85/14

Sc N 85/320 A

Université de Nancy I

U.E.R. Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

UN SYSTEME DE RECONNAISSANCE DE PAROLE CONTINUE POUR LA SAISIE DE TEXTES LUS

BIBLIOTHEQUE UNIVERSITE NANCY SCIENCES

THESE

présentée et soutenue publiquement le 11 septembre 1985

A L'UNIVERSITE DE NANCY I

pour l'obtention du titre de
DOCTEUR D'UNIVERSITE EN INFORMATIQUE
par
François CHARPILLET

Composition du Jury:

Président:	Jean-Paul HATON
Rapporteurs:	Daniel COULON
	Guy PERENNOU
Examinateurs:	Anne-Marie DEROUAULT
	Jean-Sylvain LIENARD
	Jean-Marie PIERREL

Université de Nancy I

U.E.R. Sciences Mathématiques

Centre de Recherche en Informatique de Nancy

UN SYSTEME DE RECONNAISSANCE DE PAROLE CONTINUE POUR LA SAISIE DE TEXTES LUS

THESE

présentée et soutenue publiquement le 11 septembre 1985

A L'UNIVERSITE DE NANCY I

pour l'obtention du titre de
DOCTEUR D'UNIVERSITE EN INFORMATIQUE
par
François CHARPILLET

Composition du Jury:

Président:	Jean-Paul HATON
Rapporteurs:	Daniel COULON
	Guy PERENNOU
Examinateurs:	Anne-Marie DEROUAULT
	Jean-Sylvain LIENARD
	Jean-Marie PIERREL

à mes parents
à Isabelle et Eric

à Gisèle, Anne et Dominique

Je tiens à exprimer ma profonde gratitude à Mr le Professeur Haton qui a bien voulu m'accueillir dans son laboratoire. Je lui suis infiniment redevable d'avoir su inspirer et diriger ce travail.

Que Mr le professeur Pierrel trouve ici l'expression de ma reconnaissance pour le bienveillant intérêt qu'il a porté à mes recherches et l'appui amical qu'il m'a toujours manifesté.

Je remercie Messieurs les Professeurs Coulon et Perennou qui ont bien voulu juger ce travail malgré l'importance des nombreuses tâches qu'ils assument.

Tous mes remerciement vont également à A.M Derrouault, ingénieur du Centre Scientifique IBM de Paris, et à J.S Lienard, directeur de recherche CNRS, pour avoir accepté de participer à ce jury.

Merci enfin à tous mes camarades du CRIN pour l'aide précieuse et amicale qu'ils m'ont tous apportée lors de la réalisation de ce travail.

PLAN

INTRODUCTION

PARTIE A

LA RECONNAISSANCE DE PAROLE

CHAPITRE 1
Introduction

CHAPITRE 2
Les systèmes de la reconnaissance globale de la parole

I. Présentation 13
II. Programmation dynamique et mots isolés 14
III. Mots enchainés 19
IV. Aspect multilocuteur 21
a) Approche globale 22
b) La quantification vectorielle 23

CHAPITRE 3
Le décodage acoustico-phonétique

I. Introduction 24
II. Le décodage acoustico-phonétique mis en oeuvre dans notre système 26
III. Le projet SYSTEXP 31
IV. Modélisation du décodage par un langage de contrôle à base de Frames 33

CHAPITRE 4
Les sources de connaissances dans un système de reconnaissance de parole continue

I. Introduction 34
II. Les connaissances phonologiques 34
a) Les règles d'assimilation
b) Les liaisons à la jonction des mots
c) Les accents régionaux
d) Les règles d'allongement
e) Elision du schwa
III. Le lexique 37
IV. La syntaxe 38
V. La sémantique 39
VI. La pragmatique 40
VII. La prosodie 40

CHAPITRE 5
Présentation de notre projet

PARTIE B

LES SOURCES D'INFORMATION DANS LE MODE EDITION PAROLE

CHAPITRE 1
Introduction

CHAPITRE 2
Le lexique

I. Introduction 46
II La structure du lexique 46
1. Définition du langage de description 47
i) La notation de Backus-Naur étendue
ii) La grammaire du langage de description
2. Représentation du lexique 52
1. Le lexique phonétique
a) Representation d'un mot
b) Représentation générale du lexique
c) La solution mise en oeuvre
d) La structure de donnée
2. Le lexique grammatical
III. La conjugaison automatique des verbes 60
1. Objectif 60
2. Réalisation 61
a) Introduction
b) Syntaxe et sémantique des règles
3. L'interpréteur de règles 65
IV. Le logiciel d'aide à la création du lexique 68
1. Introduction 68
2. Etat du projet 68
3. Perspectives 69
4. La base de données lexicales du "GRECO communication parlée " 69

CHAPITRE 3
Le modèle syntaxique

I. Introduction 71
II. Le modèle positionnel 73
1. Introduction 73
2. L'apprentissage automatique du modèle 74
1. Introduction 74
2. Nature des contraintes positionnelles extraites d'un texte 75
a) Formalisation
b) Les règles de résolution lexicale
c) Les matrices de précédence
3. L'étiquetage automatique 78
a) Définition du problème
b) Etiquetage avec les régles de résolution
c) Etiquetage par les matrices de précédence
d) La solution mise en oeuvre dans notre système
III. Les règles de grammaire 82
1. Introduction 82
2. La structure des règles 82
1. La partie identification 82
2. La partie action 84
a) Introduction
b) Structure des actions
3. Exemples d'application 86
1. Règles prédictives 86
a) Accord en genre et en nombre du nom avec l'article défini
b) Accord du verbe avec le pronom personnel sujet
2. Règles conditionnelles et rétroactives 86
a) Choix de l'auxiliaire pour la conjugaison d'un verbe à un temps
b) La construction des verbes
3. Implantation du sytème de règles 91
1. Edition des règles 91
2. Compilation des règles 93
3. Validation des règles sur le corpus 93
4. Conclusion 93

CHAPITRE 4
Les règles de phonologie et d'évaluation des erreurs du décodage acoustico-phonetique

I. Objectif 94
II. Structure des règles 94
III. Quelques exemples de règles 96
IV. Conclusion 102

PARTIE C

LA STRATEGIE ET LES ALGORITHMES MIS EN OEUVRE DANS LE MODE EDITION-PAROLE

CHAPITRE 1
Présentation générale et architecture adoptée

I. Introduction 104
II. Architecture mise en oeuvre 106

CHAPITRE 2
L'analyse lexicale

I. Objectif 110
II. Méthode de résolution dans Myrtille I. 110
a) Description générale 110
i) Hypothèse d'élision
ii) Hypothèse de substitution
iii) Hypothèse d'insertion
b) Algorithme 113
i) Notation
ii) Algorithme
c) Conclusion 115
III. Recherche lexicale et programmation dynamique 115
1. Introduction 115
2. Quelques éléments utiles sur la programmation dynamique 116
1. Le principe d'optimalité de Bellman 116
2. Application à la recherche dans un graphe 116
3. Application à la reconnaissance de mots isolés 118
1. Analyse mot par mot 118
a) Introduction
b) La fonction de recalage
c) Les contraintes de déplacement
d) Les formules de programmation dynamique
e) Evaluation des pénalisations associées aux arcs
1) Evaluation d'un arc diagonal
2) Evaluation d'un arc vertical
3) Evaluation d'un arc horizontal
f) Contraintes supplémentaires sur la fonction de recalage
g) L'algorithme
h) Introduction d'un flou sur les conditions initiales et finales
2. Analyse sur une structure arborescente 131
a) Introduction
b) Notations
c) Domaine de définition de W
d) Algorithme
4. Application à la reconnaissance de parole continue 135
1. Introduction générale 135
2. Formalisation 137
a) Notations
b) Les contraintes
c) Les relations de programmation dynamique
3. Algorithme 141

CHAPITRE 3
Introduction de contraintes syntaxiques

I. Introduction 149
II. Nécessité d'une stratégie en faisceaux 160
1. Introduction 160
2. Nombre élevé d'états 160
3. Indéterminisme du modèle 151
III. Solution mise en oeuvre 152
1. Programmation dynamique en faisceaux 154
2. La base des superformes 158
3. Le filtrage syntaxique 160
a) Introduction
b) L'hypothétiseur
c) Filtrage après la reconnaissance
IV. Conclusion 163

PARTIE D

LE MODE COMMANDE-PAROLE

DE L'EDITEUR

CHAPITRE 1
Introduction

CHAPITRE 2
Les sources de connaissances

I. Introduction 167
II. La définition syntaxique du langage 167
III. Définition des actions associées au langage de commande 170
1. Introduction 170
2. Les A.T.N et R.T.N 170
3. Mise en oeuvre dans la traductition de requêtes 171
a) Le langage de commande
b) La traduction
c) Un exemple d'application
d) Définition déclarative du R.T.N.
4. Le lexique 174

CHAPITRE 3
Techniques de reconnaissance employées

I. Introduction 177
II. Architecture du système 177
III. Programmation dynamique et syntaxe 180
1. Introduction 180
2. Description formelle 181
1. Sur un automate d'états finis 181
a) Les contraintes internes
b) Les contraintes externes
2. Sur une grammaire hors-contexte 184
a) Introduction
b) Principe général
c) Formalisation
d) Relations de programmation dynamique

CONCLUSION

ANNEXES

ANNEXE 1
Résultats du système de reconnaissance

ANNEXE 2
Les règles de conjugaison

ANNEXE 3
Les règles grammaticales

INTRODUCTION

INTRODUCTION

La parole représente certainement pour l'homme le mode de communication le plus spontané et le plus naturel. Les recherches sur la synthèse et la reconnaissance de la parole constituent de ce fait un enjeu capital susceptible de modifier considérablement les relations homme-machine.

La machine à écrire à commande vocale représente l'un des domaines d'application les plus prometteurs du traitement de la parole. Selon une étude réalisée aux Etats Unis par Strategic Inc, le marché des machines à dicter automatiques atteindrait en 1989 3500 millions de dollars, soit plus de la moitié du marché total des systèmes de reconnaissance de la parole. Les recherches entreprises dans ce domaine sont nombreuses à travers le monde. Des prototypes sont même déjà opérationnels pour la langue japonaise qui présente l'avantage d'être relativement simple d'un point de vue phonétique. Aux Etats Unis, une équipe de recherche IBM s'est également interessée à ce problème avec une approche de la dictée par mots isolés [Jelinek 85]. En France, plusieurs équipes coordonnées par le Gréco travaillent dans ce domaine. Citons le LIMSI à Orsay [Bellity 85] et le CERFIA à Toulouse [Perennou 80]. Pour notre part, au CRIN, nous travaillons depuis 1982 à partir des acquis de Myrtille 1 et Myrtille 2 à la réalisation d'un prototype de machine à dicter automatique orientée vers le traitement de texte. Dans ce système, l'entrée clavier est complétée par une entrée vocale et un système de désignation de type souris.

