Référence de la Communication : CIA 100

RAPACE : UN SYSTEME DE RECONNAISSANCE ANALYTIQUE DE PAROLE CONTINUE

Louis SAUTER

Laboratoires de MARCOUSSIS
Centre de Recherches de la **C. G. E.**

Route de Nozay

91460 MARCOUSSIS

1 - Introduction

Le problème de la reconnaissance de phrases simples, construites avec quelques dizaines, voire une centaine de mots, et obéissant à une syntaxe artificielle a été abordé de deux manières :

- Approche globale (1-2-3) : les différents mots sont mémorisés sous forme globale, et la reconnaissance se fait en cherchant la suite de mots de référence qui ressemble le plus à la phrase prononcée. La distance utilisée est en général une forme de comparaison dynamique.

- Approche analytique (4 par exemple) : la phrase prononcée est d'abord convertie en une suite d'éléments (phonèmes ou autres). Le résultat de cette analyse fait l'objet d'une recherche lexicale, suivie d'une analyse syntaxique.

Rapace utilise principalement l'approche analytique. En effet, la parole est d'abord transcrite sous forme d'une chaîne de pseudo-phonèmes. Les composantes lexicales et syntaxiques sont alors effectuées en une seule passe par un algorithme voisin de ceux développés pour l'approche globale (1-2). L'avantage principal du système obtenu réside dans la réduction importante de l'encombrement mémoire et de la complexité des calculs nécessaires pour mener à bien la reconnaissance, ce qui nous a notamment permis de réaliser des simulations en temps réel sur un mini-ordinateur.

L'organisation du système **Rapace** est décrite par la figure 1. Nous allons examiner successivement les composants acoustiques, phonétiques et lexico-syntaxiques.

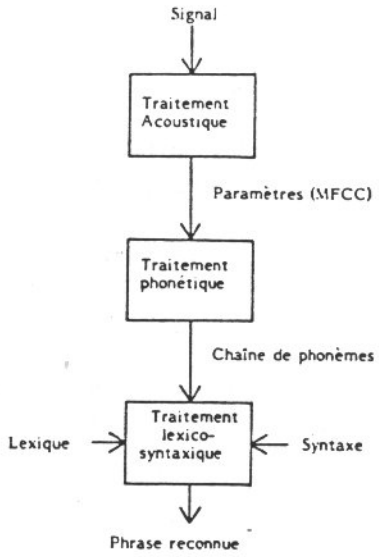

Fig. 1 - Organisation de Rapace

2 - Traitement acoustique

Différentes représentations paramétriques du signal acoustique ont été comparées dans le cadre de la reconnaissance de pseudo-phonèmes. Les meilleurs résultats ont été obtenus avec des paramètres cepstraux utilisant l'échelle de fréquence Mel (Mel Frequency Cepstrum Coefficients : MFCC). Nos résultats sont donc en accord avec ceux publiés dans (3) et (5), quoique ceux-ci aient été obtenus dans le cadre de l'approche globale.

L'échelle Mel est une mesure de la fréquence qui tient compte de la variation de la bande critique de l'oreille. Elle peut être approximée par une fonction linéaire entre 0 et 1000 Hz et logarithmique au-delà. Les MFCC sont obtenus en calculant la transformée de Fourier inverse du logarithme du spectre-mel. Celui-ci est obtenu à l'aide d'un banc de filtres dont les fréquences centrales sont distribuées régulièrement sur l'échelle Mel, c'est-à-dire selon une progression arithmétique jusqu'à 1000 Hz et une progression géométrique au-delà.

Le traitement acoustique pour **Rapace** est résumé ci-dessous :

* Le signal est filtré puis numérisé (8000 échantillons par seconde, 12 bits).

* 10 MFCC sont calculés toutes les 8 ms.
  - Multiplication par une fenêtre de Hamming (32 ms),
  - Transformée de Fourier rapide,
  - Evaluation de la sortie du banc de filtres,
  - Transformée inverse (10 valeurs).

* L'énergie est calculée toutes les 8 ms.

Remarques :

* Le spectre (Log) étant une fonction paire et réelle, la transformée de Fourier inverse se simplifie en transformée en cosinus inverse.

* Le banc de filtres, simulé ici à l'aide d'une transformée de Fourier, pourrait être remplacé par un vocodeur à canaux. L'ensemble du calcul des MFCC pourrait alors être réalisé sur un µ-processeur spécialisé en traitement du signal.

* Dans la simulation actuelle, les MFCC sont calculés sur un processeur vectoriel.

