a t1 a

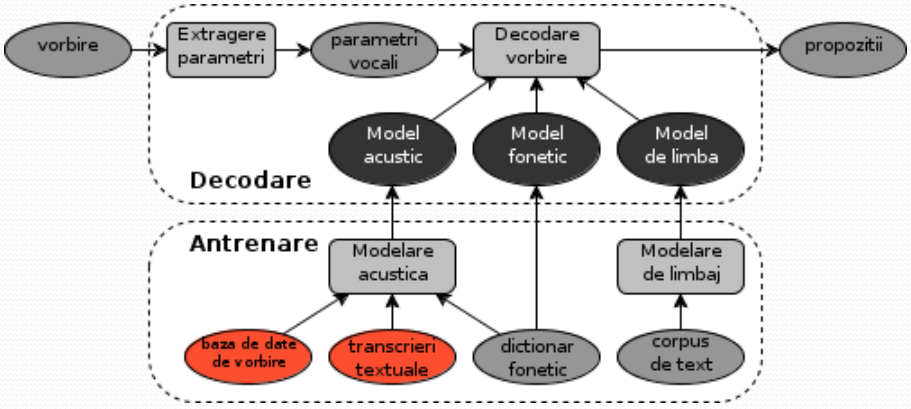
Consoanele limbii române (II)

Fonemul			Exemple de cuvinte	
Nr.	**Simbol IPA**	**Simbol intern**	**Forma scrisă**	**Forma fonetică**
26	v	v	**v**apor	v a p o r
27	h	h	**h**arta	h a r t a
28	ʒ	j	a**j**utor	a j u t o r
29	ʃ	s1	co**ş**	k o s1
30	l	l	**l**ac	l a c
31	m	m	**m**ăr	m a1 r
32	n	n	**n**as	n a s
33	s	s	**s**are	s a r e
34	z	z	**z**ar	z a r
35	r	r	**r**isc	r i s k
36	ʦ	t1	**ţ**ăran	t1 a1 r a n

Construcția dicționarului fonetic

- Fonetizarea cuvintelor din vocabularul sistemului de RAV
  - Manuală, în cazul în care vocabularul este redus
  - Automată, pentru RAV cu vocabulare mari
- Fonetizarea automată: sisteme bazate pe reguli
  - Formalizarea regulilor și excepțiilor de fonetizare
  - Formalizarea regulilor și excepțiilor de despărțire în silabe
- Fonetizarea automată: sisteme de învățare automată
  - Bazate pe rețele neurale, traducere automată statistică, etc.
- Performanțe uzuale: WER între 3% și 5%

Baza de date de vorbire într-un sistem de RAV



Caracteristicile bazei de date de vorbire

- Dimensiunea bazei de date:
  - numărul de ore de vorbire, numărul de vorbitori

Sarcina RAV	Sistem dependent de vobitor	Sistem independent de vorbitor
comandă şi control (RVC-VR)	1 oră de înregistrări, 1 vorbitor	5 ore de înregistrări, 200 de vorbitori
dictare (RVC-VE)	10 ore de înregistrări, 1 vorbitor	50 ore de înregistrări, 200 de vorbitori

- Stilul vorbirii
  - cuvinte izolate, vorbire continuă citită, vorbire conversațională
- Variabilitatea
  - calitatea înregistrărilor, zgomotul de fundal, variabilitatea vorbitorilor

Metode de achiziție a bazei de date de vorbire

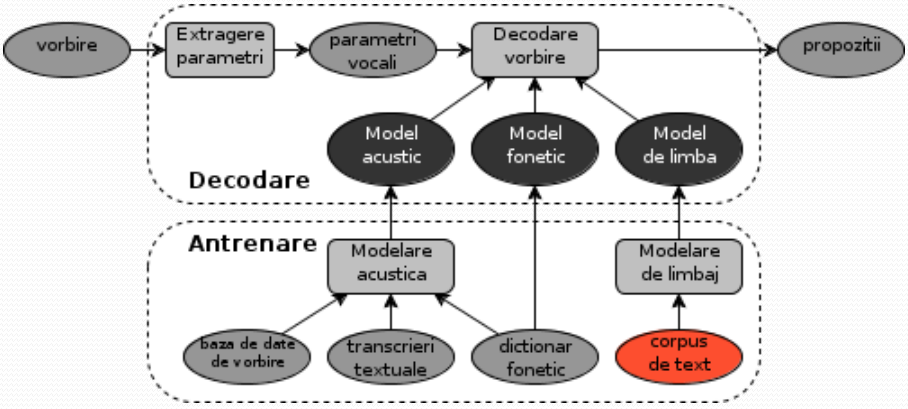
- Înregistrarea unor texte predefinite
  - Control asupra textului, vorbitorilor, etc.
  - Vorbire continuă-citită
  - Eficiență dpdv al timpului de obținere
  - Erori de pronunție
- Etichetarea unor clipuri audio ce conțin vorbire
  - Orice tip de vorbire
  - Erori de etichetare