Nous nous sommes intéressés aux deux aspects inhérents à tout système de traitement de texte: l'édition proprement dite des textes et la manipulation ou la correction des documents saisis. Ces deux modes d'utilisation nous ont conduits dans ce projet à deux approches de la reconnaissance de la parole continue motivées par les différences de complexité du langage traité.

Dans le premier mode d'utilisation de l'éditeur, le système autorise la dictée de textes du français courant de la langue écrite, sans restrictions majeures. Les mots utilisés doivent néanmoins avoir été préalablement définis dans un lexique. Dans l'état actuel du projet, le dictionnaire comporte environ mille racines de mots (les conjugaisons des verbes, les adjectifs et les noms aux différents genres et nombres sont générés automatiquement). Cependant, l'entrée clavier peut toujours suppléer à la restriction sur le nombre de mots connus du système. Parmi les autres caractéristiques du système, notons qu'il est facile d'amener l'utilisateur à suivre diverses contraintes telles que: élocution soignée, ponctuation de la dictée par des pauses entre les groupes syntaxico-sémantiques, etc... Les traitements mis en oeuvre sont spécifiques au problème de la dictée automatique. Ils s'appuient sur une analyse syntaxique locale faisant référence à des matrices de précédence biclasse et triclasse et sur des règles de grammaire. Le niveau sémantique est amplement simplifié et n'intervient plus qu'en tant que source d'information d'appoint. En effet la saisie de textes lus ne nécessite pas la compréhension complète des énoncés. De ce fait, la sémantique n'apparait plus que sous forme de traits généraux. Les ambiguités provenant d'homophones sont levées manuellement à l'aide du système de désignation intégré à l'éditeur.

Dans le mode commande de l'éditeur, le langage traité est artificiel et peut être décrit par une grammaire hors contexte. Son objectif est la correction et la manipulaton de textes édités. Il agit en interaction avec le clavier et le système de désignation de type souris. Les mots à reconnaitre sont isolés lorsqu'il s'agit de mots généraux, ou enchaînés lorsqu'il s'agit de mots de commande. L'apport de la parole est ici important. Elle permet l'appel rapide de menus, ou même l'exécution de commandes complètes. Dans ce mode, la résolution de la reconnaissance est réalisée par

programmation dynamique et est guidée par une analyse syntaxique descendante.

Avant d'en venir aux méthodes que nous avons adoptées pour notre système, nous présenterons de manière générale les sources de connaissances intervenant dans un système de reconnaissance de parole continue (partie A).

Ensuite, nous traiterons dans la partie B les différentes sources d'informations que nous avons intégrées au mode parole-édition du système réalisé. Dans la partie C, nous aborderons les traitements mis en oeuvre et la stratégie d'utilisation des différentes sources d'informations. Nous verrons que les techniques largement utilisées dans le domaine des mots isolés s'avèrent efficaces pour la parole continue. La programmation dynamique et les traitements dérivés en sont un bon exemple. Nous appliquerons la programmation dynamique à deux problèmes importants en reconnaissance de la parole continue:

-la localisation de mots dans un continuum de parole à partir de la transcription phonétique fournie par un décodeur acoustico-phonétique. Cette transcription se présente sous la forme d'un treillis intégrant les diverses solutions plausibles. Plusieurs algorithmes de localisation de mots ont déjà été proposés. La plupart travaillent sur une représentation spectrale de la parole. Les algorithmes présentés ici sont particulièrement adaptés aux traitements ultérieurs mis en oeuvre dans un système de compréhension de la parole.

-la reconnaissance de phrases complètes. Nous verrons plus particulièrement comment généraliser les algorithmes de reconnaissance de mots isolés à la parole continue. Nous insisterons tout spécialement sur l'introduction de contraintes syntaxiques locales. Nous présenterons une heuristique permettant de résoudre les problèmes introduits à la fois par la complexité et l'indéterminisme du modèle.

Dans la partie D, nous décrirons le système de reconnaissance pour le mode parole-commande de l'éditeur. Ici encore, les techniques de programmation dynamique se révèlent particulièrement performantes et

permettent d'atteindre une solution optimale avec des temps de calcul raisonnables. Nous présenterons un analyseur syntaxique descendant de phrases parlées. Ces phrases correspondent à un langage restreint (langage de commande) formalisé par une C-grammaire.

PARTIE A

LA RECONNAISSANCE

DE

PAROLE

1. Introduction

2. Les systèmes de reconnaissance globale de la parole

3. Le décodage acoustico-phonétique

4. Les sources de connaissances linguistiques en reconnaissance de parole continue

5. Présentation de notre projet

CHAPITRE 1

INTRODUCTION

La grande complexité de la reconnaissance automatique de la parole provient de certaines caractéristiques spécifiques du signal vocal qui le rendent difficiles à analyser.

Tout d'abord, la parole se caractérise par une grande variabilité à la fois interlocuteur et intralocuteur. En effet, le conduit vocal présente des différences physiologiques d'un locuteur à l'autre. Par ailleurs, les limitations d'ordre mécanique de l'appareil phonatoire conduisent à une forte dépendance contextuelle des sons émis. Pour s'en convaincre, il suffit de comparer les spectrogrammes des sons /ch/ et /s/ des logatomes /chi/ et /sou/. La différenciation de ces deux sons est impossible si le contexte est inconnu. En effet, la position articulatoire du /i/ tend à rapprocher le /ch/ du /s/, et vice versa le /u/ en ramenant la langue contre le palais approche le /s/ d'une position articulatoire voisine de celle du /ch/. Les phénomènes articulatoires, le contrôle des cordes vocales, le volume sonore ne sont pas précis. Ils varient en fonction du contexte, l'émotion ou la fatigue. Lorque la vitesse d'élocution est trop élevée, les positions articulatoires normales ne sont pas atteintes. Il en résulte de grandes variations difficiles à analyser. Pour une même personne, l'élocution d'une phrase ou d'un mot n'est jamais parfaitement répétitive. En effet, la diction fluctue avec l'humeur, l'état de santé, l'environnement dans lequel évolue le locuteur. Ainsi, la vitesse d'élocution, la fréquence fondamentale du signal de parole, l'énergie du son émis, le rythme sont susceptibles d'évoluer de façon significative. cette variabilité est augmentée lorsque l'on passe d'un locuteur à l'autre. Parmi les facteurs introduisant de grandes différences de prononciation entre différents locuteurs, citons : l'âge, le sexe, l'origine

géographique. Cette grande variabilité de la parole ne permet pas d'effectuer la reconnaissance par une comparaison biunivoque entre le signal vocal et un petit nombre de formes connues une fois pour toute. Le décodage de la parole est un processus bien plus complexe que la lecture de textes écrits pour lesquels la connaissance des lettres de l'alphabet, des chiffres et de quelques symboles suffit au décryptage. Un système de reconnaissance vocale devra comporter une description complète de chaque mot susceptible d'être prononcé. La reconnaissance s'effectue alors par le calcul d'un taux de similitude entre le signal vocal et les éléments de description connus du système.

Une autre caractéristique du signal de parole est l'absence de marques de segmentation entre les mots d'un même énoncé. De ce fait, la localisation des mots dans le signal vocal est une opération difficile et indéterministe nécessitant le recours à des connaissances linguistiques.

Enfin, notons le grand débit nécessaire à la numérisation du signal vocal. Pour une restitution fidèle de la parole, il est nécessaire d'échantillonner le signal à des fréquences de l'ordre de 10 kHz conduisant à un débit supérieur à 100 000 bits par seconde. En effet, la parole véhicule en plus du message sémantique un grand nombre d'informations telles que les caractéristiques du locuteur (âge, sexe,...), l'intonation de la voix, l'émotion, le rythme, etc...

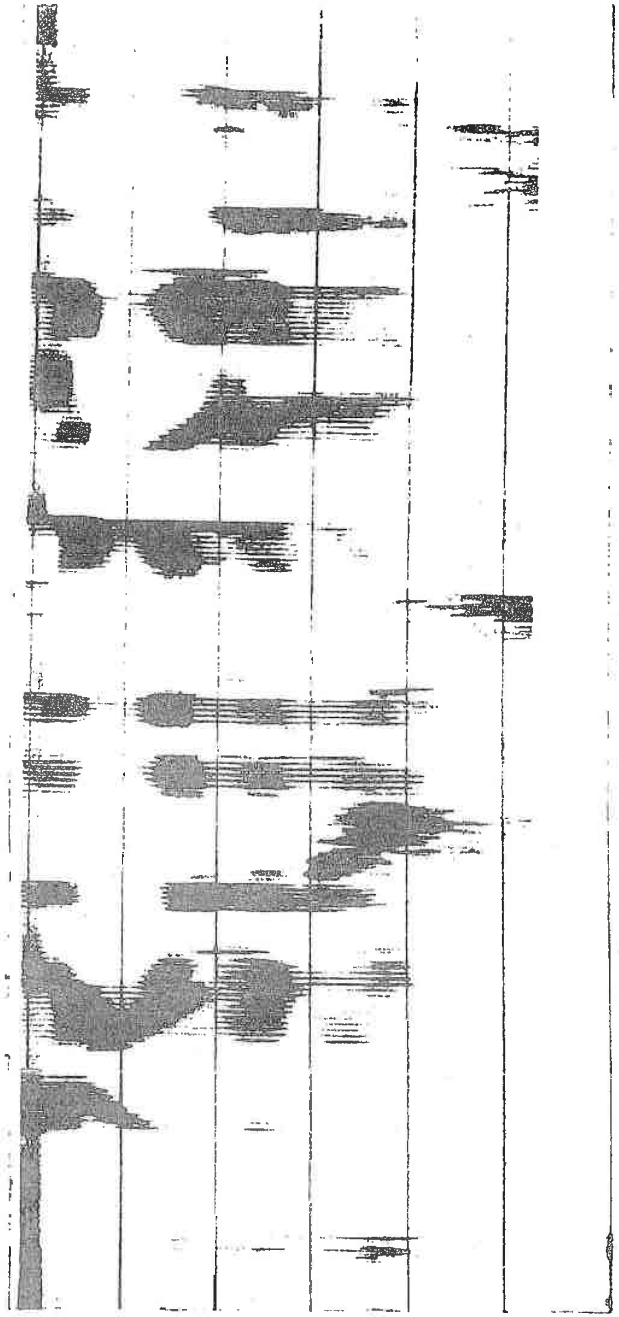
Cependant, si seuls les éléments sémantiques du message sont retenus, le débit informationnel se ramène à quelques dizaines de bits par seconde (codage phonémique ou graphémique du message). Si cette redondance confère une grande robutesse au signal vocal face aux perturbations externes, elle constitue néanmoins l'un des obstacles majeurs au traitement automatique de la parole. En effet, la quantité d'informations à analyser dépasse les capacités de traitement et de stockage des ordinateurs de gamme moyenne. L'une des étapes importantes de la reconnaissance sera donc la réduction du signal vocal par extraction des paramètres les plus pertinents. Ce traitement recouvre ce qui est généralement désigné sous le nom d'analyse acoustique du signal. Les diverses méthodes se regroupent en deux catégories :

- les méthodes de type temporel : elles consistent à extraire du signal temporel un certain nombre de paramètres tels que les passages par zéro, les maxima d'énergie ou les coefficients LPC.
- les méthodes de type fréquentiel : elles ont pour but de représenter le signal dans le domaine fréquentiel. La parole étant non stationnaire, il est nécessaire de recourir à une analyse spectrale à court terme afin de tenir compte de l'évolution temporelle du signal. Dans ce cas, le signal est découpé en tranches temporelles sur lesquelles s'effectue une analyse spectrale. La figure 1 donne un exemple d'un tel spectre.