* L'énergie est obtenue en sommant les sorties des filtres correspondant aux fréquences supérieures à 500 Hz. Une autre valeur, correspondant aux fréquences supérieures 700 Hz est également utilisée pour détecter les barres de voisement.

3 - Traitement phonétique

La composante phonétique de **Rapace** comprend 3 modules :

- Un module de segmentation en pseudo-syllabes,
- Un module d'étiquetage qui donne un nom de phonème à chaque vecteur de MFCC,
- Un module de transcription qui identifie chaque noyau vocalique et chaque groupe de consonnes.

3-1 - Module de segmentation

Ce premier module assure les fonctions suivantes :

- Détection début/fin de parole,
- Segmentation en pseudo-syllabes, c'est-à-dire localisation des noyaux vocaliques et des groupes de consonnes inter-vocaliques.

Principe : recherche des minimums et des maximums successifs de l'énergie qui correspondent respectivement aux groupes de consonnes et aux noyaux vocaliques.

Différentes méthodes (6-7) ont été proposées pour éliminer les extremums non significatifs. Celle décrite ci-dessous donne des résultats satisfaisants.

* L'énergie dans la bande de fréquences au-dessus de 500 Hz est évaluée toutes les 8 ms et légèrement lissée.

* Une suite d'extremums de sens opposé est recherchée vérifiant les contraintes suivantes :

  - Le temps entre deux extremums sucessifs (correspondant à une demi-syllabe) doit dépasser tmin (tmin environ 40 ms).
  - Le rapport des valeurs des deux extremums successifs doit être supérieur à smin (smin est de l'ordre de 2 à 3).
  - Les maximums doivent correspondre à une énergie supérieure à emin (il s'agit des noyaux vocaliques).

On obtient ainsi une segmentation en syllabes. Les performances de cette méthode ont été mesurées sur 10 phrases extraites d'un corpus de phrases phonétiquement équilibrées (8). Le pourcentage de syllabes supplémentaires est inférieur à 5 %, celui des syllabes non détectées de 6 %.

La détection de début (fin) de parole est faite en examinant la région précédant (suivant) le premier (dernier) noyau vocalique.

3-2 - Module d'étiquetage

Ce module attribue à chaque vecteur MFCC (c'est-à-dire toutes les 8 ms) une étiquette comportant un ou plusieurs noms de phonèmes spectralement voisins.

Nous avons comparé plusieurs méthodes de classement utilisant toutes les mêmes données d'apprentissage (soit 10 exemples de chaque phonème, extraits manuellement de parole continue). Les meilleurs résultats ont été obtenus avec des fonctions discriminantes linéaires (9) tenant compte de la probabilité a priori de chaque phonème (8).

Le cas particulier des plosives est traité comme suit : l'énergie dans la bande de fréquences supérieures à 700 Hz est utilisée pour détecter les zones de silence et les barres de voisement. Les classes correspondant aux plosives sont autorisées uniquement pendant les transitions silence ou barre de voisement vers parole.

Note : les références phonémiques ont été obtenues à partir de textes lus, indépendemment des phrases qui ont servi à effectuer les tests.

3-3 - Module de transcription

Ce module utilise les informations produites par les modules précédents. Il émet des hypothèses sur les noms des phonèmes qui constituent chaque noyau vocalique et chaque groupe de consonnes. Dans les deux cas, des règles permettent d'évaluer la vraisemblance des suites de phonèmes possibles en français. Pour les groupes de consonnes contenant une zone de silence ou une barre de voisement, les suites de phonèmes possibles sont obtenues en concatenant un **groupe de consonnes final** et un **groupe de consonnes initial** (6). Pour les autres groupes de consonnes ainsi que pour les noyaux vocaliques, une liste est utilisée, énumérant les suites de phonèmes possibles. La longueur de cette liste est de l'ordre de quelques dizaines seulement (groupes de consonnes sans plosive, noyaux vocaliques contenant une ou plusieurs voyelles).

Les suites de phonèmes les plus vraisemblables sont alors hypothésées.

3-4 - Performances

Le taux de reconnaissance de phonèmes a été évalué dans différents contextes, pour un locuteur.

Test A : élocution peu soignée. 20 phrases phonétiquement équilibrées (8) soit environ 450 phonèmes.

Test B : élocution lente et soignée. 10 phrases appartenant à un langage artificiel, soit 330 phonèmes.

Les performances, données dans le tableau 1, sont assez satisfaisantes. Néanmoins, la reconnaissance des phonèmes reste un point faible.

