APLICACIONS DE RECONEIXEMENT DE LA PARLA: SISTEMES DE RESPEAKING III

Memòria del projecte de final de carrera corresponent als estudis d'Enginyeria Superior en Informàtica presentat per Néstor Pérez Rivero i dirigit per Javier Serrano García.

Bellaterra, Juny de 2010

El firmant, Javier Serrano García , professor del Departament d'Enginyeria de la Informació i de les Comunicacions de la Universitat Autònoma de Barcelona

CERTIFICA:

Que la present memòria ha sigut realitzada sota la seva direcció per Javier Serrano García

Bellaterra, Juny de 2010

Firmat: Javier Serrano García

I helped Planet Express wreck a nice beach
(I helped Planet Express recognize speech)

Agraïments

En aquest punt vull ser just, per tant, començaré agraint l'ajuda que m'ha donat l'Héctor. Sense la seva paciència i consells hauria tingut molts problemes per desenvolupar aquest projecte. També vull agrair la llibertat que m'ha donat el meu director de projecte, Javier, per treballar al meu ritme sense pressionar-me. No m'oblidaré del meu cap a Telefónica Investigación y Desarrollo, Xavier, per ajudar-me a compaginar la feina de becari amb la universitat.

I per últim, vull expressar el meu deute amb tots aquells que sempre han estat al meu costat durant aquests anys d'universitat que conclouen amb el present projecte: els meus pares, la meva germana i els meus amics de tot arreu. Gràcies per suportar-me tot aquest temps d'estrès acadèmic.

Índex

Índex de figures

Capítol 1

Introducció

El llenguatge parlat és sense cap tipus de dubte el mètode de comunicació més natural, intuïtiu i eficient pels éssers humans. Durant dècades, la idea d'interaccionar amb màquines com si de persones es tractés, ha fascinat a enginyers, científics i, sobretot, a escriptors de ciència ficció.

Les tecnologies de la parla, enteses com el conjunt de disciplines científiques i d'enginyeria que s'ocupen del tractament de la parla per part de les màquines en sentit general, tenen un interès cada vegada més gran per part de la comunitat científica i la societat en general. Entre aquestes tecnologies, el reconeixement automàtic de la parla presenta un dels camps més atractius d'investigació i desenvolupament.

Fins fa pocs anys, els sistemes de reconeixement de mitjana i gran complexitat, estaven disponibles casi únicament com a prototips de laboratori. En la actualitat, les prestacions dels ordinadors comercials i l'accés a eines com HTK o SRILM ha permès l'explotació d'aquesta àrea per una gran quantitat de gent i organitzacions.

Les aplicacions més comunes que s'han creat són [8]:

- Control per comandes: La seva funció és donar ordres a un ordinador. Són sistemes que reconeixen un vocabulari molt reduït i específic, cosa que incrementa el seu rendiment.

- Telefonia: Utilitzat per navegar per menús sense utilitzar les tecles del telèfon.

- Sistemes portàtils: Sistemes de petita mida com poden ser els telèfons mòbils que incorporen la marcació per veu.

- Sistemes dissenyats per discapacitats: Sistemes que permeten a la gent amb problemes de mobilitat deixar d'utilitzar teclats o persones amb sordesa rebre trucades telefòniques.

- Sistemes de re-speaking: Molt utilitzat en el món de la televisió, sobretot des de la implantació de la Televisió Digital Terrestre. Es tracta de la incorporació de subtítols "on live".

- Dictat automàtic: Ús més comú dels reconeixedors de la parla. Al mercat podem trobar molts editors de text basats en el reconeixement de la parla que ens permeten escriure sense utilitzar el teclat. En aquests projecte desenvoluparem un sistema de dictat automàtic senzill que ens mostrarà una aplicació pràctica del reconeixedor de la parla que crearem.

1.1 Objectius

El principal objectiu d'aquest projecte és mostrar el desenvolupament d'un reconeixedor de la parla i la seva aplicació als sistemes de dictat. Bàsicament, es tracta de crear una eina computacional capaç de processar el senyal de veu emesa per l'ésser humà i reconèixer la informació continguda en aquesta. Després, una aplicació recull aquesta informació i la introdueix en un processador de text. D'aquesta manera tindrem un programa que ens permetrà escriure parlant i corregir les errades del reconeixedor.

S'ha de ser molt conscient de les limitacions que tindrà un reconeixedor de la parla desenvolupat amb tants pocs recursos humans i temporals. Per tant, no ens fixarem un objectiu inicial determinat en quant al percentatge d'encert del reconeixement. La intenció és obtenir un reconeixedor de la parla en català, veure amb detall com és el seu procés de creació, determinar quines són les seves limitacions, per quin motiu hi són i com les podríem superar.

Actualment no es troben masses sistemes reconeixedors de parla en català. Això és degut a causes molt diverses que no entrarem a discutir. Hem optat doncs

per aportar el nostre petit gra de sorra en aquest aspecte, cosa que ha suposat un repte donat que la obtenció de textos en català és més complicada que en castellà o anglès.

1.2 Estructura de la memòria

Aquesta memòria està estructurada en 8 capítols, el primer dels quals l'estem llegint ara i és la introducció al tema principal del projecte i l'objectiu del mateix.

En segon lloc, hi ha el capítol que tracta del capítol dedicat al reconeixedor de la parla en general, on trobarem una explicació sobre que és, la seva base matemàtica, el seu funcionament i una explicació dels seus tres components: model acústic, model de llengua i el lexicon. Un cop explicat el reconeixedor complet, trobem el capítol sobre la part del projecte que més temps i esforç ha comportat: el procés de creació que hem seguit per construir el model de llengua i les proves realitzades. Aquí és on trobarem l'aportació que realitza el nostre treball de recerca a la construcció de models de llengua.

Un cop obtingut el reconeixedor, trobem quatre capítols dedicats a l'aplicació de dictat automàtic (Thot) que hem desenvolupat. Concretament, en el quart fem una explicació de l'anàlisi de requisits, en el cinquè hi trobem el disseny, el sisè ens il·lustra els aspectes més rellevant de la implementació i l'últim d'aquesta par mostra les proves realitzades sobre l'aplicació.

Per acabar, tenim els dos últims capítols de la memòria. El penúltim extrau conclusions sobre tot el que hem vist al projecte i com podríem millorar els resultats que hem anat obtenint i l'ultim exposa la bibliografia emprada en la elaboració de tot el projecte.

Capítol 2

Reconeixedor de la parla

2.1 Introducció

Com ja hem dit abans, un reconeixedor de la parla és un dispositiu que automàticament ens transforma una senyal acústica en text. En aquests dispositius, generalment s'assumeix que el senyal acústic és una realització d'un missatge codificat en seqüències d'un o més símbols [1]. Aquest fet es pot observar a la figura 2.1.

Per a reconèixer aquests símbols, l'ona contínua de so és converteix en una seqüència de vectors de paràmetres equidistants. També s'assumeix que aquesta seqüència de vectors de paràmetres és una representació exacta de l'ona de so perquè, en el temps que cobreix un vector (en el nostre cas 25ms), l'ona de so pot ser considerada com estacionària. Tot i que això no és estrictament veritat, és raonablement aproximat. Normalment, representacions paramètriques d'ús comú es basa en espectres suavitzats o coeficients de predicció lineals més altres representacions derivades d'aquesta.

La funció dels reconeixedors és mapejar seqüències de vectors de parla amb seqüències dels símbols subjacents. Dos problemes fan aquesta tasca difícil. El primer, el mapeig de símbols a parla no és un-a-un ja que diferents símbols subjacents poden donar lloc a similars sons de la parla. A més, hi ha grans variacions en la parla tals com la varietat de l'emissor, l'entonació, l'ambient, etc.

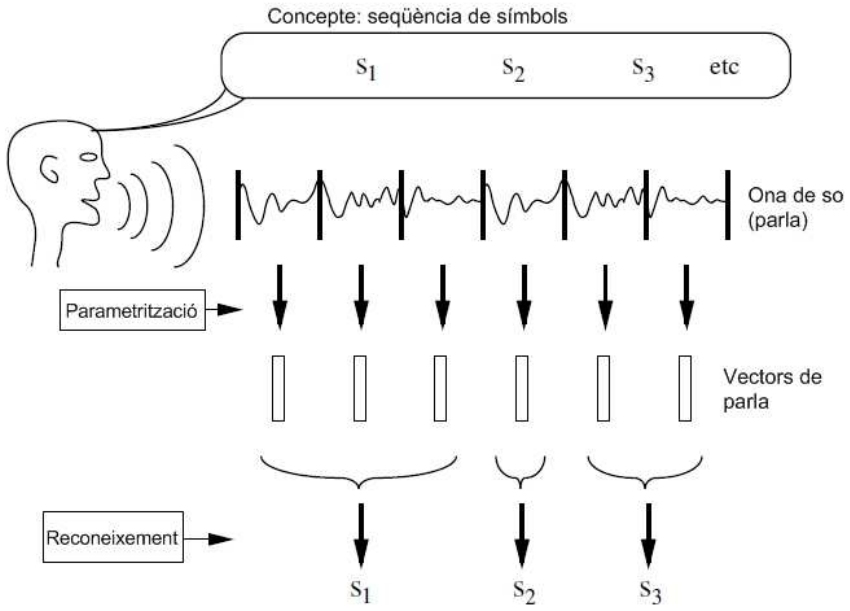

Figura 2.1: Codificació i descodificació

En segon lloc, els límits entre símbols no poden ser identificats explícitament de la ona de so. Per tant, no és possible tractar aquesta ona de so com una seqüència de patrons estàtics encadenats.

Aquest últim problema, no saber el límit de les paraules, pot ser evitat restringint la tasca a un reconeixement aïllat de paraules. Això implica que l'ona de so correspon a un únic símbol subjacent (per exemple, una paraula) triat del vocabulari fixe. Tot i el fet que aquest problema simple és una mica artificial, té un gran rang d'aplicacions pràctiques.

Cada paraula pronunciada es representa mitjançant una seqüència de vectors o observacions O, definit com $[O = o_1, o_2, .., o_T]$ on o_t és un vector observat al temps t. El problema de reconeixement aïllat de paraules pot ser considerat com el càlcul de $argmax_i P(w_i|O)$ on w_i és la ièssima paraula del vocabulari. Aquesta probabilitat no és directament calculable a no ser que usem la Regla de Bayes (ja que hauríem d'obtenir la probabilitat d'una elocució donades totes les possibles

seqüències d'observació) [1].

$$argmaxP(w_i|O) = argmax(\frac{P(O|w_i)P(w_i)}{P(O)}) \tag{2.1}$$

$P(O)$ és la probabilitat de la seqüència d'observació acústica. És difícil d'estimar perquè no se sap quina és la seqüència d'observació de totes les possibilitats dins del domini, però se sap que s'està examinant la mateixa seqüència d'observació O per cada elocució en potència i, per tant, és tindrà la mateixa $P(O)$ per a totes les elocucions possibles. Al ser el mateix denominador per cada elocució candidata, l'eliminem perquè no influirà en el càlcul del màxim. De manera que l'expressió ens queda:

$$argmaxP(w_i|O) = argmaxP(O|w_i)P(w_i) \tag{2.2}$$

Així, la paraula més probable dependrà de $P(O|w_i)$ i de $P(w_i)$ (del seu producte, concretament). $P(O|w_i)$, la probabilitat d'observació, es calcula a través del model acústic i $P(w_i)$, la probabilitat a priori, es calcula a través del model de llengua.

