UNIVERSITATEA "POLITEHNICA" din BUCUREŞTI
FACULTATEA: Electronică, Telecomunicații şi Tehnologia Informației
CATEDRA: Dispozitive, Circuite şi Aparate Electronice

TEZA DE DOCTORAT

-REZUMAT-

INTERFEȚE DE COMUNICARE PRIN VOCE CU DISPOZITIVE PORTABILE

VOICE COMMUNICATION INTERFACES FOR MOBILE DEVICES

Autor:

Ing. Cătălin Ungurean

Conducător de doctorat:

Prof. dr. ing. Corneliu Burileanu

Bucureşti – 2011

CUPRINS

1. Introducere

1.1 Motivarea alegerii temei

Scopul final urmărit în proiectarea interfețelor de comunicare prin voce este acela de a uşura comunicarea om-maşină, fie că acest lucru este făcut într-un singur sens (numai recunoaştere sau numai sinteză) sau în ambele sensuri (atât recunoaştere cât şi sinteza vorbirii). Implementarea tehnologiei vorbirii poate fi făcută atât sub forma unor aplicații special destinate şi create pentru acest scop, cât şi sub forma unor pachete de dezvoltare care sunt destinate pentru extinderea unor aplicații existente (al căror scop final poate fi altul decât cel al prelucrării vorbirii) sau pentru crearea de noi aplicații.

Interfețele de comunicare prin voce vor ocupa cu siguranță un loc central în conceptul de interfață universală către care evoluăm [THO07]. Conceptul de universalitate în proiectarea interfețelor de comunicare prin voce se referă la: tipurile de aplicații, la tipul de utilizatori țintă, la caracteristicile sonore ale mediului ambiant în care se desfăşoară dialogul, la infrastructura hardware pe care este implementată interfața şi la tipurile de interacțiune dintre interfața de comunicare prin voce cu celelalte moduri de interacțiune om-maşină.

Pentru a obține atributul de interfață universală, o abordare *multimodală* a interfețelor de comunicare este binevenită, în ciuda faptului că interfețele de acest tip sunt încă destul de rar întâlnite. Astfel de sisteme aduc interfeței robustețe în funcționare şi ajută utilizatorii în corectarea erorilor, adaugă tipuri distincte de canale de comunicație şi medii de funcționare. Folosirea interfețelor *multimodale* poate stimula înțelegerea modului de funcționare a sistemelor şi chiar crea alte moduri de interacțiune [SMI96].

În ceea ce priveşte universalitatea interfețelor de comunicare prin voce abordată din perspectiva infrastructurii hardware de implementare rezultă patru categorii mari de interfețe: de tip *PC-based*, centrate pe rețea, distribuite şi de tip *embedded*. Trebuie menționat că, odată cu dezvoltarea tehnologică, datorită convergenței eforturilor de cercetare în domeniile computerelor, comunicațiilor şi a prelucrării semnalelor, este posibil să asistăm la existența simultană, pe un singur dispozitiv de tip *smartphone* sau PDA (*Personal Digital Assistant*) a tuturor tipurilor de interfețe de comunicare prin voce. Universalitatea se referă şi la sistemele de operare (SO) care stau la baza funcționării unor astfel de dispozitive; astfel, se preconizează că toate SO serioase vor fi dotate standard cu module de recunoaştere şi de sinteză.

De ce dispozitive portabile? O altă direcție către care se îndreaptă societatea tehnologică la ora actuală este spre conceptul de pervasive computing, în care dispozitivele portabile ocupă un loc central, către care se îndreaptă eforturile de cercetare în domeniul IT&C şi care este susținut de dezvoltarea tehnologică prin creşterea puterii de calcul a microprocesoarelor, simultan cu evoluția tehnologiilor de comunicație. Dispozitivele care intră în această clasă pot avea diferite forme şi dimensiuni, de la unități portabile, de tipul telefoanelor mobile, PDA etc. şi până la cele aproape invizibile, omniprezente în mediul ambiant. Interfețele om-maşină de comunicare prin voce, chiar şi cele de tip comandă-control, sunt inspirate din dialogul interuman. Dialogul natural cu o maşină reprezintă încă *Holy Grail*-ul către care se îndreaptă eforturile cercetărilor în domeniu deoarece discursul, în sens general, conține atât elemente verbale cât şi non-verbale. Astfel, deşi o mare cantitate de informație este conținută în mesajul vorbit în sine, intonația, gesturile, pauzele dintre cuvinte, retorica, aduc aportul lor la transmiterea mesajului.

Ideea centrală de la care am plecat în cercetare a fost dezvoltarea unui sistem de sinteză pornind de la text (TTS – *Text-To-Speech*) pentru limba română în care vorbirea sa fie cât mai fluentă şi mai naturală. Cercetările efectuate în acest domeniu în ultimii ani au demonstrat cu prisosință faptul că realizarea acestui deziderat nu poate fi o reuşită individuală şi că este necesară segmentarea problemei în cauză pe cel puțin două paliere mari: prelucrarea limbajului natural respectiv generarea semnalului vorbit.

1.2 Obiectul lucrării

Scopul acestei lucrări a fost de a aduce contribuții în domeniul tehnologiei vorbirii în general şi în cel al sintezei vorbirii pornind de la text pentru limba română în mod particular, având ca segment de aplicabilitate finală dispozitivele portabile.

Obiectivele lucrării au fost următoarele:

- Reliefarea importanței majore pe care o are nivelul de prelucrare a limbajului natural (NLP – *Natural Language Processing*) într-un sistem de comunicare prin voce şi explicarea rolului elementelor componente ale NLP pentru un sistem de sinteză a vorbirii pornind de la text pentru limba română.
- Obținerea unor resurse lingvistice care să creeze infrastructura pe care să se desfăşoare etapa de implementare propriu-zisă. Aceasta se referă la: corpusuri de texte de antrenare – testare pentru limba română, dicționare de despărțire în silabe, dicționare de poziționare a accentelor lexicale şi de conversie fonetică, dicționare de excepții pentru diverşii algoritmi de prelucrare lingvistică etc.
- Dezvoltarea întregului nivel de prelucrare a limbajului natural, etapă obligatorie pentru obținerea unui sistem de sinteză a vorbirii pornind de la text de foarte bună calitate.
- Implementarea întregului etaj NLP pe un dispozitiv portabil (ca reprezentant al clasei *embedded*), în vederea punerii în evidență. a eventualelor constrângeri arhitecturale aduse de aceste tipuri de dispozitive şi a găsirii soluțiilor posibile de rezolvare a acestora. În plus, deoarece această etapă este necesară în vederea obținerii ulterioare a unui sistem complet de sinteză a vorbirii pe astfel de dispozitive, s-a dorit să se obțină o abordare conformă cu specificațiile *Speech Synthesis Markup Language* (SSML) care tind să devină la ora actuală un standard în domeniul realizării interfețelor de sinteza vorbirii pornind de la text.

1.3 Organizarea lucrării

Lucrarea a fost structurată în zece capitole astfel:

În capitolul 2 autorul realizează o introducere în tehnologia vorbirii şi a interfețelor de comunicare prin voce şi descrie arhitectura sistemului TTS pentru limba română care este în curs de implementare la această oră.

În capitolul 3 autorul explică unele elemente ale teoriei matematice care stau la baza modelelor de prelucrare lingvistică de tip statistic (cu referire la teoria *n*-gramelor), în perspectiva utilizării lor într-un sistem TTS pentru limba română.

În capitolul 4 este descris un algoritm original de refacere a semnelor diacritice pentru textele scrise în limba română.

În capitolul 5 autorul face o comparație între metodele bazate pe reguli şi cele de tip statistic, pretabile a fi folosite pentru rezolvarea problemei predicției accentului lexical pentru limba română, şi descrie un algoritm propriu, de tip statistic, pentru rezolvarea acestei probleme.

În capitolul 6 sunt explicate elementele caracteristice ale limbii române în ceea ce priveşte despărțirea automată în silabe, etapă necesară în vederea obținerii unei prozodii de calitate. De asemenea este explicat modul în care poate fi implementat un algoritm hibrid (după criterii statistice şi pe bază de reguli) de despărțire automată în silabe, ca parte componentă a unui nivel de prelucrare a limbajului natural dintr-un sistem TTS.

În capitolul 7 autorul descrie aspectele legate de conversia fonetică pentru limba română, etapă esențială pentru un algoritm de sinteza vorbirii pornind de la text indiferent de tehnologia de sinteză folosită, şi abordează o soluție în esență bazată pe reguli în vederea conversiei de la grafeme la alofone.

În capitolul 8 autorul realizează un algoritm pentru preprocesarea şi normalizarea textelor furnizate la intrarea sistemelor TTS, dezvoltând o versiune deja existentă şi raportată în [BUR99a] şi extinde acest algoritm cu alte noi elemente.

Capitolul 9 este dedicat în întregime realizării unei interfețe de implementare a tuturor modulelor de prelucrare a limbajului natural, descrise anterior, pe un dispozitiv de tip *embedded*, ca parte componentă a unei aplicații de citire a mesajelor de poştă electronică, în vederea înglobării ulterioare într-un sistem complet de sinteză pornind de la text.

Capitolul 10 este destinat concluziilor şi direcțiilor de cercetare viitoare.

Lucrarea cuprinde 151 de pagini. Textului lucrării îi sunt asociate 24 de figuri şi 19 tabele. Numărul referințelor bibliografice este de 120, realizatorul tezei de doctorat fiind coautor la 9 dintre acestea.

2. Interfețe de comunicare prin voce

2.1 Aspecte generale

2.1.1 Multidisciplinaritate în proiectarea interfețelor de comunicare prin voce

Proiectarea unei interfețe de comunicare prin voce este o sarcină multidisciplinară, la fel ca şi cea a realizării altor interfețe de tip om – maşină. Deoarece sunt necesare cunoştințe aprofundate din diferite discipline, o echipă de proiectare într-un astfel de domeniu ar trebui să aibă un caracter mixt, colaborând la realizarea diferitelor module care compun un astfel de sistem, lucru care de multe ori este ignorat, sarcina considerându-se ca fiind realizabilă în totalitate de către ingineri.

2.1.2 Interfața vocală, parte componentă a interfeței *multimodale*

Interacțiunile *multimodale* de tip om-computer pot fi abordate din mai multe perspective. Astfel, se poate face o diferențiere între sistemele de interacțiune aflate de partea umană şi cele de partea calculatorului sau a maşinii . Modurile de intrare dinspre utilizator sunt următoarele: vorbirea, gura, privirea, ochii, fața, atingerea, gesturile, mâna, scrierea (desenul), mirosul.

2.2 Clasificarea interfețelor de comunicare prin voce

2.2.1 Clasificarea interfețelor din punct de vedere al aplicațiilor

A. Interfețe de recunoaştere a vorbirii şi a vorbitorului
B. Interfețe pentru sinteza vorbirii
C. Codarea vorbirii

2.2.2 Tipuri de interfețe din punct de vedere al arhitecturii de implementare

A. Interfețe de comunicare prin voce de tip *PC-based*
B. Interfețe de comunicare prin voce centrate pe rețea
C. Interfețe de comunicare prin voce de tip *embedded*

2.3 Un sistem de sinteză pornind de la text pentru limba română

2.3.1 Descriere generală a sistemului TTS

Etapele de bază ale sintezei pornind de la text pot fi descrise printr-un număr de transformări succesive ce trebuie aplicate asupra şirului de caractere ce reprezintă textul de intrare [BUR99b][JIT02] până la obținerea mesajului vorbit. Mesajul vorbit de ieşire trebuie să transmită conținutul informațional inserat în textul furnizat sistemului, la o calitate, exprimată prin inteligibilitate şi naturalețe, cât mai ridicate. Pentru obținerea acestor parametri, sistemul nostru TTS pentru limba română cuprinde două niveluri de prelucrare. Astfel, există o primă etapă, de prelucrare a limbajului natural, destinată pentru analiza textului de intrare, conversia fonetică şi estimarea prozodiei şi o a doua de generare a semnalului vocal care funcționează prin concatenarea de segmente acustice.

2.3.2 Descrierea rolului componentelor etajului NLP

Obținerea unui nivel de prelucrare a limbajului natural de cea mai bună calitate a fost sarcina principală a autorului acestei lucrări în vederea obținerii unui sistem TTS pentru limba română în care vorbirea să fie cât mai inteligibilă şi naturală. Rolul etajului de prelucrare a limbajului natural din sistemul nostru de sinteză este de a transforma textul de intrare într-o reprezentare fonetică şi prozodică, care trebuie să descrie cât mai fidel posibil pronunția sa. Acest lucru poate fi realizat parcurgând mai multe etape succesive, care au fost reprezentate în figura 2.1.

Vom prezenta în cele ce urmează principalele caracteristici şi roluri ale etajelor de prelucrare a limbajului natural pentru un sistem de sinteză performant, majoritatea acestora fiind deja implementate în sistemul nostru TTS. Astfel, am propus o abordare modulară, în care diverse etaje NLP fac schimb de date pentru a obține rezultatul final.

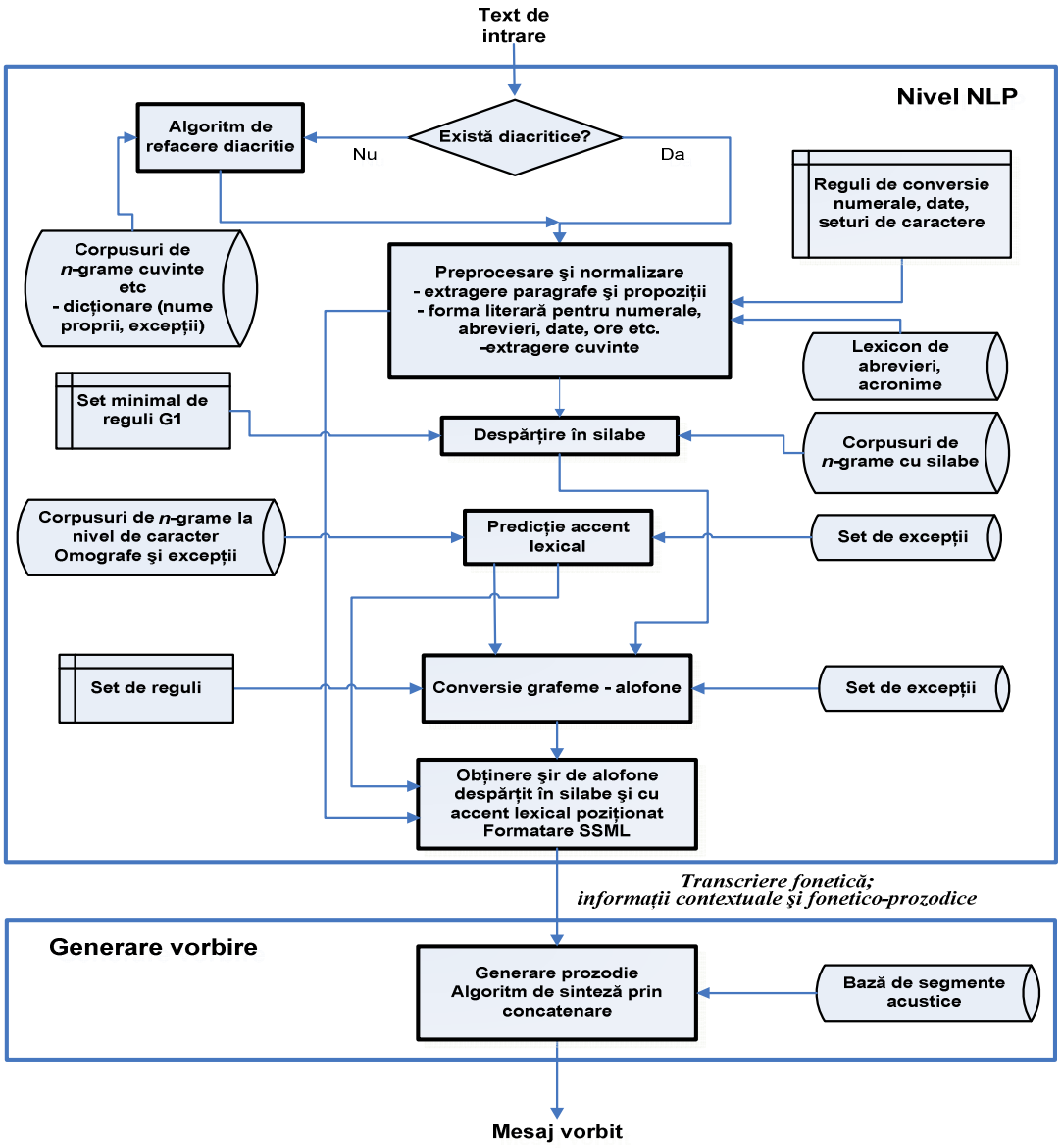

Figura 2.1 Schema sistemului TTS pentru limba română, cu evidențierea etajului de prelucrare a limbajului natural

A. Modulul de refacere a diacriticelor

Rolul său este acela de a detecta dacă un text de intrare este scris sau nu cu semne diacritice şi de a declanşa algoritmul de refacere a acestor semne.

B. Modulul de normalizare şi preprocesare a textului de intrare

Sarcina acestui *modul de normalizare şi preprocesare* este de a transforma textul de intrare într-o formă care să poată fi corect prelucrată ulterior.

C. Modulul de despărțire automată în silabe

Rolul acestui modul este multiplu. În primul rând el furnizează informații pentru etajul de generare a prozodiei deoarece împreună cu modulul de predicție a accentului lexical ajută la poziționarea acestui tip de accent la nivelul silabelor, pentru a le evidenția pe cele accentuate. În al doilea rând, acest modul, împreună cu modulul de conversie fonetică, ajută la obținerea formei finale de transcriere fonetică, care conține accentul lexical corect poziționat pe forma convertită fonetic şi despărțită în silabe a fiecărui cuvânt. Mai mult, despărțirea automată în silabe mai foloseşte şi pentru deducerea informațiilor contextuale utile pentru alegerea segmentelor acustice în momentul concatenării.