TEST	Phonèmes classés en 1ère position	Phonèmes classés dans les 3 premières positions
A	58 %	71 %
B	68 % Consonnes : 61 % Voyelles : 74 %	80 % Consonnes : 69 % Voyelles : 91 %

Exemple extrait du test A : "La vaisselle propre est mise sur l'évier".

(L) ET (V) (ET) (S) (ET) (L)(R) (O) (R) (ET) (M)(I) ET L (S) E (R)

Z E R E R E M S AN S I N E SH ET etc

P (A) M Y F UN J K ON F V Y R AN

Tab. 1 - % de phonèmes correctement détectés après segmentation, étiquetage et transcription

4 - Analyse lexicale et syntaxique

Nous avons déjà indiqué que les analyses lexicale et syntaxique de **Rapace** sont effectuées en une passe par un algorithme unique. Nous allons maintenant examiner cet algorithme en détail.

4-1 - Représentation du lexique

Le lexique comporte des mots, ainsi que des groupes de mots indivisibles pour le langage considéré. Dans un premier temps, chaque élément du lexique est représenté par une suite unique de phonèmes, correspondant à la manière habituelle de le prononcer. Nous verrons plus loin comment tenir compte des variations phonologiques.

4-2 - Comparaison d'une forme inconnue aux éléments du lexique

La chaîne de phonèmes correspondant à la prononciation d'un élément du lexique contient en général des erreurs de trois types : élisions, insertions, substitutions.

Nous définissons la distance $\tilde{d}$ entre une chaîne de phonèmes et un élément du lexique comme le coût minimal pour transformer l'un en l'autre à l'aide des opérations d'élision (coût : p_0), d'insertion (coût : p_1) et de substitution (coût : p_2) (Distance de Levenshtein pondérée (13-14)).

Cette distance peut être calculée par un algorithme de type programmation dynamique (10) : si M est la longueur de la chaîne de phonèmes et N la longueur de la référence lexicale, $\tilde{d}$ est obtenu de la manière suivante :

a. $d(o,o) = 0$

b. $d(o,j) = j \times p_o$ pour $j = 1 \ldots N$

c. $d(i,o) = i \times p_1$ pour $i = 1 \ldots M$

d. pour $i = 1$ à M
pour $j = 1$ à N

$$d(i,j) = \min \begin{cases} d(i,j-1) + p_o \\ d(i-1,j) + p_1 \\ d(i-1,j-1) + p_2 \end{cases} \quad (1)$$

e. $\tilde{d} = d(M,N)$

Dans la suite, nous noterons $(\hat{i},\hat{j})$ les arguments de d dans la ligne de (1) correspondant au minimum $(\hat{i},\hat{j}) = (i,j-1)$ ou $(i-1,j)$ ou $(i-1,j-1)$.

Dans le cas où la chaîne de phonèmes est de largeur supérieure à 1, p_2 pourra être variable pour tenir compte des différentes hypothèses.

p_2 peut également varier en fonction de la vraisemblance d'une substitution (ainsi, p_2 sera plus élevé pour la substitution d'une voyelle par une fricative que pour celle d'une fricative par une autre fricative). Les paramètres p_o, p_1 et p_2 ont été choisis expérimentalement.

4-3 - Représentation de la syntaxe

L'algorithme qui est utilisé par **Rapace** pour l'analyse lexico-syntaxique nécessite que la syntaxe soit représentée de la manière suivante :

Le langage, supposé fini, est accepté par un automate à états finis $A(Q,V,\delta, q_o, Z)$ où Q est un ensemble de NE+1 états : $Q = \{q_o, q_1, \ldots q_{NE}\}$, V est un vocabulaire, $\delta \subset Q \times V \times Q$ est un ensemble de transitions : $\delta = \{t_1, t_2, \ldots\}$, q_o est l'état initial et $Z \subset Q$ l'ensemble des états finals.

On suppose que les états de Q sont numérotés de manière que :

$$\forall v \in V, q_i, q_j \in Q : (q_i, v, q_j) \in \delta \Rightarrow i < j$$

Ceci est possible car nous avons supposé que le langage est fini (11).