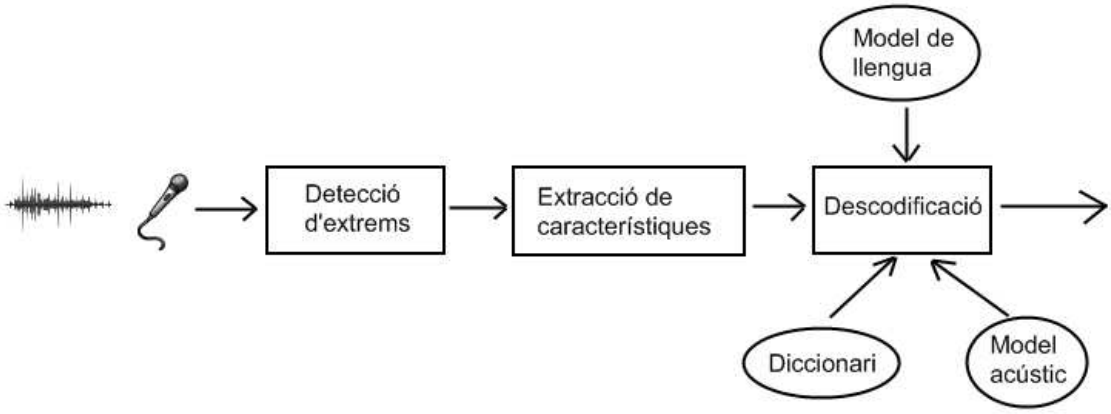

Figura 2.2: Esquema general

La figura 2.2 ens mostra com es relacionen aquests dos elements, junt amb el diccionari fonètic, en el reconeixedor de la parla. El funcionament d'un

sistema reconeixedor de la parla compren dues etapes: una d'entrenament i una de reconeixement [3]. Durant l'etapa d'entrenament, se li proporciona al sistema certa quantitat de pronunciacions que volem que el sistema tingui o "memoritzi". El que fa el sistema és emmagatzemar les propietats d'un conjunt i no les pronunciacions en si. Durant l'etapa de reconeixement, el sistema identifica la pronunciació que més probablement s'assembla a les pronunciacions que hi ha a la memòria del sistema.

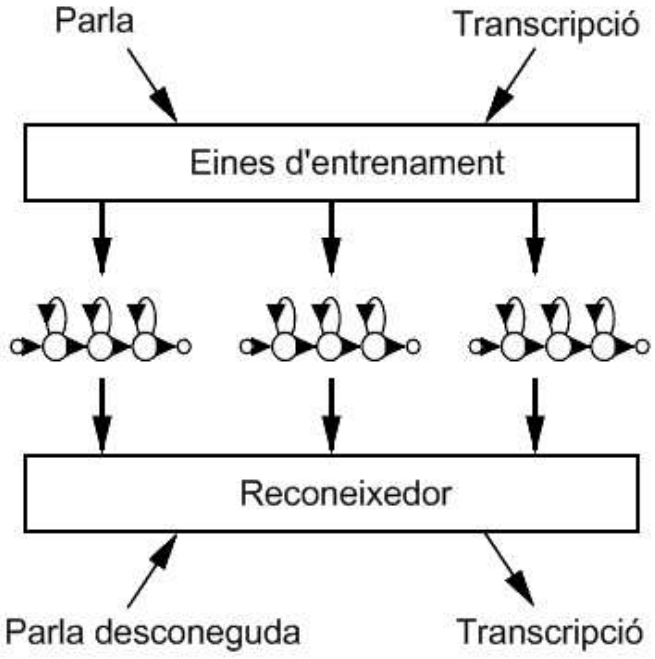

Figura 2.3: Entrenament i reconeixement

La figura 2.3 ens mostra un petit esquema del funcionament d'un reconeixedor de la parla basat en models ocults de Markov, que són els que hem utilitzat nosaltres. El motiu és molt simple, si assumim un model paramètric de producció de paraules com els Models de Markov, la estimació de les dades és possible ja que el problema d'estimació de $P(O|w_i)$ es converteix en un problema més simple: estimar els paràmetres del model de Markov.

Un model de Markov simple (o autòmat de pesos) és especificat per un conjunt d'estats, un conjunt de probabilitats de transició, un estat inicial, un, o més, estats

finals i un conjunt de probabilitats d'observació.

Un procés estocàstic es defineix com un conjunt de variables aleatòries X_t la distribució del qual varia respecte a un paràmetre, normalment, el temps. La variable t pren valors d'un subconjunt de nombres enters o reals no negatius. Les variables aleatòries X_t prenen valors en un conjunt anomenat espai d'estats.

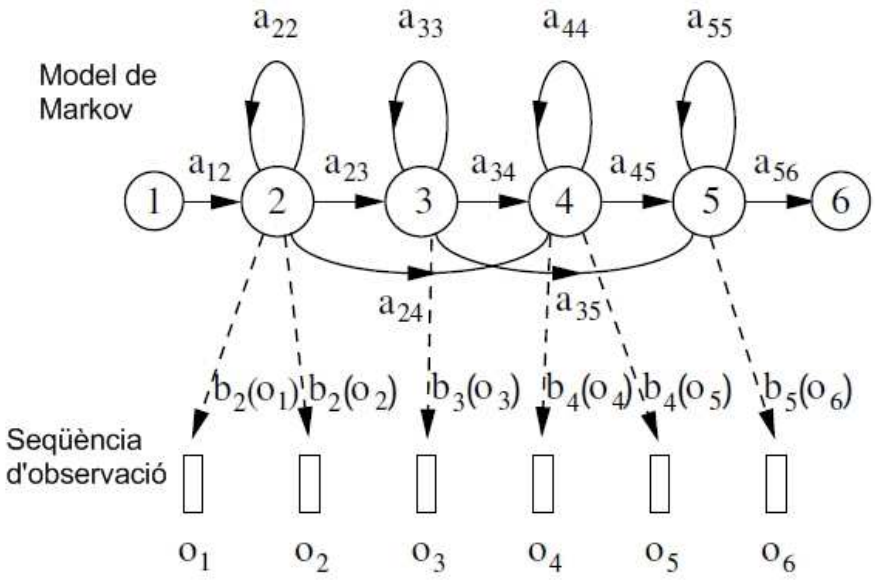

Figura 2.4: HMM

Un model de Markov ocult (HMM) és un procés estocàstic on el canvi d'estat només depèn de l'estat actual i no dels anteriors; és una màquina d'estats finits en la qual el següent estat depenent únicament de l'estat actual i associat a cada transició entre estats es produeix un vector d'observacions o paràmetres. Es diu ocult degut a que porta associat dos processos: un ocult (no observable directament) corresponent a les transicions entre estats, i un altre observable (directament relacionat amb el primer), les realitzacions del qual són els vectors de paràmetres que es produeixen des de cada estat i que formen el patró a identificar [3].

Per a crear el nostre reconeixedor de la parla hem utilitzat HTK. Es tracta d'un conjunt d'eines per construir HMMs desenvolupat pel Speech Vision and Robotics

Group del departament d'enginyeria de la Universitat de Cambridge i el seu ús és totalment gratuït. Tot i que el nucli de HTK és de propòsit general, està bàsicament dissenyat per construir eines de processament de la parla basades en models ocults de Markov, en concret reconeixedors. Així, molta de la infraestructura de suport a HTK està dedicada a aquesta tasca [14].

2.2 Model acústic

El model acústic captura les propietats acústiques de la senyal d'entrada, obté un conjunt de vectors de característiques que després compara amb un conjunt de patrons que representen símbols d'un alfabet fonètic i dóna els símbols que més s'hi assemblen. Aquesta és la intuïció del procés matemàtic probabilístic anomenat model ocult de Markov, que com ja hem dit, hem utilitzat per crear el nostre reconeixedor. A més, el model acústic conté el diccionari de pronunciació en el qual es defineix la descomposició de les paraules en unitats més petites que corresponen a les unitats dins de l'alfabet fonètic definit.

L'anàlisi acústic amb el qual extraiem els vectors de característiques s'ha realitzat mitjançant la tècnica MFCC (Mel Frequency Ceptral Coefficients). Per calcular aquests coeficients o característiques espectrals la tècnica consisteix en els següents passos (segons L.Rabiner i Juang 1993):

1. Es divideix el senyal en fotogrames de cert nombre de milisegons (25).
2. Per cada fotograma s'obté l'espectre d'amplitud.
3. Es calcula el logaritme de l'espectre.
4. Es converteix a l'espectre de Mel.
5. Es calcula la transformada del cosinus discret.

Inicialment partim del monofons (fonemes) i obtenim un conjunt de 44 models. A partir d'aquests models crearem els trifons. Primer s'analitzen les transcripcions de tots els arxius del corpus d'entrenament i es formen tots els trifons "cross-word" que hi apareixen. Això vol dir que no només agafem els fonemes de tres en tres per cada paraula, sinó que també "unim" les paraules utilitzant els últims i els primers fonemes de les paraules. Si, per exemple,

tinguéssim una gravació com “trucada per número”, els trifons que obtindríem són:

sil-t+r, t-r+u, r-u+k, u-k+a, k-a+D, a-D+@, D-@+p, @-p+@, p-@+rr, @-rr+n, rr-n+u, n-u+m, u-m+@, m-@+r, @-r+u, r-u+sil

Veiem el procés que hem anomenat abans: el trifon D-@+p agafa els últims dos fonemes de “trucada” i el primer de “per”, és a dir, “dap”.

En aquest punt, enlloc de tenir 44 models, en tenim molts més, concretament 10887 models. Són moltes dades, tenint en compte que cada model té 3 estats amb matrius de transició, vectors de mitjanes i covariàncies, i més endavant diverses components gausianes. Per treballar amb tot això el que fem és “lligar” les matrius de transició. Això consisteix en aprofitar que molts trifons tenen sons en comú. A nivell molt bàsic podem dir que “traiem factor comú”, de manera que els models que tenen sons en comú comparteixen les matrius de transició. D’aquesta manera ens estalviem repetir moltíssimes dades.

No és massa difícil veure com amb els trifons que només apareixen al corpus, no obtenim tots els possibles trifons que es poden produir a la llengua catalana. Per solucionar això, sintetitzem totes les possibles combinacions de trifons que es poden produir amb els 44 fonemes que teníem inicialment. Això ens permet tenir models suficients per utilitzar qualsevol diccionari. Ara ens trobem amb un altre problema, no tenim dades d’entrenament per tots els trifons sintetitzats. Per solucionar aquest problema, tornem a “lligar” aquests models però aquest cop no només les matrius de transició, sinó tots els paràmetres, a algun dels models que si es troben al corpus d’entrenament. Es realitza mitjançant arbres de decisió (tied-state triphones). Bàsicament, es basa en fer preguntes sobre els contexts esquerra i dret de cada trifon. L’arbre de decisió intenta trobar els contexts que tenen la diferència acústica més gran i que, per tant, distingeixen agrupacions.

D’aquesta manera hem obtingut 10887 models físics on la resta, “lògics, estan lligats a algun dels físics. Per últim cal dir que s’han dividit les gausianes en diverses components per tal que els models s’adaptin millor als sons. Concretament vam rebre dues versions dels models acústics: amb 20 i 6 components gausianes. Vam fer proves utilitzant 20 components gausianes però

els resultats no eren tan satisfactoris com amb 6 components, per tant, ens vam quedar amb la segona versió.