D. Modulul de conversie grafeme-alofone (transcriere fonetică)

Modulul de transcriere fonetică (conversia litere – foneme, sau, mai corect pentru limba română, *grafeme – alofone*), transformă secvențele de caractere ortografice în secvențe fonetice (foneme de bază împreună cu anumite variante de pronunție ale lor).

E. Elemente de analiză prozodică

O problemă importantă pentru analiza prozodică în sistemele TTS este aceea a poziționării corecte a accentelor, fiind evident faptul că un accent fixat incorect poate reduce mult naturaleţea rostirii, sau chiar inteligibilitatea sa (cazul cuvintelor omografe). Din punct de vedere lingvistic, accentul se poate încadra în două categorii principale:

- accent lexical;
- accent la nivel de frază.

2.3.3 Etajul de generare a semnalului vocal folosit în sistemul TTS pentru limba română

În ceea ce priveşte etapa de generare a semnalului vocal trebuie precizat faptul că în cadrul colectivului de cercetare din care autorul acestei lucrări a făcut parte au fost realizate până la ora actuală mai multe implementări. Astfel, prima versiune de sistem, descrisă în [BUR99b], a fost realizată prin concatenare de difoneme, şi reprezintă prima implementare a unui sistem TTS pentru limba română.

Pentru sistemul actual de sinteză sunt folosite în continuare tehnici de concatenare, însă apar câteva diferențe majore față de sistemul inițial. Prima diferență constă în faptul că unitățile de selecție pentru concatenare nu mai sunt uniforme, putând exista atât difoneme cât şi polifoneme, în plus existând mai multe instanțe pentru unitățile de acelaşi tip. A doua diferență majoră este că în versiunea actuală s-a decis folosirea unui model de sinteză mai evoluat şi anume *Harmonic plus Noise Model* (HNM) pentru o mai bună concatenare şi generare prozodică. [NEGR07][NEGR09][BUR10a].

3. Tehnici de prelucrare statistică: *n*-grame

3.1 Introducere. Tipuri de modele lingvistice

Un model de limbaj reprezintă un element deosebit de important al oricărui sistem de prelucrare a vorbirii (recunoaştere sau sinteză automată), care îmbunătăţeşte performanțele sistemului prin aplicarea unor constrângeri deprinse (deduse) prin antrenare, de regulă prin înglobarea unor informații însuşite pe baza unor corpusuri de antrenare (colecții mari de texte). De exemplu, în tehnologia vorbirii rolul modelelor lingvistice este de a elimina acele succesiuni grafemice (cuvinte, silabe, caractere etc.) care au o semnificație mai puțin probabilă dată de un model.

Cele mai cunoscute modele de limbaj sunt fie bazate pe reguli fie sunt statistice. În capitolul de față am descris unul dintre cele mai importante modele statistice de limbaj, şi anume pe cel al *n*-gramelor, dar trebuie spus că există şi alte astfel de modele (de exemplu HMM-*Hidden Markov Model*) care au fost folosite pentru realizarea unor sarcini de prelucrare a limbajului natural.

În sinteza vorbirii pornind de la text rolul etajului de prelucrare de limbaj, după cum precizam în capitolul anterior, este acela de a transforma textul de intrare într-o reprezentare fonetică şi prozodică, care trebuie să descrie cât mai fidel posibil pronunția sa.

3.2 Modele lingvistice statistice bazate pe *n*-grame

Aceste modele au o răspândire largă nu doar în domeniul recunoaşterii şi sintezei vorbirii ci şi în recunoaşterea caracterelor scrise de mână, corecția erorilor de ortografie sau în sistemele de traducere automată.

În formularea un model matematic de limbaj se pleacă de la formula de calcul a probabilităților condiționate şi de la teorema lui Bayes. Astfel, probabilitatea ca un eveniment *A* să se producă – *P(A)*, în condițiile în care ştim că s-a produs evenimentul *B* poate fi scrisă ca:

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)} \tag{3.1}$$

$$P(A \cap B) = P(B)P(A \mid B) = P(A)P(B \mid A) \tag{3.2}$$

în care P(B) este probabilitatea producerii evenimentului B, iar $P(A \cap B)$ este probabilitatea producerii ambelor evenimente simultan.

Teorema lui Bayes ne permite schimbarea ordinii de dependențe între evenimente şi este o consecință firească a ecuației 3.2:

$$P(B \mid A) = \frac{P(B \cap A)}{P(A)} = \frac{P(A \mid B)P(B)}{P(A)} \tag{3.3}$$

Din perspectiva aplicațiilor de prelucrare a vorbirii din acest punct lucrul poate fi îndreptat atât către aplicațiile de recunoaştere cât şi către cele de sinteză. Astfel, să presupunem că avem un sistem TTS care primeşte la intrare un text W şi care trebuie să producă la ieşire un sistem de vectori acustici A. Vectorii acustici sunt aleşi astfel încât să maximizeze probabilitatea condiționată *P*(*A*|*W*).

$$A^* = \arg\max_A \{P(A \mid W\} = \arg\max_A \{\frac{P(W \mid A)P(A)}{P(W)}\} = \arg\max_A \{P(W \mid A)P(A)\}, \tag{3.4}$$

unde *P*(*A*) reprezintă modelul de sinteză iar *P*(*W*|*A*) reprezintă probabilitățile calculate apriori.

Teoria statistică fundamentală care a stat la baza implementării etajelor noastre de NLP este cea a *n*-gramelor. Un model lingvistic poate fi implementat la nivel de cuvânt, caractere, silabe etc. Pentru a crea şi descrie un astfel de model trebuie să dispunem de o serie de elemente cum ar fi un vocabular V şi un corpus (colecție) de texte de antrenare T. Astfel, în cazul unui model implementat la nivel de caracter, V este chiar alfabetul folosit, la nivel de cuvânt este un întreg dicționar ş.a.m.d. Pe de altă parte, în cazul implementării la nivel de cuvânt, corpusul de antrenare, poate fi o colecție de texte, din diverse domenii, de dimensiuni cât mai mari. Considerând evenimentele ca fiind evenimente aleatoare discrete, pe baza corpusului de texte de antrenare T trebuie să asociem, fiecărei entități w_i din vocabularul V, un set corespunzător de probabilități P(wi), astfel încât ele să satisfacă condițiile generale pentru o distribuție de probabilitate:

$$\sum_{w_i \in V} P(w_i) = 1 \tag{3.5}$$

$$P(w_i) \geq 0, \forall w_i \in V \tag{3.6}$$

Pentru modelele de *n*-grame, va trebui să calculăm distribuția de probabilitate $P(w_1^N)$ a unei secvențe de cuvinte (sau alte elemente luate în calcul, de exemplu caractere): $w_1^N = (w_1, w_2, ..., w_N)$. Probabilitatea $P(w_1^N)$ poate fi recalculată folosind probabilități condiționate, astfel [MAN00]:

$$\begin{aligned} P(w_1^N) &= P(w_1)P(w_2 \mid w_1)P(w_3 \mid w_1 w_2)...P(w_N \mid w_1 w_2 ... w_{N-1}) = \\ &= \prod_{i=1}^{N} P(w_i \mid w_1 w_2 ... w_{i-1}) = \prod_{i=1}^{N} P(w_i \mid w_1^{i-1}) \end{aligned} \tag{3.7}$$

Se observă că pentru estimarea unei astfel de secvențe de evenimente, ar trebui cunoscut tot lanțul de probabilități condiționate componente. În practică acest lucru nu este productiv şi se apelează la prezumția Markov, conform căreia probabilitatea unui eveniment depinde numai de n-1 evenimente precedente, (termen cunoscut sub titlul de istoria unui cuvânt) ajungându-se la modelul n-gramelor.

Ecuația 3.7 poate fi aproximată pe baza lanțurilor Markov astfel:

$$P(w_1^N) = \prod_{i=1}^{N} P(w_i \mid w_{i-n+1}^{i-1}) \tag{3.8}$$

3.2.1 Modelul unigramelor

Pentru modelul unigramelor se foloseşte prezumția că toate cuvintele sunt independente, altfel spus, nu se ia în considerare istoria unui cuvânt la estimarea probabilității sale de apariție într-un text de intrare.

$$P(w_k \mid w_1^{k-1}) \approx P(w_k) \tag{3.9}$$

Dacă înlocuim relația (3.9) în ecuația (3.8), rezultă relația de calcul a probabilităților pentru modelul lingvistic cu unigrame:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k) \tag{3.10}$$

Probabilitatea fiecărui cuvânt P(wk) se poate exprima prin frecvența relativă a respectivului cuvânt în colecția de texte de antrenare T:

$$P(w_k) = \frac{n_k}{N} \tag{3.11}$$

În relația de mai sus n_k este numărul de apariții al cuvântului w_k, iar N este numărul total de cuvinte din setul de antrenare.

3.2.2 Modelul bigramelor

În cazul modelului cu bigrame se consideră, că orice cuvânt depinde doar de cuvântul care îl precede. Astfel istoria cuvântului are lungimea unu.

$$P(w_k \mid w_1^{k-1}) \approx P(w_k \mid w_{k-1}) \tag{3.12}$$

Înlocuind această prezumție în ecuația (3.8), obținem formula de calcul pentru modelul lingvistic cu bigrame:

$$P(w_1^n) \approx \prod_{k=1}^{n} P(w_k \mid w_{k-1}) \tag{3.13}$$

Pentru evaluarea probabilităților condiționate de forma $P(w_k \mid w_{k-1})$, folosim metoda MLE (*Maximum Likelihood Estimation*), numărând aparițiile secvențelor (w_{k-1},w_k) şi a secvențelor ce încep cu cuvântul notat [CHE96].

$$P(w_k \mid w_{k-1}) = \frac{P(w_{k-1}, w_k)}{P(w_{k-1})} = \frac{C(w_{k-1}, w_k)}{\sum_{w_k} C(w_{k-1} w_k)} = \frac{C(w_{k-1}, w_k)}{C(w_{k-1})}, \tag{3.14}$$

unde $C(n_x)$ reprezintă numărul de apariții pentru n-grama n_x.

3.2.3 Modelul trigramelor

În cazul modelului trigramelor, pentru evaluarea probabilității unui cuvânt se iau în considerare două cuvinte precedente (istoria cuvântului este de lungime doi). Prezumția Markov utilizată în cazul acestui model poate fi scrisă:

$$P(w_k \mid w_1^{k-1}) \approx P(w_k \mid w_{k-1}, w_{k-2}) = P(w_k \mid w_{k-2}^{k-1}) \tag{3.15}$$

Înlocuind din nou în ecuația 3.8, obținem relația de calcul a probabilităților dată de modelul trigramelor:

$$P(w_1^n) = \prod_{k=1}^{n} P(w_k \mid w_{k-1}, w_{k-2}) \tag{3.16}$$

Pentru evaluarea probabilităților condiționate, putem folosi relațiile date de modelul bigramelor, sau relația:

$$P(w_k \mid w_{k-1}, w_{k-2}) = \frac{C(w_{k-2}, w_{k-1}, w_k)}{C(w_{k-2}, w_{k-1})} \tag{3.17}$$

3.3 Metode de netezire a probabilităților pentru modelele lingvistice

Filozofia algoritmilor de netezire [CHE96][MAN00] este de a reestima probabilitățile evenimentelor, în special pentru probabilitățile nule sau foarte mici, prin alocarea către acestea a unei mase de probabilitate nenulă, extrasă din evenimentele cu probabilitate mai mare şi cu conservarea masei totale de probabilitate.

3.3.1 Estimatorul Good – Turing

Estimatorul GT (Good – Turing) este fundamental pentru multe dintre tehnicile de netezire actuale. Metoda pleacă de la ipoteza că pentru fiecare *n*-gramă care apare de *r* ori în corpusul de antrenare, putem pretinde că aceasta apare de r^* ori unde acest termen se calculează după formula:

$$r^* = (r+1)\cdot\frac{n_{r+1}}{n_r}, \tag{3.18}$$

unde n_r este numărul de *n*-grame care apar de exact *r* ori în corpusul de antrenare.

Probabilitatea dată de modelul GT – p_{GT}, calculată cu noile valori netezite va fi dată de relația:

$$P_{GT} = \frac{r^*}{N} \text{ unde } N = \sum_{r=0}^{\infty} n_r r^* \tag{3.19}$$

3.3.2 Metoda Jelinek – Mercer (interpolarea liniară)

Metoda Jelinek – Mercer, interpolează *n*-gramele de ordin superior cu modelele de ordin inferior deoarece, atunci când nu există suficiente date pentru estimarea probabilității modelului de ordin superior, modelul de ordin inferior poate aduce informații utile.

3.3.3 Metoda de netezire Katz (metoda cu revenire)

Algoritmul Katz [KAT87] extinde metoda de netezire Good – Turing, prin combinarea modelelor *n*-gramelor de ordin superior cu cele de ordin inferior. Vom explica metoda plecând în primul rând de la modelul bigramelor.

Pentru o bigramă w_{i-1}^i care are un număr de apariții în corpusul de antrenare de $r = c(w_{i-1}^i)$ numărul de apariții netezit este calculat cu formula:

$$c_{katz}(w_{i-1}^i) = \begin{cases} d_r r & dacă\ r > 0 \\ \alpha(w_{i-1})\, p_{ML}(w_i) & dacă\ r = 0 \end{cases} \tag{3.20}$$

Cu alte cuvinte bigramele cu un număr de apariții diferit de zero sunt diminuate după un factor de diminuare d_r dat de algoritmul Good-Turing, şi are valoare aproximativă $\frac{r^*}{r}$. Probabilitatea extrasă din evenimentele cu un număr de apariții nenule este distribuită evenimentelor (bigramelor) cu o frecvență nulă de apariție de ordin imediat inferior. Factorul de multiplicare $\alpha(w_{i-1}^i)$ este ales astfel încât numărul total de evenimente din distribuție rămâne nemodificat, deci $\sum_{w_i} c_{katz}(w_{i-1}^i) = \sum_{w_i} c(w_{i-1}^i)$.

Valoarea lui $\alpha(w_{i-1}^i)$ este dată de formula:

$$\alpha(w_{i-1}) = \frac{1-\sum_{w_i:c(w_{i-1}^i)>0} P_{katz}(w_i \mid w_{i-1})}{\sum_{w_i:c(w_{i-1}^i)=0} P_{ML}(w_i)} = \frac{1-\sum_{w_i:c(w_{i-1}^i)>0} P_{katz}(w_i \mid w_{i-1})}{1-\sum_{w_i:c(w_{i-1}^i)>0} P_{ML}(w_i)} \tag{3.21}$$

Probabilitatea calculată de algoritmul cu revenire este:

$$P_{katz}(w_i \mid w_{i-1}) = \frac{c_{katz}(w_{i-1}^i)}{\sum_{w_i} c_{katz}(w_{i-1}^i)} \tag{3.22}$$

Factorul de diminuare d_r se obține astfel: frecvențele de apariție mari, peste un prag *k*, sunt considerate corecte, iar d_r =1. În algoritmul Katz original, valoarea lui *k* a fost propusă 5. Pentru *n*-gramele din corpusul de antrenare întâlnite de un număr de ori mai mic decât *k*, d_r este dat de formula:

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)n_{k+1}}{n_1}}{1 - \frac{(k+1)n_{k+1}}{n_1}} \tag{3.23}$$

3.3.4 Metoda de netezire Kneser – Ney

Netezirea inițială introdusă de Kneser şi Ney se mai numeşte şi **netezire absolută**, deoarece se bazează pe metoda scăderii absolute. Scăderea absolută înseamnă de fapt, că se scade o valoare constantă ***d*** din fiecare număr de apariții ale secvențelor de *n*-grame mai mare decât 0.

3.4 Reducerea dimensiunilor modelelor lingvistice

Pentru a putea fi folosite eficient, este necesară reducerea, micşorarea sau compresia modelelor, denumită în literatura de specialitate *n-gram pruning*. Metodele de tip *data pruning* îşi pot dovedi eficiența mai ales pentru sistemele de tip embedded, sau pentru cele caracterizate prin resurse de memorie şi putere de calcul limitate.

3.4.1 Reducerea prin eliminare

Este cea mai simplă şi utilizată metodă de reducere a modelelor lingvistice şi presupune eliminarea *n*-gramelor a căror frecvență de apariție, calculată pe baza unui corpus de antrenare *T*, este sub o valoare de prag *k*.

3.4.2 Reducerea probabilistică

Este o metodă naturală de reducere a modelelor lingvistice. Deoarece perplexitatea este cea mai bună caracteristică de evaluare a calității unui model, reducerea probabilistică se bazează pe aceasta.

3.4.3 Reducerea bazată pe scor

Metoda de față constă în ideea de a scoate dintr-un model toate secvențele de cuvinte care după eliminare schimbă un scor mai puțin decât un prag impus.

4. Modulul de restaurare a diacriticelor

4.1 Introducere

În afara constrângerilor specifice ale aplicațiilor de sinteza vorbirii (lărgime de bandă telefonică limitată, degradarea semnalului produsă de canalul de comunicație), diversele aplicații implementate cu ajutorul unui sistem TTS trebuie să realizeze de multe ori şi o altă sarcină importantă – cea a refacerii caracterelor scrise cu semne diacritice [KAR06]. Trebuie precizat faptul că sinteza vorbirii pe baza unui text scris fără diacritice, într-o limbă vorbită care foloseşte astfel de semne distinctive, va produce fie mesaje vocale neinteligibile, fie va introduce ambiguități semantice şi sintactice care vor face ca întregul mesaj să fie neinteligibil.

4.2 Restaurarea automată a diacriticelor

4.2.1 Preliminarii

Una dintre sarcinile autorului acestei lucrări a fost aceea de a obține un algoritm automat care să refacă cu o cât mai bună acuratețe caracterele scrise cu diacritice, în perspectiva realizării unei sarcini de sinteză de tip TTS de cât mai bună calitate.