Parce qu'elle s'apparente le plus au fonctionnement de l'appareil auditif humain, la représentation la plus utilisée dans le codage de la parole est une représentation fréquentielle du signal. Elle permet de ramener le débit d'informations à quelques milliers de bits par seconde. Les méthodes fréquentielles présentent de plus l'avantage d'éliminer la phase du signal, mesure à laquelle l'oreille humaine est peu sensible (les haut-parleurs haute fidélité ne conservent pas la phase des sons qu'ils transmettent).

Le but ultime de la reconnaissance de la parole serait la possibilité d'utiliser un vocabulaire illimité, sans contraintes particulières ni sur le domaine d'application du discours, ni sur l'élocution de l'utilisateur. Le traitement complet de cet objectif serait, pour le moment, voué à l'échec. C'est pourquoi les systèmes développés jusqu'à présent ne traitent que des sous-ensembles abordables du dialogue oral. Les simplifications qui en découlent imposent des contraintes supplémentaires au locuteur qui, par exemple, ne pourra utiliser qu'un nombre restreint de mots, devra prononcer les mots isolément ou devra passer par une phase préalable d'apprentissage avant l'utilisation du système de reconnaissance.

Pour analyser un mot dans le signal vocal, il est nécessaire de connaître les caractéristiques acoustiques et/ou phonétiques de ce mot. Ainsi tous les systèmes de reconnaissance de la parole comportent un lexique dans lequel tous les mots susceptibles d'être prononcés sont définis. La taille du vocabulaire est une limite des systèmes actuels dont il faudra s'affranchir pour de nombreuses applications réelles. Elle constitue un facteur important qui affecte la complexité d'un sytème de reconnaissance. Lorque le nombre de mots du lexique croît, il devient nécessaire de compacter les formes de référence. Il faut également recourir à des techniques proches de l'intelligence artificielle afin de réduire le nombre de mots devant être testé à chaque étape du processus de reconnaissance. La limitation est plus ou moins draconienne selon que l'approche utilisée en reconnaissance est globale ou analytique. Ces deux types de résolution correspondent aux deux modes adoptés dans la représentation des unités morphologiques. Historiquement, la première approche a été une représentation globale des mots. Ceux-ci sont dans ce cas considérés comme une suite de prélévements de type acoustique effectués sur le signal (par exemple, la sortie de bancs de filtres, des coefficients LPC, une FFT). Le vocabulaire de l'application est stocké dans un dictionnaire de formes.



Spectrogramme de la phrase

"Les cigognes passent au-dessus de la route"

Figure 1

La création du lexique nécessite alors une phase d'apprentissage pendant laquelle un ou plusieurs locuteurs prononcent isolément chaque mot à intégrer au lexique.

Cette représentation n'est plus satisfaisante dès que le nombre de mots de l'application croît ou que les énoncés traités ne sont plus ponctués par des pauses atificielles. En effet, le stockage des formes de référence devient trop important. De plus, il est nécessaire d'enregistrer plusieurs représentants de chaque mot afin de tenir compte des variations contextuelles. L'approche analytique résoud ce problème. Elle consiste à représenter chaque mot par une séquence d'unités phonétiques de phonèmes, diphonèmes, de syllabes, etc... Les unités sont dans ce cas décrites une fois pour toutes. Cette description présente deux avantages principaux par rapport à l'approche globale:

- elle est moins dépendante du locuteur. Il est possible de prendre en compte automatiquement les différentes variantes de prononciation grâce à des règles de phonologie,
- la place mémoire nécessaire au stockage du lexique est bien plus faible.

La grande variabilité interlocuteur de la parole est un des problèmes les plus mal résolus. Les systèmes multilocuteurs actuellement disponibles sont assez rares et n'autorisent généralement la reconnaissance que de quelques dizaines de mots. La variabilité est traitée par des méthodes de classification statistique. Elles reposent sur l'enregistrement d'un grand nombre de locuteurs. L'objectif de l'analyse statistique sera de choisir parmi les représentations obtenues celles qui sont les plus représentatives de chaque mot à reconnaître.

Il devient nécessaire alors de segmenter le signal vocal en éléments acoustiques invariants qui devront être étiquetés par des phonèmes, des diphonèmes ou des syllabes selon la représentation adoptée.

La prononciation en mots isolés constitue une simplification notable du problème de la reconnaissance. Dans ce cas, le dialogue se restreint à une suite de mots séparés les uns des autres par des pauses artificielles. La difficulté sur la localisation des mots est alors levée. Le principe de reconnaissance est ramené à un problème classique de reconnaissance des formes. Il s'agit de comparer successivement toutes les formes de référence du lexique à la forme inconnue que constitue le mot à reconnaitre. La forme lexicale qui satisfait au mieux le critère de comparaison avec la forme à identifier est retenue si le taux de ressemblance dépasse un certain seuil.

En parole continue, plusieurs difficultés supplémentaires doivent être prises en compte : Les frontières des mots sont inconnues, les caratéristiques acoustiques des sons varient en fonction du contexte (problèmes de coarticulation). Par ailleurs, le signal de parole ne contient pas toutes les informations nécessaires pour décoder le message de façon non ambigue. En effet, le processus de compréhension fait appel à des niveaux multiples de connaissance afin de lever toutes les ambiguités apparues. Il est nécessaire d'utiliser des connaissances sur le langage, l'environnement et le contexte pour comprendre une phrase. Les sytèmes de reconnaissance calquent ce processus dès qu'ils abordent la reconnaissance de la parole continue ou la reconnaissance en mots isolés sur de grands vocabulaires. Pour résoudre les ambiguités et contrôler la recherche combinatoire des solutions partielles pouvant interpréter les phrases prononcées, il devient nécessaire, pour parvenir à un résultat, de faire coopérer diverses sources de connaissances :

- acoustiques: pour la paramétrisation du signal vocal et la détection parole - non parole,
- phonétiques: pour la détection du signal vocal en unités phonétiques,
- phonologiques: pour la prise en compte des différentes variantes de prononciation aux frontières de mots ou à l'intérieur d'un même mot,
- prosodiques: pour définir les phénomènes de variation du rythme, de l'intensité et de la mélodie de la voix,
- lexicales: pour la définition des caractéristiques des mots de l'application (phonétiques, phonologiques, syntaxiques, sémantiques...),
- syntaxiques: pour rendre compte des structures du langage,
- sémantiques: pour définir le sens des mots et leurs fonctions d'interprétation,
- pragmatiques: pour définir les connaissances particulières du domaine d'application. Elles sont importantes pour la mise en oeuvre des procédures de dialogue.

Ces sources de connaissances ont pour objet de contrôler la recherche combinatoire des solutions pouvant interpréter l'énoncé oral analysé. La connaissance partielle que l'on a de ces différents niveaux de connaissance ne permet pas de définir de manière exhaustive les énoncés appartenant à un langage aussi général qu'une langue naturelle. De ce fait, les systèmes de reconnaissance sont limités à des domaines d'application restreints. On

commence seulement à entrevoir des applications générales appartenant au langage courant grâce aux progrès des sytèmes de décodage acoustico-phonétique. Une définition partielle des niveaux de connaissances ci-dessus est alors suffisante pour interpréter le signal vocal dès lors que la compréhension des énoncés n'est pas nécessaire. La dictée automatique s'insère dans ce cadre.

Nous allons présenter dans le chapitre suivant les différentes techniques de reconnaissance de parole. Nous aborderons d'abord rapidement les systèmes de reconnaissance globale de la parole par mots isolés et enchainés. Ceux-ci sont opérationnels d'un point de vue industriel. Puis nous traiterons l'approche analytique de la reconnaissance. Nous indiquerons les différents niveaux de traitement. Enfin, nous présenterons rapidement l'objectif que nous nous sommes fixés.

CHAPITRE 2

LES SYSTEMES DE LA RECONNAISSANCE GLOBALE DE LA PAROLE

I. Présentation.

Les systèmes de reconnaissance globale de la parole se caractérisent par l'absence d'une étape de décodage phonétique. En effet, dans cette approche, le mot constitue l'unité à extraire du signal. Dans le cas où le vocabulaire est limité, cette technique est employée avec succès et permet d'atteindre des taux de reconnaissance supérieurs à 90%.

Le signal est ici directement traité dans sa représentation acoustique (par exemple sortie d'un banc de filtres, coefficients LPC, coefficients cepstraux). Les formes vocales se présentent généralement comme une suite de vecteurs dont les composantes sont les paramètres acoustiques tirés du signal. Le fonctionnement d'un tel système comprend deux phases: une phase d'apprentissage et une phase de reconnaissance. Pendant l'apprentissage chacun des mots de l'application est prononcé isolément par un ou plusieurs locuteurs selon que le système est mono ou multilocuteur. Les formes acoustiques de chaque mot sont alors mémorisées dans un dictionnaire de formes. La reconnaissance s'effectue ensuite par comparaison de la représentation acoustique des mots prononcés à chacune des formes de référence enrégistrées durant la phase d'apprentissage. Le mot reconnu correspond à l'étiquette du mot du lexique qui présente la meilleure mise en correspondance avec la forme inconnue. Un seuil de dissemblance permet de rejeter un mot si la comparaison est incertaine.