Per aplicar la teoria dels HMMs al reconeixement de veu, es representa cada paraula del vocabulari del reconeixedor com un conjunt de models de trifons que acabem de calcular a la fase d'entrenament. Per fer això, s'assumeix que durant la pronunciació d'una paraula, l'aparell fonador només pot adoptar un nombre finit de configuracions articulatòries o estats, i que des de cada un d'aquests estats es produeixen un o més vectors d'observació les característiques espectrals dels quals dependran (probabilísticament) de l'estat en el que hagi sigut generat. Vista la generació de la paraula, les característiques espectrals de cada fragment de senyal depenen de l'estat actiu a cada instant, i les de l'espectre del senyal durant la pronunciació d'una paraula depenen de la funció de transició entre estats.

Donada una seqüència d'observació $[O = o_1, o_2, .., o_T]$ i creats els models acústics, $P(O|w_i)$ es calcularia obtenint totes les possibles seqüències d'estats de longitud T. A cada temps $t = 1, 2, ..., T$ es tenen N possibles estats arribables, per tant N^T operacions. L'algorisme forward és una manera eficient per calcular aquestes probabilitats ja que calcula la probabilitat de la seqüència d'observació parcial $o_1, o_2, ..., o_t$ a l'estat i fins al temps t, donat el model de Markov. D'aquesta manera el nombre d'operacions es redueix a N^2T [9].

Per a triar la seqüència que sigui òptima en algun sentit s'utilitza l'algorisme Viterbi [9]. Es tracta d'una simplificació de l'algorisme forward on només es consideren les probabilitats acumulades als camins òptims, és a dir, només es considera la seqüència d'estats que resulta de maximitzar les expressions de l'algorisme forward [7].

Per ajustar els paràmetres del model per així poder maximitzar $P(O|w_i)$ s'utilitza l'algorisme Baum-Welch que és un cas especial de l'algorisme de maximització de la probabilitat. Permet entrenar les probabilitats de transició entre els estats i les probabilitats que emet cada estat de manera que iterativament utilitza els algorismes forward i backward.

Aquí resideix l'elegància i el poder del marc de treball dels HMM. Donat un conjunt d'entrenament corresponent a un model concret, els paràmetres d'aquest

model poden ser determinats automàticament per un robust i eficient procediment de re-estimació.

Actualment la majoria dels reconeixedors en funcionament es basen en tècniques estadístiques perquè són les que requereixen menys memòria física i tenen un millor temps de resposta. Tot i això, necessiten una fase d'entrenament més lenta i costosa però no té massa importància donat que només es realitza una vegada.

2.2.1 Corpus d'entrenament

Per entrenar els models acústics hem utilitzat la base de dades de consum creada per la UPC i ATLAS. La Universitat Politècnica de Catalunya (UPC) i Applied Tecnologies on Language and Speech (ATLAS) han enregistrat i processat una gran base de dades oral dins del projecte finançat per la Generalitat de Catalunya.

El corpus conté la veu de 550 informants i cada un enregistrat en una sessió. Els informants han estat seleccionats seguint el criteri:

- Equilibri entre dialectes: un mínim de 97 informants per regió.

- Equilibri pel que fa al sexe.

- Equilibri pel que fa a l'edat: tres grups d'edat 16-30, 31-45, 46-60 estan igualment representats a la base de dades.

S'han definint 5 ambients:

- Oficina: 200 persones. Una oficina, és a dir, una habitació on la gent treballa amb escriptoris, normalment o probablement amb un ordinador.

- Entreteniment (ambient domèstic): 75 persones. Sala d'estar, és a dir, una habitació amb alguns mobles i llocs on la gent hi pot seure. Una taula, un televisor o algun equip de so hi són presents. Enlloc d'una sala d'estar, una habitació d'hotel també és possible.

- Cotxe: 75 persones. Vehicle per 4 o 5 passatgers.

- Lloc public: 200 persones. Un vestíbul gran o espai obert. El vestíbul ha de tenir almenys 3 parets i un sostre; amb gent més o menys ocupada però no massa silenciosa. Un espai obert no té parets i tampoc un sostre tancat. Evidentment, pot estar delimitat per les parets dels edificis del voltant.

Entre tots els escenaris d'enregistrament, els micròfons de "mitja distància" i "llarga distància" estan encarats a l'informant. La persona enregistrada s'asseu a una cadira durant tota la sessió. Els dos micròfons de 'curta distància' estan muntats sobre el mateix informant i els micròfons de 'mitja' i 'llarga distància' estan situats a una alçada mitja de 1.2 metres, permetent una desviació de 50 cm. Pel que fa a les propietats de reverberació d'un lloc, la posició dels informants relativa a objectes reflectors, com les parets, és important. Les etiquetes de posició diferencien en categories aquestes posicions de forma genèrica. Per cada lloc d'enregistrament i posició específica, la resposta impulsional de l'habitació és mesurada. Per cada sessió, un nivell de soroll és mesurat. [13]

2.2.2 Proves del model acústic

Per provar el model acústic hem creat una gramàtica lliure de context que simula les instruccions que entendria una possible casa domòtica. Compren ordres del tipus "obre la porta" o "tanca la finestra". És molt simple i la fem servir com a model de llengua per a poder provar el model acústic. Diverses persones han gravat frases que compren aquesta gramàtica i que hem utilitzat com a test. Utilitzant les eines HDecode i HResults d'HTK, que explicarem amb deteniment a les proves del reconeixement, hem obtingut els següents resultats.

```
====================== HTK Results Analysis =======================
  Date: Fri Feb 26 15:12:54 2010
  Ref : testref.mlf
  Rec : recoutNormal.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=99.38 [H=159, S=1, N=160]
WORD: %Corr=99.72, Acc=99.72 [H=718, D=0, S=2, I=0, N=720]
===================================================================
```

Figura 2.5: proves model acústic

És evident que les condicions de gravació del corpus d'entrenament, explicades anteriorment, i les condicions de les probes són diferents. Tot i això, la figura 2.5 ens mostra uns resultats molt bons. De les 160 frases que s'han analitzat,

el 99.38% han sigut correctament corregides. Aquestes frases estaven formades per 720 paraules en total el 99.72% han estat ben reconegudes.

2.3 Lexicon

El lexicon, o diccionari de pronunciació, especifica l'adequada seqüència de sons (representats mitjançant un conjunt de símbols) que componen cada paraula. Aquests símbols els hem format usant un alfabet fonètic, concretament del SAMPA (Speech Assessment Methods Phonetic Alphabet).

La font per crear el diccionari ha sigut el corpus d'entrenament del model de llengua. El que hem fet ha sigut agafar les paraules del corpus que surten dues o més vegades i ordenar-les alfabèticament. Hem desestimat les que només apareixien una vegada perquè són molt propenses a contenir errors. Com més paraules tingui el diccionari (i el model de llengua) menys restrictiu serà el reconeixedor. En el nostre cas hem obtingut un vocabulari de casi 50.000 paraules.

Un cop tenim totes les paraules del diccionari, hem creat el model de llengua. Per poder provar tot el reconeixedor cal que l'ampliem amb la transcripció fonètica de cada paraula. Per obtenir aquesta transcripció hem utilitzat l'aplicació Segre.

Es tracta d'un transcriptor automàtic per totes les variants de la llengua catalana. Se li passa un text ortogràfic i dóna una transcripció fonètica (cadena d'al·lòfons amb silůlabització i accentuació). El transcriptor està basat en regles externes que es subministren al programa en forma de fitxers. La versió que es distribueix del programa inclou els fitxers de regles vàlides pel català central (que és el que hem utilitzat nosaltres), els dialectes nord-occidentals, balear i valencià, encara que les regles per aquests tres últims no són tan detallats.

Segre va ser desenvolupat al Centre TALP de la UPC amb la col·laboració del Laboratori de Fonètica del Departament de Filologia Espanyola de la Universitat Autònoma de Barcelona, i també del Grup de Recerca en Variació Dialectal del Departament de Filologia Catalana de la Universitat de Barcelona, en el marc del CREL (Centre de Referència en Enginyeria Lingüística) de la Generalitat de

Catalunya [4].

Cal tenir en compte que fins ara teníem tots els fitxers en codificació UTF-8 i Segre treballa en ISO-8859-15. Per tant, cal recodificar el diccionari usant el script utf2iso.sh. Ara ja només cal corregir alguns detalls per acabar d'encaixar amb el model acústic i ajuntar les paraules i les transcripcions en un nou fitxer.

2.4 Model de llengua

El model de llengua conté les propietats lingüístiques del llenguatge i ens dona la probabilitat a priori d'una seqüència de paraules. Endevinar la següent paraula, anomenat predicció de paraula, $P(w_i)$, és una subtasca essencial en el reconeixedor [2]. Això està relacionat amb el problema de calcular la probabilitat d'una seqüència de paraules. Els algorismes que assignen probabilitats a un enunciat poden ser utilitzats per assignar la probabilitat a la següent paraula en un enunciat incomplet o al revés.

Aquest model de predicció de la paraula s'anomena n-grama. Definim un n-grama com una seqüència de n símbols, en el nostre cas, paraules. Un model de llengua basat en n-grames s'utilitza per predir cada símbol en una seqüència donats els $n - 1$ símbols anteriors. En el nostre cas particular, treballarem amb unigrames, bigrames i trigrames, que tindran en compte cap, la última i les dues últimes paraules anteriors, respectivament. Es basa en la idea de que la probabilitat d'un n-grama específic que apareix en un text desconegut pot ser estimada a partir de la freqüència de les seves aparicions en un text d'entrenament. Anomenem aquest text corpus d'entrenament.

La construcció d'un model de llengua basat en n-grames consta de tres parts diferenciades tal i com podem veure a la figura 2.6. En primer lloc, el corpus d'entrenament és escanejat, es compten els seus n-grames i es guarden en arxius de grames. En segon lloc, algunes paraules són mapejades en una classe anomenada fora de vocabulari, o un altre tipus de classe, i després s'utilitzen els arxius de grames per calcular les probabilitats dels n-grames. Aquestes són, bàsicament, el model de llenguatge i es guarden a l'arxiu de model de llenguatge.

En tercer lloc i últim, la qualitat del model de llenguatge es pot estimar calculant la perplexitat en un test desconegut. Bàsicament, es pot dir que un model de llenguatge és millor com més baixa sigui la perplexitat obtinguda. Més endavant tenim les proves que hem realitzat on es calcula la perplexitat [1].

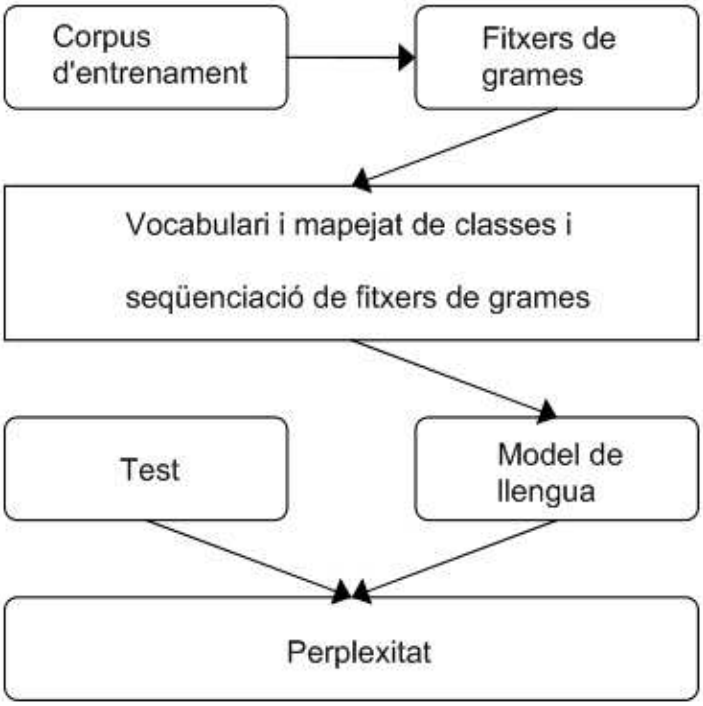

Figura 2.6: Esquema general del model de llengua

Tot i que el principi bàsic dels models de llenguatges basats en n-grames és molt simple. A la pràctica, hi ha més n-grames potencials dels que es poden recollir a l'hora de construir una estimació robusta de freqüències. A més a més, per a l'aplicació real com els reconeixedors de la parla, la utilització d'un corpus d'entrenament finit fa difícil generar un model de llengua que encaixi bé amb una gran varietat de textos de prova. Per exemple, un model de llengua entrenat amb textos dels diaris anirà molt bé per predir textos periodístics però no pas per a cartes personals o converses comunes entre persones. Una dificultat final és que el vocabulari d'un model de llenguatge basat en n-grames és finit i fix en el temps de construcció. D'aquesta manera, si el model de llenguatge està basat en paraules

com és el nostre cas, només podrà predir les que estiguin en el seu vocabulari i per afegir-ne més, s'haurà de reconstruir tot el model de llenguatge.