D'autre part, les transitions de δ sont numérotées de manière que :

si $t_i = (q_{i1}, v_i, q_{i2})$ et $t_j = (q_{j1}, v_j, q_{j2})$ alors $i < j \Rightarrow i_1 < j_2$

4-4 - Algorithme de reconnaissance lexico-syntaxique

Nous cherchons la suite de mots vérifiant la syntaxe et qui correspond au minimum de distance "cumulée". Plus formellement, appelons $(r_1, r_2 \ldots r_M)$ la chaîne de sortie de l'analyseur phonétique correspondant à la phrase prononcée et $\tilde{d}(i,j,v)$ la distance entre la sous-chaîne $(r_i \ldots r_j)$ et l'élément lexical v ; alors nous cherchons la suite $(v_1 v_2 \ldots v_k)$ vérifiant la syntaxe et qui rend minimum la fonction :

$$\tilde{D}(v_1 \dots v_k) = \min_{i_1, i_2 \dots i_k} \left[\sum_{e=1}^{k} \tilde{d}(i_{e-1}, i_e, v_e) \right]$$

avec $i_o = 0$, $i_k = M$ (la longueur totale de la chaîne de phonèmes) et $i_e \leqslant i_{e+1}$ pour tout e.

Notons que les i_e correspondent aux frontières entre mots.

$\tilde{D}$ peut être calculé par un algorithme de type programmation dynamique. Définissons $Q' = \{(s,i),\ s \in Q,\ i = 0, 1, 2 \dots M\}$; alors $\tilde{D}$ est obtenu de la manière suivante :

a. $D(q_o, 0) = 0$

b. pour $(s,i_k) \in Q'$:

$$D(s,i_k) = \min_{\substack{(q,i_{k-1}) \in Q' \\ (q,v_k,s) \in \delta \\ i_{k-1} \leqslant i_k}} (D(q,i_{k-1}) + \tilde{d}(i_{k-1}, i_k, v_k))$$

c. $\tilde{D}(v_1 \dots v_k) = \min_z D(z,M)$ où z est un état final ($z \in Z$)

4-5 - Mise en oeuvre de l'algorithme

MYERS et LEVINSON (1) et BRIDLE, BROWN et CHAMBERLAIN (2) ont proposé des méthodes pour mettre en oeuvre cet algorithme. Nous utilisons la variante suivante :

initialisation : $FQ(q_o) = 0$

$D_t(o,j) = j \times p_o$ si la transition t part de q_o

∞ sinon

pour i allant de 1 à M (longueur de la chaîne de phonèmes) faire :

$DQ(q_o) = (i-1) \times p_1$

pour $t \in \delta$; $t = (q,v,s)$ faire

Dt(i,o) = DQ(q)

Ft(i,o) = FQ(q)

pour j allant de 1 à N_v (longueur de la référence v)

faire :

Dt(i,j) calculé selon l'équation 1

Ft(i,j) = Ft(î,ĵ)

Fin

si $Dt(i,N_v)$ correspond à la plus petite valeur obtenue pour des transitions aboutissant à l'état s alors

$DQ(s) = Dt(i,N_v)$

$FQ(s)$ = pointeur vers une cellule contenant $Ft(i,N_v)$ et v

Fin

Fin

Fin

pour le meilleur score obtenu par un état final, utiliser le contenu des cellules pour retrouver le chemin optimal.

Fin

4-6 - Améliorations apportées à l'algorithme

a) Représentation lexicale

Chaque mot est représenté par un graphe de référence (4) contenant à la fois la transcription phonétique usuelle et des variantes dues aux altérations phonologiques et aux erreurs fréquentes du frontal phonétique. Par exemple, le groupe de mots "parler à" peut être représenté par la fig. 2.

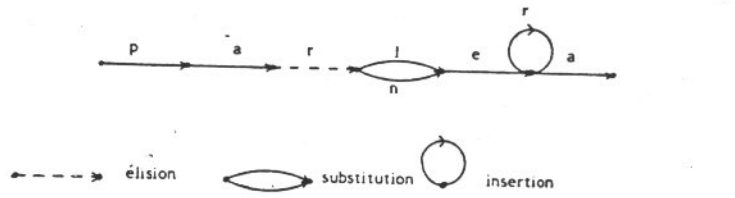

Fig. 2 - Exemple de graphe de référence

L'équation (1) peut alors être modifiée pour annuler le coût d'une insertion, élision ou substitution prévue.

b) Transition Joker

Le langage peut comporter des transitions dont le contenu sémantique est faible. On peut donc prévoir des "jokers", c'est-à-dire des éléments de V tel que d(i,j,v) ne dépend que de (j-i) et non des sorties effectives de l'analyseur phonétique.

c) Retour en arrière partiel

Pendant que l'algorithme examine la chaîne de phonèmes, il est possible d'éli-

miner à chaque instant les cellules non référencées. Ainsi, il est possible de connaître le début du chemin optimal avant d'avoir atteint la fin de la phrase. En utilisant un seuil pour abandonner les chemins "peu prometteurs", il est possible d'amplifier ce comportement ; la reconnaissance se fait alors avec un retard de quelques mots seulement.