Primul aspect de noutate introdus în metoda noastră constă într-un proces de filtrare secvențială, pe mai multe niveluri, care se bazează pe contextul de apariție a cuvintelor şi pe probabilitatea de apariție, fiecare dintre nivelurile de filtrare aducând un câştig în plus, potrivit unui prag de filtrare prestabilit. În implementarea noastră am ales o abordare pe trei niveluri de filtrare: cel al unigramelor, al bigramelor şi al trigramelor dar pot fi propuse şi alte etaje – cu prefixe, POS etc.

Un al doilea element de noutate introdus este cel de filtrare minimală prin care se îndepărtează numai formele asupra cărora se decide că sunt *sigur greşite*, menținându-se totuşi cât mai mult formele ambigue.

4.2.2 Descrierea algoritmului

Etapa de antrenare presupune realizarea următorilor paşi:

i. *Construirea manuală a unui dicționar D1 care conține cele mai frecvent folosite cuvinte ale limbii române şi care să conțină de asemenea cât mai multe forme flexionate ale acestora.*
ii. *Pe baza lui D1 se construieşte o structură dicționar D2 (hash table) care mapează fiecare formă din D1, cu diacriticele înlocuite cu caracterele de bază, cu toate formele posibile, extrase din D1, care produc aceeaşi formă fără diacritice.*
iii. *Pe baza unui corpus de antrenare A, care conține forme corecte scrise cu diacritice, se construiesc la nivelul cuvintelor, un set de trei dicționare, astfel: o structură U care conține unigramele extrase din dicționarul de antrenare împreună cu frecvențele lor de apariție, o structură B care conține toate bigramele posibile extrase din corpusul A şi o structură T, care conține trigramele la nivelul sufixelor cuvintelor din A.*

În ceea ce priveşte etapa de testare a algoritmului, aceasta este compusă în principal din trei procese de filtrare în cascadă, în care datele de la ieşirea fiecărui etaj superior constituie date de intrare în etajul imediat următor. Intrarea generală în algoritm constă în secvențe de text în care semnele diacritice au fost înlăturate. Algoritmul de testare este următorul:

a. *Pe baza dicționarelor D1 şi D2 sunt inserate formele care nu conțin diacritice şi formele care nu introduc ambiguități, altfel spus, cuvintele de tipul 1 respectiv de tipul 2. Etapa este una de tip determinist şi presupune o înlocuire biunivocă a formei fără diacritice cu cea corectă.*
b. *Pentru fiecare pereche de cuvinte consecutive din textul test, care conține cel puțin o formă ambiguă, se formează setul de variante posibile cu toate aceste cuvinte; dacă cel puțin o pereche din acest set este găsită în corpusul de bigrame B, sunt eliminate toate perechile de variante care nu apar în B, altfel sunt lăsate toate variantele nemodificate.*
c. *Din textul de intrare care conține variantele de ieşire de la pasul 2, se formează toate grupurile de câte trei cuvinte consecutive şi din acestea se extrag sufixele; se obțin astfel perechi alternative de triplete de tip cuvinte-sufixe; dacă există cel puțin o tripletă de sufixe în dicționarul T atunci renunță la toți membrii din setul de cuvinte corespunzătoare tripletelor de sufixe care nu sunt în T şi furnizează spre ieşire doar variantele pentru care sau găsit membri în T; dacă nu s-a găsit niciuna dintre variante în T, lasă tot setul de cuvinte neschimbat.*
d. *Folosind setul de unigrame U, păstrează pentru fiecare cuvânt doar varianta cea mai probabilă, pe baza frecvențelor de apariție calculate din corpusul de antrenare, dintre variantele furnizate la pasul 3.*

Studiile realizate de colectivul nostru au arătat de asemenea că în limba română cele mai semnificative probleme pentru refacerea diacriticelor sunt cele create de ambiguitatea **a-ă** la finalul cuvintelor. Acest fenomen este produs cu o frecvență mare de apariție de unele construcții lingvistice scurte precum **ca/că**, **sa/să**, **sau/său** etc. şi de formele articulate/nearticulate ale substantivelor feminine: **casa/casă**, **piața/piață** etc.

4.3 Experimente

4.3.1 Contextul de evaluare

În algoritmul descris anterior, în secțiunea 4.2, antrenarea a fost făcută pe un corpus extras din texte literare scrise în limba română care conține peste 10^7 cuvinte[1]. Aceste texte au contribuit de asemenea la extinderea dicționarului D1, construit inițial manual, pe baza **Dicționarului Explicativ al Limbii Române**. Din păcate acest ultim dicționar conține numai formele de bază ale cuvintelor, adică aproximativ 70.000 de cuvinte diferite, iar pentru rularea algoritmului am avut nevoie de un dicționar care să conțină cât mai multe forme morfologice flexionate. La ora actuală dicționarul conține aproximativ 330.000 de forme şi include aproximativ 6.000 de nume proprii. Dicționarele (vom folosi mai departe şi denumirea de corpusuri) de unigrame, bigrame şi trigrame au fost extrase din textele de antrenare şi salvate ca fişiere text separate.

Corpusul de testare a fost corectat manual şi conține propoziții extrase din articole, texte literare şi teze de doctorat din mai multe domenii de activitate, disponibile în format electronic în limba română. Acest corpus are următoarele caracteristici: (i) numărul de cuvinte de intrare: 1.200.000, (ii) procentul de cuvinte scrise fără diacritice: 57,41%, procentul de cuvinte scrise întotdeauna cu diacritice: 16,33%; procentul de cuvinte ambigue la refacerea diacriticelor: 26,26%.

4.3.2 Măsurarea performanțelor

Rezultatele testării algoritmului au fost sintetizate în tabelul 4.1

Clasa de ambiguitate	***Precizie* (%)**	***Recall* (%)**	***Măsura-F* (%)**	***Măsura-F medie* (%)**
Baza de comparație				
a / ă / â	78,03 / 92,69 / 99,50	99,65 / 35,27 / 61,86	87,53 / 51,10 / 76,29	77,11
i / î	98,63 / 99,67	99,96 / 89,54	99,30 / 94,34	98,71
s / ş	94,83 / 97,89	99,48 / 81,48	97,10 / 88,93	95,25
t / ţ	95,30 / 91,25	98,82 / 71,63	97,03 / 80,25	94,57
Bigrame şi unigrame				
a / ă / â	97,82 / 91,58 / 99,24	96,49 / 94,66 / 99,57	97,15 / 93,10 / 99,40	96,19
i / î	99,97 / 99,57	99,94 / 99,82	99,95 / 99,69	99,93
s / ş	99,93 / 99,49	99,85 / 99,76	99,89 / 99,63	99,83
t / ţ	99,86 / 98,07	99,66 / 99,22	99,76 / 98,64	99,60
Bigrame, trigrame şi unigrame				
a / ă / â	98,22 / 93,30 / 99,26	97,22 / 95,62 / 99,55	97,72 / 94,45 / 99,41	96,93
i / î	99,98 / 99,56	99,94 / 99,82	99,96 / 99,7	99,93
s / ş	99,93 / 99,51	99,85 / 99,77	99,89 / 99,64	99,84
t / ţ	99,87 / 98,2	98,69 / 99,27	99,78 / 98,74	99,63

Tabel 4.1 Măsurarea performanțelor algoritmului de restaurare a diacriticelor

[1] Datele de antrenare au fost extrase din următoarele surse publice online: http://www.liternet.ro, http://www.romanialibera.ro, http://www.romanialiterara.ro, http://www.wikisource.org.

Au fost calculate următoarele mărimi de performanță standard:

1. **Precizia**: definită ca raportul dintre numărul total de diacritice inserate corect şi numărul total de diacritice inserate.
2. **Recall**: definit ca raportul dintre numărul total de diacritice inserate corect şi numărul de diacritice din baza de test corectată manual.
3. **Măsura-F**: este definită ca media armonică dintre precizie şi recall.

Rezultatele au fost obținute pentru trei versiuni de rulare ale algoritmului nostru, astfel:

- Ca bază de comparație s-a luat metoda de restaurare a diacriticelor pe baza dicționarului *D*2, pentru cuvintele care nu prezintă ambiguități, fără a fi necesare date de antrenare.
- A doua metodă foloseşte bigramele extrase din corpusul de antrenare şi probabilitățile unigramelor calculate pe baza aceluiaşi corpus.
- Algoritmul complet, care constă din toate cele trei etaje de filtrare cascadate: etajul bigramelor, urmat de cel al trigramelor cu sufixe iar ca nivel final de filtrare este cel al unigramelor.

Tot în tabelul 4.1 am prezentat de asemenea valorile medii statistice ponderate ale *măsurii-F* pentru fiecare clasă de ambiguitate, în care ponderile sunt calculate la nivel de caracter. Se observă că se obține o valoare medie ponderată pentru *măsura-F* de 99,34%. În ceea ce priveşte rata de performanță obținută la nivel de cuvânt am obținut o rată de eroare la nivel de cuvânt de 1,4% pe baza aceloraşi date de test, calculată ca diferența față de *măsura–F*.

4.3.3 Exemplu de interfață de refacere a diacriticelor

Sarcina de refacere a diacriticelor poate fi utilă şi pentru alte aplicații, de aceea am decis obținerea unei aplicații complet independente de acest fel, pentru a rula pe o configurație de tip *PC-based*. Interfața este realizată în Visual Studio dar algoritmul de tip NLP bazat pe metoda descrisă anterior, a fost realizată în Perl şi convertită în fişier executabil cu ajutorul programului *perl2exe* [IND]. în acest fel nu mai este nevoie de instalarea separată a interpretorului Perl pe sistemul de operare deoarece toate bibliotecile Perl vor fi incluse în executabilul obținut. Rolul interfeței (figura 4.2) este acela de introducere şi afişare a textului de intrare, de apelare a executabilului şi de afişare şi salvare a rezultatelor conversiei.

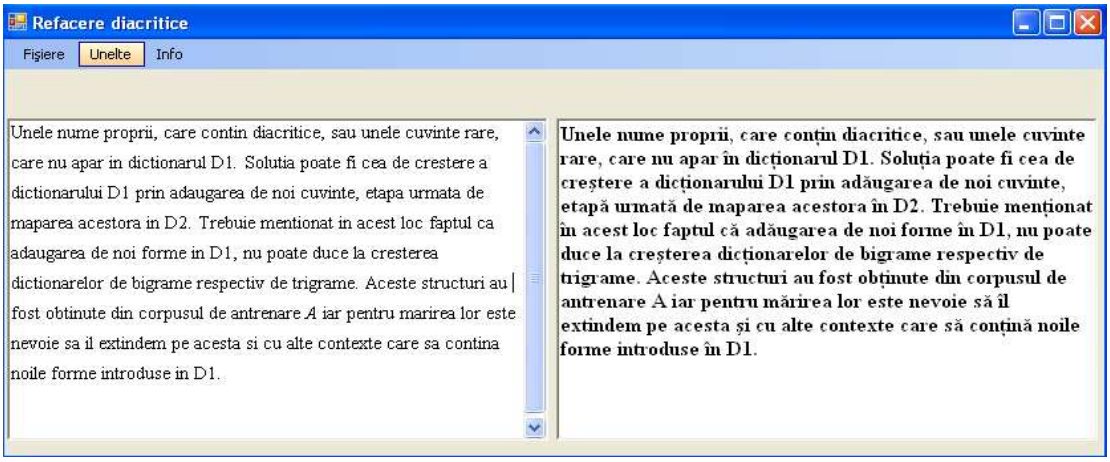

Figura 4.2 Interfață independentă de refacere a diacriticelor

4.4 Concluzii

Algoritmul de refacere a diacriticelor pentru cuvintele scrise în limba română prezentat anterior a dus la obținerea următoarele măsuri de performanță: o rată de eroare la nivel de caracter de 0,65%, iar la nivel de cuvânt de 1,4%, ambele calculate ca diferență față de valorile *măsurii-F*. Aceste rezultate sunt

mai bune decât cele raportate până în acest moment în literatura de specialitate şi discutate în secțiunea 3.2. pentru limba română cu referire la [MIH02], [NEM08] şi [TUF99].

Metoda descrisă este implementată la ora actuală în sistemul nostru TTS dar a fost testată şi într-o aplicație de tip TTS e-mail reader, mai exact înaintea etajului de preprocesare, fiind inclusă după etajele de segmentare textuală şi de detecție a semnelor de punctuație. În plus, pe baza aceleiaşi metode, a fost realizată şi o aplicație independentă de refacere a semnelor diacritice, care poate fi utilă pentru corectarea rapidă a unor texte scrise în limba română, fără diacritice.

5. Modulul de poziționare automată a accentelor lexicale

5.1 Introducere

În multe limbi vorbite accentul lexical reprezintă o componentă fundamentală a pronunției unui cuvânt iar determinarea poziționării corecte a acestuia devine o sarcină principală în predicția şi generarea corectă a prozodiei pentru un sistem TTS de înaltă calitate. Capitolul de față descrie o metodă statistică, bazată în principal pe *n*-grame construite la nivel de caracter, destinată poziționării automate a accentelor lexicale, în vederea folosirii în sistemul nostru de sinteză pornind de la text pentru limba română.

5.2 Noțiuni fundamentale; rezultate anterioare

În ceea ce priveşte limba română, singura încercare de soluționare a problematicii plasării accentelor lexicale din perspectiva folosirii în sistemele TTS, este descrisă în [OAN02]. Autorii propun şi evaluează experimental o metodă bazată pe reguli prin care calculează poziția silabelor accentuate în funcție de mulți parametri morfologici, fonetici şi lexicali, şi anume: numărul de caractere şi silabe, tipul silabelor, secvențele de trei caractere de la începutul şi sfârşitul cuvintelor. Aceşti parametri sunt folosiți în vederea antrenării sistemului prin împărțirea cuvintelor de intrare pe clase iar în vederea recunoaşterilor accentelor cuvintele de test sunt clasificate după clasa de apartenență. A fost raportată o eroare maximă (*Word Error Rate*) de 6% pentru un corpus de test de 4.500 cuvinte, bază de test care a fost folosită şi pentru generarea regulilor. Înainte de a descrie pe larg algoritmul propus, vom explica unele dintre particularitățile accentului lexical pentru limba română. Unii autori [SUT93] sugerează ideea că o regulă destul de generală pentru română este următoarea: *dacă un cuvânt se încheie cu o consoană, accentul este pe ultima silabă iar dacă un cuvânt se încheie cu o vocală, accentul este pe penultima silabă.* Trebuie să menționez că o regulă atât de simplă nu poate fi o soluție validă pentru un sistem TTS de bună calitate, datorită numeroaselor exemple care o contrazic, totuşi aceasta ar putea fi folosită în alte aplicații limitate.

5.3 Descrierea algoritmului de poziționare a accentului lexical

5.3.1 Formularea problemei

Predicția poziției accentului lexical pentru sistemul nostru TTS pentru limba română este realizată statistic, pe baza unui algoritm axat pe *n*-grame, modelarea de limbaj fiind realizată la nivel de caracter [UNG11a][UNG11b]. Abordarea a fost aleasă pe baza experienței acumulate în folosirea metodelor statistice de prelucrare a limbajului natural [UNG08] şi de asemenea pe baza faptului că este dificil de construit un set de reguli care să soluționeze problematica în discuție pentru limba română.

5.3.2 Algoritmul de antrenare

Pentru antrenarea modelelor de *n*-grame am folosit un corpus *T*1, compus din cuvinte având accentul lexical poziționat manual iar delimitatorii de start şi stop poziționați automat. Un corpus de validare (*held-out*), denumit *T*2 a fost folosit în etapa de antrenare în vederea optimizării parametrilor aleşi în etapa de netezire a probabilităților *n*-gramelor.

Pentru fiecare cuvânt din *T*1 am extras toate variantele posibile de *n*-grame şi am calculat probabilitățile acestora. Pentru reducerea efectelor nedorite cauzate de insuficiența datelor de antrenare, am implementat în algoritmul nostru o metodă de netezire a probabilităților de tip Katz cu revenire, care extinde algoritmul Good – Turing de estimare a probabilităților prin combinarea modelelor de *n*-grame de ordin mai mare cu cele de ordin mai mic [KAT87] [CHE96]. Frecvențele de apariție pentru *n*-gramele

generice l_{i-n+1}^{i} sunt modificate după formula:

$$c_{katz}(l_{i-n+1}^{i}) = \begin{cases} d_r r & dacă \quad r > 0 \\ \alpha(l_{i-n+1}^{i-1}) c_{katz}(l_{i-n+2}^{i}) & dacă \quad r = 0 \end{cases}, \tag{5.1}$$

unde *r* este numărul de *n*-grame observate:

$$r = c(l_{i-n+1}^{i}), \tag{5.2}$$

iar $\alpha(l_{i-n+1}^{i-1})$ sunt coeficienți de ponderare care asigură că suma probabilităților este egală cu 1.

În algoritmul Katz original, coeficienții de ponderare $\alpha(l_{i-n+1}^{i-1})$ sunt calculați pentru fiecare istorie l_{i-n+1}^{i-1} astfel încât numărul total de apariții să se conserve, ceea ce se traduce prin $\sum_{l_i} c_{katz}(l_{i-n+1}^{i}) = \sum_{l_i} c(l_{i-n+1}^{i})$. Am propus un algoritm simplificat dar eficient care pleacă de la ipoteza că pentru aceeaşi lungime a *n*-gramelor coeficienții de ponderare $\alpha(l_{i-n+1}^{i-1})$ pot fi aceeaşi. Alegerea lor se face pe baza corpusului de tip de validare *T*2 prin alegerea acelor coeficienți $\alpha_{held-out}(n)$ care minimizează eroarea de predicție a accentului lexical. Se va observa totuşi că, deşi algoritmul nostru nu garantează o sumă a distribuțiilor de probabilitate egală cu 1, acest lucru nu afectează performanța globală a metodei propuse.