Le principal problème est d'établir un taux de similitude entre les formes prononcées et les formes de référence. Afin de tenir compte des variations en volume et en vitesse entre deux productions, il est nécessaire d'effectuer deux types de normalisation:

- une normalisation en énergie.
- une normalisation temporelle.

Les différences en énergie peuvent être traitées directement par le système d'acquisition ou par une normalisation dans le calcul des distances inter-formes.

La principale difficulté est provoquée par des variations de la parole sur l'axe temporel. La normalisation temporelle a été un thème de recherche important en reconnaissance globale de la parole. Les premiers algorithmes effectuaient une transformation linéaire entre les formes de référence et les formes acoustiques des mots prononcés. Cette approche n'était pas satisfaisante, en effet les distortions temporelles entre deux productions ne sont pas toujours linéaires. Les algorithmes de programmation dynamique donnent une solution satisfaisante à ce problème. Les differences temporelles entre les formes vocales sont éliminées, dans cette approche, en dilatant ou contractant, pas à pas, l'axe temporel de sorte que la distance entre les deux formes soit minimisée.

II. Programmation dynamique et mots isolés.

Les techniques de programmation dynamique introduites dans le cadre plus général de la commande optimale sont très largement utilisées en reconnaissance globale de la parole. Le rôle essentiel de ces algorithmes est de synchroniser les échelles de temps entre deux formes vocales.

Les formes acoustiques de deux mots s'expriment comme deux séquences de vecteurs.

Soit :

$$T=T(1),T(2),\ldots,T(I)$$ la forme à tester

et

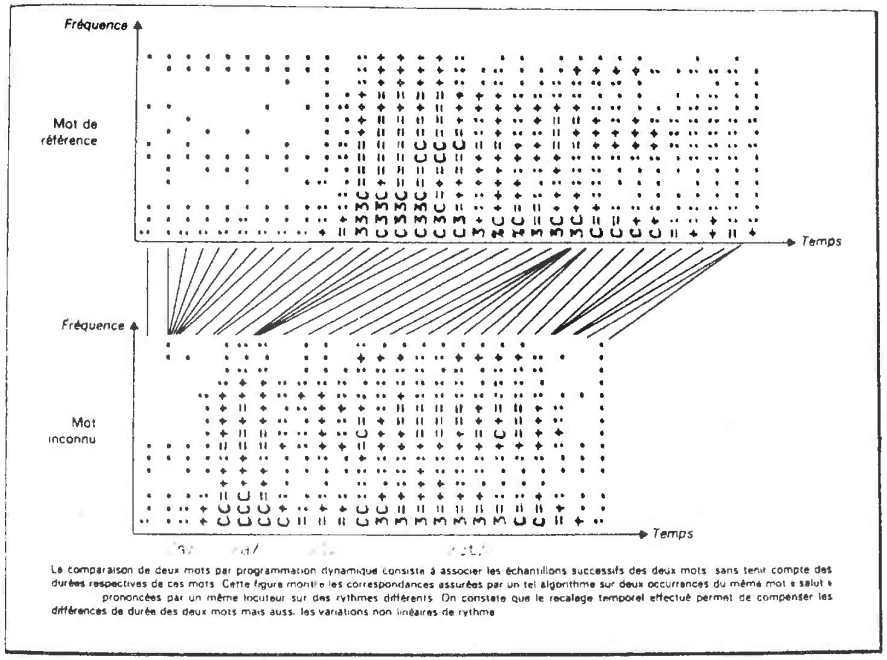
$$R=R(1),R(2),\ldots,R(J)$$ la forme de référence

où I et J représentent respectivement les longueurs en nombre d'échantillons de T et de R.

Le principe de l'alignement temporel est représenté sur la figure 1. Si les formes acoustiques T et R sont respectivement associées à un axe i et j du plan de comparaison, le recalage temporel se ramène à la recherche d'un chemin W:

$$W=W(1),W(2),\ldots,W(K)$$

où $W(k)=[i(k),j(k)]$ représente un point du plan de comparaison par lequel passe le chemin W.



La comparaison de deux mots par programmation dynamique consiste à associer les échantillons successifs des deux mots sans tenir compte des durées respectives de ces mots. Cette figure montre les correspondances assurées par un tel algorithme sur deux occurrences du même mot « salut » prononcées par un même locuteur sur des rythmes différents. On constate que le recalage temporel effectué permet de compenser les différences de durée des deux mots mais auss. les variations non linéaires de rythme

Comparaison par programmation dynamique

Figure 1 tiré de [Haton 85]

La représentation spectrographique des mots utilisée ici est fournie par un Vocoder à 15 canaux couvrant la plage de fréquences 200-5000 Hz. Chaque colonne correspond à 20 ms de parole. L'intensité sonore dans chaque canal est codée de 0 à 7, soit 8 valeurs représentées graphiquement par 8 caractères depuis /blanc/ (pour 0) jusqu'à /%/ (pour 7).

W est appelé la fonction de recalage temporel.

A chaque fonction W du plan de comparaison, il est possible d'associer une métrique permettant d'évaluer un facteur de ressemblance entre les deux formes.

soit:

$$D(W) = \sum_{k=1}^{K} d(W(k)).P(k)$$

où

d(W(k)) représente la distance locale entre les vecteurs i(k) et j(k) des formes T et R

P(k) est un facteur de pondération qui diffère en fonction des transitions locales W(k)-->W(k+1).

Cette métrique D(W) atteint une valeur minimale lorsque le recalage temporel entre les deux formes T et R est parfaitement réalisé. La distance résiduelle entre ses deux formes correspond alors à un taux de ressemblance qui fait abstraction des fluctuations d'ordre temporel.

La normalisation temporelle s'exprime par:

$$(1) \quad D(T,R) = \min_{W} \sum_{k=1}^{K} \frac{d(W(k)).P(k)}{N(k)}$$

où

N(k) est un facteur de normalisation dont le rôle est de rendre indépendante la métrique D(T,R) du nombre de points de la fonction de recalage W.

Des contraintes particulières sont imposées sur la fonction W. En effet, elle doit respecter les structures essentielles du signal vocal: continuité, monotonie, transition d'un segment à l'autre.

Ces restrictions s'expriment par des conditions supplémentaires sur les points :

$$W(k)=[i(k),j(k)]$$

1) monotonie.

$$\begin{cases} i(k-1) < i(k) \\ j(k-1) < j(k) \end{cases}$$

2) continuité

$$\begin{cases} i(k)-i(k-1) < 1 \\ j(k)-j(k-1) < 1 \end{cases}$$

3) conditions aux frontières.

$$\begin{cases} i(1)=1,\ j(1)=1 \\ i(K)=I,\ j(K)=J \end{cases}$$

4) domaine de définition de W.

$$|i(k)-j(k)| < r$$

Cette condition permet de restreindre le recalage temporel dans des limites réalistes.

5) contraintes locales.

Elles se traduisent par des contraintes supplémentaires sur la fonction de recalage W. Elles permettent de limiter localement les taux de dilatation et de compression des formes vocales. Parmi les contraintes les plus utilisées, citons les contraintes de Sakoe et Chiba [Sakoe 71], les contraintes d'Ikatura [Ikatura 75], et les contraintes simples. (cf. figure 2).

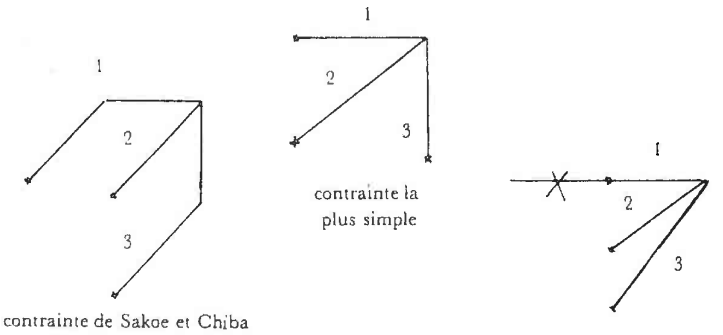


Figure 2 exemples de contraintes

6) les coefficients de pondération.

Il existe deux types de fonctions de pondération:

- les formes symétriques où

$$P(k)=(i(k)-j(k-1))+(j(k)-j(k-1))$$

- les formes asymétriques où

$$P(k)=i(k)-i(k-1)$$

Ces deux formes permettent d'extraire de la formulation (1) le dénominateur W(k) qui devient constant, soit:

$W(k)=I+J$ pour les contraintes symétriques

$W(k)=I$ pour les contraintes asymétriques.

Le problème de minimisation se ramène dans ce cas à chercher:

$$D(T,R) = \min_{W} \sum_{k=1}^{K} d(W(k)).P(k)$$

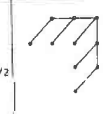
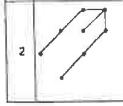
Cette simplification permet de résoudre la minimisation par programmation dynamique de la façon suivante:

$$\begin{cases} g_1(W(1)) = d(W(1)).P(1) \\ g_k(W(k)) = \min_{W(k-1)} [\, g_{k-1}(W(k-1)) + d(W(k)).P(k) \,] \end{cases}$$

où gk correspond à la distance cumulée optimale aboutissant au point W(k).