Com ja hem dit, els n-grames són molt dependents del corpus d'entrenament, és molt probable que moltes combinacions de paraules apareguin poques vegades, i tinguin una probabilitat molt petita, o que ni tan sols apareixin, i tinguin probabilitat zero. La tasca d'avaluar aquestes probabilitats i assignar-lis valors diferents de zero s'anomena "smoothing" o suavitzat.

Els algorismes de suavitzat proporcionen una millor manera d'estimar la probabilitat de n-grames que no ocorren ja que els hi agreguen una mica de la massa de probabilitat. En la creació del nostre model de llengua hem utilitzat Good-Turing discounting.

Good-Turing discounting es basa en re-estimar la quantitat de massa de probabilitat per assignar-la als n-grames amb valor cero o molt baix, observant el nombre de n-grames amb valor alt [2][9].

Capítol 3

Recerca

Després de l'explicació completa d'un reconeixedor de la parla i com es va obtenir i testejar el model acústic, aquest capítol mostra la feina de recerca realitzada en aquest projecte. En concret l'origen del corpus d'entrenament utilitzat pel model de llengua, el procés de creació d'aquest model i les proves realitzades tant al model de llengua com al reconeixedor sencer.

3.1 Corpus d'entrenament

La disponibilitat d'un corpus per la creació de models de llengua en Català és encara molt limitada i és el component més important a l'hora d'obtenir el model de llenguatge, ja que d'aquí obtindrem les probabilitats dels n-grames i el diccionari. Com més grandària tingui, més varietat de paraules hi haurà i més combinacions d'aquestes. A més, com ja hem dit anteriorment, ens condicionarà el tipus de reconeixedor que tindrem. Si utilitzem, per exemple, llibres de medicina, obtindrem un reconeixedor molt particular que no tindrà bons resultats fora d'aquest àmbit. Per tant, aquests són els dos factors principals a l'hora d'escollir l'origen del corpus d'entrenament: la grandària i el tipus de llenguatge.

Per solucionar el problema d'obtenir un corpus vam tenir dues idees. La primera va ser obtenir-lo utilitzant la Viquipèdia ja que aquesta conté moltíssimes paraules en català [6]. A priori semblava una bona opció però dos inconvenients

ens van fer descartar-la. En primer lloc, el tipus de llenguatge que utilitza. Un model de llenguatge entrenat amb aquest corpus estaria molt restringit a aquest tipus de llenguatge. A no ser que volguéssim crear una aplicació per escriure articles per la Viquipèdia, no tenia massa sentit. El segon inconvenient va ser la enorme tasca de neteja. La Viquipèdia conté moltíssims errors i símbols que no serveixen per a crear un corpus i que, per tant, cal netejar. Estem parlant de coses com les faltes d'ortografia, sigles, les abreviatures, els nombres romans, els nombres, lletres d'altres abecedaris (grec, ciríl·lic...), símbols i formules matemàtiques. Es tracta d'una quantitat de feina titànica que necessitaria d'uns recursos humans i temporals impressionants i que òbviament no teníem per la realització del present projecte.

La segona opció, soluciona el problema del tipus de llenguatge i la tasca de neteja i preparació del corpus. Vam optar per obtenir el corpus d'entrenament a partir de 10 novel·les:

- La vall dels cavalls de J. M. Auel
- El codi Da Vinci de Dan Brown
- Els homes que no estimaven a les dones de Stieg Larsson
- L'església del mar de Ildefonso Falcones
- Kafka a la bora d'Haruki Murakami
- La mà de Fàtima de Ildefonso Falcones
- Els Pagesos de Josep Pla
- Els pilars de la terra de Ken Follet
- La reina al palau dels corrents d'aire de Stieg Larsson
- Harry Potter i les relíquies de la mort de J. K. Rowling

Aquesta opció presenta molts punts positius. Es tracta d'una redacció acurada i revisada per professionals, té un llenguatge molt més planer i comú que la Viquipèdia i l'aparició de símbols que no ens interessen és molt més reduïda. Es tracta de la nostra aportació personal a l'àmbit dels models de llengua.

Tot i això no és perfecte i té els seus inconvenients. La seva dimensió és molt menor. Per obtenir un corpus proper als dos milions i mig de paraules, hem hagut d'utilitzar 10 novel·les. Ja ens podem fer una idea de la quantitat que hauríem de

fer servir per obtenir 20 o 100 milions de paraules. Hem creat una sèrie de scripts en Perl que han convertit les novel·les en un corpus d'entrenament llest per ser utilitzat. La seva feina ha sigut:

- Conversió de dígits: passar-los de números a lletres. Utilitzant el mòdul Lingua::CA::Numeros de Perl i adaptat al català per Héctor.

- Conversió de nombres romans: traduir a lletra nombres com els segles o els reis.

- Expansió d'abreviacions i sigles: escriure senceres les que més apareixen.

- Corregir paraules partides pel canvi de pàgina (separades per guió).

- Eliminació caràcters estranys fora de l'alfabet llatí.

- Corregir errors produïts pel pas de pdf a txt.

- Eliminació signes de puntuació.

- Substitució de símbols com +, @, =, °i % per més, arrova, igual, graus i per cent.

- Flanquejar cada frase amb les etiquetes <s> i </s>, com ens demana HTK. Les etiquetes d'inici i final de frase es poden canviar. Tot i que hi ha una frase per cada línia, això no és obligatori però ho deixa més clar a l'hora de llegir. També ho hem passat tot a minúscules ja que el model acústic ens ho exigeix així.

3.2 Creació model de llengua

Aquesta secció descriu els passos concrets que hem seguit per construir el nostre model de llengua utilitzant les eines proporcionades per HTK. El model es construeix des de zero considerant que el corpus d'entrenament i el diccionari ja els tenim preparats tal i com hem explicat anteriorment. Totes les següents comandes es troben recollides en un script bash per poder crear el model en una sola crida i poder realitzar canvis ràpidament.

El primer pas és utilitzar l'eina LGPrep per escanejar el corpus i produir un conjunt preliminar de fitxers n-grames ordenat. Els guardarem a la carpeta modelo.0, per tant, cal crear-la.

$ mkdir model.0

Les eines d'HTK mantenen un mapeig de paraules acumulatiu on cada paraula nova és afegida i té assignada un identificador únic. Això vol dir que es poden afegir nous arxius d'n-grames sense haver de reconstruir els que ja tenim, sempre que comencem pel mateix mapeig de paraules, assegurant-nos així que cada identificador segueix sent únic. Per preparar el mapeig de paraules inicial, que es trobarà buit, utilitzem l'eina LNewMap. Necessita el nom que usarem internament en el mapeig de paraules i l'arxiu on hi escriurem. L'opció -f ens afegeix un camp addicional que ens indicarà quantes vegades s'ha trobat cada paraula fins a la data.

$ LNewMap -f WFC Model empty.wmap

Ara ja estem preparats per processar el corpus.

$ LGPrep -T 1 -a 3000000 -b 200000 -d model.0/ -n 4 -s "Model"empty.wmap corpus.txt

La opció -a estableix el nombre màxim de paraules noves que es poden trobar al corpus. Cal tenir compte d'augmentar el nombre per defecte (100.000) perquè és molt probable que sigui insuficient. La opció -b determina la mida del buffer dels n-grames intern. Els requeriments de memòria els podem calcular $membytes = (n + 1) * 4 * b$ (on n és la opció -n i b la -b).

En el nostre cas, hem optat per treballar amb bigrames i trigrames (n=3) perquè és el màxim que després ens acceptarà HDecode. La opció -T 1 ens mostra informació sobre l'execució de la comanda i la utilitzarem en totes les eines d'HTK.

Un cop ha acabat l'execució, en el directori indicat tindrem arxius del tipus gram.nombre i wmap. Aquest últim és un mapeig de paraules que ha crescut per incloure les noves paraules i identificadors assignats a elles.

Si examinem els arxius gram.* veurem que con el contingut està ordenat però els arxius entre ells no estan seqüenciats. Cadascun té un conjunt de n-grames independent de la resta. Utilitzarem l'eina LGCopy per tenir una seqüència ordenada de fitxers de n-grames on no hi haurà cap grama repetit en dos fitxers. Guardarem aquests nous fitxers en un altre directori que cal crear.

$ mkdir model.1

$ LGCopy -T 1 -o -m vocabulari/vocabulari.wmap -a 10000 -b 2000000 -d

vocabulari -w dicutf model.0/wmap model.1/data.*

Els fitxers de n-grames resultants, junt amb el mapeig de paraules, ja poden ser utilitzats per generar el nostre model de llengua per un vocabulari específic. Tot i que aquest pas que acabem de realitzar no és estrictament obligatori, és molt recomanable per evitar problemes de rendiment.

Creem el directori on guardarem el nostre model de llengua:

$ mkdir vocabulari

Si fem memòria, recordarem que el nostre diccionari no conté totes les paraules del corpus, ja que vam eliminar les que tenien una freqüència de només una aparició. Tornem a utilitzar LGCopy, aquest cop per filtrar aquestes paraules que no es troben en el diccionari i posar-les a la classe "unknown". El seu símbol per defecte és !!UNK, però pot ser canviat.

$ LGCopy -T 1 -o -m vocabulari/vocabulari.wmap -a 10000 -b 2000000 -d vocabulari -w dicutf model.0/wmap model.1/data.*

Com que hem usat la opció -o tots els n-grames que continguin OOVs seran extrets dels fitxers d'entrada i les paraules mapejades com OOV a la classe "unknown" i guardats a vocabulari/data.*. Tindrem un nou mapeig de paraules amb la classe "unknown" i les paraules que apareixen al diccionari al fitxer vocabulari/vocabulari.wmap.

Per utilitzar l'algorisme Turing-Good discounting, explicat a la secció 2.4, pel nostre model n-grama, necessitem crear una taula de freqüència de freqüències del nostre vocabulari. No és necessari però si interessant per podar baixes freqüències. Ho realitzarem a l'eina LFoF.

$ LFoF -T 1 -n 3 -f 32 vocabulari/vocabulari.wmap vocabulari/vocabulari.fof model.1/data.* vocabulari/data.*

Després de l'execució, la taula es troba guardada al fitxer vocabulari/vocabulari.fof que mostra el nombre de vegades que una paraula és trobada amb una certa freqüència. També li podem passar un arxiu de configuració (opció -C) perquè la taula ens mostri el nombre de n-grames que es quedarien fora per a cada poda que apliquéssim.

I ara ja estem en disposició de crear el nostre model de llengua. Es pot

construir en passos diferents (unigrames, bigrames i trigames) o en un de sol. Les diferències entre les dues opcions són insignificants.