Formula de calcul a probabilității finale este:

$$P_{katz}(l_{i-n+1}^{i}) = \begin{cases} P_{katz}^{*}(l_{i-n+1}^{i}) & dacă \quad c(l_{i-n+1}^{i}) > 0 \\ \alpha_{held-out}(n) \cdot p_{katz}(l_{i-n+2}^{i}) & dacă \quad c(l_{i-n+1}^{i}) = 0 \end{cases} \tag{5.3}$$

Într-un asemenea model recursiv probabilitățile de ordin *n* sunt calculate pe baza probabilităților de ordin *n*-1. Algoritmul se opreşte la nivelul unigramelor prin alegerea distribuției de tip MLE, astfel încât în această etapă probabilitățile bigramelor sunt calculate direct din probabilitățile unigramelor de tip MLE.

$$P_{katz}(l_{i-1}^{i}) = \begin{cases} P_{katz}^{*}(l_{i-1}^{i}) & dacă \quad c(l_{i-1}^{i}) > 0 \\ \alpha_{held-out}(n) \cdot p_{MLE}(l_i) & dacă \quad c(l_{i-1}^{i}) = 0 \end{cases} \tag{5.4}$$

Pentru a putea realiza o recursie completă trebuie numărate toate aparițiile *n*-gramelor, până la nivelul unigramelor. Frecvențele de apariție sunt salvate în fişiere separate şi apoi re-estimate prin algoritmul Good – Turing.

Pentru a însuma explicațiile anterioare, algoritmul de antrenare are următoarele etape de bază:

1. *Din corpusul de antrenare T1 (compus din cuvinte cu accentele poziționate corect şi cu delimitatorii de start şi stop pentru fiecare cuvânt), sunt extrase toate n-gramele, la nivel de caracter. Pentru o istorie de lungime n, sunt extrase de asemenea n-gramele de ordin n-1, n-2,...,1.*
2. *Se numără toate aparițiile n-gramelor de ordin n, n-1,...,1 se calculează şi se salvează probabilitățile de tip ML (sau numărul de aparții) rezultate.*
3. *Pe baza corpusului held-out T2 se estimează coeficienții de ponderare $\alpha_{held-out}(n)$ care minimizează eroarea de predicție a accentului lexical în T2 (care în această etapă joacă rolul de facto al corpusului de test).*
4. *Folosind coeficienții de ponderare estimați în etapa 3, se recalculează probabilitățile potrivit metodei Katz backoff. Se renunță la datele de antrenare anterioare.*

5.3.3 Algoritmul de testare

În etapa de test fiecare cuvânt din textul de intrare este comparat cu toate intrările dintr-un dicționar de forme omografice; dacă un cuvânt este găsit în dicționar, acesta va fi lăsat nemodificat şi trimis către un alt modul al sistemului TTS pentru o procesare ulterioară. În cazurile rare în care un cuvânt intră în algoritmul de poziționare a accentelor, rezultatul poate să nu fie o formă corectă semantic.

Dacă un cuvânt are cel puțin două silabe atunci acesta este procesat după următorul algoritm:

i. *Se pleacă de la forma* $\mathbf{w} = l_1 l_2 ... l_N$, *și se obține forma* $\underline{\mathbf{w}}$ *prin plasarea delimitatorilor de start și stop,"<" respectiv ">":* $\underline{\mathbf{w}} = l_0 l_1 l_2 ... l_i ... l_N l_{N+1}$, *în care* $l_0 = <$ *și* $l_{N+1} = >$.

ii. *Pentru fiecare formă* $\underline{\mathbf{w}}$, *se generează toate variantele posibile accentuate* $\mathbf{g}_j$, $j = 1,...,M$ *(în care M este numărul de vocale din* $\underline{\mathbf{w}}$*), prin inserarea unui caracter de accent înaintea fiecărei vocale.*

iii. *Din fiecare variantă* $\mathbf{g}_j$ *extrage n-gramele de ordin n, n-1,…,1 și calculează probabilitățile finale după formula generală din algoritmul de netezire de tip Katz cu revenire, în care probabilitățile condiționate sunt cele obținute din datele salvate în timpul etapei de antrenare.*

iv. *Varianta* $\mathbf{g}_{j,\max}$ *care are probabilitatea maximă este furnizată la ieșirea sistemului.*

5.4 Experimente şi rezultate

Datele inițiale folosite în experimentele noastre de evaluare au fost extrase din texte literare scrise în limba română, pe baza corpusului descris în capitolul anterior, totalizând peste 10 milioane de cuvinte, iar dicționarul *T* de start conține peste 330.000 de cuvinte diferite (cuprinzând multe forme flexionate ale cuvintelor de bază). Acest dicționar a fost procesat mai departe prin plasarea manuală a semnelor corespunzătoare accentelor lexicale înaintea vocalelor accentuate. După această etapă au fost eliminate cuvintele omografe prin eliminarea automată a cuvintelor care au două sau mai multe forme accentuate, din aceste cuvinte rezultând un dicționar al omografelor. În plus, cuvintele monosilabice nu au fost introduse în algoritm, după cum am explicat anterior. În final, *T* a fost împărțit în trei părți egale: *T*1, *T*2 şi *T*3, potrivit tabelului 5.1; *T*1 a fost folosit pentru antrenarea modelelor *n*-gramelor, *T*2 a fost folosit ca dicționar de validare (*held-out*) pentru estimarea coeficienților de ponderare în algoritmul de tip Katz implementat iar *T*3 a fost folosit pentru testare. Trebuie să subliniem încă o dată faptul că cele trei dicționare amintite sunt complet diferite, altfel spus, intersecția mulțimilor de cuvinte reprezentate de dicționarele *T*1,*T*2 şi *T*3 este mulțimea vidă.

Evaluarea a fost făcută în mod automat, prin compararea rezultatelor obținute în diferite condiții de testare, pe corpusul *T*3 cu şi fără *markerii* de accente la intrare.

Corpus	**Denumire**	**Mărime din T(%)**	**Mărime în cuvinte**
Antrenare	*T1*	75	234.700
Validare	*T2*	10	31.300
Testare	*T3*	15	47.000

Tabel 5.1 Corpusurile extrase din dicționarul *T*

Algoritmul a fost testat mai întâi cu diferite lungimi ale *n*-gramelor extrase la nivel de caracter, până la lungimea maximă de 8 caractere pentru care am obținut o rată de eroare minimă la nivel de cuvânt de 0,89% pentru sarcina de predicție a accentului lexical. Peste această lungime, potrivit datelor experimentale, nu am obținut o creştere de performanță.

5.5 Concluzii

Am prezentat un algoritm eficient (comparativ cu cele existente) şi precis de predicție a poziției accentelor lexicale, pentru un modul TTS în limba română. Am propus o abordare statistică, pe baza modelului *n*-gramelor la nivel de caracter folosind totodată un algoritm de netezire a probabilităților de tip Katz cu revenire pentru diminuarea efectelor nedorite cauzate de insuficiența datelor de antrenare. Pentru un corpus de test compus din aproximativ 47.000 de cuvinte am obținut o rată de eroare minimă la nivel de cuvânt de 0,89%. Această mărime de performanță poate fi clasificată ca fiind foarte bună din punct de vedere al complexității sarcinii în discuție pentru limba română dar şi comparată cu cele mai bune

rezultate în domeniu (după cum am precizat în Subcapitolul 5.2). Dacă o comparație reală de performanță nu poate fi făcută raportată la alte contribuții similare din alte limbi vorbite din cauza particularităților acestora, rezultatele noastre le depăşesc pe cele raportate în [OAN02] pentru limba română(o acuratețe de aproximativ 94% la nivel de cuvânt pe o bază de test de 4500 de cuvinte – acestea fiind de altfel, din ceea ce cunoaştem, singurele rezultate notabile în domeniu).

6. Modulul de despărțire în silabe

6.1 Introducere

În capitolul de față am abordat o altă problematică importantă în vederea obținerii unui sistem TTS de calitate: cea a despărțirii automate în silabe.

6.2 Particularități ale limbii române şi dificultăți ale sarcinii de despărțire automată în silabe

Sunetele sunt cele mai mici unități ale limbii şi apar, în orice limbă, într-un număr limitat fiind clasificate în **vocale** şi **consoane**. Unele vocale devin, în anumite contexte, **semivocale**. O vocală sau o vocală împreună cu una sau mai multe consoane sau o vocală şi cel mult două semivocale constituie, în anumite condiții, o unitate fonetică superioară: **silaba**.

În limba română, orice silabă conține o vocală şi numai una. Silaba poate să conțină semivocale pe lângă vocala principală. O vocală şi o semivocală aflate în aceeaşi silabă reprezintă un **diftong**. O vocală şi două semivocale aflate în aceeaşi silabă constituie un **triftong**. Vocalele consecutive ale unui cuvânt nu pot fi situate decât în silabe diferite şi sunt vocale în hiat. Silabele pot fi accentuate sau neaccentuate iar accentul are uneori rol semantic, distinctiv, şi poartă numele de accent lexical, după cum am menționat în capitolul 5.

6.3 Descrierea algoritmului de despărțire în silabe

Algoritmul implementat este de tip hibrid. Astfel, există două seturi de reguli: inițial şi final, între care se aplică un algoritm statistic, bazat pe *n*-grame.

6.3.1 Setul inițial de reguli

Setul inițial de reguli a fost extras din literatura de specialitate [DEX98] şi cuprinde un număr de situații clare referitoare la despărțirea cuvintelor în silabe.

6.3.2 Algoritm statistic. Etapa de antrenare

1. *Dintr-un corpus de antrenare T_1 (compus din cuvinte despărțite corect în silabe şi având delimitatorii de start şi stop pentru fiecare cuvânt), sunt extrase toate n-gramele, la nivel de caracter. Pentru o istorie de lungime n, sunt extrase de asemenea n-gramele de ordin n-1, n-2,...,1.*
2. *Se numără toate aparițiile n-gramelor de ordin n, n-1,...,1 pentru n de la 1 la n*
3. *Se calculează şi se salvează probabilitățile de tip MLE rezultate*
4. *Se netezesc probabilitățile după metoda Good-Touring, pentru n mai mare strict decât*

6.3.3 Algoritm statistic. Etapa de testare

1. *Se pleacă de la forma* $\underline{\mathbf{w}}$ ***a unui cuvânt***

   $\underline{\mathbf{w}} = l_1 l_2 ... l_i ... l_N$, în care $l_1 = <$ şi $l_N = >$ iar l_i **sunt litere**

2. *Pentru fiecare formă* $\underline{\mathbf{w}}$*, se generează toate variantele posibile de despărțire în silabe* $\mathbf{g}_j$, $j = 1,...,2^L$*, prin inserarea caracterului despărțitor înaintea fiecărui caracter din* $\underline{\mathbf{w}}$ *iar L este lungimea cuvântului.*
3. *Din fiecare variantă* $\mathbf{g}_j$ *extrage n-gramele de ordin n, n-1,…,1.*

4. *Se calculează emisiile de probabilitate finale după formula generală din algoritmul Katz cu revenire, fără aplicarea unui algoritm de netezire, în care probabilitățile condiționate sunt cele obținute din datele salvate în timpul etapei de antrenare, iar pragul k de decizie pentru Katz este egal cu 1 pentru lungimi mai mici decat trei.*
5. *Varianta* $\mathbf{g}_{j,\max}$ *care are probabilitatea maximă este furnizată la ieşirea sistemului.*

6.3.4 Set final de reguli

R.1. Consoanele singure nu pot forma silabe.
R.2. Grupul ***na-şi*** *la final de cuvânt trece în* ***naşi***
R.3. Grupul ***na-ți*** *la final de cuvânt trece în* ***nați***
R.4. Grupul ***na-i*** *la final de cuvânt trece în* ***nai***
R.5. Grupul ***re-au*** *la final de cuvânt trece în* ***reau***
R.6. Grupul ***s-e*** *la final de cuvânt trece în* ***se***

6.4 Rezultate şi concluzii

Ca bază de start, pentru efectuarea testelor, am folosit un dicționar *T* de 56.284 cuvinte corect despărțite în silabe. Bazele de antrenare/testare au fost obținute prin extragerea aleatoare a cuvintelor din baza de start, menținând în permanență raportul 80%/20% între dimensiunile de antrenare/testare. Pentru o estimare cât mai corectă a performanțelor metodei descrise, au fost efectuate un număr de trei iterații de tip antrenare – testare prin extragerea a diferite baze de antrenare – testare, pe mulțimi disjuncte de cuvinte (niciunul dintre cuvintele din corpusul de antrenare nu se află în corpusul de test), menținând acelaşi raport între dimensiunile corpusurilor.

Am rulat algoritmul pentru *n*-grame cu 2, 3, 4, 5, 6 caractere. Rezultatele au fost sintetizate în tabelul 6.1:

Lungime *n*-grame	**WER (%)**
2	28.62
3	15.54
4	6.30
5	**2.86**
6	3.92

Tabel 6.1 Rezultate obținute în funcție de lungimea *n*-gramelor

Practic, se observă o îmbunătățire bruscă a rezultatelor până la o istorie de mărime 5, după care apare o scădere semnificativă din cauza lipsei datelor de antrenare coroborată cu specificul algoritmului nostru.Cele mai bune rezultate pentru toate cele trei teste amintite sunt menționate în tabelul 6.2 şi au fost obținute pentru aceeaşi lungime maximă a n-gramelor şi anume 5.

	Cuvinte test	**Număr de erori**	**WER (%)**
Test 1	11,256	347	3.08
Test 2	11,257	331	2.94
Test 3	11,257	322	2.86

Tabel 6.2 Rezultate obținute pentru diferite corpusuri de test

După cum se poate observa, cele trei valori rezultate sunt apropiate, ceea ce ne face să credem că un eventual fenomen de tip *overtraining* a fost evitat. Performanța finală, măsurată în WER a fost calculată ca medie a celor trei rezultate parțiale şi este de **2.96%**.

6.5 Exemplu de interfață NLP generală

Cu algoritmii descrişi până în acest moment a fost implementată, în scopuri de cercetare-dezvoltare, o interfață completă care să demonstreze funcționalitatea şi performanțele algoritmilor propuşi. A fost realizată o implementare modulară a diverselor etaje de prelucrare a limbajului natural, iar în final rezultatele diverselor niveluri de prelucrare concură pentru realizarea unui şir NLP unic (ulterior am inclus şi etapa de conversie fonetică, după cum vom vedea în capitolul următor). Aceeaşi abordare modulară a fost folosită şi pentru a separa diversele module componente, ceea ce permite o înlocuire ulterioară facilă a acestora. Implementarea de față este rezultatul unor dezvoltări succesive şi poate fi oricând modificată pentru a satisface diverse cerințe. Astfel, se poate renunța la algoritmul de refacere a diacriticelor dacă cerințele de memorie sunt prea restrictive sau dacă beneficiarul nu solicită acest lucru.

7. Modulul de conversie fonetică

7.1 Introducere

Fonetica este ştiința care studiază sunetele vorbirii. Din punct de vedere al mecanismului actului comunicării verbale, fonetica are trei ramuri: articulatorie, acustică şi neuroperceptivă.

Din punct de vedere fonologic spunem că două sau mai multe unități de expresie nonechivalente şi utilizate ca unități minimale în vorbire se numesc **foneme.** Două sau mai multe unități de expresie echivalente şi utilizate ca unități minimale în vorbire se numesc **alofonele** unui fonem. Relația de nonechivalență poate fi explicată prin intermediul noțiunii de comutare. Astfel, dacă unei corelații dintre cel puțin două unități de expresie îi corespunde o corelație între două unități de conținut, atunci unitățile de expresie sunt în raport de comutare.

7.2 Problemele conversiei fonetice pentru limba română

Una dintre cele mai dificile situații pentru transcrierea fonetică în limba română apare în cazul conversiei **vocalelor** (şi al **grupurile vocalice**) care introduc ambiguități de pronunție. Variațiile mari de pronunție ale vocalelor în funcție de context fac de asemenea extrem de dificilă segmentarea unităților acustice ce le includ şi utilizarea acestor unități în etapa generare a semnalului de vorbire obținută prin concatenarea de unități acustice.

O altă problemă, care se suprapune în mare măsură cu aceea a ambiguităților de pronunție, este legată de alinierea literelor cu grafemele. Un grafem poate fi format din una sau mai multe litere (de exemplu în limba română există grafemele **ch** şi **gh**). Pentru o conversie automată pornind de la text este necesară identificarea grafemelor înaintea realizării conversiei propriu-zise [BUR99b].

Alfabetul fonetic folosit la ora actuală în cadrul colectivului nostru conține 34 de simboluri pentru a identifica tot atâtea alofone. Dintre acestea, 29 formează setul de foneme de bază, iar 5 sunt alofonele vocalelor **e, i, o, u**: există deci patru semivocale la care se mai adaugă un **i-final surd**, neaccentuat, ca în cuvintele **poți**, **comori**.

Trebuie menționat în acest punct faptul că pentru a face legătura între diversele module ale sistemului şi pentru a putea refolosi eventual alte module existente, am decis să apelăm la un limbaj de tip *mark-up*, şi anume SSML (*Speech Synthesis Markup Language*), după cum vom descrie în capitolul următor. Deoarece SSML stabileşte ca necesitate existența codificării IPA, pentru asigurarea conceptelor de interoperabilitate şi de internaționalizare, modulul de transcriere fonetică pe care l-am realizat poate realiza conversia imediată către codificarea IPA (*International Phonetic Alphabet*).

7.3 Descrierea algoritmului de conversie fonetică

Algoritmul de conversie fonetică pe care l-am propus şi implementat, foloseşte în principal modulul de despărțire în silabe descris în capitolul 6 dar şi pe cel de poziționare a accentelor lexicale descris în capitolul 5. În acest fel se rezolva mult mai uşor, într-o manieră segmentală, problemele legate de

identificarea hiaturilor după care diftongii şi triftongii sunt mult mai uşor de evidențiat, ca mulțime disjunctă în grupurile de caractere vocalice. După acest prim nivel, în urma unei analize proprii făcute, am ajuns la concluzia că se pot identifica exact, pe bază de reguli, tipurile vocalice particulare (diftongii ascendenți sau descendenți, tipurile de triftongi) şi se poate deci realiza conversia fonetică.