Înregistrarea textelor predefinite

- Obținerea unei fluențe în pronunție (fără ezitări/respirații)
  - fraze corecte dpdv lexical şi gramatical, cu topică uzuală
  - fraze scurte (maxim 15-20 de cuvinte)
  - fraze conectate semantic
  - utilizarea cuvintelor frecvente în limbajul de zi cu zi
- Evitarea ambiguității pronunției
  - utilizarea cuvintelor frecvente în limbajul de zi cu zi
  - evitarea cuvintelor împrumutate din limbi străine
  - evitarea interjecțiilor
  - scrierea numerelor cu litere

17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 31

Corpusul de text într-un sistem de RAV



17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 32

Utilizarea corpusului de text

- Sisteme RAV de comandă și control
  - Utilizează gramatici cu reguli
  - Construcția gramaticilor cu reguli nu necesită un corpus de text
- Sisteme RVC cu vocabular mare
  - Utilizează modele de limbă statistice
    - Un corpus de text este absolut necesar pentru extragerea de statistici
    - Calitatea modelului de limbă depinde de dimensiunea corpusului

Achiziția corpusului de text

- Surse de text utilizabil în modelarea lingvistică statistică
  - Cărți în format electronic
  - Știri de pe site-urile ziarelor on-line
  - Subtitrări de filme
  - Articole personale (blog-uri) on-line
- Dificultatea normalizării textelor depinde de sursa de achiziție
  - Fiecare carte are formatul ei specific (antet, cuprins, etc.)
  - Subtitrările conțin ștampile de timp (în diverse formate)
  - Site-urile de știri conțin meniuri, reclame, etc.

Normalizarea corpusului de text

- Eliminarea etichetelor HTML
- Expandarea abrevierilor
- Transcrierea numerelor scrise cu cifre în numere scrise cu litere
- Tratarea parantezelor și a caracterelor speciale
- Tratarea semnelor de punctuație
- Conversie uppercase-to-lowercase

Ieri, dl. Stan a obținut 6,5% din voturi. Felicitări!

Ieri, domnul Stan a obținut 6,5% din voturi. Felicitări!

Ieri, domnul Stan a obținut șase virgulă cinci % din voturi. Felicitări!

Ieri, domnul Stan a obținut șase virgulă cinci la sută din voturi. Felicitări!

ieri domnul stan a obținut șase virgulă cinci la sută din voturi felicitări

17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 35

Restaurarea diacriticelor în corpusul de text

- 30 – 40% dintre cuvintele limbii române conțin diacritice!
  - Cuvinte neambigue dpdv al diacriticelor:
    - alb, astfel, fabricând, științific
  - Cuvinte ambigue:
    - sărmana/sărmană, pana/pană/până, tari / țări / țari / târî

diacriticele limbii române: ă, â, î, ș, ț sunt câteodată înlocuite cu: a, a, i, s, t

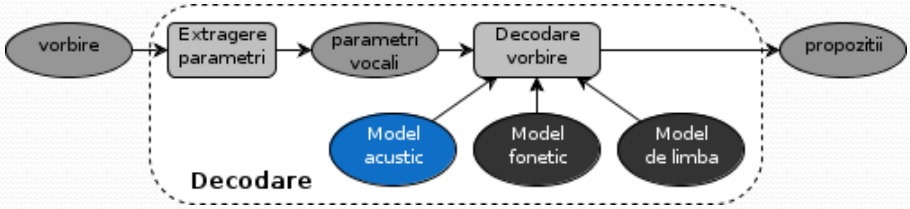
- Textul de la ieșirea sistemului de RAV este de obicei scurt și eliptic, iar lipsa diacriticelor îl poate face de neînțeles!
- Restaurarea diacriticelor este obligatorie pentru textele colectate de pe Internet
- Restaurarea diacriticelor se face automat, prin metode statistice