$ LBuild -T 1 -c 2 1 -c 3 1 -n 3 -f TEXT vocabulari/vocabulari.wmap vocabulari/tg2-1_1 model.1/data.* vocabulari/data.*

Al fitxer vocabulari/tg2-1_1 ja hi tenim el nostre model de llengua basat en trigrames. Les opcions -c ens permeten fer podes, en aquest cas de freqüència 1. La opció -f ens crea el model en format de text que ens permet visualitzar-lo. Per defecte, es crea en codi binari propi d'HTK.

3.3 Proves del model de llengua

Per provar el model calcularem la seva perplexitat. Això no ens dirà necessàriament com de bé el model de llenguatge realitzarà una tasca en el reconeixedor, perquè no té en compte les similaritats acústiques o capritxos d'un sistema en concret, però si que ens dirà com de bé és modelat un text pel nostre model de llengua.

La perplexitat pot ser entesa com una mesura mitjana de quantes paraules equiprobables poden seguir a una paraula donada. Perplexitats petites representen models de llengua millors, tot i que només té una certa correlació amb el rendiment d'un reconeixedor de la parla ja que no té l'habilitat de prendre nota de la semblança, o no, acústica de les paraules [1].

La mida del diccionari pot ser fàcilment vista com rellevant perquè el nombre de paraules possibles decreixerà proporcionalment a mesura que decreix el diccionari, per tant la probabilitat en promig ha de créixer i la perplexitat decréixer. Com a referència d'un valor de perplexitat, podem veure la menor perplexitat que s'ha publicat del Brown Corpus (consta d'un milió de paraules en anglès americà de diferents temes i generes) és de 247 per paraula [5]. És possible aconseguir una menor perplexitat amb corpus més especialitzats ja que aquests són més previsibles.

Utilitzem l'eina HPlex, que ens proporciona HTK, que calcula la perplexitat i les estadístiques de fora de vocabulari d'un text i un model de llengua donat

[1]. De la mateixa manera que tenim el corpus d'entrenament, el text ha provar està posat per línies que comencen i acaben amb els símbols <s> i </s>. Les paraules fora de vocabulari del test es poden manejar de dues maneres diferents. Per defecte, la probabilitat de n-grames que contenen paraules fora del diccionari (OOVs) no es calcula. Això és útil per provar models de vocabulari tancat en textos que contenen OOVs. La segona manera és utilitzant la opció -u. En aquest cas, com en el primer, els n-grames que tenen OOV en els seus predecessors són descartats però la probabilitat de paraules del diccionari pot ser condicionada pel context, incloent les OOV.

Per a realitzar les proves hem utilitzat dos tests diferents. El primer conté frases extretes directament del corpus d'entrenament i el segon conté frases extretes d'una altra novel·la que no hem utilitzat per entrenar el model. Concretament es tracta de La filla de l'hostalera de Anna Tortajada. Els dos test tenen el mateix nombre de frases (30) i paraules (500). Hem calculat la perplexitat de la segona manera.

La primera part dels resultats mostra informació general sobre el nombre de frases i paraules trobades, estadístiques de paraules predites i fora del diccionari. La segona part dels resultats dóna informació explícita sobre les estadístiques del model.

La figura 3.1 ens mostra els resultats pel test de frases extretes del corpus. Podem observar diferents coses. En primer lloc, veiem com tenim dos conjunts de resultats: dels bigrames i dels trigrames. En cada cas, se'ns diuen les paraules que han sigut mapejades com desconegudes i la perplexitat. Cal recordar que vam eliminar del diccionari les que només apareixien una vegada, per això n'hi ha tan poques. Veiem com la perplexitat que obtenim dels bigrames és molt superior a la obtinguda a través dels trigrames. És comprensible que hi hagi un nombre més gran de reconeixements quan treballem amb bigrames i trigrames que només amb bigrames. La segona perplexitat és de 137. Es tracta d'una molt bona mesura. Com ja hem dit, aquesta perplexitat inclou la predicció de paraules del context que inclogui OOV.

La taula ens diu com s'han trobat els bigrames i els trigames en el test. La

```
$ LPlex -u -n 2 -n 3 -T 4 -t vocabulario/tg2-1_1 test_corpus.txt
LPlex test #0: 2-gram
Processing text stream: test_corpus.txt
mapping OOV: vagareja
mapping OOV: d'ashio
mapping OOV: s'esforça
mapping OOV: lamentar-se
perplexity 200.7712, var 10.4584, utterances 30, words predicted 466
num tokens 500, OOV 4, OOV rate 0.85% (excl. </s>)

Access statistics for vocabulario/tg2-1_1:
Lang model  requested  exact backed    n/a     mean    stdev
    bigram        466  83.3%  14.6%   2.1%    -5.30     3.23
   trigram          0   0.0%   0.0%   0.0%     0.00     0.00
LPlex test #1: 3-gram
Processing text stream: test_corpus.txt
mapping OOV: vagareja
mapping OOV: d'ashio
mapping OOV: s'esforça
mapping OOV: lamentar-se
perplexity 141.8806, var 11.9701, utterances 30, words predicted 466
num tokens 500, OOV 4, OOV rate 0.85% (excl. </s>)

Access statistics for vocabulario/tg2-1_1:
Lang model  requested  exact backed    n/a     mean    stdev
    bigram        262  70.2%  26.0%   3.8%    -6.45     3.49
   trigram        466  43.8%  21.0%  35.2%    -4.95     3.46
```

Figura 3.1: Test extret del corpus

primera columna mostra el percentatge que estaven exactament igual al corpus d'entrenament. La segona, el percentatge que s'han calculat fent "backward" dels bigrames respectius. I la tercera els que s'han trobat simplement acurtant el context de paraules.

La figura 3.2 ens mostra els resultats pel test de frases extretes de la novel·la no inclosa en el corpus que hem anomenat anteriorment. Si comparem els dos resultats obtinguts, veiem com hi ha diferències significatives. En primer lloc, hi ha més paraules mapejades com desconegudes en el segon test. Això és natural ja que ara el text no pertany al corpus i, per tant, ens trobem més paraules desconegudes pel diccionari.

En segon lloc, veiem l'evolució de la perplexitat és manté. En els dos test baixa en els trigrames respecte els bigrames però en el primer els resultats són millors que el segon. És raonable que en el test no extret del corpus hi hagi una quantitat mitja de paraules més gran que poden seguir a una paraula donada perquè el model no ha après amb aquest test. Tot i l'augment del segon cas, no es tracta d'un resultats dolents donada la mida del corpus. Les gran empreses com

```
$ LPlex -u -n 2 -n 3 -T 4 -t vocabulario/tg2-1_1 test_original.txt
LPlex test #0: 2-gram
Processing text stream: test_original.txt
mapping OOV: emocionades
mapping OOV: petronella
mapping OOV: iu
mapping OOV: mostassaf
mapping OOV: rasclet
mapping OOV: cardava
mapping OOV: rosseta
mapping OOV: renunciari
mapping OOV: llustrant

perplexity 225.1648, var 10.4992, utterances 30, words predicted 461
num tokens 500, OOV 9, OOV rate 1.91% (excl. </s>)

Access statistics for vocabulario/tg2-1_1:
Lang model  requested  exact backed    n/a     mean    stdev
    bigram        461  80.0%  18.9%   1.1%    -5.42     3.24
   trigram          0   0.0%   0.0%   0.0%     0.00     0.00

LPlex test #1: 3-gram
Processing text stream: test_original.txt
mapping OOV: emocionades
mapping OOV: petronella
mapping OOV: iu
mapping OOV: mostassaf
mapping OOV: rasclet
mapping OOV: cardava
mapping OOV: rosseta
mapping OOV: renunciari
mapping OOV: llustrant

perplexity 201.7901, var 11.6226, utterances 30, words predicted 461
num tokens 500, OOV 9, OOV rate 1.91% (excl. </s>)

Access statistics for vocabulario/tg2-1_1:
Lang model  requested  exact backed    n/a     mean    stdev
    bigram        303  69.6%  28.7%   1.7%    -6.33     3.44
   trigram        461  34.3%  30.8%  34.9%    -5.31     3.41
```

Figura 3.2: Test no extret del corpus

Microsoft treballen amb corpus de 250.000 milions de paraules. És evident com nosaltres treballem amb una molt petita part d'aquest volum dades i tot això els resultats han sigut prou bons.

3.4 Proves del reconeixedor

HDecode és l'eina de descodificació per grans vocabularis de HTK dissenyada per reconeixedors de la parla. Entre les restriccions que té, ens interessen:

- Només funciona amb trifons de paraules creuades.
- Suporta n-grames fins a trigrames.

Molt similar al HVite, la millor hipòtesis de transcripció serà generada en el Master Label File (MLF) format. Té dos modes principals de treball. El primer, i que utilitzem nosaltres, és completa descodificació utilitzant un model de llenguatge basat en n-grames. El segon és per re-puntuar l'enreixat (lattice) [1].

Per provar el nostre reconeixedor, tenim dos conjunts de proves. Aquests tenen el mateix nombre de frases (30), el mateix nombre de paraules (350), la mateixa veu de gravació (masculina) però dos orígens diferents. El primer conjunt està extret directament del corpus d'entrenament. El segon d'una novel·la que no es va utilitzar per crear el corpus i que hem mencionat abans.

Abans de poder executar HDecode, cal que preparem bé tots els elements que necessitarà per poder realitzar les proves. En primer lloc necessitarà un conjunt de prova. Per tant, primer gravarem els arxius d'àudio. Això ho hem fet amb el programa Audacity, concretament a 16000Hz i una freqüència de mostreig de 16 bits. Cada arxiu d'àudio conté una frase del test.

També necessitem un fitxer que relacioni cada arxiu d'àudio amb els vectors respectius de característiques. Aquest fitxer té l'extensió scp (script) i el podem fer a mà amb qualsevol editor de text.

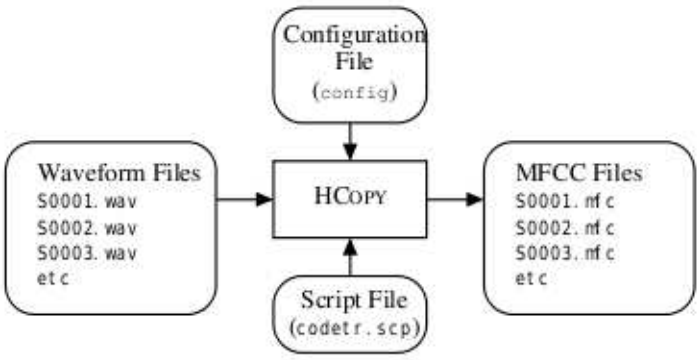

Figura 3.3: HCopy

Ja podem extreure els vectors de característiques de cada una perquè HDecode els pugui processar. Utilitzarem els Mel Frequency Cepstral Coefficients

(MFCCs), que es deriven de la transformada ràpida de Fourier. Per fer això, usem l'eina HCopy que, com ens mostra la figura 3.3, rep els arxius d'àudio, un arxiu de configuració i un arxiu de relaciona cada arxiu d'àudio amb els seus vectors. Retorna els fitxers amb els coeficients que hem esmentat abans. L'arxiu de configuració té tots els paràmetres amb els valors per defecte, el posem per si es volgués canviar algun d'ells.

L'execució és molt simple:

$ HCopy -T 1 -C config -S codetr.scp

Ara ja tenim les dades a punt per provar el reconeixedor. La comanda d'execució és una mica llarga donat que li hem de passar els models acústics (hmmdefs, macros), la ruta dels vectors de característiques (test.scp), l'arxiu de configuració (config.hdecode), l'arxiu de sortida per l'execució (recout.mlf), el model de llengua (TG2-1_1), el diccionari (diccionari.hdecode) i la llista de tots els models (tiedlist).