După cum precizam anterior, prin alăturarea, în cadrul unui cuvânt, a două sau trei grafeme din grupa (**a, e, i, o, u, ă, î** (sau **â**)) pot rezulta diftongi, triftongi sau hiaturi. Deşi aceste structuri sunt greu de evidențiat într-o manieră automată se observă totuşi că pentru limba română există o structură destul de regulată (în limita a câtorva excepții) a modului în care aceste grafeme se pot asocia. Astfel, în tabelele 7.2 şi 7.3 am exemplificat cazurile de apariție pentru diftongi sau triftongi precum şi eventualele ambiguități care pot să apară, luând în considerare şi hiatul.

	a	**e**	**i**	**o**	**u**	**ă**	**î**
a	H	H	H ↓	H	H ↓		
e	H ↑	H	H ↓	H ↑	H ↓		H
i	H ↑	H ↑	H ↓	H ↑	H ↑ ↓		H
o	H ↑	H	H ↓	H	H ↓		H
u	H ↑	H	H ↓	H	H	H ↑	↑
ă			H ↓	H	H ↓		
î			H ↓		H ↓		

Tabel 7.1 Cazuri de perechi vocalice ambigue, cu evidențierea diftongilor şi a hiaturilor

În cazul ambiguităților dintre diftongi şi hiat, reprezentate sugestiv în tabelul 7.1, pot să apară următoarele situații de alăturări, înainte de despărțirea în silabe:

1. nici hiaturi nici diftongi – fără simbol
2. numai hiaturi – simbolizat cu H
3. hiaturi şi diftongi ascendenți simbolizat cu H ↑
4. hiaturi şi diftongi descendenți simbolizat cu H ↓
5. hiaturi şi diftongi ascendenți sau descendenți simbolizat cu H ↑ ↓

O analiză similară a posibilelor alăturări vocalice a fost realizată şi pentru triftongi.

În vederea implementării considerentelor discutate anterior, am decis să pun în evidență hiaturile, prin folosirea modulului de despărțire în silabe descris în capitolul anterior. Astfel, cel puțin teoretic, alăturările vocalice care formează hiat vor fi îndepărtate, deoarece fac parte din silabe diferite, iar în interiorul silabelor evidențiate vor rămâne numai cazurile de diftongi sau triftongi. Principiul de la care am plecat este că într-o silabă putem avea o singură vocală. Celelalte grafeme vocalice alăturate, din interiorul unei silabe, marchează semivocale şi formează cu vocala diftongi sau triftongi. Odată identificată prezența unei asemenea alăturări, care nu face parte dintr-un triftong, problema conversiei grafem-fonem este rezolvabilă determinist, cu o singură excepție deja amintită: cazul **iu**, care poate fi atât diftong ascendent cât şi descendent, şi pentru care am decis o abordare separată, cu ajutorul modulului de predicție a accentului lexical, ca în figura 7.1.

Algoritmul este următorul :

1. Pentru fiecare cuvânt de test parcurge următorii paşi:

- *a. Obține forma despărțită în silabe a cuvântului inițial prin trecerea cuvântului prin modulul de despărțire în silabe.*
- *b. Poziționează accentul lexical la nivelul vocalei accentuate cu ajutorul modulului de poziționare a accentului lexical. Folosind forma despărțită în silabe a cuvântului inițial obține forma despărțită în silabe şi cu accentul lexical poziționat la nivelul silabei accentuate care conține vocala accentuată identificată anterior.*
- *c. Transformă **y** în **i**, **ke** în **che**, **ki** în **chi**, **k** în **c**, **x** în **cs** şi **w** în **v**.*

2. *Pentru fiecare silabă din cuvânt identifică tipul de diftongi sau triftongi interni ai silabei. Realizează conversia fonetică pe baza informațiilor referitoare la vocale / semivocale şi a setului de 45 de reguli deduse din analiza diftongilor şi a triftongilor.*

3. *Se înlocuiesc grafemele **ce/ci**, **che/chi, ge/gi,** şi **ghe/ghi** cu alofonele corespunzătoare pe baza informațiilor obținute anterior.*

4. *Pentru silabele în care apare perechea vocalică ambiguă **iu**:*
   a. *Dacă accentul lexical este poziționat pe litera **i**, **i** devine vocală şi **u** se transformă în semivocală.*
   b. *Dacă accentul lexical lipseşte sau este poziționat pe litera **u**, **u** devine vocală iar **i** devine semivocală.*

5. *Pentru silabele finale care se finalizează în litera **i** precedată de o consoană:*
   a. *Dacă accentul lexical este poziționat pe litera **i**, **i** devine o vocală.*
   b. *Dacă **i** nu poartă accent lexical, **i** este înlocuit de simbolul alofonului **i-final surd**.*

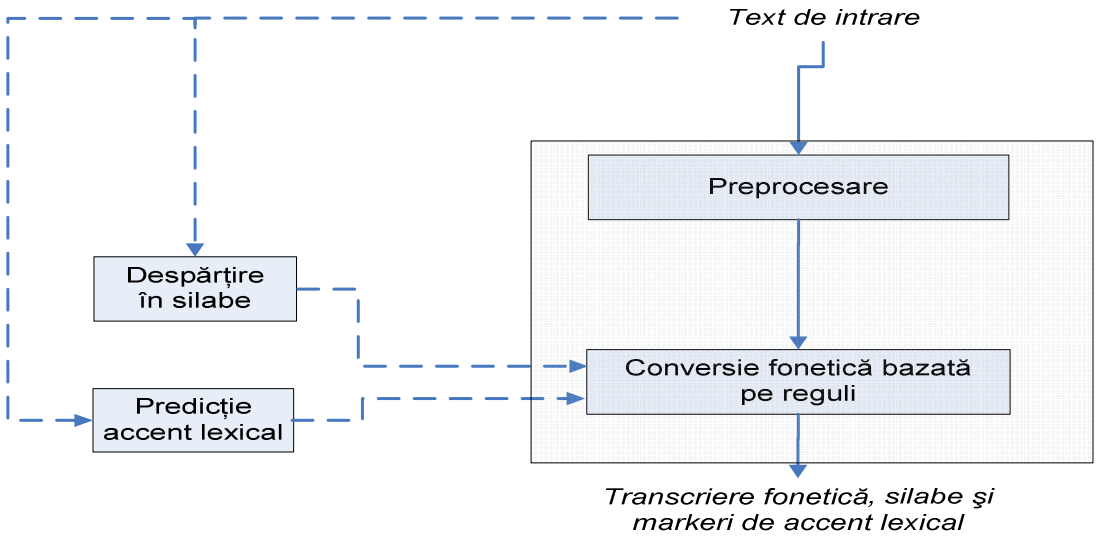

Figura 7.1 Implementarea algoritmului de conversie fonetică

7.4 Rezultate şi concluzii

În vederea testării algoritmului amintit anterior, am avut la dispoziție un corpus de test construit manual alcătuit din 11.819 de cuvinte, pentru care a fost realizată conversia fonetică. Rezultatul obținut pe acest corpus de test este de 3,01% WER.

8. Modulul de preprocesare şi normalizare

8.1 Descrierea problemei. Definiții

Un text furnizat la intrarea sistemului TTS poate să conțină, pe lângă reprezentările grafemice regulate şi o serie de secvențe care sunt de obicei anormale din punct de vedere lingvistic față de majoritatea cuvintelor din text şi care, din punct de vedere al implementării în sistemul de sinteză trebuie mai întâi transformate într-un format ce poate fi recunoscut de partea de prelucrare de limbaj. Pentru ca informația extrasă din textul de intrare să fie cât mai relevantă pentru următoarele niveluri de prelucrare, trebuie să avem, pe lângă o etapă de extragere a cuvintelor şi a semnelor de punctuație (de **segmentare** a textului), de un proces de uniformizare, de aducere a tuturor cuvintelor la forma **ortografică**. Acestea sunt cele două roluri ale etajului de preprocesare/normalizare. Deşi poate părea o problemă simplă, preprocesarea/normalizarea unui text ridică o multitudine de dificultăți. Astfel, nu este întotdeauna posibilă determinarea marginilor unei fraze pe baza semnelor de punctuație. Punctul (**.**) poate să apară

atât la sfârşitul frazei cât şi în abrevieri, acronime, numerale sau indicația că se omite un anumit fragment de text (**...**). O sarcină dificilă este şi conversia anumitor secvențe de simboluri în cuvinte care să poată fi analizate lingvistic. Dacă unele abrevieri uzuale pot fi "expandate" imediat, cu ajutorul unui tabel de echivalențe, există multe situații în care secvențe de simboluri care nu se pot distinge pe baza ortografierii lor cer tipuri diferite de conversii; de exemplu, numărul format din şapte cifre **1234567** poate reprezenta un număr întreg sau un număr de telefon şi va trebui citit diferit în cele două situații. În general, prezența şirurilor de numere în text ridică numeroase dificultăți, deoarece ele pot să apară în diferite contexte: ore, date, numere de telefon, expresii aritmetice etc. Trebuie observat că aceste ambiguități create de natura multifuncțională a semnelor de punctuație sau de modul diferit de citire a aceloraşi secvențe de simboluri, pot avea implicații majore asupra acurateței întregului proces de prelucrare lingvistică şi în final asupra pronunției corecte a textului de către sistemul de sinteză [BUR99b].

Din perspectiva implementării sistemului de sinteză a vorbirii pornind de la text am definit următoarele elemente de prelucrare aplicate asupra textului furnizat la intrarea sistemului:

- **Etapa de normalizare** este activitatea de a transformare a literelor şi semnelor ortografice într-o formă comună, care să poată fi ulterior prelucrată de către modulele NLP. Prin formă comună înțelegem atât forma ortografică a unor cuvinte cât şi, de exemplu, forma rostită a unor semne ortografice (sunt situații în care sistemul de sinteză trebuie să rostească astfel de semne).
- Etapa **de preprocesare** este activitatea de segmentare a unui text de intrare, de împărțire a lui în fraze şi propoziții şi de extragere a cuvintelor.

8.2 Descrierea algoritmului de preprocesare şi normalizare

Algoritmul pe care l-am implementat cuprinde următoarele etape:

1. Introducerea textului ce urmează să fie transformat în vorbire.

2. Separarea textului în segmente de cuvinte sau caractere alfanumerice despărțite prin spații albe. Semnele de punctuație se includ temporar în segmente, fiind apoi folosite pentru introducerea markerilor prozodici.

*4. Eliminarea semnelor de punctuație care nu vor fi folosite la generarea prozodiei. Acestea sunt - **"** şi **'** .*

*5. Detecția numerelor de telefon, fixe şi mobile, se face atât după un pattern de zece cifre, cât şi după existența unor cuvinte cheie în imediata vecinătate a numărului. Astfel, de exemplu dacă într-o fereastră de cinci cuvinte în stânga numărului apar o serie de cuvinte cheie: **sun**, **telef**, **apel**, **form** sistemul decide că este vorba despre un număr de telefon şi realizează o conversie prin concatenarea cifrelor expandate, altfel acesta este interpretat ca un numeral.*

*6. Expandarea numeralelor cardinale si ordinale. Se folosesc reguli simple, unui număr fiindu-i asociat un şir de litere (**99 - nouăzeci şi nouă**, **al 3-lea** devine **al treilea**). Implementarea a fost făcută până la 999 de miliarde, într-o manieră recursivă, cu detecția ordinului de mărime (zeci, sute, mii ş.a.m.d). În plus, sunt abordate şi numerele fracționare.*

7. *Expandarea abrevierilor şi acronimelor; segmentarea textului în fraze şi grupuri prozodice şi adăugarea markerilor prozodici.*

3. Conversia majusculelor în minuscule.

8.3 Integrarea SSML în sistemul TTS pentru limba română

8.3.1 Necesitatea existenței unui standard de reprezentare internă pentru sinteza vorbirii

Aplicațiile de sinteza vorbirii pot fi implementate după diverse arhitecturi. Astfel, după cum am descris în capitolul 2, pot exista de exemplu sisteme de tip *embedded* cât şi sisteme cu o arhitectură distribuită – având ca suport o infrastructură de tip rețea (IP sau telefonie).

Este cunoscut faptul că încă din primele sisteme TTS implementate, dezvoltatorii de aplicații au folosit un limbaj propriu de comunicare inter-module şi de interfațare cu utilizatorul final. Acest lucru face, pe de o parte ca modulele componente să nu poată fi folosite în alte sisteme de acelaşi tip iar pe de

altă parte apare necesitatea ca interfețele finale să fie re-implementate în conformitate cu fiecare pachet API pus la dispoziție.

8.3.2 Locul SSML într-un sistem TTS

Pentru a veni în întâmpinarea problemelor menționate anterior, consorțiul W3C a elaborat, prin *Voice Browser Working Group*, recomandările SSML (*Speech Synthesis Markup Language*) ajunse la versiunea 1.1 (septembrie 2010). În mod ideal SSML este independent de orice sistem TTS particular, care trebuie să transforme un fişier adnotat într-o reprezentare internă specifică unui anumit tip de sinteză. Elementele SSML capătă relevanță din perspectiva stivei unui sistem TTS şi aduc claritate pentru un sistem de sinteză prin faptul că transmit către acesta date şi comenzi care ajută procesul în sine. Astfel, de exemplu, la nivelul etajului de preprocesare / normalizare este realizată segmentarea textului în paragrafe, fraze, cuvinte, semne de punctuație, sunt expandate abrevierile etc. Aceste elemente obținute de către preprocesor sunt imbricate cu ajutorul elementelor de tip *markup* (de adnotare) SSML pentru a furniza etajelor următoare cât mai multe informații utile pentru ca vorbirea rezultată să fie mai naturală. De asemenea, propozițiilor li se pot adăuga elemente SSML de control prozodic iar la nivelul cuvintelor şirurile de conversie fonetică pot fi obținute fie dintr-o referință de tip lexicon sau printr-o conversie fonetică, ambele marcate prin elemente de adnotare.

8.3.3 Istoricul apariției SSML

Au existat mai multe încercări, de tip proprietar, de a controla un sistem TTS folosind un limbaj de marcare: astfel enumerăm: prima versiune de SSML (ediție elaborată la Universitatea Edinburg), STML, JSML, SABLE , SAPI, SSML (W3C). Deşi există diferențe sintactice intre aceste limbaje, ele funcționează într-o manieră similară şi sunt independente de tipul de sinteza vorbirii. Un sistem TTS trebuie să traducă notațiile de tip ML în reprezentări interne specifice, în vederea controlului procesului de sinteză.

8.3.4 Caracteristicile SSML

SSML oferă controlul sintezei vorbirii la nivel de cuvânt, fonem dar şi la nivel de formă de undă pentru a satisface o gamă cât mai largă de aplicații.

Etapele pe care le implică transformarea unui text pentru a fi adus la o formă utilă unui proces de sinteză folosite de un proces de sinteză pentru sunt următoarele:

- **Crearea documentului**, într-o manieră automată sau manuală
- **Procesarea documentului**, implică şase niveluri de prelucrare
- **Analiza XML** pentru extragerea structurii documentului şi a conținutului acestuia.
- **Analiza structurii textului** (în esență extragerea paragrafelor şi a propozițiilor).
- **Normalizarea textului** – aici se fac înlocuirile datelor, numerelor, abrevierilor etc cu forma lor ortografică.
- **Conversia text – foneme**. Folosirea unui lexicon poate fi foarte utilă atunci când există excepții. Un lexicon trebuie să respecte specificațiile PLS – *Pronunciation Lexicon Specification.*
- **Analiza prozodică** permite controlul de pitch, al pauzelor, al vitezei vorbirii, evidențierea cuvintelor etc.
- **Producerea formei de undă** se realizează pe baza informațiilor fonemice şi prozodice.

Un sistem de tip TTS trebuie să poată analiza şi să traducă elementele de tip *mark-up* într-o reprezentare internă în vederea controlului procesului de sinteză. Pentru aceasta pot fi utilizate diverse *lexicoane* (dicționare) de conversie fonetică, orientate pe un anumit domeniu sau pe o anumită limbă. Conform specificațiilor SSML se impune recunoaşterea alfabetului fonetic IPA dar se poate apela şi la alte conversii, de tip proprietar. În implementarea pe care am realizat-o am folosit un astfel de alfabet pentru limba română, cu mențiunea că se poate face conversia imediată în codificarea IPA.

8.3.5 Implementarea unei aplicații de sinteza vorbirii, prin folosirea etajelor de prelucrare de limbaj şi codificare SSML

Aplicația pe care am dezvoltat-o este de tip embedded, este implementată pe un terminal mobil, conține toate modulele de prelucrare a limbajului natural descrise anterior şi este destinată implementării într-un sistem de sinteză de tip e-mail sau file reader. În figura 8.1 am reprezentat modul în care elementele SSML sunt trecute prin stiva NLP, până la nivelul etajului de selecție a unităților vocalice. Dezvoltările specificațiilor SSML ulterioare sau alte implementări de tip proprietar ar putea continua dezvoltarea şirului XML dincolo de acest nivel.