17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 36

Cuprins

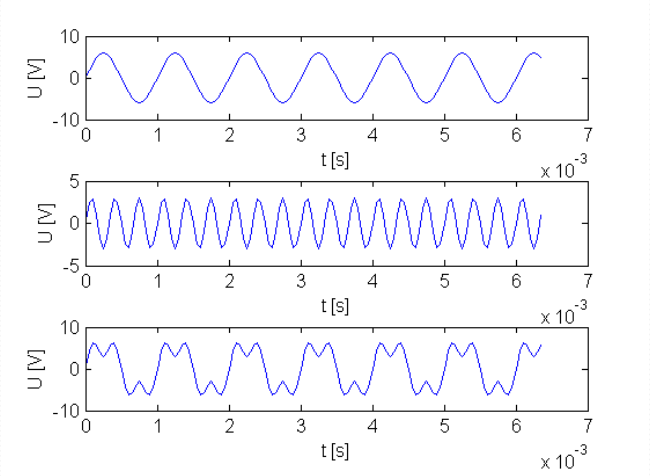
- Recunoaşterea automată a vorbirii (RAV): introducere
- Recunoaşterea automată a vorbirii: principii generale
- Resurse fonetice, acustice şi lingvistice
- Modelarea acustică în sisteme de RAV
  - Semnalul de vorbire
  - Platforma HMM-GMM
  - Algoritmi de antrenare şi decodare
- Modelarea limbajului în sisteme de RAV
- Concluzii, discuţii şi demonstraţie practică

Rolul modelului acustic într-un sistem de RAV

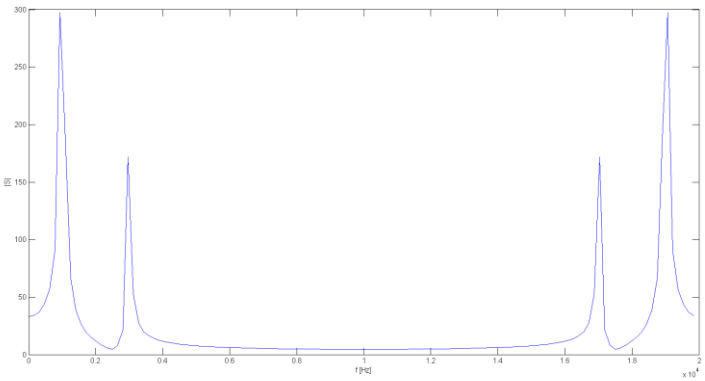


$$W^* = \arg\max_W p(W \mid X) = \arg\max_W \frac{p(X \mid W)p(W)}{p(X)} = \arg\max_W p(X \mid W)p(W)$$