$ HDecode -T 1 -H hmmdefs -H macros -S test.scp -t 220.0 220.0 -C config.hdecode -i recout.mlf -w tg2-1_1 -p -5.0 -s 15.0 diccionari.hdecode tiedlist

A l'arxiu de configuració indiquem que es tracta d'un test (MFCC_0_D_A) i que el model de llengua està comprimit (gz). Les opcions -p, -s i -t són per especificar el word insertion penalty, LM scale factor i pruning beam width, respectivament. Els resultats que mostrem més avall, figura 3.4, els hem obtingut amb valors de p = -5 i s = 15. Hem fet diverses proves fins a arribar a aquests valors que ens permeten obtenir paraules no massa curtes i donar un pes més gran al model de llengua, respectivament.

Per extreure estadístiques dels resultats utilitzem l'eina HResults. És molt útil per poder valorar els resultats de les proves que hem fet. Rep tres paràmetres dels quals un encara no el tenim. Cal crear una transcripció dels arxius de so que hem gravat, en el format que necessita HTK: Master Label File (mlf). Bàsicament, cada frase té una capçalera que indica el format i el nom de l'arxiu. A més, cada paraula es troba en una línia diferent i cada frase ha d'acabar en un punt. Així HResult podrà comparar la transcripció original de l'àudio amb la sortida obtinguda amb HDecode. També necessita la llista de tots els models.

$HResults -I transcripcio.mlf tiedlist recout.mlf

La figura 3.4 mostra els resultats de les dues proves. La primera part mostra la data i el nom dels arxius utilitzats. La línia SENT mostra el nombre total de frases senceres que s'han reconegut correctament. La línia WORD mostra les estadístiques de reconeixement per les paraules individualment.

```
====================== HTK Results Analysis =======================
  Date: Sat Jun  5 13:36:38 2010
  Ref : wordsCorpus.mlf
  Rec : recoutCorpus.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=30, N=30]
WORD: %Corr=68.86, Acc=38.00 [H=241, D=12, S=97, I=108, N=350]
===================================================================


====================== HTK Results Analysis =======================
  Date: Sat Jun  5 11:00:49 2010
  Ref : wordsOriginal.mlf
  Rec : recoutOriginal.mlf
------------------------ Overall Results --------------------------
SENT: %Correct=0.00 [H=0, S=30, N=30]
WORD: %Corr=74.29, Acc=47.14 [H=260, D=11, S=79, I=95, N=350]
===================================================================
```

Figura 3.4: Test extret del corpus

Podem veure com en els dos casos obtenim un bon percentatge d'encert a nivell de paraula: 68.86% i 74.29%. El percentatge d'encert de frases és 0% perquè no s'ha reconegut cap frase bé totalment sencera. La diferència de percentatges entre el test extret del corpus i el test original ens fa pensar que el model acústic té més pes en el reconeixement de paraules, ja que no sembla afavorir els resultats el fet de gravar frases que el model de llengua "ha après". Això suposa un factor positiu, ja que l'element més potent del nostre reconeixedor és, sense dubte, el model acústic. El motiu és que l'hem creat a partir d'un corpus d'entrenament molt bo. El lexicon i el model de llengua, en canvi, són més febles ja que es van crear utilitzant un corpus d'entrenament molt més limitat (2,4 milions de paraules extretes de les novel·les). A més, cal recordar que els arxius d'àudio que hem utilitzat com a test es van gravar amb una situació molt diferent a la creació del corpus del model acústic. Cal tenir en compte la diferència de tipus

de micròfon, la qualitat del canal de transmissió, l'eco, el soroll, la distància a la que la persona sosté el micròfon, etc.

Tot i aquests elements, hem obtingut resultats prou bons. Segurament no són percentatges suficientment alts com per poder utilitzar el nostre reconeixedor en una aplicació comercial. Però si que ens mostra el potencial del procés que hem seguit. No es ingenu pensar que amb més recursos es podria arribar a obtenir un reconeixedor de la parla molt potent seguint la filosofia que ha introduït aquest projecte.

Capítol 4

Anàlisi de requisits

Com ja hem esmentat a la introducció, l'objectiu d'aquest projecte és mostrar la creació d'un reconeixedor de la parla i la seva aplicació als sistemes de dictat. En les seccions anteriors s'ha vist com hem creat el reconeixedor i quins resultats han donat les proves realitzades. Ha arribat el moment d'utilitzar-lo en una aplicació.

En aquest capítol es detallen els requisits que creiem ha de tenir la nostra aplicació de dictat automàtic. Bàsicament, es tracta de la unió de dos conceptes: un reconeixedor de la parla i un editor de text. Per tant, aquests són els dos requisits principals de l'aplicació: l'escriptura de text mitjançant la parla i l'edició de text mitjançant el teclat.

4.1 Requisits funcionals

4.1.1 Reconeixement "on live"

Fins ara hem fet el reconeixement de la parla en "playback", és a dir, primerament hem gravat la senyal d'àudio en fitxers de so i després els hem passat pel reconeixedor. És evident que aquesta manera de treballar no ens serveix per un sistema de dictat automàtic. Ara ens interessa dictar pel micròfon, és a dir, que vagi escrivint a mesura que anem parlant. Anomenem aquesta manera de treballar "on live".

Com acabem d'introduir, la nostra aplicació ha de permetre escriure mitjançant la nostra veu, utilitzant el reconeixedor. També ha de ser capaç d'escriure mitjançant el teclat per tal que no sigui obligatori utilitzar el micròfon i a més poder corregir els errors que s'hagin produït durant el reconeixement. A més, ha de proporcionar la possibilitat de comprovar que el que hem dit correspon amb el que està escrit. La figura 4.1 ens mostra les funcionalitats esmentades.

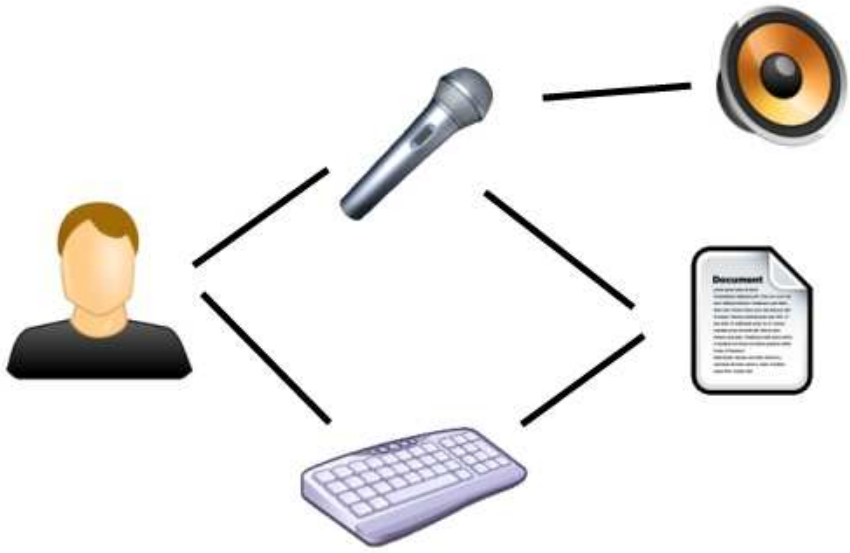

Figura 4.1: Funcionalitat de l'Aplicació

4.1.2 Editor de text

Al tractar-se d'un editor de text l'aplicació ha de comptar amb les funcionalitats més típiques d'aquests programes. Tant pel que fa al tractament de fitxers:

- crear de nou

-obrir els ja existents

- desar els canvis

- imprimir

Com pel que fa a les opcions d'edició del text:

- tallar

- copiar
- enganxar
- buscar
- reemplaçar
- mostrar el text i el fons en diferents colors.

La figura 4.2 ens les resumeix.

Figura 4.2: Edició de text

4.2 Requisits no funcionals

4.2.1 Interfície d'usuari

La interfície haurà de permetre a l'usuari utilitzar les funcionalitats que acabem de descriure, tant editar text a través del teclat com pel micròfon. Ha de ser una interfície fàcil d'utilitzar perquè el reconeixedor és independent del parlant i tothom hauria de ser capaç d'utilitzar-la.

4.2.2 Portabilitat

HTK és un conjunt d'eines multiplataforma i el reconeixedor que hem obtingut és, bàsicament, un conjunt fitxers de text. Podem dir, doncs, que és portable i aquesta

portabilitat no la volem perdre. Les dues plataformes més utilitzades a Europa són Windows i Linux, per tant, la nostra aplicació haurà de ser portable per les dues.

4.2.3 Hardware

Com és lògic es necessitarà un ordinador on executar l'aplicació. Aquest haurà de tenir la suficient capacitat computacional, tant en processament com en memòria, per permetre l'execució d'una manera correcte. A més, necessitarà els perifèrics necessaris per poder escriure mitjançant la parla, el teclat i comprovar que el que s'ha dictat coincideix amb el que s'ha escrit.

4.2.4 Legalitat

La legalitat és un requisit molt important en el desenvolupament de software en el temps actual. Hem de poder estar segurs que no incomplim cap tipus de norma legal en el desenvolupament de l'aplicació i la utilització de tots els seus elements.

4.2.5 Fiabilitat

Una aplicació que funcionés un cop de cada dos seria totalment inútil. L'aplicació ha de proporcionar les funcionalitats abans descrites sense provocar errors. Ha de ser capaç de fer treballar tots els seus elements coordinadament de manera que es pugui exprimir el reconeixedor de la parla i relacionar-lo amb l'edició de text. És a dir, obtenir un sistema de dictat automàtic que funcioni correctament.

Capítol 5

Disseny

En aquest capítol expliquem quines decisions de disseny s'han pres per complir els requisits que s'han explicat en el capítol anterior. L'objectiu doncs és obtenir un disseny complet de l'aplicació a partir del qual es pugui extreure una implementació viable, cosa que farem en el següent capítol.

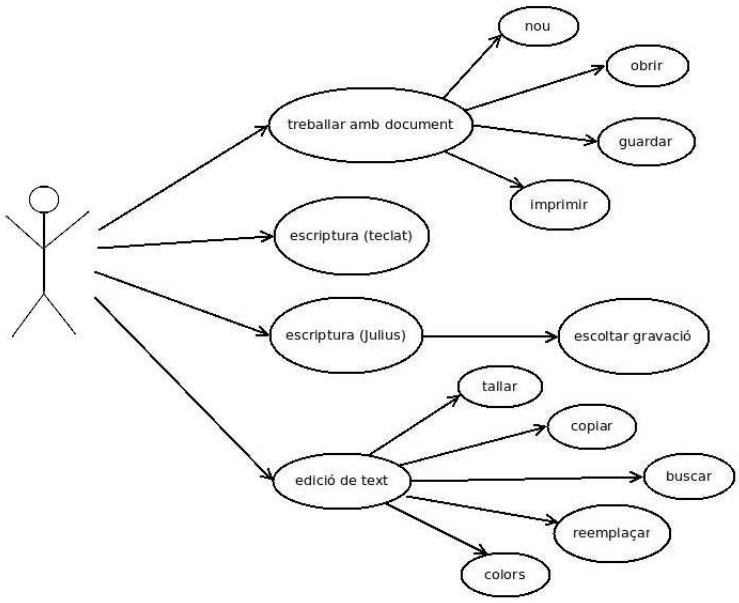

Figura 5.1: Diagrama de casos d'ús

En primer lloc, li hem donat un nom a l'aplicació: Thot. La raó és que aquest

era el nom del Déu egipci de l'escriptura, la saviesa i la música [16]. Considerem que es tracta d'un nom força adient donades les seves funcionalitats i que a més no s'ha utilitzat mai per descriure un software semblant.