4-7 - Quelques résultats

Nous avons effectué plusieurs simulations, toutes en temps réel. L'utilisation du retour en arrière partiel a effectivement permis la reconnaissance du début de la phrase avant que la fin de celle-ci ait été prononcée.

a) Simulation avec la syntaxe de MYRTILLE I (4) (standard téléphonique)

Sur 10 phrases (88 mots) utilisant toutes les constructions possibles de ce langage, il y a eu une erreur (correspondant à un taux de reconnaissance par mots de plus de 98 %).

b) Simulation avec un langage de commande de robot

(vocabulaire d'une vingtaine de mots, commandes simples du type "posez la petite pyramide sur le grand cube")

Le taux de reconnaissance pour 50 phrases a été de 92 % (pour les phrases) et de 98 % (pour les mots).

Ces simulations ont été effectuées sur un miniordinateur HP 1000. Nous rappelons que le calcul des paramètres représentatifs était réalisé par un processeur spécialisé.

4-8 - Conclusion

Nous avons décrit le système de reconnaissance de la parole **Rapace.** L'approche adoptée pour ce système ne peut vraisemblablement pas être étendu aux langages pseudo-naturels ou naturels. Néanmoins, dans le cadre des langages artificiels utilisant des vocabulaires de petite taille, **Rapace** donne un exemple de communication orale fiable et en temps réel plus naturelle que celle permise par les systèmes de reconnaissance de mots isolés, mais moins coûteuse que la reconnaissance globale de mots enchaînés.

Références

[1] **C.S. Myers** et **S.E. Levinson,** "Connected Word Recognition Using a Syntax Directed Dynamic Programming Temporal Alignment Procedure", Proc. ICASSP-81, pp. 956-959, 1981.

[2] **J.S. Bridle, M.D. Brown** et **R.M. Chamberlain,** "An algorithm for Connected Word Recognition", Proc. ICASSP-82, pp. 899-902, 1982.

[3] **Ch. Gagnoulet** et **M. Couvrat,** "Seraphine : A Connected Speech Recognition System", Proc. ICASSP-82, pp. 887-890.

[4] **J.M. Pierrel,** "Contribution à la reconnaissance automatique du discours continu", Thèse de Doctorat de Spécialité, Université de Nancy, 1975.

[5] **S.B. Davis** et **P. Mermelstein,** "Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences", IEEE Trans. on Acoust., Speech, and Sign. Proc., Vol. ASSP-28 n° 4, pp. 357-366, août 1980.

[6] **Th. Schotola,** "Silbenanlautende und Silbenauslautende Konsonantenfolgen als Entscheidungseinheiten für die automatische Spracherkennung", Thèse de Docteur Ingé-

nieur, Université de Münich, 1980.

[7] **P. Mermelstein,** "Automatic Segmentation of Speech into Syllabic Units", J. Acoust. Soc. Amer., Vol. 58 n° 4, pp. 880-883, octobre 1975.

[8] **P. Combescure,** "20 listes de phrases phonétiquement équilibrées", Revue d'Acoustique n° 56, pp. 34-42, 1981.

[9] **T.W. Anderson,** "Introduction to Multivariate Statistical Analysis", John Wiley and Sons, Sect. 6.6-6.8, 1958.

[10] **J.P. Haton,** "Contribution à l'analyse, la paramétrisation et la reconnaissance automatique de la parole, Université de Nancy, 1975.

[11] **R.E. Crochière et A.V. Oppenheim,** "Analysis of linear digital networks", Proc. IEEE, Vol. 63, pp. 581-595, 1975.

[12] **P.F. Brown, J.C. Spohrer, P.M. Hochschild et J.K. Baker,** "Partial Traceback and Dynamic Programming", Proc. ICASSP'82, pp. 1629-1632.

[13] **V.I. Levenshtein,** "Binary codes capable of correcting deletion, insertions and revearsals", Sov. Phy. Dokl. 10, pp. 707-710, 1966.

[14] **K.S. Fu, S.Y. Lu,** "A clustering procedure for syntactic patterns", IEEE Trans. SMC-7, pp. 734-742, 1977.