En intégrant les conditions aux contraintes dans la formulation g(i,j)=g(W=(i,j)) s'exprime par: [Sakoe 78]

SYMMETRIC AND ASYMMETRIC DP-ALGORITHMS WITH SLOPE CONSTRAINT CONDITION $P = 0, \frac{1}{2}, 1,$ AND 2

P	Schematic explanation	Symmetric / Asymmetric	DP-equation $g(i,j) =$
0		Symmetric	$\min \begin{bmatrix} g(i,j-1)+d(i,j) \\ g(i-1,j-1)+2d(i,j) \\ g(i-1,j)+d(i,j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i,j-1) \\ g(i-1,j-1)+d(i,j) \\ g(i-1,j)+d(i,j) \end{bmatrix}$
1/2		Symmetric	$\min \begin{bmatrix} g(i-1,j-3)+2d(i,j-2)+d(i,j-1)+d(i,j) \\ g(i-1,j-2)+2d(i,j-1)+d(i,j) \\ g(i-1,j-1)+2d(i,j) \\ g(i-2,j-1)+2d(i-1,j)+d(i,j) \\ g(i-3,j-1)+2d(i-2,j)+d(i-1,j)+d(i,j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-1,j-3)+\{d(i,j-2)+d(i,j-1)+d(i,j)\}/3 \\ g(i-1,j-2)+\{d(i,j-1)+d(i,j)\}/2 \\ g(i-1,j-1)+d(i,j) \\ g(i-2,j-1)+d(i-1,j)+d(i,j) \\ g(i-3,j-1)+d(i-2,j)+d(i-1,j)+d(i,j) \end{bmatrix}$
1		Symmetric	$\min \begin{bmatrix} g(i-1,j-2)+2d(i,j-1)+d(i,j) \\ g(i-1,j-1)+2d(i,j) \\ g(i-2,j-1)+2d(i-1,j)+d(i,j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-1,j-2)+\{d(i,j-1)+d(i,j)\}/2 \\ g(i-1,j-1)+d(i,j) \\ g(i-2,j-1)+d(i-1,j)+d(i,j) \end{bmatrix}$
2		Symmetric	$\min \begin{bmatrix} g(i-2,j-3)+2d(i-1,j-2)+2d(i,j-1)+d(i,j) \\ g(i-1,j-1)+2d(i,j) \\ g(i-3,j-2)+2d(i-2,j-1)+2d(i-1,j)+d(i,j) \end{bmatrix}$
		Asymmetric	$\min \begin{bmatrix} g(i-2,j-3)+2\{d(i-1,j-2)+d(i,j-1)+d(i,j)\}/3 \\ g(i-1,j-1)+d(i,j) \\ g(i-3,j-2)+d(i-2,j-1)+d(i-1,j)+d(i,j) \end{bmatrix}$

III. Mots enchaînés.

La reconnaissance de mots enchaînés constitue un progrés sensible pour l'utilisateur d'un tel système. En effet, la contrainte consistant à ponctuer les séquences de mots par des pauses artificielles disparait. Les principales applications de ces systèmes sont la reconnaissance des 10 chiffres, ou bien des lettres de l'alphabet (possibilité d'épeler un mot). Les systèmes de reconnaissance de mots enchaînés présentent la particularité par rapport aux systèmes de reconnaissance de parole continue de traiter le signal vocal sous sa forme globale.

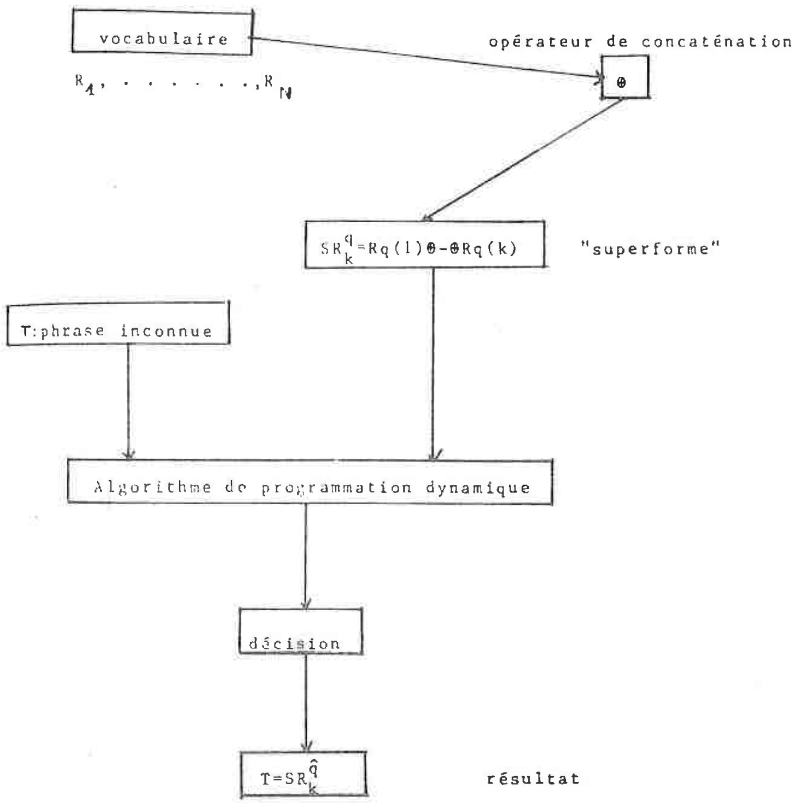
La reconnaissance fonctionne, comme pour les mots isolés, par comparaison de la représentation acoustique de la séquence prononcée à chacune des formes de référence du dictionnaire. Les références ayant été prononcées isolement, la principale difficulté consiste à minimiser les effets de coarticulation aux frontières des mots.

Le principe général de la programmation dynamique s'applique efficacement à ce problème. Il s'agit dans ce cas de trouver parmi l'ensemble des concaténations possibles de mots (les superformes), celles qui interprètent le mieux la forme acoustique inconnue (la phrase prononcée) au sens du critère de comparaison. Le processus général de comparaison tel qu'il a été défini par Sakoe est donné figure 3.

Plusieurs algorithmes ont été proposés. Citons Sakoe qui a développé un algorithme en deux passes [Sakoe 79]:

1) comparaison de chacune des formes de référence à toutes les partitions de la forme inconnue.
2) calcul de la séquence de mots optimale en utilisant les calculs de l'étape 1.

L'algorithme de Myers (the level building DP mathing) présente une amélioration de l'algorithme de Sakoe [Myers 81]. En effet la comparaison dynamique ne s'effectue plus qu'en une seule passe. Le principe de résolution consiste à construire à chaque niveau n de l'algorithme toutes les superformes de référence possibles qui comportent n mots. Pour chacune d'elles, et pour chaque prélèvement de la forme inconnue, il évalue la distance cumulée associée à la mise en correspondance avec les i premiers prélèvements de la forme à tester. A chaque niveau, une minimisation est effectuée, de sorte que pour chaque segment de la forme inconnue, seule la superforme optimale soit retenue.



Principe de la reconnaissance de mots enchaînés d'après Sakoe

Figure 3

La phrase reconnue correspond à la superforme optimale mise en corespondance jusqu'au dernier prélèvement de la forme vocale inconnue.

Enfin plus récemment, Bridle et Nakagawa ont, chacun de leur côté, optimisé l'algorithme précédent. Celui-ci présentait l'inconvénient de calculer plusieurs fois certaines distances locales en raison des retours arrière qui sont effectués sur la forme testée [Bridle 82] [Nakagawa 83]. Ce nouvel algorithme présente de plus l'avantage d'être entièrement orienté temps réel.

IV. Aspect multilocuteur.

Un système de reconnaissance multilocuteur doit être capable de reconnaître un mot ou une phrase indépendamment du locuteur. Vue la grande diversité qui caractérise la parole, cet aspect est encore mal résolu. Les premiers systèmes de reconnaissance globale de la parole impliquaient un passage obligatoire par une phase d'apprentissage complet du vocabulaire pour chaque nouveau locuteur. Même si cette procédure est simple, une telle approche n'est pas satisfaisante et comporte de nombreux inconvénients. En effet, les systèmes de reconnaissance globale visent des applications qui ont pour vocation d' être utilisées par un grand nombre de locuteurs (reconnaissance des chiffres, alphabet). L'enregistrement des formes de référence de chaque locuteur, bien qu'il soit rapide, serait fort couteux.

Par ailleurs, pour les applications autorisant des vocabulaires supérieurs à 50 mots, la phase d'apprentissage serait trop longue et fastidieuse.

De plus, les caractéristiques vocales d'un locuteur peuvent évoluer de façon significative dans le temps, ce qui nécessite une mise à jour constante des références.

La recherche d'invariants, ou la mise en oeuvre de procédures d'adaptation transformant un ensemble de formes de référence fixes constituerait une démarche satisfaisante, proche du processus de compréhension propre à l'homme. La connaissance partielle que l'on a de ces processus ne permet pas encore d'adopter en reconnaissance de parole une telle approche. Les systèmes actuels mettant en oeuvre l'aspect multilocuteur traitent la variété interlocuteur par une approche statistique. Elle consiste à enregistrer un grand nombre de locuteurs ayant des caractéristiques vocales différentes (hommes, femmes, jeunes, vieux...).

Actuellement, deux approches permettent le traitement des formes vocales ainsi obtenues:

- une approche de type globale.
- une approche analytique par quantification vectorielle et apprentissage par des modèles de Markov cachés.

a) approche globale.

Le fonctionnement général d'un tel système se décompose en deux phases distinctes:

- apprentissage qui s'effectue une fois pour toute.
- la reconnaissance proprement dite.

Lors de l'apprentissage, un grand nombre de locuteurs prononcent isolément chaque mot de l'application. Les formes acoustiques obtenues sont comparées les unes aux autres grâce à un algorithme de comparaison dynamique globale. Le but est de ne retenir parmi les formes enrégistrées qu'un nombre limité de représentants, afin d'optimiser la place nécessaire et les temps de calcul des traitements ultérieurs. Deux types de méthodes sont couramment utilisés pour construire le dictionnaire de formes représentatives:

- les méthodes de nuées dynamiques (clustering).
- les méthodes incrémentales (covering).

La reconnaissance d'un mot prononcé s'effectue par comparaison dynamique entre la forme acoustique inconnue et l'ensemble des formes de référence issues du traitement statistique. Un mot est retenu s'il se trouve dans la zone d'influence d'un représentant du dictionnaire.

A Nancy, nous travaillons sur un projet de réponse vocale via le téléphone pour une banque. Le dialogue est fortement dirigé. L'initiative du locuteur se limite à répondre aux questions par ''oui'' ou ''non'', ou bien à énoncer une séquence de chiffres [Divoux 85].

b) la quantification vectorielle.

La quantification vectorielle est une technique utilisée pour la compression d'un signal lorsqu'il est représenté par une suite de vecteurs. D'abord dans une phase d'apprentissage, un ensemble de prototypes représentant chaque variété élémentaire du signal est déterminé par une analyse statistique. Ensuite, le signal est segmenté en partitions qui sont comparées à chacun des prototypes. Chaque segment peut se réduire alors à l'étiquette du prototype qui lui est le plus proche.

De telles techniques sont utilisées en reconnaissance de parole. Le dictionnaire de prototypes est construit automatiquement à partir d'un corpus d'apprentissage.

Chaque mot de l'application peut alors se représenter comme une suite d'étiquettes renvoyant à un prototype du dictionnaire.

Une façon de prendre en compte la variabilité inter et intralocuteur, est de modéliser les unités lexicales par un modèle de Markov caché. Rappelons qu'un tel modèle est constitué par un automate d'états finis. A chaque état est associée la probabilité qu'a un prototype de se réaliser à l'instant t, compte tenu de l'état courant. Chaque transition de l'automate est également affectée d'une probabilité[Levinson 83].