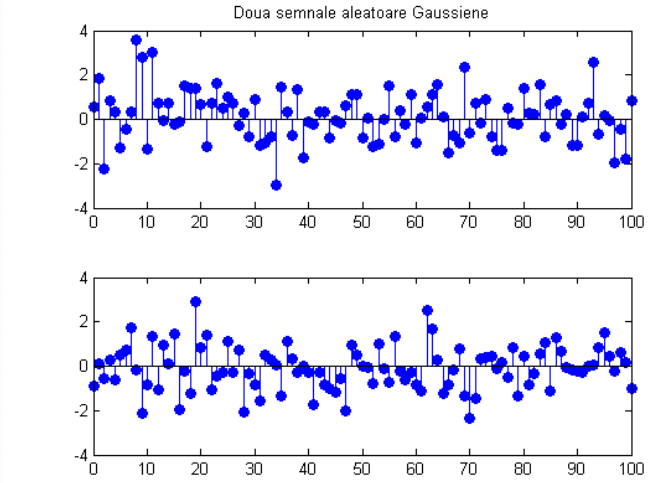
Semnale deterministe



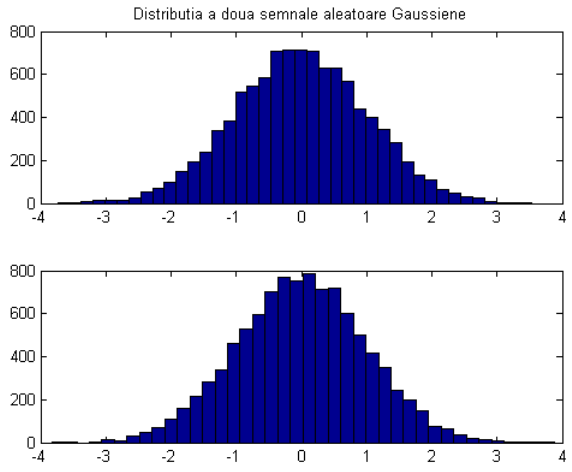
Spectrul semnalului determinist



Semnale aleatoare

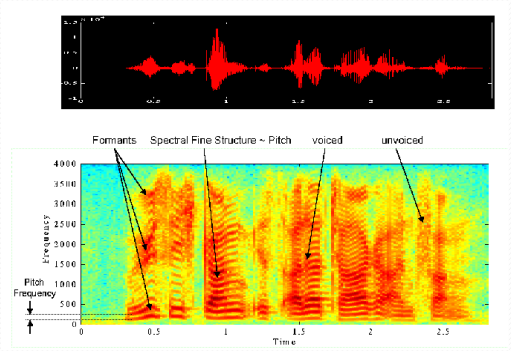


Distributia semnalelor aleatoare

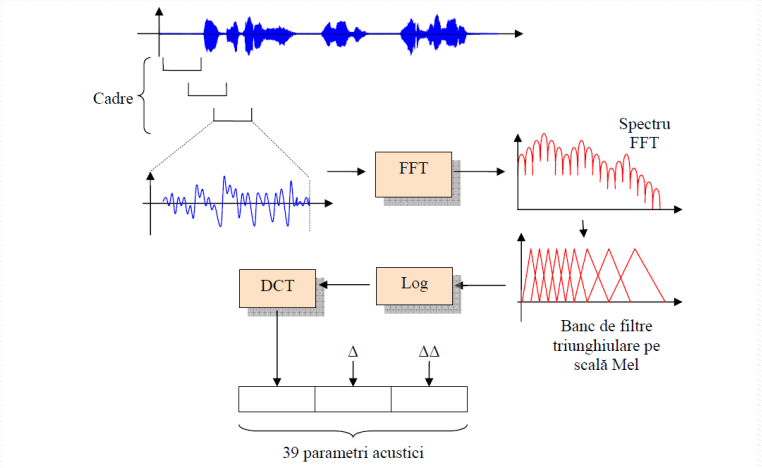


Spectrul semnalului vocal

- Segmentarea semnalului vocal în ferestre (10-30 ms)
- Ferestrele sunt suprapuse
- Transformata Fourier rapidă (FFT)

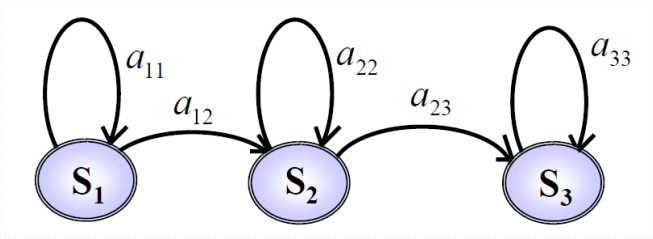


Parametrii semnalului vocal - MFCC



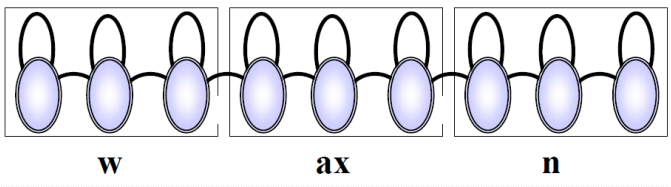
HMM de foneme

- Fiecare fonem este reprezentat de un HMM cu 3 stări cu
- S1 modelează începutul unui fonem, S2 mijlocul, iar S3 ultima parte a sunetului