Com ja hem esmentat anteriorment, es tracta d'un sistema de dictat i necessitem una eina que ens permeti utilitzar el reconeixedor de la parla "on live". Per satisfer aquest requisit hem triat Julius. Es tracta d'un descodificador d'alt rendiment de dues passades orientat a reconeixedors de la parla de gran vocabulari. Està basat en N-grames i HMM, i treballa casi en temps real, raons per les quals serveix molt bé als nostres propòsits. Actualment està desenvolupat pel grup japonès Interactive Speech Technology Consortium (ISTC) [12].

En quant a la vessant d'editor de text de Thot, hem triat Java i en concret el paquet Swing com a llenguatge d'implementació. Es tracta d'una biblioteca gràfica per Java que inclou "widgets" per interfície d'usuari tals com caixes de text i botons [11]. Ens permetrà implementar les funcionalitats típiques d'aquests programes que hem esmentat abans. La figura 5.1 ens mostra el diagrama de casos d'ús de l'aplicació que mostra que pot fer l'usuari amb Thot.

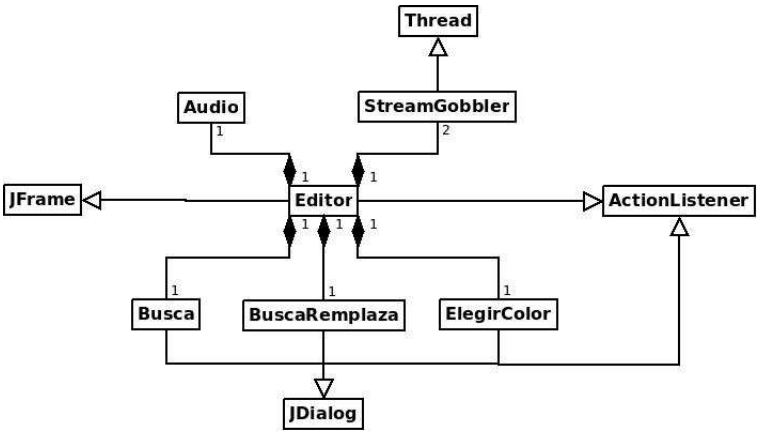

Figura 5.2: Diagrama de classes

La figura 5.2 ens mostra el diagrama de classes que ens descriu quines classes formaran l'aplicació i com es relacionen entre elles. La classe principal es diu Editor que estén la classe JFrame i implementa la ActionListener, ambdues del paquet Swing. S'encarrega de crear tots els menús, botons i objectes necessaris.

Des d'aquí cridarem totes les funcions de l'aplicació. Bàsicament podem dir que és la classe que fa funcionar tota l'aplicació. Forma una composició d'un a un amb les classes Busca, BuscaRemplaza i ElegirColor que també estenen JFrame i implementen ActionListener. El seu nom ja ens diu quina és la seva funció: buscar, reemplaçar i triar el color del text i del fons, respectivament. Editor també forma composició amb Audio i StreamGobbler, aquesta última estén Thread. La primera s'encarrega de gravar i reproduir l'àudio mentre la segona es comunica amb Julius.

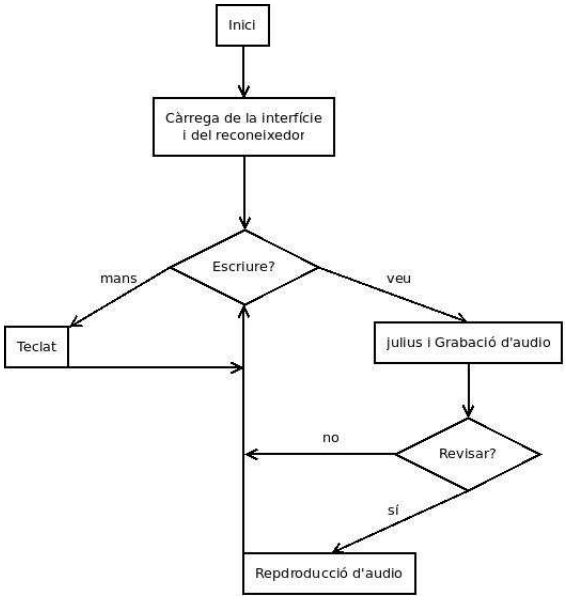

Figura 5.3: Diagrama de flux

Per acabar amb els diagrames, la figura 5.3 ens mostra el diagrama de flux. Aquest representa el flux d'execució bàsic de l'aplicació. Tot comença iniciant Thot que carregarà la interfície i el reconeixedor, és a dir, carregarà els models. Un cop tot està preparat l'usuari ja pot començar a escriure tant amb el teclat com amb la veu. Després d'escriure mitjançant el reconeixedor pot triar a escoltar el que ha dit per repassar el text o seguir escrivint en qualsevol de les dues formes.

5.0.6 Interfície d'usuari

Com ja hem dit es tractarà d'una interfície típica del editors de text. Amb una àrea principal per escriure, una petita que indicarà la última, o últimes paraules reconegudes, i les opcions d'edició ja comentades accessibles a través de botons i menús. Això facilitarà la seva utilització per qualsevol tipus d'usuari.

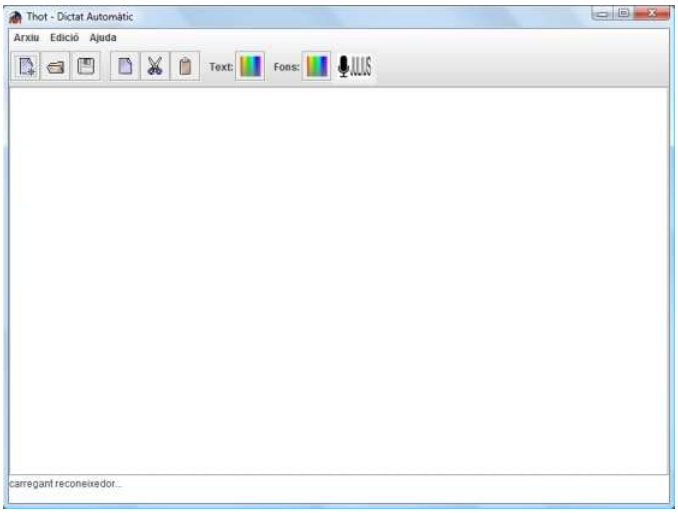

Figura 5.4: Thot

La figura 5.4 és una captura de pantalla de l'aplicació. Veiem com es tracta d'un tipus interfície molt comuna al món dels editors de text i que no sorprendrà a cap usuari.

5.0.7 Portabilitat

Volíem que l'aplicació fos portable respecte dues plataformes: Linux i Windows. Ho complirem satisfactòriament, ja que tant Java [11] com Julius són compatibles amb aquests dos sistemes [12].

5.0.8 Hardware

Un cop triades les tecnologies que usarem per implementar l'aplicació, ja podem prendre les decisions de disseny sobre el hardware. Observant els requisits

de Julius i Java i l'espai que ocupa el reconeixedor que hem obtingut, podem determinar els requeriments que tindrà la nostra aplicació en quant a memòria. Julius treballa amb una memòria RAM mínima de 64 MBytes [12] i Java de 128 MBytes [11]. Per tant serà recomanable disposar d'un ordinador amb una memòria RAM de 256 MBytes. A més, també necessitarem un espai al disc dur de 100 MBytes per poder-hi posar l'aplicació i el reconeixedor. En quant a processador, el mínim necessari és de 166 MHz tot i que seria recomanable una freqüència més gran.

A més, necessitarem un micròfon per poder introduir la nostra veu al reconeixedor. Qualsevol micròfon amb unes característiques bàsiques ens servirà ja que no necessitem res d'especial. I, per acabar, un sistema d'altaveus també convencional si volem escoltar el que hem dit quan dictàvem.

5.0.9 Legalitat

A nivell legal no tenim cap problema ja que tant Java [11] com Julius es distribueixen sota llicència "open source" [12]. Això ens permet desenvolupar la nostra aplicació sense cap tipus de preocupació, sobretot tenint en compte que no té l'objectiu ni la vocació de convertir-se en una aplicació comercial. Ha de ser entesa com una demostració del potencial que té la línia iniciada per aquest projecte.

5.0.10 Fiabilitat

Java Swing ens permet crear tota la interfície d'usuari i la utilització de Julius amb total garantia de fiabilitat. Es tracta d'un paquet molt utilitzat en programació que ha demostrat ser molt útil i segur.

Com hem vist a les proves que hem realitzat en els apartats anteriors, el percentatge d'encert utilitzant HDecode es trobava entre el 68% i el 74% aproximadament. Aquests percentatges ens poden servir com a orientació però no són pas un requisit de fiabilitat estricte ja que, com acabem de dir, aquestes proves es van realitzar en "playback". Farem noves proves per veure com es comporta el

reconeixedor “onlive” utilitzat per Julius.

5.0.11 Resum

Un cop repassats tots els requisits podem veure com el disseny que proposem és assequible i ens permet la seva implementació. Com a resum d’aquest capítol direm que la interfície d’usuari es basarà en botons i menús seguint el disseny habitual dels editors de text als quals tots els usuaris d’ordinador ja hi estan acostumats. Java i Julius ens proporcionen una gran portabilitat en plataformes majoritàries a Europa com són Windows i Linux. És evident que qualsevol ordinador comercial podrà complir sense problemes els requeriments físics o de hardware. A més, no patirem per la legalitat del software que desenvoluparem ni de l’ús que en farem després. Per últim, no tenim uns requeriments específics sobre la fiabilitat de l’aplicació. Partim dels resultats obtinguts amb les eines HTK per tenir una idea aproximada del que ens trobarem i farem proves utilitzant Julius per determinar la fiabilitat de l’aplicació.

Capítol 6

Implementació

En aquest capítol veurem els problemes i els aspectes a destacar que ens hem trobat durant la implementació del disseny proposat al capítol anterior.

El primer que cal destacar és que Julius necessita d'un model de llengua "al revés" o "backward", ja que si recordem, és de dues passades. La creació d'aquest model "backward" és molt senzilla ja que només cal invertir l'ordre de les paraules del corpus d'entrenament per cada frase i seguir els passos descrits al apartat 3.2.

Com ja hem comentat, la implementació s'ha realitzat en Java per permetre la portabilitat amb Windows i Linux. En els dos casos s'ha desenvolupat l'aplicació utilitzant el mateix programa: Eclipse. Tot i això i com és natural, no ens hem trobat les mateixes dificultats i problemes en les dues plataformes.

Començarem parlant de la implementació en entorn Linux. El problema principal que ens podem trobar és tractar amb la targeta de so. En especial per treballar amb el micròfon. No totes les targetes de so i els seus drivers treballen correctament en entorns Linux, Ubuntu 10.04 en el nostre cas. Podem trobar diversos problemes a l'hora de seleccionar l'origen de gravació de so, en especial en ordinadors portàtils que tinguin un micròfon integrat. A més també podem tenir problemes a l'hora de gravar i reproduir so mitjançant Java pel mateix motiu.