CHAPITRE 3

LE DECODAGE ACOUSTICO-PHONETIQUE

I. INTRODUCTION

Comme nous l'avons déjà souligné précédemment, deux méthodes conceptuellement différentes permettent d'aborder la reconnaissance de parole. Toutes deux sont fondées sur la segmentation du signal en unités:

- mots pour les méthodes globales
- syllabes, diphones, phonèmes pour les méthodes analytiques.

Pour la reconnaissance de vocabulaires limités, l'extraction directe de mots dans le signal a été utilisée avec succès. Cependant, dès lors que la reconnaissance de grands vocabulaires ou de parole continue est envisagée, il devient nécessaire d'extraire du signal des unités plus fines: c'est le rôle du décodage acoustico-phonétique.

La transformation du signal vocal en une suite d'étiquettes phonétiques est une opération fondamentale dans les systèmes analytiques de reconnaissance automatique de la parole. De cette étape du traitement dépendent les performances globales du système. En effet, le décodage constitue le premier maillon du processus de reconnaissance. L'efficacité des autres niveaux de traitement (lexical, syntaxique, sémantique..) est étroitement liée à la qualité du décodage. La chaîne ou le treillis phonétique qui en résulte, contient toute l'information extraite du signal. Toute erreur à cette étape augmente donc de façon significative l'indéterminisme des traitements ultérieurs.

Historiquement, la première approche de ce problème a été la mise en oeuvre de techniques de reconnaissance des formes par extraction et classification de paramètres acoustiques tels que l'énergie dans certaines bandes de fréquences, la position des formants pour la reconnaissance des voyelles, les centres de gravité (détection des fricatives), les valeurs de la fréquence fondamentale pour dissocier la reconnaissance de sons voisés et non voisés, les passages par zéro, etc...

A cette recherche d'indices acoustiques généraux, le décodeur, selon les approches, complète l'analyse par l'utilisation alternative ou simultanée de techniques de comparaison à des prototypes. Ceux-ci se calculent lors d'une phase d'apprentissage pendant laquelle un locuteur prononcera plusieurs fois, dans des contextes différents, chacun des phonèmes de la langue choisie. Un prototype sera par exemple constitué par un spectre moyen du phonème qu'il décrit.

Un tel traitement n'est cependant pas totalement satisfaisant. En effet, ces décodeurs se caractérisent par un nombre élevé de paramètres liés au locuteur. De ce fait, l'aspect multilocuteur est difficilement intégrable, même s'il existe des procédures d'adaptation automatique [Pister 84].

Par ailleurs, le phonème est une unité trés variable et fortement influencée par le contexte phonétique dans lequel il apparait. En effet, pour des problèmes liés à l'articulation, l'image acoustique d'un phonème sera modifiée par les deux phonèmes antérieur et postérieur du contexte. Dans certains cas, deux phonèmes différents peuvent se réaliser quasi identiquement, et seul le contexte permet la discrimination (exemple du /ʃ/ et du /s/ dans les logatomes /chi/ et /sou/). Dans une approche de type reconnaissance des formes, la prise en compte de phénomènes contextuels nécessite de recourir à un nombre prohibitif de références. Le français comportant environ 30 phonèmes, il faudrait étudier de l'ordre de 30x30x30 contextes différents pour chaque locuteur.

Des méthodes de décodage en unités plus grandes que le phonème permettent de prendre en compte partiellement les problèmes de coarticulation. Parmi ces méthodes, citons l'analyse par diphonèmes [Schwartz 1979][Mariani 1981] ou par demi-syllabes [Rabiner 1981], [Ruske 81]

Dans tous ces systèmes, les taux de réussite sont encore insuffisants (moins de 70% en monolocuteur). Si cette approche est satisfaisante dans les systèmes de reconnaissance travaillant sur les langages artificiels très contraints, le passage à la reconnaissance de grands vocabulaires et

l'utilisation de langages quasi-naturels nécessitent des taux de reconnaissance bien supérieurs.

Une nouvelle voie de recherche s'est récemment développée. Elle consiste à incorporer dans les décodeurs des connaissances phonétiques de la parole. Cette orientation vers les techniques d'intelligence artificielle est encouragée par les performances obtenues par des experts humains en phonétique dans la lecture de spectrogrammes. De l'observation d'un ensemble de lecteurs de spectrogrammes, il ressort que:

- le signal acoustique est riche en informations phonétiques,
- de telles informations peuvent être extraites à partir d'une représentation spectrographique du signal,
- le processus d'extraction de l'information fait référence à des règles explicites qui peuvent être enseignées.

[ZUE 81]

De ce fait, l'amélioration des algorithmes de décodage peut être entreprise par l'analyse et la modélisation du savoir faire d'un expert en lecture de spectrogrammes. En effet, les performances atteintes par un expert humain dépassent les 80% en contexte multilocuteur. Nous donnons figure 1 un exemple de spectrogramme de parole avec les transcriptions phonétiques fournies par un expert phonétitien extrait du projet SYSTEXP développé dans notre équipe [Carbonnell 85] (cf paragraphe III).

II. Le décodage acoustico-phonétique mis en oeuvre dans notre système.

Le décodeur acoustico-phonétique que nous utilisons est celui mis en oeuvre dans le système Myrtille II [LAZREK 83,84]. Il procède à la segmentation du signal en phonèmes. Afin d'augmenter le taux de réussite de la reconnaissance, le décodage fournit pour chaque segment identifié jusqu'à trois candidats. Le système de transcription phonétique s'effectue à partir de spectres à court terme fournis en temps réel par un vocodeur à 16 canaux. Le principe de la reconnaissance peut se schématiser comme indiqué figure 3 et 4. La transcription s'effectue en deux étapes:

1) pour chaque échantillon de parole de 10 ms:

- calcul du trait voisé / non voisé,
- calcul de paramètres acoustiques (cf. figure2),
- recherche, par un calcul de distance entre le prélèvement et les prototypes définis lors d'une phase d'apprentissage, des trois phonèmes les plus proches. L'identification du segment pourra

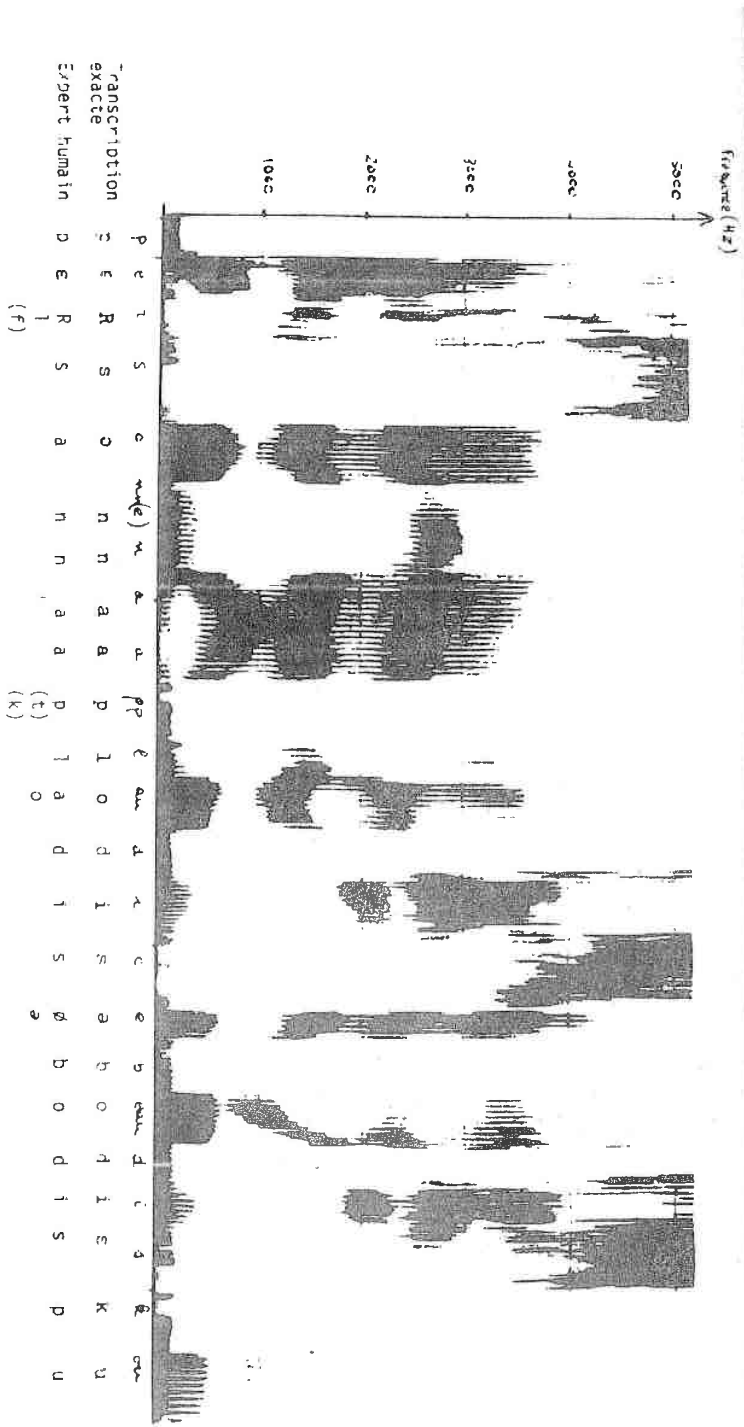


Figure1 : exemple de spectrogramme de parole avec la transcription exacte et la transcription fournie par l'expert

dans certains cas s'effectuer à partir de traits suffisamment caractéristiques pour désigner un phonème.

2) regroupement des étiquettes par un processus de décision majoritaire parmi les trois phones retenus à chaque centiseconde.

Le processus d'identification est orienté grâce à des algorithmes de segmentation du signal en trois macroclasses: voyelles, consonnes voisées, consonnes non-voisées.

En premier lieu, la détection du trait voisé/ non-voisé donne lieu à une première partition entre:

- les échantillons appartenant à la classe des consonnes non-voisées (f,s,ʃ ,p,t,k)
- les échantillons de type voyelle ou consonne voisée (v,Z,z,b,d,g,n,m,R,L)

Ensuite, la phase d'identification des voyelles, secondée par un algorithme de détection des noyaux vocaliques, parachève la segmentation en macro-classes. Chacun des échantillons du signal de type voisé est identifié par deux critères:

- un critère énergétique,
- un critère de similitude par rapport aux prototypes des voyelles.

Le critère de similitude nécessite le calcul de la plus petite distance entre le prélèvement et les références des voyelles. Si la distance est trop grande, l'échantillon est affecté à la classe des consonnes voisées.