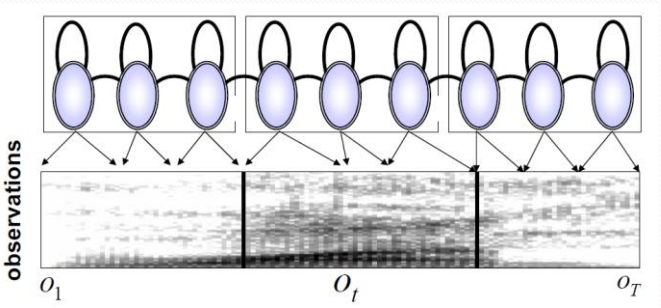
Modelarea cuvintelor și a frazelor

- Se construiește HMM la nivel de cuvânt sau frază din concatenarea modelelor pentru foneme. De exemplu cuvântul în engleză “ONE” cu pronunția “W AX N”:



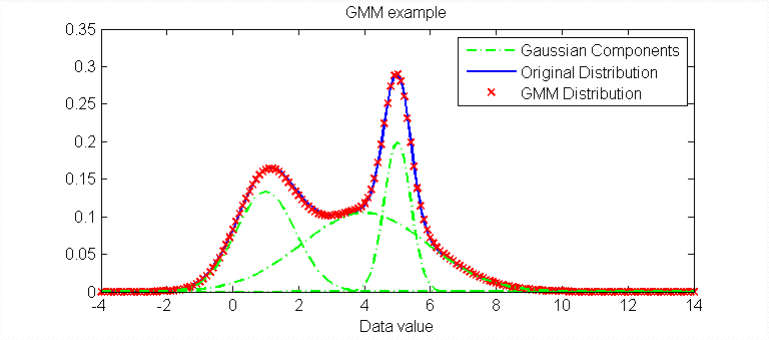
Generarea vorbirii

- Un semnal vocal este generat (produs) de o secvență de HMM cu o anumită probabilitate



Gaussian Mixture Models

- Every distribution can be approximated with a linear combination of Gaussian distributions



$$g(x|\theta) = \sum_{i=1}^{G} w_i g_i(x|\theta_i)$$

Viterbi decoding

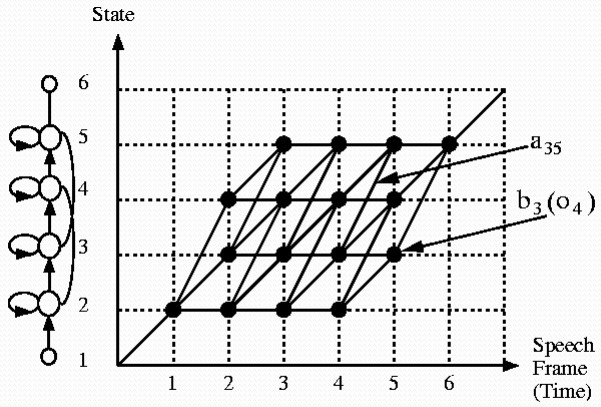


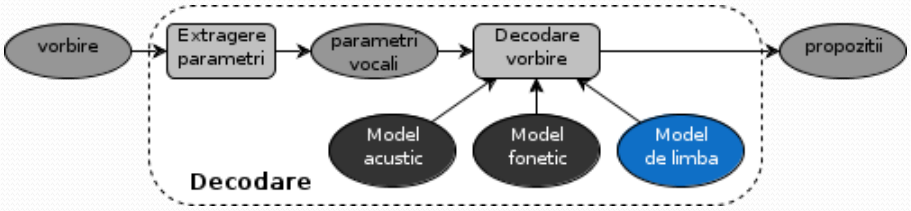
Fig. 1.6 The Viterbi Algorithm for Isolated Word Recognition

Cuprins

- Recunoaşterea automată a vorbirii (RAV): introducere
- Recunoaşterea automată a vorbirii: principii generale
- Resurse fonetice, acustice şi lingvistice
- Modelarea acustică în sisteme de RAV
- Modelarea limbajului în sisteme de RAV
  - Modele de limbă statistice
  - Gramatici cu reguli
- Concluzii, discuţii şi demonstraţie practică

Rolul modelului de limbă într-un sistem de RAV

Casa ta e mare? sau *Casat a Ema re?*



$$W^* = \arg\max_W p(W \mid X) = \arg\max_W \frac{p(X \mid W)p(W)}{p(X)} = \arg\max_W p(X \mid W)p(W)$$