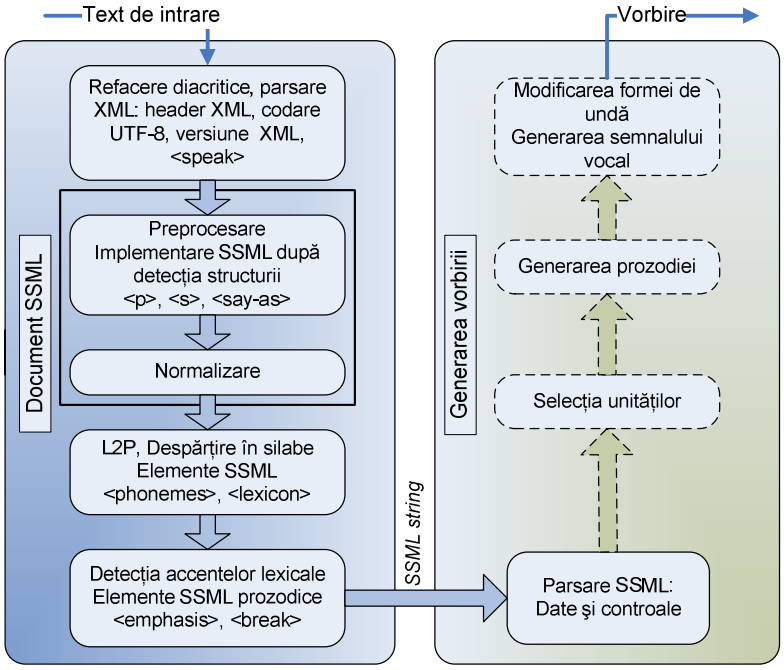

Figura 8.1 Implementarea stivei SSML sistemul TTS [după UNG11b]

8.4 Concluzii

Ne-am propus implementarea prin intermediul modulelor de NLP a unui etaj complet de procesare lingvistică care să respecte specificațiile SSML. În acest fel putem ataşa rezultatele prelucrării naturale de limbaj la ieşire orice fel de etaj de generare a vorbirii care respectă specificațiile menționate, recomandate de W3C, care tind să devină un standard pentru sinteza vorbirii. Totodată, tipul de tratare modulară la care am optat permite o depanare şi o dezvoltare ulterioară facilă a întregii aplicații.

9. O implementare a etajului de prelucrare a limbajului natural pentru un sistem TTS pe dispozitive portabile

Rolul pe care autorul acestei lucrări l-a avut în cadrul colectivului de cercetare a fost acela de a realiza într-o manieră nouă, diferită de cea existentă deja, întregul etaj de prelucrare a limbajului natural pentru un sistem TTS în limba română şi de a realiza o implementare pe un dispozitiv portabil care să pună în valoare stiva NLP obținută, în vederea realizării unor interfețe de comunicare prin voce cu un astfel de dispozitiv. Implementarea a fost realizată în Perl.

9.1. Caracteristicile clasei *embedded*

Introducerea tehnologiei vorbirii în dispozitivele portabile actuale este văzută ca următorul pas către pervasive computing, stare în care procesarea datelor şi comunicarea vor exista pretutindeni între elementele de mediu [BUR07][POS06]. Spre deosebire de clasa PC-urilor, caracterizată oarecum de uniformitate în materie de procesoare şi memorie (arhitecturile Intel pe 32 / 64 de biți sunt cele mai

răspândite), clasa embedded este caracterizată de neuniformitate / diversitate şi de folosirea unor arhitecturi hard & soft de tip proprietar.

Principalele limitări ale acestor sisteme sunt de regulă traduse prin putere de calcul scăzută şi memorie scăzută, cele două cerințe fiind de regulă într-o relație complementară [HUA07][BUR04c][IBM08B][NÉM08]. Este evident că din cauza rulării pe astfel de configurații hardware, arhitectura interfeței trebuie să fie simplificată. O altă limitare este adusă de alimentarea pe acumulatori.

9.2. Aspecte generale legate de implementare

Prima etapă urmată în realizarea interfeței a fost aceea de a implementa, cu performanțe cât mai bune, întreaga stivă NLP descrisă la modul general în capitolul 2, pe o arhitectură generică de tip PC şi de a estima cerințele de memorie şi putere de calcul pentru fiecare dintre modulele implementate.

Pentru realizarea etajului NLP pe dispozitive de tip *embedded*, o a doua etapă a fost cea de alegere şi de pregătire a mediului de lucru pentru versiunea *embedded* pentru care trebuie asigurată şi o altă cerință legată de necesitatea prezenței interpretorului Perl. În cazul în care acest lucru nu se poate asigura este necesară convertirea aplicațiilor într-un alt format, de regulă în fişiere executabile. Trebuie menționat faptul că Perl este inclus implicit în majoritatea sistemelor de operare de tip Linux, prin urmare pe astfel de maşini rularea aplicației nu este o problemă. Deoarece astfel de sisteme de operare au fost portate pe o mare varietate de arhitecturi de tip *embedded* (*ARM, AVR32, EXTRAS CRIS, m68K, MIPS, SuperH, Xtensa* etc.),avem garanția că aplicația va putea rula fără probleme pe acestea şi în viitor, tendința actuală fiind aceea de a se renunța la sistemele de operare (SO) de tip proprietar în favoarea de tip *open*. Pentru SO de tip Windows (*XP, WIN7*) am reuşit conversia aplicațiilor din Perl în fişiere de tip *exe*, folosind aplicația perl2exe [IND]. Pentru sistemele de tip Windows Mobile suportul adus pentru Perl s-a oprit la nivelul anului 2003, de aceea am optat pentru terminalele mobile cu un SO de tip *open* şi anume Android. Legat de arhitectura hardware aleasă, din cauza cerințelor destul de ridicate în special pe partea de memorie RAM s-a optat pentru un telefon mobil de ultimă generație, model *Nexus S* produs de firma Samsung, care are următoarele caracteristici: procesor RISC Hummingbird pe 32 de biți model S5PC110 la 1GHz, 512 MB RAM de tip Mobile DDR, memorie flash de 16GB de tip *iNAND*, ecran super LCD de 100 de mm şi sistem de operare Android 2.3. (Gingerbread).

Cea de-a treia etapă, cea de implementare propriu-zisă va fi descrisă în subcapitolul următor.

9.3. Interfață de prelucrare a limbajului natural implementată pe un dispozitiv portabil

Sistemul de operare Android (vom folosi de aici încolo această denumire generică pentru stiva de aplicații Android) permite instalarea Perl ca aplicație independentă [PERLA], şi rularea acestora prin intermediul ASE (*Automatic Scripting Environment*), alături de Python, JRuby, Lua şi BeanShell. ASE este un *layer* de *scripting* dezvoltat special pentru acest sistem de operare care facilitează accesul la funcțiile din *kit*-ul de dezvoltare [ASE]. Din păcate versiunea de Perl pentru Android existentă la ora actuală este incomplet implementată, şi multe dintre modulele acestui interpretor, care apar în implementările de tip Strawberry Perl sau Active Perl lipsesc, inclusiv cele care asigură accesul la CPAN (*The Comprehensive Perl Archive Network*). O versiune mai amplă de Perl pentru Android este la ora actuală în curs de dezvoltare [PERLN]. Din aceste considerente, autorul acestei lucrări a propus următoarea soluție:

- Accesul la *kernelul* de tip Linux pe care rulează sistemul de operare, aflat la baza stivei de aplicații Android; pentru aceasta trebuie obținut accesul la *userul* de tip *root*, care va permite rularea unei maşini virtuale de tip Ubuntu.
- Instalarea unei maşini virtuale de tip Ubuntu, compilată ARM32, o distribuție Linux completă care, după cum aminteam anterior, vine cu Perl instalat implicit. Procedura de creare a unei imagini ARM32 este descrisă în [ANDU].
- Lansarea Ubuntu virtuale prin aplicația *chroot*, aplicație care permite rularea unei maşini virtuale, alta decât cea existentă sub Android.
- Instalarea pe maşina virtuală a unui server de tip SSH (*Secure Shell*) pentru transportul fişierelor

sursă şi pentru a avea accesul la o fereastră de tip *command prompt* pe maşina virtuală, de pe un PC.

- Instalarea unui server VNC (*Virtual Network Connections*) pe maşina virtuală, dacă se doreşte captura ecranului Ubuntu.
- Copierea aplicațiilor NLP, prin serverul SSH, de pe PC pe maşina virtuală.
- Rularea aplicațiilor şi afişarea rezultatelor. Lansarea aplicațiilor se poate realiza în trei moduri: direct de pe telefon; prin serverul SSH; prin serverul VNC.

Reamintim faptul că aplicația noastră finală a fost optimizată pentru un maxim de performanță. Aceste lucru s-a realizat în următoarele moduri:

- Rularea stivei de module NLP într-o manieră *multithreading* pe două fire de execuție de tip multiproces, implementare realizată prin intermediul funcției *fork*. Un proces este rezervat pentru refacerea diacriticelor, iar cel de-al doilea este destinat pentru restul de module; cele două procese fac schimb de semnale prin IPC(*Inter Process Communications*), mai exact prin tehnici de tip *pipes*. În acest fel pe de o parte se scurtează timpii de execuție, prin încărcarea simultană a tuturor modelelor de limbaj antrenate în prealabil şi se poate beneficia eventual de o configurație multiprocesor (de altfel o astfel de configurație pentru dispozitive portabile a fost deja anunțată de firma LG prin telefonul Optimus 2X, cu procesor Tegra II, lansat şi în România în luna aprilie 2011).
- Evitarea prelucrărilor redundante pe datele de antrenare şi salvarea rezultatelor intermediare în format binar; acest lucru a fost realizat prin salvarea tuturor datelor de antrenare în forma finală utilă pentru procesare, renunțându-se astfel la fişierele de tip text în favoarea celor în format binar, obținute în Perl prin apelarea funcțiilor *store* şi *retrieve*, care pot acționa direct asupra unor structuri de date (şiruri, arii, liste etc.), permițând salvarea respectiv recuperarea acestora din memorie. Ca exemplu, acest lucru s-a dovedit benefic mai ales pentru algoritmul de refacere a diacriticelor unde se lucrează cu fişiere de *n*-grame de dimensiuni mari (de ordinul zecilor de MB). Acestea sunt transformate în interiorul algoritmului mai întâi în variabile de tip şir, după care aceste şiruri sunt transformate în structuri de tip *hash* care fac corespondența între *n*-grame şi probabilitățile lor de apariție în corpusul de antrenare. Pentru evitarea prelucrărilor succesive ulterioare am salvat direct sub formă de fişiere toate aceste structuri, fapt ce a dus la o scădere simțitoare a timpilor de inițializare a algoritmilor de la ordinul a 10-15 minute la ordinul secundelor.
- Reducerea dimensiunilor fişierelor de *n*-grame. După testarea performanțelor stivei NLP pe o arhitectură hardware de tip *PC-based* şi optimizarea algoritmilor în sensul scăderii timpilor de execuție, am constatat că necesarul de memorie RAM depăşea cu mult spațiul de memorie oferit de arhitectura pe care s-a decis implementarea. Am decis reducerea dimensiunii fişierelor de *n*-grame, prin renunțarea la *n*-gramele cu cele mai mici frecvențe de apariție. Metoda, cunoscută sub numele de *data pruning*, a fost descrisă în capitolul 3.
- La nivelul aplicației, după comenzile furnizate prin intermediul interfețelor, textul de intrare trebuie orientat către modulele de prelucrare. Acesta poate fi citit direct de pe mediile de stocare sau poate fi adus în timp real de pe Internet – de la o adresă de poştă electronică, *RSS feeder*, pagină WEB etc. Pentru a exemplifica rularea algoritmilor descrişi în capitolele anterioare, am propus o aplicație care citeşte mesajele de la o adresă de *e-mail* şi aplică algoritmul NLP pentru câmpurile extrase din mesaj. Aplicația realizează următorii paşi:
  - Se *loghează* automat la un cont de *e-mail* de tip YAHOO, cu nume de utilizator şi parolă
  - Citeşte şi numerotează automat mesajele noi
  - Extrage textul care marchează expeditorul mesajului
  - Extrage textul privind subiectul mesajului
  - Extrage mesajul în sine – corpul mesajului
  - Normalizează şi preprocesează toate elementele componente ale mesajului
  - Realizează refacerea diacriticelor dacă acestea lipsesc
  - Poziționează accentul lexical
  - Realizează despărțirea în silabe
  - Realizează conversia fonetică
  - Poziționează *markerii* de tip SSML

Rularea proceselor:

- Se lansează fereastra de tip *Terminal emulator* de pe *smartphone*.
- Se intră în mod *root* cu comanda **su**
- Se lansează maşina virtuală cu comanda **bootubuntu**
- Se lansează serverul **ssh**
- Se copie fişierele program printr-un server de tip ftp lansat odată cu serverul **ssh**
- Se lansează aplicația de prelucrare NLP a mesajelor de poştă electronică

Am efectuat transferul datelor folosind aplicația WinSCP. Spațiul de memorie ocupat de aplicație se poate analiza utilizând comanda *htop*. Cele două procese de care aminteam anterior rulează în paralel şi necesită 132 respectiv 109 MB.

Trebuie amintit faptul că bazele de antrenare au fost reduse în scopul rulării pe dispozitivul *embedded*. Pentru ca performanțele etajului NLP să nu scadă prea mult procesul de reducere a dimensiunilor a continuat iterativ şi s-a oprit în momentul în care s-a reuşit rularea întregului algoritm fără depăşirea spațiului de memorie avut la dispoziție – acest eveniment neplăcut a provocat de fiecare dată o resetare a dispozitivului. În ceea ce priveşte scăderea performanțelor algoritmilor, cauzată de micşorarea dimensiunilor modelelor de limbaj, aceasta se traduce printr-un rezultat final de 3.3% WER la nivel de cuvânt în cazul algoritmului de refacere diacriticelor. La celelalte module nu se mai pune problema reducerii performanțelor metodei deoarece bazele de antrenare au rămas aceleaşi.

10. Concluzii şi perspective

Colectivul de cercetare din care a făcut parte autorul acestei lucrări şi-a propus să proiecteze un sistem de sinteză a vorbirii pornind de la text (TTS – *Text-To-Speech*) în limba română de foarte bună calitate (vorbire expresivă, vocabular nelimitat de cuvinte) şi să realizeze trei versiuni distincte de implementare.

Obiectivele principale urmărite de autorul acestei lucrări au fost următoarele:

a. **Reliefarea importanței** majore pe care o are nivelul de prelucrare a limbajului natural (NLP – *Natural Language Processing*) într-un sistem de comunicare prin voce om – maşină şi explicarea rolului elementelor componente ale NLP pentru un sistem de sinteză a vorbirii pornind de la text pentru limba română.

b. **Pregătirea infrastructurii** în vederea implementării modulelor de prelucrare a limbajului natural. Una dintre cele mai importante dificultăți întâmpinate în implementarea etajului de prelucrare a limbajului natural pentru limba română a fost legată de lipsa unor studii lingvistice complete şi a instrumentelor corespunzătoare (dicționare transcrise fonetic, studii acustico-fonetice detaliate, corpusuri de vorbire naturală etichetate, corpusuri de silabe, cercetări privind prozodia etc.) care să permită tratarea automată cu ajutorul calculatorului a diferitelor sarcini legate de sinteza TTS. Etapa de realizare a unor astfel de corpusuri de către autorul acestei lucrări a fost una dintre cele mai mari consumatoare de timp.

c. **Realizarea integrală a etajului NLP şi implementarea modulelor NLP într-o manieră iterativă**

Extragerea cunoştințelor lingvistice relevante din textele de antrenare şi apoi implementarea completă a principalelor module ce compun etajul de prelucrare a limbajului natural în vederea folosirii acestuia într-un sistem TTS s-a dovedit nu numai o sarcină laborioasă în sine, dar şi extrem de dificilă în contextul particularităților legate de limba română, menționate anterior.

Pentru fiecare dintre algoritmii implementați a fost nevoie de mai multe iterații, traduse prin numeroase reantrenări şi testări, ajustări de parametri, corectări manuale ale dicționarelor şi corpusurilor de antrenare, extragerea sau obținerea unor dicționare separate de nume proprii, cuvinte despărțite în silabe, omografe, excepții.

d. **Implementarea etajului NLP pe un dispozitiv de tip *embedded***, care să demonstreze funcționalitatea sistemului, capacitatea acestuia de a fi ajustabil după o configurație hardware cu constrângeri şi care să asigure un nivel suficient de generalitate pentru a fi folosit pe post de *driver* pentru

un motor de sinteză (etajul de generare a vorbirii) aflat în faza de dezvoltare.

Principalele aspecte abordate în lucrare sunt detaliate în cele ce urmează:

În capitolul 2 autorul realizează o introducere în tehnologia vorbirii şi a interfeţelor de comunicare prin voce. S-a accentuat asupra caracterului multidisciplinar pe care îl implică proiectarea acestor interfeţe şi s-a realizat o clasificare a aplicaţiilor de comunicare prin voce după destinaţia finală şi după arhitectura de implementare. A fost realizată o caracterizare generală a unui sistem TTS pentru limba română şi au fost descrise pe larg componentele acestuia, insistând asupra rolului şi importanţei modulelor de prelucrare a limbajului natural.

În capitolul 3 sunt descrise fundamentele modelelor de limbaj în vederea introducerii lor în sistemele de tip TTS şi sunt prezentate modelele lingvistice bazate pe *n*-grame. Tot în acest capitol au fost evidenţiate conceptele matematice folosite în procesele de netezire a probabilităţilor *n*-gramelor şi în cele de reducere a dimensiunilor corpusurilor de antrenare, în perspectiva folosirii modelelor de limbaj pentru aplicaţii TTS pe dispozitive de tip *embedded*, caracterizate prin dimensiuni reduse ale spaţiului de memorie.

În capitolul 4 autorul descrie un algoritm original de refacere a semnelor diacritice pentru aplicaţiile de sinteză care folosesc texte scrise fără diacritice în limba română. Procesul este necesar a fi implementat în sistemele TTS pentru ca vorbirea generată să fie inteligibilă iar mesajul să nu fie alterat. Algoritmul este realizat prin metode statistice, pe bază de *n*-grame la nivel de cuvânt şi terminaţii ale acestora. Metoda rulează după principiul filtrărilor minimale succesive şi permite folosirea sau testarea ulterioară şi a altor criterii de filtrare. Astfel, la ora actuală autorul acestei lucrări dezvoltă încă un nivel de filtrare pe bază de părţi de vorbire (POS – *Part-of-Speech*), rezultatele preliminare indicând o creştere de performanţă.

Capitolul 5 este destinat descrierii algoritmului de predicţie a accentului lexical ca parte componentă a nivelului de generare a prozodiei într-un sistem TTS. Tot în acest capitol se face o comparaţie între metodele bazate pe reguli şi cele de tip statistic folosite pentru rezolvarea poziţionării accentului lexical şi se descriu problemele legate de poziţionarea accentului lexical în limba română. Metoda implementată de autorul acestei lucrări este originală prin faptul că foloseşte tehnici statistice bazate pe *n*-grame la nivel de caracter, cu netezirea probabilităţilor, mai exact un algoritm de tip *Katz cu revenire modificat*. Tot în acest capitol s-a analizat influenţa unui prag de decizie variabil asupra rezultatelor finale ale sistemului TTS în vederea reducerii unor efecte nedorite cauzate de poziţionarea eronată a accentului lexical.