Dans chaque macro-classe, l'identification des unités phonétiques se réalise par trois processus complémentaires:

- identification par le calcul de la distance du prélèvement par rapport à chaque prototype de la macro-classe. Le segment est étiqueté par les trois références les plus proches.
- segmentation par le calcul d'une distance inter-spectrale entre deux échantillons successifs et l'extraction de traits.
- regroupement des prélevements par un processus de décision majoritaire.

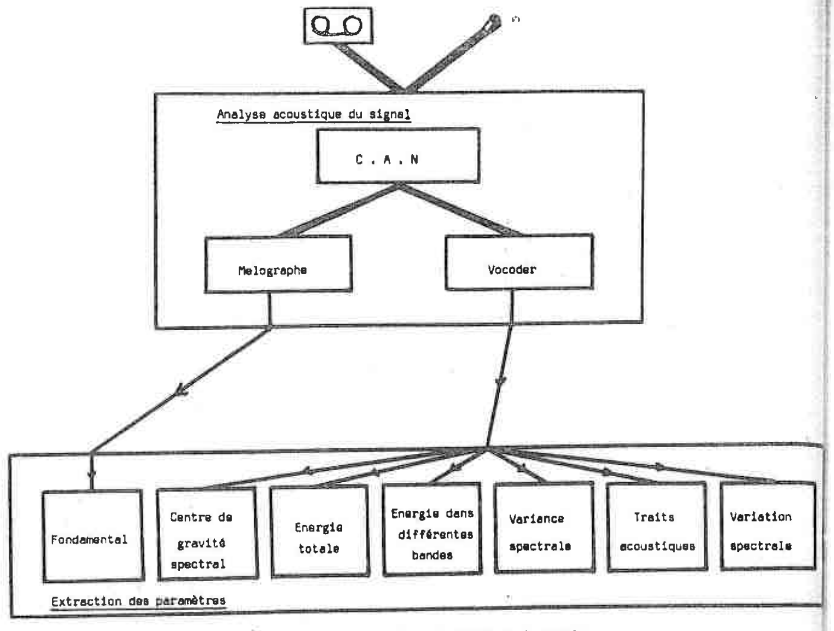
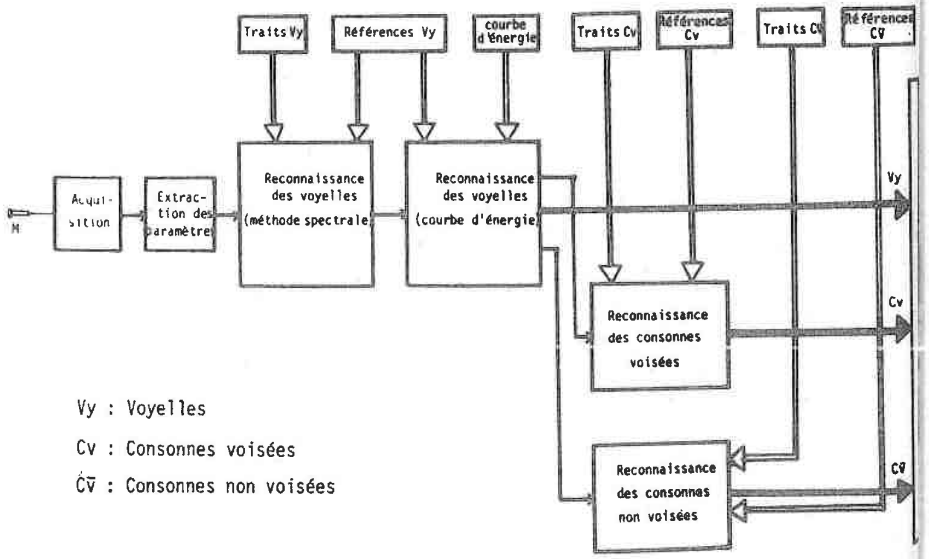


Figure2 Analyse acoustique et extraction des paramètres du signal de la parole

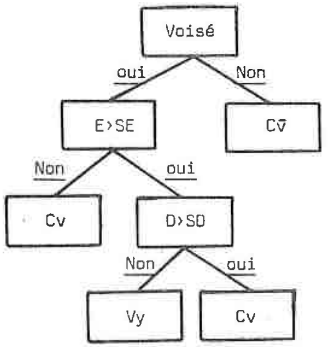


Vy : Voyelles

Cv : Consonnes voisées

Cv̄ : Consonnes non voisées

Figure3 Schéma général du système de reconnaissance phonétique



avec -

E : énergie d'un échantillon

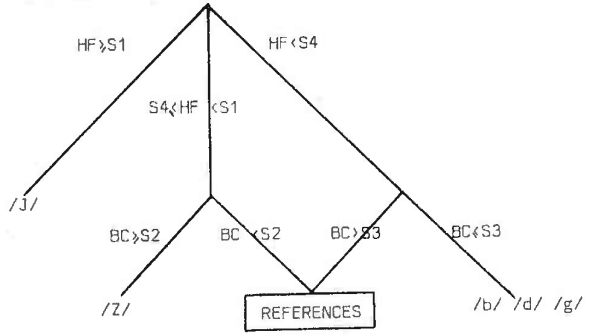
D : distance entre l'échantillon et les références des voyelles

Cv : consonne voisée

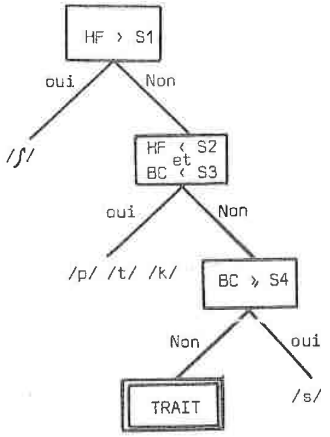
Cv̄ : consonne non voisée

Vy : voyelle.

processus de reconnaissance des voyelles



reconnaissance des consonnes voisées



où HF : l'énergie dans les hautes fréquences

BC : centre de gravité de la distribution d'énergie dans le spectre

Reconnaissance des consonnes sourdes

Figure 4

RESULTATS

Si l'on prend en compte, pour chaque segment identifié, les trois candidats phonétiques possibles, les taux de reconnaissance avoisinent les 70%. On est cependant encore loin des performances nécessaires à une bonne compréhension des phrases prononcées. Une étude de Lienard (1975), consistant à faire entendre à des auditeurs humains des chaines phonétiques entachées d'erreurs, a montré que la bonne compréhension d'une phrase nécessite la présence d'au-moins 85% de bons phonèmes.

Pour notre part, l'évaluation de notre système montre qu'au-dessus de 20% d'erreurs dans le treillis phonétique, l'utilisation des niveaux supérieurs de traitement ne permettent plus de compenser les insuffisances du décodage acoustico-phonétique.

Cependant, dans le cadre de langages artificiels restreints tel celui défini dans Myrtille I [Pierrel 75], les performances d'un tel décodeur sont nettement améliorées. En effet, il est dans ce cas possible, lors de l'apprentissage des prototypes, de référencer la quasi totalité des contextes phonétiques.

III. Le projet SYSTEXP.

Depuis 1982, dans notre équipe, un groupe de travail a été mis en place dans le but de réaliser un système expert en décodage phonétique. Ce système est déstiné à intégrer, l'expertise mise en oeuvre par un phonétitien (François LONCHAMP) lors de la lecture de spectrogrammes.

L'analyse de l'expert en lecture de spectrogrammes permet d'enrichir les procédures de décodage:

- en augmentant le nombre et la qualité des informations acoustiques pertinentes relatives à la segmentation et à l'identification phonétique.
- en mettant en évidence des stratégies d'analyse efficaces.

L'expertise a été recueillie à partir de 50 spectrogrammes de phrases. Cinq locuteurs ont enregistré chacun 10 phrases phonétiquement équilibrées du corpus de Combescure [Combescure 81]. Les performances de l'expert (près de 80%) se sont avérées être largement supérieures à celles des décodeurs actuels.

Le système réalisé comprend

- un ensemble de modules de prétraitement.
- une base de connaissance
- un moteur d'inférence.

i) Les modules de prétraitement.

Ils consistent à réaliser une première segmentation du signal en macro-classes. Les procédures de traitement ne faisant pas appel à des informations contextuelles sont implantées procéduralement. Trois algorithmes permettent d'effectuer la segmentation:

- Novoca pour la détection des noyaux vocaliques et le calcul de la durée vocalique moyenne.
- Plosi pour la détection des plosives.
- Frica pour la détection des fricatives.

ii) La base de connaissance.

Les connaissances expertes sont formalisées sous la forme de régles de production du type:

prémisses - - - > conclusions.

Les prémisses sont constituées soit par des conclusions obtenues lors d'une étape précédente du raisonnement, soit par des mesures effectuées sur le signal. Ces dernières sont obtenues grâce à l'appel de procédures de traitement (calcul de burst, formants, coefficients LPC, etc...)

Il existe deux types de règles:

- des règles dont les conclusions donnent lieu à une liste de phonèmes.
- des règles dont les conclusions donnent lieu à une action qui modifie soit la segmentation proposée par le module de prétraitement, soit l'arbre de déduction (treillis phonétique).

iii) Le moteur d'inférence.

La stratégie de déduction mise en oeuvre doit se rapprocher le plus possible de celle de l'expert. Elle doit permettre

- la remise en cause de la segmentation à tout moment,
- le déroulement en parallèle de plusieurs lignes de raisonnement,

- l'introduction d'un intéterminisme dans le raisonnement.

En effet, les règles expertes sont contextuelles. Elles portent à la fois sur le contexte gauche et le contexte droit du segment analysé. L'analyse étant de gauche vers la droite, les éléments du contexte droit sont inconnus et donnent lieu à des hypothèses qu'il conviendra par la suite de vérifier. Par ailleurs, un contexte gauche peut être constitué de plusieurs hypothèses qu'il conviendra de confirmer ou d'infirmer.

Pour répondre à ces différentes exigences, le mode de chaînage est mixte:

- chaînage avant (analyse guidée par les indices présents dans le signal),
- chaînage arrière pour vérifier la validité des hypothèses.

Le résultat de l'analyse est constitué par un treillis phonétique qui rend compte des différentes alternatives envisagées: aussi bien les identifications multiples et leurs axes que les segmentations parallèles retenues.

IV. Modélisation du décodage par un langage de contrôle à base de Frames

L'implantation d'un système expert en décodage acoustico-phonétique pose des problèmes d'efficacité surtout en ce qui concerne les contraintes de fonctionnement en temps réel. Par ailleurs, une fois l'expertise acquise, une telle approche n'a plus d'intérêt si ce n'est par le côté didactique. C'est pourquoi, en complément du projet Systexp présenté ci-dessus, notre équipe travaille également sur l'élaboration d'un décodeur par l'utilisation d'un langage de contrôle à base de frames [Damestoy 85]. Une approche similaire limitée à l'extraction d'indices acoustiques a été mise en oeuvre par R. Demori [Demori 1983].