En entorn Windows, en canvi, no ens hem trobat cap tipus de problema a l'hora de seleccionar el micròfon. Tampoc en quant a gravar la nostra veu i reproduir-la després per comprovar el que s'ha escrit. Si que és diferent però, la codificació de

caràcters. Tant els models de llengua (“forward i “backward”), com el diccionari i el model acústic els tenim en UTF-8 però la sortida predeterminada de Windows és en ISO-8859. Per tant, hem hagut de fer una conversió per tal que les paraules amb accents s’escriguessin correctament i no obtinguéssim caràcters estranys. A més, cal remarcar un aspecte que no és en si un problema però un desavantatge respecte l’entorn Linux. La gestió de memòria en Windows, Vista en el nostre cas, fa que la càrrega inicial del reconeixedor sigui significativament més lenta que en el seu company del pingüí. Diem que no és un problema ja que la càrrega es fa només un cop quan arranquem l’aplicació.

Capítol 7

Proves

Les proves sobre l'aplicació les hem dividit en dues parts. En primer lloc hem comprovat que totes les funcionalitats estiguessin disponibles correctament. Aquesta prova a resultat totalment satisfactòria ja que hem pogut escriure mitjançant les dues vies de que disposem i hem escoltat la gravació del nostre dictat, cosa que ens ha permès corregir errors. A més a més les opcions d'edició de text també han funcionat correctament.

En segon lloc hem comprovat el rendiment que extreu Julius al nostre reconeixedor de la parla. Hem dictat les frases que formaven els dos tests de prova utilitzats al model de llengua. Si recordem, es tractava de dos tests de 350 paraules repartides en 30 frases cada un. La diferència bàsica era el seu origen, un estava extret directament del corpus d'entrenament i l'altre no. Hem comptat manualment les paraules encertades, hem calculat el percentatge respecte del total i hem obtingut les següents dades:

Test extret del corpus:

- 202 encertades de 350 paraules totals. 57,7% d'encert a nivell de paraula.
- 5 frases encertades completament respecte 30 totals. 16% a nivell de frase.

Test extret no del corpus:

- 202 encertades de 350 paraules totals. 57,7% d'encert a nivell de paraula.
- 3 frases encertades completament respecte 30 totals. 10% a nivell de frase.

Ràpidament podem veure com els resultats són casi exactament iguals. De-

duïm doncs que l'origen del test no té cap tipus d'incidència en el reconeixement. El percentatge obtingut a nivell de paraula (57,7%) és significativament inferior al obtingut a les proves en "playback" (entre el 68% i 74%). L'origen d'aquesta diferència el trobem en el fet que el reconeixement usant HDecode era molt més lent que utilitzant Julius. Evidentment en "playback" no teníem cap tipus de pressa i podíem dedicar molt temps per a obtenir bons resultats. Treballant "onlive" això no ens és possible. És important que entre el dictat i l'escriptura no hi hagi un gran període de temps, encara que això ens penalitzi el percentatge d'encert.

L'única petita diferència entre les dues proves es troba en el percentatge d'encert de frases. No és significativa però si que ens hem adonat d'un fet a remarcar. És molt més probable que reconegui una frase completa correctament si aquesta és curta. Les frases llargues són molt més complicades de reconèixer, és a dir, com més curta és una frase (3, 4 o 5 paraules) menys li costa reconeixla correctament. Si comparem amb les proves en "playback" veiem com ara els percentatges han millorat a nivell de frase. En aquelles proves no vam obtenir una sola frase sencera ben reconeguda, cosa que ara sí hem aconseguit.

Capítol 8

Conclusions

Com ja vam comentar a la introducció d'aquest projecte, no ens havíem fixat un objectiu inicial determinat en quant al percentatge d'encert del reconeixedor, tot i que els resultats obtinguts han sigut força satisfactoris. La intenció era obtenir un reconeixedor de la parla en català, mostrar el procés que hem seguit per la seva creació, introduir el corpus de les novel·les com a una opció nova d'aprenentatge i valorar els resultats obtinguts per determinar quines són les seves limitacions, per quin motiu hi són i com les podríem superar.

Doncs bé, hem aconseguit crear un reconeixedor de la parla que ha sigut integrat en una aplicació de dictat automàtic, cosa de la que podem estar molt orgullosos. El procés que hem seguit està descrit amb cura perquè pugui ser reproduït per qualsevol persona interessada en millorar, innovar o senzillament modificar la feina que hem realitzat. Els tres components principals han quedat ben explicats i s'han comportat molt bé donades les seves limitacions. Cal remarcar especialment la introducció del corpus d'entrenament del model de llengua basat en novel·les com una opció nova i vàlida. És el gra de sorra que dipositem en el món de la recerca dels models de llengua. Creiem que és una alternativa innovadora i original que encara no s'havia plantejat en aquest punt i que aporta un nou punt de vista.

A més, hem sigut capaços d'incorporar el reconeixedor a una aplicació per formar un sistema de dictat automàtic. Ha sigut realment satisfactori comprovar

com tota la feina realitzada, després de tantes hores d'esforç, podia tenir una aplicació pràctica, més enllà de la recerca. Evidentment es tracta d'una aplicació molt senzilla que no és un bon producte a comercialitzar, però si que demostra com es poden aprofitar els reconeixedors de la parla i quina pot ser una de les seves utilitats. Les proves realitzades en aquest punt també han sigut satisfactòries.

Durant la realització del present projecte hem utilitzat les eines HTK i Julius. Ens han ajudat moltíssim i observant tot el que hem pogut fer amb elles, arribem a la conclusió que és increïble la seva potència. Desenvolupar el software necessari que ens ha permès crear el nostre sistema requereix una gran quantitat de coneixements matemàtics, de programació i l'esforç de moltes persones. Sense aquestes eines seria impossible que un estudiant pogués desenvolupar reconeixedors de la parla a casa seva amb un portàtil convencional. No tenim més que paraules d'elogi cap als desenvolupadors d'HTK i Julius.

Com a resum final d'aquest projecte només ens queda dir que tot el sistema que hem desenvolupat concentra l'esforç de moltes persones. L'obtenció del corpus pel model acústic, la creació del model acústic, l'obtenció del model de llengua, l'obtenció del corpus pel model de llengua, la realització de les diferents proves, el desenvolupament de l'aplicació i moltes altres coses són el resultat de moltes hores de treball per part de moltes persones. És la unió de l'esforç, els coneixements i la dedicació de molta gent la que ha fet possible desenvolupar tot el que es mostra en aquest projecte.

8.1 Millores futures

En aquesta secció veurem quines millores es podrien aplicar en el futur per obtenir millors resultats en el nostre reconeixedor. Les proves realitzades al reconeixedor han sigut molt satisfactòries i ens duen a la primera millora: el creixement del corpus. És evident com augmentant-lo obtindríem un reconeixedor de més qualitat. Com més i millor aprengui tant el model acústic com el de llengua, millor diccionari fonètic obtindrem i millor treballaran tots plegats per formar el sistema sencer.

La segona d'elles es basa en adaptar el reconeixedor a la persona concreta. El reconeixedor que hem creat en aquest projecte és independent del parlant, si entrenem els models acústics utilitzant dades d'entrenament d'un únic usuari, obtindrem millors resultats per aquesta persona. Cal dir que també els obtindrem pitjors per a la resta. Si féssim això com s'han creat els models acústics que tenim, necessitaríem una gran quantitat de dades d'entrenament de cada usuari concret. La millor manera de fer això és adaptant els models que ja tenim a les característiques del nou usuari usant una petita quantitat de dades d'entrenament.

Una altra millora que proposem es basa en l'entrenament discriminat. L'objectiu dels models de Markov és maximitzar la versemblança. Aquest criteri proporciona una aproximació raonable al criteri de mínim error de classificació. Si tenim el suficient material d'entrenament pot existir un conjunt de paràmetres que, tot i no complit el criteri de màxima versemblança, estigui més aprop de complir el de mínim error. Aquells mètodes que tendeixen a obtenir models acústics que siguin el més aprop possible a la situació de mínim error, s'anomenen d'entrenament discriminat [10].

Bibliografia

[1] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev i P. Woodland. *The HTK Book*. Març 2009. pàgines: 3-5, 20, 51-52.

[2] X. Huang, A. Acero, Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm and System Development* Prentice Hall 2001. pàgines: 558-560.

[3] J. L. *Malodonado La estadística como herramienta para el desarrollo de sistemas automáticos reconocedores del habla* Revista Economía No. 14, 1998. Instituto de Estadísticas Aplicada y Computación. Universidad de Los Andes. pàgina: 9, 12-16.

[4] P. Pachès, C. de la Mota, M. Riera, M. P. Perea, A. Febrer, M. Estruch, J. M. Garrido, M. J. Machuca, A. Ríos, J. Llisterri, I. Esquerra, J. Hernando, J. Padrell i C. Nadeu. *SEGRE: An automatic tool for grapheme-to-allophone transcription in catalan* Maig 2000. pàgines: 2-3.

[5] S. A. Della Pietra, P. E Brown, J. C. Lai, V. J. Della Pietra, R. L. Mercer *An Estimate of an Upper Bound for the Entropy of English*. Març 1992. pàgines: 38-39.

[6] A. Montero-Asenjo, C. A. Iglesias *Turning Wikipedia into a resource for language research*. Grupo de Sistemas Inteligentes. Universidad Politécnica de Madrid. Novembre 2008. pàgines: 1-5.

[7] A. Becerra, M. Gómez, M. F. Ordóñez, B. Macas *Reconocimiento de voz mediante Modelos Ocultos de Markov*. Agost 2009. pàgines: 3-4.

[8] S. Cook *Speech Recognition HOWTO*. Abril 2002. pàgines: 5-6.

[9] D. Jurafsky, J. H. Martin. *Speech and Language Processing* Prentice-Hall 2000. pàgines 168-178, 212-214.

[10] A. N. Rodríguez *Entrenamiento Discriminativo de Modelos Ocultos de Markov de Unidad Subléxica para su Aplicación a Sistemas de Reconocimiento Automático del Habla Continua* 1999. página: 21.

[11] Java.
<http://www.java.com>

[12] Julius.
<http://julius.sourceforge.jp/en>

[13] Tecnologies de la parla en català.
<http://www.talp.upc.edu/tecnoparla/>

[14] HTK Speech Recognition Toolkit
<http://htk.eng.cam.ac.uk/>

[15] Swing y JFC (Java Foundation Classes).
<http://213.149.242.201/java/tutorial/swing/>

[16] La tierra de los faraones.
<http://www.egiptologia.org>

Firmat: Néstor Pérez Rivero

Bellaterra, Juny de 2010

Resum

En aquest projecte es fa una introducció als reconeixedors de la parla, l seu funcionament i la seva base matemàtica. Un cop tots els conceptes han quedat clars, es mostra el mètode de creació que hem seguit per obtenir el nostre propi reconeixedor de la parla, utilitzant les eines HTK, en Català . S'avaluen les seves virtuts i els seus defectes a través de diferents proves realitzades als seus components. A més a més, el projecte arrodoneix la feina implementant un sistema de dictat automàtic que explota el reconeixedor de la parla utilitzant Julius.

Resumen

En este proyecto se hace una introducción a los reconocedores del habla, su funcionamiento y su base matemática. Una vez todos los conceptos han quedado claros, se muestra el método de creación que hemos seguido para obtener nuestro propio reconocedor del habla, usando las herramientas HTK, en Catalán. Se evalúan sus virtudes y sus defectos a través de diferentes pruebas realizadas a sus componentes. Además, el proyecto redondea el trabajo implementando un sistema de dictado automático que explota el reconocedor del habla usando Julius.

Abstract

This project provides an introduction to the speech recognition, operation and mathematical basis. Once all the concepts are clear, it shows the method of creation that we followed for our own speech recognizer using HTK tools in Catalan. Assesses its strengths and weaknesses across different tests on their components. In addition, the project gets complete implementing an automatic dictation system that exploits the speech recognizer using Julius.