În capitolul 6 autorul continuă descrierea algoritmilor de prelucrare a limbajului natural pentru sistemul TTS în limba română cu despărţirea automată în silabe, etapă obligatorie pentru poziţionarea corectă a accentelor lexicale la nivelul silabelor şi în final generarea unei prozodii corespunzătoare. În plus, această etapă este esenţială, în abordarea noastră, şi pentru transcrierea fonetică a textului de intrare. Sunt discutate elementele caracteristice ale limbii române pentru acest subiect şi este descrisă o metodă statistică originală de despărţire automată în silabe, care foloseşte un algoritm hibrid, bazat atât pe reguli cât şi pe o abordare statistică.

Capitolul 7 descrie problemele conversiei fonetice pentru limba română, etapă esenţială pentru un algoritm de sinteza vorbirii pornind de la text, indiferent de tehnologia de sinteză folosită. Tot în acest capitol autorul a considerat necesar să realizeze o clasificare a diftongilor, triftongilor şi hiaturilor limbii române, ca elemente definitorii a problemei grupurilor vocalice. În urma acestei analize originale, autorul a ajuns la concluzia că se poate implementa un algoritm de conversie fonetică bazat numai pe reguli dacă se pun în evidenţă hiaturile printr-un algoritm de despărţire în silabe. Un număr limitat de ambiguităţi rămase sunt abordate folosind algoritmul de predicţie a accentelor lexicale descris în capitolul 5.

În capitolul 8 autorul descrie procedura de preprocesare şi normalizare a textelor de intrare pentru sistemul TTS în limba română. Acest modul este responsabil de următoarele aspecte importante referitoare la textul de intrare: despărţirea textului de intrare în paragrafe şi fraze, separarea cuvintelor, conversia unor structuri speciale (abrevieri, numerale cardinale şi ordinale, numere de telefon etc.) în forma lor ortografică, identificarea semnelor de punctuaţie şi salvarea acestora. Tot în acest capitol sunt descrise specificaţiile *Speech Synthesis Markup Language* (SSML) propuse de consorţiul W3C şi este descrisă implementarea acestui mod de marcare la nivelul sistemului nostru TTS în limba română.

Capitolul 9 descrie implementarea etajului de prelucrare a limbajului natural pe un dispozitiv de tip

embedded, ca parte componentă a unei aplicații de citire a mesajelor de poştă electronică. Ca dispozitiv a fost folosit un telefon mobil de ultimă generație, model Samsung Nexus S, cu sistem de operare Android.

Trebuie insistat asupra faptului că, prin prisma folosirii unor algoritmi preponderent statistici, pentru obținerea unui model de limbaj cât mai corect, antrenarea trebuie făcută pe un spațiu bine reprezentat, atât în ceea ce priveşte corectitudinea datelor de antrenare (a corpusurilor de antrenare) cât şi în ceea ce priveşte dimensiunea şi diversitatea acestora. Prima caracteristică a spațiului de antrenare este necesară în vederea obținerii unei relaționări corecte între modelul de limbaj şi spațiul soluțiilor. Cea de-a doua caracteristică are în vedere consistența datelor de antrenare din perspectiva modelului de limbaj implementat, fiind posibilă chiar alegerea unui domeniu mai restrâns de antrenare a modelelor de limbaj (medical, miliar, politic etc.), în ideea asigurării unei mai bune consistențe.

Mai mult, pentru a se minimiza o serie de efecte nedorite, concretizate prin apariția în timpul rulării testelor a unor evenimente neobservate la antrenare (este cazul cuvintelor noi) şi care pot duce fie la rezultate eronate fie la erori de algoritm, trebuie luate unele măsuri speciale de abordare pentru astfel de situații, concretizate de cele mai multe ori prin procese de netezire a probabilităților, care să aloce probabilități diferite de zero evenimentelor neîntâlnite pe timpul antrenării. În ceea ce priveşte spațiul de testare, trebuie insistat asupra faptului că, indiferent de tipul de algoritm (statistic sau pe bază de reguli), nu pot fi raportate rezultate corecte dacă se face evaluarea metodelor implementate pe baze de test extrase din corpusurile de antrenare sau care conțin elemente comune cu aceasta.

În implementările descrise în lucrare spațiile de antrenare respectiv de căutare a soluțiilor au o reprezentare diferită în funcție de tipul problemei abordate. A fost interesant să observ că pentru rezolvarea sarcinilor etajului NLP din perspectiva folosirii lui în sistemele TTS, spațiul soluțiilor poate fi atât dedus pe timpul antrenării cât şi generat automat la testare. De exemplu, în cazul refacerii diacriticelor, spațiul de antrenare este reprezentat de colecții mari de texte, scrise corect cu diacritice. Spațiul soluțiilor este reprezentat de variantele posibile de cuvinte scrise cu diacritice, variante obținute pe timpul antrenării. Un alt exemplu poate fi dat pentru problema poziționării accentului lexical: spațiul de antrenare este reprezentat de un corpus de forma unui dicționar, format din cuvinte cu accentual lexical corect poziționat iar spațiul soluțiilor este reprezentat de o serie de variante generate automat prin poziționarea markerului de accent lângă fiecare vocală.

Trebuie făcută precizarea că pentru modelele de limbaj realizate la nivelul cuvintelor întregi în limba română corpusurile de antrenare/testare sunt extrase din colecții mari de texte. Contextele de antrenare sunt localizate în interiorul frazelor sau propozițiilor, iar pentru corectitudinea rezultatelor trebuie respectată această segmentare. Pe de altă parte, dacă algoritmii statistici implementați acționează la nivel de caracter, corpusurile de antrenare/testare sunt reprezentate de dicționare formate din cuvinte distincte, contextele de antrenare fiind localizate chiar în interiorul acestora.

Contribuții originale

1. Am propus ca modulele unui etaj de prelucrare a limbajului natural din cadrul sistemului TTS pentru limba română să facă schimb de mesaje pentru rezolvarea unor situații de ambiguitate. Astfel, modulul de despărțire în silabe furnizează informații către modulul de poziționare a accentului lexical şi către modulul de conversie fonetică. În plus, modulul de conversie fonetică primeşte date şi de la modulul de predicție a accentului lexical pentru rezolvarea unor situații de ambiguitate.
2. Am reuşit o tratare modulară a întregului nivel NLP, sub forma unei stive. În acest fel, depanarea programelor este mult mai facilă, iar ulterior se poate interveni numai asupra anumitor module, în vederea îmbunătățirii performanțelor sau portării pe alte sisteme sau aplicații. Fiecare modul are un format unic de intrare-ieşire, sub forma unor fişiere text codificate UTF8, folosind markeri SSML.
3. Algoritmii implementați sunt capabili să prelucreze orice text de intrare. Astfel, în afara unor limitări create de modulul de preprocesare/normalizare a cărui dezvoltare va rămâne un subiect deschis în permanență, nivelul NLP poate aborda cazul cuvintelor neîntâlnite pe timpul antrenării, cu condiția ca acestea să respecte regulile lingvistice ale limbii române. O problemă aparte apare

în cazul împrumuturilor din alte limbi, pentru abordarea căreia se simte nevoia constituirii unor dicționare de excepții.

4. Am arătat că abordările de tip hibrid (statistic plus reguli) pot îmbunătăți substanțial atât rezultatele prelucrării NLP cât şi timpii de execuție.
5. Am realizat, în urma rulării algoritmilor implementați, un corpus compus din cuvinte convertite fonetic, despărțite în silabe şi cu accentul lexical poziționat, care numără peste 330.000 de intrări. Acesta poate fi folosit atât cu rol de lexicon într-un sistem TTS cât şi în rol de corpus de antrenare pentru alți algoritmi statistici destinați conversiei fonetice, poziționării accentului lexical, de despărțire în silabe etc. Un astfel de lexicon, poate reduce dramatic efortul de calcul sau efortul de cercetare deoarece o serie întreagă de etape sunt eliminate însă trebuie spus că nu se poate renunța la etajul NLP care este esențial în vederea abordării cuvintelor din afara lexiconului. Trebuie precizat totuşi că acuratețea obținută cu datele din acest lexicon nu o poate depăşi pe cea a modulelor descrise anterior.
6. Am obținut un corpus de antrenare, **pentru algoritmul de refacere a diacriticelor**, de dimensiuni medii (10.000.000 de cuvinte) compus din texte în limba română, cu diacritice corect poziționate. Pentru a avea un corpus de antrenare echilibrat am folosit texte extrase din diferite surse: cărți în format electronic şi pagini Web.
7. Am obținut un dicționar propriu, care numără peste 330.000 de cuvinte diferite. Acest dicționar conține multe forme flexionate ale cuvintelor din limba română, şi a fost realizat prin adăugarea cuvintelor extrase din corpusul de antrenare la un dicționar inițial de aproximativ 65.000 de intrări. Acest dicționar serveşte la găsirea variantelor de obținere a diacriticelor dar poate avea şi alte aplicații [UNG08].
8. Am propus şi implementat un algoritm original de refacere a semnelor diacritice pentru texte fără diacritice scrise în limba română, bazat pe probabilitatea de apariție a cuvintelor scrise cu diacritice şi pe contextul în care acestea apar. Am demonstrat astfel că o abordare statistică, folosind n-grame, conduce la rezultate foarte bune.
9. Am introdus principiul filtrărilor succesive cumulat cu cel al filtrărilor minimale pentru refacerea semnelor diacritice. Primul principiu asigură un criteriu de complementaritate şi consistență iar al doilea de precizie. Prin prisma bazei de antrenare, compusă dintr-un număr mediu de cuvinte, am observat apariția fenomenului de inconsistență de antrenare, fenomen prezent cu precădere la folosirea n-gramelor la nivel de cuvânt [UNG08].
10. Am analizat diverse niveluri de filtrare, pentru refacerea diacriticelor. Am demonstrat că în condițiile existenței unui corpus de antrenare de dimensiuni medii n-gramele realizate cu terminațiile cuvintelor pot aduce un plus de performanță pentru refacerea diacriticelor. Mai mult, în aceleaşi condiții, folosirea trigramelor din cuvinte întregi nu aduce rezultate semnificative, din cauza insuficienței datelor de antrenare. Pe de altă parte, nici folosirea începutului cuvintelor nu aduce un câştig semnificativ metodei, dar această observație nu este legată de corpusul de antrenare ci de specificul limbii române. Am arătat că lungimea optimă a finalului cuvintelor pentru antrenarea trigramelor este egală cu 4 şi am arătat că ordinea în care se aplică filtrările succesive pentru algoritmul de refacere a diacriticelor influențează rezultatele algoritmului [UNG08].
11. Am realizat o aplicație independentă de poziționare a diacriticelor printr-o implementare a algoritmului de poziționare a diacriticelor care păstrează forma grafică a textului de intrare. Acest lucru a fost realizat prin folosirea unui sub-nivel propriu de preprocesare a textului de intrare, care detectează semnele ortografice, caracterele alfa-numerice, desparte textul în propoziții/fraze şi extrage cuvintele.
12. Rezultatele finale obținute pentru algoritmul de poziționare a diacriticelor, pe un corpus de test de peste 2.000.000 de cuvinte extrase din cuvântările ținute în Parlamentul European [EURO] sunt: o rată de eroare la nivel de cuvânt (WER) de 1,4%, respectiv de 0,4% la nivel de caracter. În plus, la ora actuală există deja o serie de rezultate pozitive în ceea ce priveşte implementarea unui nivel de filtrare suplimentar, bazat pe părți de vorbire, primele rezultate indicând o scădere a ratei de eroare la nivel de cuvânt la aproximativ 1% [UNG08][UNG11b].

13. Am realizat un corpus de antrenare-testare pentru algoritmul de despărțire în silabe, compus dintr-un dicționar de cuvinte corect despărțite în silabe, care numără aproximativ 56.000 de intrări.
14. Am propus şi implementat un algoritm hibrid de despărțire în silabe compus din două seturi de reguli între care este intercalată o metodă de prelucrare statistică. Algoritmul statistic este original, bazat pe n-grame, fiind un algoritm Katz cu revenire, modificat în sensul că nu s-au folosit în mod deliberat tehnici de netezire a n-gramelor pentru istorii mai mici decât 3. La acest nivel am considerat că antrenarea este suficient de bine făcută pentru ca lipsa unor n-grame de acest ordin să semnaleze de fapt o despărțire incorectă în silabe. Astfel, algoritmul se încheie mai rapid pentru variantele incorecte, ceea ce are un efect benefic asupra timpului de calcul. Decizia finală este făcută, pentru variantele generate care parcurg întregul algoritm, prin calcularea probabilităților finale de emisie şi alegerea variantei de probabilitate maxime. Performanțele algoritmului implementate sunt de 2,96% WER [UNG11a].

15. Am realizat un corpus de antrenare-testare-validare pentru un algoritm de poziționare a accentelor lexicale compus din cuvinte cu accentul lexical corect poziționat care numără peste 330.000 de intrări. Acest corpus a fost realizat din dicționarul amintit anterior, la care s-au poziționat manual accentele lexicale.
16. Am propus şi implementat un algoritm original de poziționare a accentelor lexicale. Metoda utilizează o strategie statistică, bazată în principal pe n-gramele construite la nivel de caracter şi foloseşte un algoritm de tip de netezire de tip Katz cu revenire, modificat după următorul principiu: din fiecare variantă posibilă se extrag n-gramele de ordin n, n–1,…,1 şi se calculează emisiile de probabilitate finale, probabilitățile condiționate fiind cele obținute în timpul etapei de antrenare. În urma rulării algoritmului de poziționare a accentelor lexicale am observat că performanțele maxime se obțin pentru o lungime maximă a n-gramelor de 8 caractere [UNG09].
17. În vederea reducerii efectelor nedorite cauzate de poziționarea incorectă a accentului lexical în vorbirea sintetizată, am propus utilizarea unui prag de decizie variabil, care va poziționa accentul lexical numai pentru acele cuvinte pentru care raportul probabilităților între primele două variante de ieşire depăşeşte o valoare de prag. În situațiile de ambiguitate în care pragul de decizie nu este depăşit cuvântul este lăsat neaccentuat. Acest lucru se poate dovedi benefic pentru sistemul TTS în ansamblu, prin reducerea numărului de variante cu accent lexical incorect poziționat. Pragul de decizie optim va fi stabilit pe timpul rulării testelor de sinteză, cel mai probabil ca valoare care maximizează un rezultat de tip Mean Opinion Score (MOS).
18. În urma unei analize proprii am demonstrat că pentru realizarea sistemului TTS cuvintele monosilabice pot fi considerate ca fiind fără accent lexical. Detecția cuvintelor monosilabice este făcută prin algoritmul de despărțire în silabe. Rezultatul final obținut pentru algoritmul de poziționare a accentelor lexicale este de 0,89% WER [UNG09]. Pe baza corpusului de antrenare destinat pentru poziționarea accentelor lexicale am realizat un algoritm original de detecție a formelor omografice şi am creat un dicționar de forme omografice.

19. Am demonstrat, în urma unei analize privitoare la fonemele limbii române, că se poate implementa un algoritm de conversie fonetică pe bază de 45 de reguli, dacă se pun în evidență hiaturile cu ajutorul modulului de despărțire în silabe. Am implementat un modul de conversie fonetică bazat numai pe reguli care abordează majoritatea situațiilor dificile pentru limba română, cu două excepții: cazul diftongilor iu, care pot fi ascendenți sau descendenți şi a vocalei cazului i din finalul cuvintelor care apare înaintea consoanelor. Am arătat că informația referitoare la poziționarea accentului lexical poate fi folosită în procesul de conversie fonetică pentru a rezolva cele două ambiguități amintite anterior: diftongul iu şi i din finalul cuvintelor care apare înaintea consoanelor [UNG11a].
20. Am realizat o analiză detaliată a grupurilor vocalice specifice limbii române (diftongi, triftongi şi hiaturi) şi am realizat o clasificare a acestora în vederea extragerii regulilor de conversie fonetică amintite anterior. Rezultatul final obținut raportat pentru algoritmul de conversie fonetică este o rată de eroare de 3,11% WER la nivel de cuvânt [UNG11a].