Modele de limbă statistice de tip n-gram

- Se folosesc în sistemele de recunoaștere automată a vorbirii cu vocabular mare de peste 20 de ani
- Estimează probabilitatea unei secvențe de cuvinte *W* astfel:

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

- Istoria de cuvinte precedente - limitată la *m* cuvinte
  - *m* depinde de cantitatea de text de antrenare
  - *m* influențează eficiența computațională a sistemului
  - În mod tipic *m*=1 (bi-grame) sau *m*=2 (trigrame)

Modele de limbă de tip n-gram - construcție

- Exemplu pentru un model de tip bi-gram (secvențe de 2 cuvinte)
- Probabilitățile $p(w_j|w_i)$ pentru fiecare pereche (w_i, w_j) se estimează utilizând un corpus de text de antrenare, astfel:

$$p(w_j \mid w_i) = \frac{count(w_i, w_j)}{\sum_w count(w_i, w)}$$

- Dimensiuni tipice pentru corpusul de text de antrenare:
  - Zeci de mii de cuvinte pentru sisteme RAV adaptate la un domeniu restrâns
  - Sute de milioane de cuvinte pentru sisteme RAV fără constrângeri
- Metode de evitare a data sparseness: metode de netezire și back-off

Modele de limbă de tip n-gram - evaluare

- În contextul unui sistem de RAV
  - WER (rata de eroare la nivel de cuvânt)
- Independent, fără un sistem de RAV
  - Perplexitatea (măsoară capacitatea de predicție a unui model)
  - Rata de cuvinte în afara vocabularului (OOV-rate)

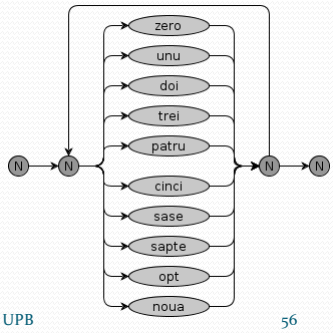
Gramatici cu reguli

- Aplicații de recunoaștere a vorbirii continue cu vocabular mare -> modele de limbă de tip n-gram
- Aplicații RAV de comandă și control -> gramatici cu reguli
- Gramaticile cu reguli specifică explicit:
  - Vocabularul de cuvinte posibile
  - Secvențele de cuvinte permise
  - Probabilitățile de apariție a cuvintelor în anumite contexte

Gramatici cu reguli – moduri de reprezentare

- Exemplu: sarcina de recunoaștere a cifrelor conectate
  - Vocabular de 10 cuvinte: zero, unu, doi, ..., nouă
  - Reguli: orice cuvânt poate apărea de oricâte ori, în orice context
- JSGF – Java Speech Gramar Format(http://java.sun.com/products/java-media/speech/forDevelopers/JSGF/index.html)

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



- FSG – Finnite State Grammar
- altele

Cuprins

- Recunoaşterea automată a vorbirii (RAV): introducere
- Recunoaşterea automată a vorbirii: principii generale
- Resurse fonetice, acustice şi lingvistice
- Modelarea acustică în sisteme de RAV
- Modelarea limbajului în sisteme de RAV
- Concluzii, discuţii şi demonstraţie practică
  - De reţinut despre RAV
  - Limitările şi direcţiile viitoare în RAV
  - Primul sistem RAV cu vocabular extins pentru limba română

Concluzii

- Recunoaşterea automată a vorbirii este abordată statistic!
- Componentele principale ale unui sistem de RAV sunt:
  - Modelul acustic ("urechea")
  - Modelul de limbă ("creierul")
  - Modelul fonetic ("vocabularul")
- Complexitatea RAV depinde de:
  - Dimensiunea vocabularului
  - Calitatea înregistrării
  - Variabilitatea semnalului vocal

Limitări şi direcții de cercetare

- Nu există sisteme de RAV cu acuratețe 100%
- Direcții de cercetare
  - Creșterea robusteții față de zgomot
  - RAV cu vocabular deschis
  - RAV pentru limbi cu resurse puține
  - RAV pentru vorbitori non-nativi
- Alte domenii de cercetare care includ RAV:
  - Înțelegerea limbajului vorbit
  - Traducerea limbajului vorbit
  - Căutarea în baze de date de vorbire
  - RAV multi-limbă

17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 59

Demo & Discuții

17.06.2013 Laboratorul de cercetare Speech & Dialogue | UPB 60