21. Am realizat implementarea întregii stive NLP folosind markeri de tip SSML în vederea asigurării consistenței, interoperabilității şi generalității sistemului de sinteză. Consider că limbajul SSML tinde să devină un standard de facto în ceea ce priveşte sinteza de tip TTS. Pentru a obține o implementare cât mai modulară am folosit elementele SSML pentru a realiza schimbul de date cât şi de comenzi, atât intern între diferitele module NLP cât şi extern între cele două niveluri deja menționate. Fiecare dintre modulele NLP implementate va adăuga un plus de semnificație şirului textual sau SSML furnizat la intrare, într-o manieră incrementală, în scopul obținerii unei reprezentări fonetice şi prozodice cât mai apropiate de semnificația textului prelucrat. O asemenea abordare oferă o mai mare flexibilitate deoarece oricare dintre modulele NLP sau întregul etaj de generare a vorbirii pot fi înlocuit în scopul obținerii unor performanțe mai bune. [UNG11b]

22. Am reuşit implementarea întregii stive NLP pe o platformă de tip embedded, pentru o aplicație de citire a unor mesaje de poştă electronică.
23. Am implementat toate modulele de prelucrare a limbajului natural în condițiile unui spațiu limitat de memorie, printr-o reducere succesivă a dimensiunilor modelelor lingvistice (numai pentru etajul de refacere a diacriticelor), după criteriul frecvențelor de apariție, dar păstrând dimensiunea maximă a modelelor de limbaj pentru care s-a reuşit rularea. În acest fel s-a asigurat un maxim de consistență a modelelor iar reducerea performanțelor a fost minimă, în ceea ce priveşte rata de eroare la nivel de cuvânt.
24. Din perspectiva implementării pe clasa embedded am arătat că cele mai importante constrângeri sunt create de modelul de limbaj utilizat pentru refacerea diacriticelor, care utilizează în special n-grame la nivel de cuvânt.
25. În vederea rulării proceselor în timp real, am implementat o serie de tehnici de prelucrare originale, centrate pe arhitectura avută la dispoziție, în măsură să scadă foarte mult timpii de execuție. Astfel am implementat următoarele metode:
26. Evitarea proceselor redundante prin salvarea în memorie a modelelor de limbaj în forma finală necesară prelucrării.
27. Rularea multiproces. Astfel am separat sarcina de refacere a diacriticelor care rulează într-un singur proces, un al doilea fir de execuție fiind destinat celorlalte module NLP rămase. Cele două procese se sincronizează reciproc şi fac schimb de date de prelucrare.
28. Folosirea buffer-lor de procesare. Aceasta s-a dovedit foarte benefică, în special pentru algoritmul de refacere a diacriticelor. Deoarece pot exista situații (pe timpul testării au fost simulate astfel de cazuri) în care la intrarea nivelului NLP să fie furnizate texte de dimensiuni foarte mari (sute de mii de propoziții) am constatat că din punct de vedere al timpilor de calcul este benefică utilizarea unui buffer format din maxim 1000 de propoziții/fraze oferite simultan nivelului NLP, separate prin semne de punctuație adecvate. S-a ajuns astfel la reducerea de până la un ordin de mărime a timpilor de calcul pentru texte foarte mari (de ordinul milioanelor de cuvinte), observație care se poate dovedi utilă şi în vederea implementării unor sisteme TTS de tip client-server în care sarcina principală de prelucrare este localizată la nivelul serverului.

Posibile dezvoltări ulterioare

1. Trebuie subliniat faptul că cel puțin la ora actuală, prin prisma rezultatelor comunicate în literatura de specialitate, performanțele obținute sunt dintre cele mai bune pentru fiecare dintre modulele implementate. Implementarea finală a întregului nivel NLP rulează în timp real (sunt procesate peste 4.000 de cuvinte pe secundă) pe un calculator personal de performanțe medii. În plus nu cunoaştem în acest moment o altă abordare completă a întregului etaj NLP, folosit pentru un sistem de sinteză TTS pentru limba română bazat pe concatenare de segmente acustice, cum este a noastră. Consider că stiva NLP descrisă în lucrare asigură un cadru suficient pentru a conduce la performanțe foarte bune pentru sistemul de sinteza vorbirii pornind de la text în limba română, a cărui dezvoltare este în curs de finalizare în momentul de față.

2. Datorită faptului că dispunem la ora actuală de un *toolkit* NLP matur, capabil să rezolve majoritatea problemelor importante pentru un sistem TTS, pot fi obținute pe baza lui o serie de dicționare de antrenare şi lexicoane mai ample, eventual orientate pe anumite domenii de activitate. Intuitiv, astfel de

abordări ar trebui să ducă la o scădere a dimensiunilor modelelor de limbaj, la scăderea timpilor de execuție şi la o îmbunătățire a rezultatelor.

3. Folosirea diverselor resurse lingvistice obținute, pentru alte tipuri interfețe de comunicare prin voce, cum ar fi cele de recunoaştere a vorbirii.

4. Implementarea unui server de tip NLP într-o aplicație de sinteză distribuită a vorbirii şi testarea arhitecturii pe diferite tipuri de canale de comunicație.

5. Orientarea către o abordare *multimodală*, în care etajul de prelucrare a limbajului natural să ocupe aceeaşi poziție de *etaj driver* atât pentru un etaj de generare a vorbirii cât şi pentru alte etaje de tipul celor de generare a imaginilor, generare de gesturi etc.

6. Obținerea unor sisteme de prelucrare a limbajului natural, capabile să "înțeleagă" contextul în care a fost rostit un enunț.

7. Dezvoltarea etajului de preprocesare/normalizare, etaj la care se pot aduce în mod constant îmbunătățiri, prin creşterea numărului de abrevieri/acronime luate în calcul, extinderea lexicoanelor de cuvinte împrumutate din alte limbi, frecvent utilizate în limba română, a celor de nume proprii, a dicționarelor de excepții etc.

8. Separarea sarcinii NLP de cea TTS şi utilizarea modulelor componente pentru diferite sarcini de lingvistică computațională, de exemplu în scopul obținerii unor aplicații independente (de refacere diacritice, despărțire în silabe etc.).

9. Prin prisma specificului algoritmilor de prelucrare lingvistică obținuți, apreciez că este de asemenea posibilă folosirea modulelor de prelucrare a limbajului natural, descrise în această lucrare, pentru o apreciere cantitativă a originilor limbii române şi a împrumuturilor din alte limbi.

Bibliografie selectivă

[ANDU] http://androlinux.com/android-ubuntu-development/how-to-build-chroot-arm-ubuntu-images-for-android/

[ASE] http://code.google.com/p/android-scripting/

[AST03] Astrov, S., Bauer, J. G., Stan, S., "High Performance Speaker and Vocabulary Independent ASR Technology for Mobile Phones". In *Proceedings of ICASSP 2003*, pp 2.281-2.284, IEEE, 2003

[BAG03] Bagein, M. and Pietquin, O., *Architecture for VoiceEnabled Interfaces over Local Wireless Networks*, Faculté Polytechnique de Moins – TCTS Lab, 2003

[BER03] Bemsen, N. O., "On-line User Modelling in a Mobile Spoken Dialogue System". In *Proceedings of the 8th European Conference on Speech Communication and Technology—Eurospeech 2003*, September 1-4, Geneva, Switzerland, pp 737—740, 2003

[BUR99a] Burileanu, D., Dan, C., Sima, M, Burileanu, C., "A Parser-Based Text Preprocessor for Romanian Language TTS Synthesis". In *Proceedings of Eurospeech'99*, Budapest, Hungary, Vol. 5, pp. 2063-2066, 1999

[BUR99b] Burileanu, D., "Contribuții la Modelarea si Procesarea Semnalelor de Vorbire în Vederea Sintezei Automate Pornind de la Text în Limba Română". *Teza de doctorat*, Universitatea "Politehnica" Bucureşti, 1999

[BUR03a] Burileanu, D., Fecioru, A. and Ion, D. "On Automatic Speech Synthesis for Spoken Languages Interfaces". In *Speech Technology and Human-Computer Dialogue*, Publishing House of the Romanian Academy, Bucharest, pp. 127-138, 2003, ISBN 973-27-0963-4.

[BUR04b] Burileanu, D., Fecioru, A., Ion, D., Stoica, M., Ilas, C, "An Optimized TTS System Implementation Using a Motorola StarCore SC140-Based Processor". In *Proceedings of ICASSP 2004*, Montreal, Canada, Vol. 5, pp. 317-320, 2004

[BUR04c] Burileanu, D., "Speech Synthesis Technology Application to Intelligent Interfaces with Embedded Devices". In *Proceedings of the Romanian Academy*, Series A, Vol. 5, No. 3, Ed. Academiei Române, Bucuresti, 2004

[BUR07] Burileanu, D., "Spoken Language Interfaces for Embedded Applications. In Daryle Gardner-Bonneau & Harry E. Blanchard", *Human Factors and Voice Interactive Systems*, 2nd edition, Springer, 2007

[BUR10a] Burileanu, D., Negrescu, C. and Surmei, M., "Recent Advances in Romanian Language Text-to-Speech Synthesis". In *Proceedings of the Romanian Academy, Series A – Mathematics, Physics, Technical Sciences, Information Science, vol. 11, no. 1*, Publishing House of the Romanian Academy, Bucharest, pp. 92-99, 2010

[CHE96] Chen, S.F. and Goodman, J., "An Empirical Study of Smoothing Techniques for Language Modeling". In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, Santa Cruz (California), USA, pp. 310-318, 1996

[DEX98] *Dicţionar Explicativ al Limbii Române*, Academia Română, Institutul de lingvistică "Iorgu Iordan", Ed. Univers Enciclopedic, Bucureşti 1998

[DOO98] *Dicţionarul ortografic, ortoepic şi morfologic al limbii române*, Bucureşti, Editura Academiei, 1998

[GER06] Geruson, R., "Embedded Speech Systems in Mass Market Phones". *Voice Signal Technologies*, 2006

[HUA07] Zheng-Hua, T. and Varga, I.. "Networked, Distributed and Embedded Speech Recognition: An Overview". In Z.-H. Tan, and B. Lindberg (eds.), *Automatic speech recognition on mobile devices and over communication networks*, Springer-Verlag, London, 2007

[IND] http://www.indigostar.com/perl2exe.php

[IBM08b] IBM Research. *Embedded Conversational Solutions.* IBM Research, 2008

[JIT02] Jitca, D., Teodorescu, H.N., Apopei, V., and Grigoras, F., "Improved Speech Synthesis Using Fuzzy Methods". In *International Journal of Speech Technology, Springer Netherlands*, No. 5, pp. 227-235, 2002

[KAT87] Katz, S.M., "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer". In *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 35, No.3, pp.400-401, 1987

[KIM01] Kim, H.K., Cox, R.V., "Feature Enhancement for a Bitstream-based Front-end in Wireless Speech Recognition". In *Proceedings of ICASSP*, Salt Lake City, U, 2001

[KAR06] Karpov, E., Kiss, I, Leppanen, J., Olsen, J., Oria, D., Sivadas, S and Tian, J., "Short Message Dictation on Symbian Series S60 Mobile Phones". In *Proc WorkshopSpeech Mobile Pervasive Environments (SiMPE) Conjunction MobileHCI*, 2006

[KIS02] Kiss, I., Vasilache, M., "Low Complexity Techniques for Embedded ASR Systems", In *Proceedings of International Conference on Spoken Language Processing*, (ICSLP2002), Denver, Colorado, USA, September 2002

[MAN00] Manning, C. and Schütze, H., *Foundations of Statistical Natural Language Processing.* London: The MIT Press, 2000

[MIH02] Mihalcea, R. and Nastase, V., "Letter Level Learning for Language Independent Diacritics Restoration". In *Proceedings of CoNLL 2002*, Taipei, Taiwan, pp. 105-111, 2002

[NEGR07] Negrescu, C., Ciobanu, A., Burileanu, D., Stanomir, D., *An Improved Hybrid Time-Frequency Algorithm for Time-Scale Modification of Speech/Audio Signals.* Advances in Spoken Language Technology, Publishing House of the Romanian Academy, Bucharest, pp. 147-162, 2007

[NEGR08] Negrescu, C., Ciobanu, A., Stanomir, D., "Refined Spectral Estimation of Tonal Components". In *Proceedings of the Annual Symposium of the Institute of Solid Mechanics*, 2008

[NÉM08] Németh, G., Kiss, G.,Zainkó,C., Olaszy,G., Tóth, B., "Speech Generation In Mobile Phones". *In* Gardner-Bonneau, D., Blanchard, H., *Human Factors and Interactive Voice Response Systems*, 2nd ed., Springer US, pp. 163-191, 2008

[OAN02] Oancea, E. and Badulescu, A., Stressed Syllable Determination for Romanian Words Within Speech Synthesis Applications". *International Journal of Speech Technology*, No.5, Vol.3, pp.237–246, 2002

[PERLA] http://code.google.com/p/android-scripting/downloads/list

[PERLN] http://code.google.com/p/perldroid/

[POS06] POST. *Pervasive Computing*. The Parliamentary Office of Science and Technology, London, 2006

[SMI96] Smith, A., Dunaway, J., Demasco, P., Peischl, D., "Multimodal Input for Computer Access and Augmentative Communication". *In Annual ACM Conference on Assistive Technologies*, pp. 80-85, 1996

[SUT93] Şuteu, F., Şoşa, E., *Dicţionar ortografic al limbii române (Orthographic Corpus of Romanian Language)*. Bucharest: ATOS Publishing House, 1993

[THO07] Thomas, J., Basson, S. and Bonneau, D.G., *Accessibility and speech technology: advancing toward universal access.* IBM T.J. Watson Research Center Bonneau and Associates, and Western Michigan University, 2007

[UNG08] Ungurean, C, Burileanu, D., Popescu, V., Negrescu, C., Derviş, A."Automatic Diacritic Restoration for a TTS-based E-mail Reader Application", *UPB Scientific Bulletin*, Series C, Vol. 70, Issue 4, Ed. Politehnica Press, Bucharest, pp. 3-12, 2008, ISSN 1454-234x – *Revistă categoria B+, indexare INSPEC, SCOPUS, Ulrich's International Periodicals Directory, Elsevier Sciences's Bibliographic Database;* http://www.scientificbulletin.upb.ro/Arhiva_2008/Seria_C/Nr4C-2008.pdf

[UNG09] Ungurean C., Burileanu D., Negrescu C. and Dervis A., "A Statistical Approach to Lexical Stress Assignment for TTS Synthesis". In *International Journal of Speech Technology, Springer Netherlands, vol. 12*, no. 2-3, pp. 63-73, 2009

[UNG11a] Ungurean C., Burileanu D., Popescu V., and Dervis A., "Hybrid Syllabification and Letter-to-phone conversion for TTS synthesis". In *UPB Scientific Bulletin*, Series C, 2011.

[UNG11b] Ungurean C., Burileanu D., "An Advanced NLP Framework for High-Quality Text-to-Speech Synthesis". In *Proceedings of the 6th Conference "Speech Technology and Human-Computer Dialogue – SpeD 2011"*, Braşov, (CD-ROM), Published under the aegis of Romanian Academy, pp. 87-92, 2011

[W3O] http:www.w3.org/TR/speech-synthesis11/

Lista de lucrări

1. Ungurean, C., Burileanu, D., Popescu,V. and Dervis, A. "Hybrid Syllabification and Letter-to-Phone Conversion for TTS Synthesis", *UPB Scientific Bulletin, Series C*, Ed. Politehnica Press, Bucharest, 2010, ISSN 1454-234x (lucrare în curs de publicare) **– *Revistă categoria B+, indexare INSPEC, SCOPUS, Ulrich's International Periodicals Directory, Elsevier Sciences's Bibliographic Database***

2. **Ungurean, C.**, Burileanu, D., "An Advanced NLP Framework for High-Quality Text-to-Speech Synthesis"**,** *Proceedings of the 6th Conference "Speech Technology and Human-Computer Dialogue – SpeD 2011"*, Braşov, CD-ROM, pp. 87-92, 2011 **– Indexare INSPEC, IEEE Xplore - IEEE Catalog Number: CFP1155H-DVD ISBN: 978-1-4577-0439-0** http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5940733**,** DOI: 10.1109/SPED.2011.5940733

3. Burileanu, D., Surmei, M., Negrescu, C., **Ungurean, C.**, Dervis, A. "Multi-Access Network-Based TTS Architectures", *Romanian Journal of Information Science and Technology*, Vol. 13, No. 3, Publishing House of the Romanian Academy, Bucharest, pp. 310-320, 2010, ISSN 1453-8245 **– *Revistă cotată ISI Web of Science (Science Citation Index Expanded)***

4. Ungurean, C., Burileanu, D., Negrescu, C. and Dervis, A. "A Statistical Approach to Lexical Stress Assignment for TTS Synthesis", *International Journal of Speech Technology*, Vol. 12, No. 2-3 (Sept. 2009), Springer Netherlands, pp. 63-73, 2009, ISSN 1381-2416 **– *Revistă indexată SCOPUS, ERIH, Compendex, CompuScience, Academic OneFile, Ei Page One, SpringerLink;*** http://www.springerlink.com/content/x658743813381765/, http://dx.doi.org/doi:10.1007/s10772-009-9062-4

5. Surmei, M., Burileanu, D., Negrescu, C., **Ungurean, C.**, Derviş, A. "Real-Time Architectures for a Network-Based Text-to-Speech System Implementation", *Proceedings of the 5th Conference "Speech Technology and Human-Computer Dialogue – SpeD 2009"*, Constanța, published in *From Speech Processing to Spoken Language Technology*, Publishing House of the Romanian Academy, Bucharest, pp. 95-102, 2009, ISBN 978-973-27-1808-7 **– *Volum indexat INSPEC, IEEE Xplore - IEEE Catalog Number: CFP0955H-CDR;*** http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?reload=true&arnumber=5156178, http://dx.doi.org/doi:10.1109/SPED.2009.5156178

6. Ungurean, C, Burileanu, D., Popescu, V., Negrescu, C., Derviş, A."Automatic Diacritic Restoration for a TTS-based E-mail Reader Application", *UPB Scientific Bulletin*, Series C, Vol. 70, Issue 4, Ed. Politehnica Press, Bucharest, pp. 3-12, 2008, ISSN 1454-234x **– *Revistă categoria B+, indexare INSPEC, SCOPUS, Ulrich's International Periodicals Directory, Elsevier Sciences's Bibliographic Database;*** http://www.scientificbulletin.upb.ro/Arhiva_2008/Seria_C/Nr4C-2008.pdf

7. Surmei, M., Burileanu, D., Negrescu, C., Pîrvu, R., **Ungurean, C.**, Derviş, A. Text-to-Speech Engines as Telecom Service Enablers. *Proceedings of the 4th Conference "Speech Technology and Human-Computer Dialogue – SpeD 2007"*, Iaşi, published in *Advances in Spoken Language Technology*, Publishing House of the Romanian Academy, Bucharest, pp. 89-98, 2007, ISBN 978-973-27-1516-1

8. Burileanu, D., **Ungurean, C.**, "Metodă de predicție a accentului lexical în sinteza vorbirii pornind de la text în limba română", cerere de *Brevet de invenție* depus şi înregistrat la OSIM cu nr. A/01246 din 29.11.2010

9. Burileanu, D., **Ungurean, C.** "Metodă de inserare automată a semnelor diacritice pentru sinteza vorbirii pornind de la text în limba română în aplicații ce au ca suport rețele de date", cerere de *Brevet de invenție* depus şi înregistrat la OSIM cu nr. A/01247 din 29.11.2010