L'intérêt d'une telle approche est constitué par la possibilité de raisonner à différents niveaux de traitement (connaissances et méta-connaissances) et de structurer au mieux la base de connaissances. De plus, ce formalisme permet aisément d'attacher des procédures de traitement aux connaissances déclaratives. Ce dernier point est très important en décodage phonétique où il est nécessaire d'interconnecter connaissances déclaratives et procédurales (extraction multi-indices).

CHAPITRE 4

LES SOURCES DE CONNAISSANCES DANS UN SYSTEME DE RECONNAISSANCE DE PAROLE CONTINUE.

I. Introduction.

Le décodage phonétique permet d'extraire du signal vocal des unités phonétiques telles que les phonèmes ou les syllabes. Ce problème est résolu par l'utilisation mixte de connaissances acoustiques (extraction de traits pertinents) et phonétiques (description des unités à reconnaître). La redondance de la parole permet grâce à l'utilisation de connaissances linguistiques de pallier en partie la faiblesse actuelle de cette phase de traitement. Ces connaissances sont également nécessaires pour accéder au sens du message, ou pour lever certaines ambiguités telles que celles dûes à des homophones, ou à l'absence de marques de segmentation entre les mots : une même représentation phonétique peut correspondre à plusieurs énoncés différents:

L a m a L a d R w a

le maladroit ou *le mâle adroit* ou *le mâle a droit*

II. Les connaissances phonologiques.

Elles recouvrent l'ensemble des connaissances qui permettent d'établir les variantes des différentes prononciations admissibles d'un mot, dès lors qu'elles agissent au niveau global du phonème (par opposition aux connaissances phonétiques).

Les phénomènes étudiés regroupent:

- les règles d'assimilation : en français les assimilations de sonores et de nasalité,
- l'étude des liaisons entre mots,
- les allongements,
- l'élision du schwa,
- etc...

Dans certains systèmes, on intègre également dans ces connaissances les altérations phonologiques dûes à la conjugaison des verbes et aux déclinaisons (singulier, pluriel, féminin, masculin) pour les autres mots. [MYRTILLE II.].

a) Les règles d'assimilation.

i) les assimilations sourde-sonore.

De telles assimilations apparaissent lorsque deux consonnes se succèdent avec un trait de voisement différent, soit:

consonne sourde consonne sonore.

consonne sonore consonne sourde.

Dans ces deux cas, en français, la première consonne adoptera le trait de voisement de la seconde.

exemples:

sonore sourde

médecin	m e d ə s ɛ̃
	m e t s ɛ̃ lorsqu'il y a élision du schwa.
couverte de neige	K u v e R t ə d ə n ɛ z
	K u v e R d d n ɛ z

sourde sonore

aqueduc	a K ə d y K
	a g d y K
décevant	d e s ə v ɑ̃
	d e Z v ɑ̃
paquebot	P a k ə b o
	P a g b o

Il y a une exception à cette règle lorsque la deuxième consonne est un /R/ ou un /L/. Dans ce cas, la règle est inversée.

Dans "plein", le /L/ devient sourd.

ii) les assimilations de nasalité.

En français, elles s'expriment par:

V C occlusive C --› V C nasale C.

exemples:

banque de France --› b ɑ̃ k d ə f R ɑ̃ s

dans ce cas l'assimilation s'effectue en deux étapes :

1) assimilation de sonorité --› b ɑ̃ g d ə f R ɑ̃ s

2) assimilation de nasalité --› b ɑ̃ ɳ d ə f R ɑ̃ s

Pentecôte	p ɑ̃ t ə K o t
	p ɑ̃ n K o t
22	v ɛ̃ t d o
	v ɛ̃ n d o
subjonctif	S y b z õ K t i f
	S y b z õ ɳ t i f

b) les liaisons à la jonction des mots

Elles recouvrent à la fois les altérations dûes aux phénomènes d'assimilation mais également les liaisons apparaissant lors de la formation des pluriels :

mes amis

m e Z a m i

c) les accents régionnaux

"bandit" se prononce différement à Paris et à Marseille :

b ɑ̃ d i

b ɑ̃ ɳ d i

d) les règles d'allongement.

Si la phrase prononcée comprend plus de 6 à 9 syllabes, celle-ci est scindée en groupes (généralement syntaxiques) séparés par des pauses.

En français, les voyelles dans les syllabes finales de groupes peuvent être allongées dans deux cas:

- lorsqu'elles sont suivies des consonnes [R,Z,z,v]
- lorsque certaines voyelles [a, , ,oe], [o,o] sont suivies d'une consonne ou d'un groupe de consonnes plus L ou R, celles-ci sont allongeables.

exemple:

j'en ai marre

c) élision du schwa.

Le schwa (ə) est très souvent omis lorsque la vitesse d'élocution est rapide:

exemple: je voudrais z v u d R e

L'ensemble de ces variations phonologiques peut être regroupés en trois catégories:

- les élisions.
- les insertions.
- les substitutions.

Ce niveau de connaissances est lié à d'autres sources de connaissances:

- syntaxe.
- lexique.
- prosodie.

III. Le lexique.

Il recouvre toutes informations relatives aux mots de l'application:

- *phonétique* : pour la recherche ou la vérification d'un mot dans la représentation fournie lors du décodage acoustico-phonétique.

- *phonologique* : les différentes prononciations possibles d'un mot doivent être intégrées au lexique.

- *graphique* : le but de la reconnaissance étant de passer du signal à une représentation orthographique du message vocal, il est donc nécessaire de connaître pour chaque mot sa forme graphique.

- *syntaxique* : pour la connaissance des classes grammaticales et des traits syntaxiques des mots. Ces informations relient le lexique au modèle syntaxique.

- *logico-sémantique* : informations relatives au sens des mots.

Plusieurs représentations sont possibles selon le critère choisi. Les unités intégrées au lexique peuvent être:

- soit tous les mots à toutes les formes du singulier, du pluriel, du féminin et du masculin pour les noms et adjectifs, et à toutes les formes conjuguées pour les verbes.
- soit des radicaux et des terminaisons types à partir desquels les mots seront générés.

Les différentes variantes phonologiques d'un mot peuvent être intégrées directement dans le lexique ou engendrées au fur et à mesure des besoins.

La représentation peut être optimisée en factorisant les débuts de mots ou les fins de mots.

Le codage choisi doit être un compromis entre plusieurs critères tels que l'espace mémoire occupé, la possibilité d'accès rapide aux diverses informations (phonétiques, syntaxiques, sémantiques...).

IV. La syntaxe.

La syntaxe sert à formaliser la structure du langage traité. Elle définit toutes les séquences de mots acceptables. En reconnaissance de parole, le niveau syntaxique permet de limiter le nombre de solutions possibles pouvant interpréter le treillis phonétique. Il réduit à chaque niveau du traitement le nombre de mots autorisés.

Les langages modélisés sont de trois types:

- les langages artificiels.
- les langages pseudo-naturels.
- les langages naturels.

Parmi les modèles les plus utilisés par la représentation de la syntaxe, citons:

- les grammaires d'états finis.
- les grammaires hors contexte.
- les ATN [Woods 70].

- les RPN [Pierrel 82].
- les prédicats du premier ordre [Colmerauer 79].
- les TNF [Hayes-Roth].
- les grammaires en chaines [Harris 71] [Salkov 73].
- les grammaires transformationnelles [Chomsky 59].
- les matrices de précédences [Andrewsky 73] [Fluhr 77].
- les grammaires de cas [Fillmore 67].

La liaison de la syntaxe avec les autres niveaux de connaissances est importante avec:

- la sémantique pour accéder à la structure profonde des énoncés.
- la prosodie (par exemple, étude des pauses entre les groupes syntaxiques)
- le lexique et la phonologie.

V. La sémantique.

Elle regroupe l'ensemble des informations liées à la signification des mots et aux concepts sous-jacents. Le rôle de la sémantique est double dans un système de reconnaissance de parole continue:

- d'abord, dans la détermination du sens des énoncés. Cet aspect est important dans les procédures de dialogue. La compréhension du sens du message passe par une interprétation guidée par les connaissances sémantiques propres au domaine d'application.

- ensuite, dans la définition des suites de mots acceptables. Les contraintes sémantiques permettent de confirmer ou d'infirmer certaines hypothèses émises par d'autres niveaux de traitement. En ce sens, elle est un complément utile du niveau syntaxique. D'ailleurs, les linguistes se posent encore la question de savoir s'il existe réellement une frontière entre syntaxe et sémantique.

Le niveau sémantique permettra donc le rejet de phrases syntaxiquement correctes mais sans aucune interprétation sémantique. Par exemple, dans l'application météorologique traitée dans Myrtille II le syntagme :

orage de soleil

syntaxiquement correct sera rejeté sur un critère sémantique.

Les solutions proposées pour la compréhension des langues écrites sont adaptables au problème de la reconnaissance de parole. Citons entre autres comme méthodes de représentation et d'analyse sémantique:

- les descriptions logiques.
  - prédicats du 1er ordre [Prolog].
  - logiques multivaluées [ensemble flou].
  - logiques modales.
  - logiques non monotones.
- les descriptions procédurales [Winograd 72]
- les dépendances conceptuelles [Schank 75]
- les descriptions stéréotypées:
  - scripts [Schank 77]
  - plans
  - Frames [Minsky 75]
- les réseaux sémantiques.

Par ailleurs, dans les domaines d'application limités, certaines représentations adoptées pour la syntaxe peuvent s'avérer utilisables pour la sémantique:

- grammaires et automates.
- ATN sémantiques.

VI. La pragmatique.

Le niveau pragmatique englobe les informations relatives au contexte de l'univers et du dialogue.

Le traitement des contraintes pragmatiques doit permettre:

- l'interprétation des structures profondes en coopération avec le niveau sémantique.
- la résolution d'ambiguités résultant d'une référence pronominale.
- la prédiction partielle du comportement de l'utilisateur en s'appuyant sur le dialogue.

VII. La prosodie.

La prosodie concerne l'étude de certaines caractéristiques acoustiques de la voix telles que les variations de la fréquence fondamentale, les variations d'intensité et le rythme. Elle constitue une sorte de ponctuation de la parole intervenant à la fois pour la compréhension du message et comme source d'informations confirmant les autres niveaux du processus de

compréhension.

La prosodie est sans doute la source de connaissances la plus difficile à utiliser dans un sytème de reconnaissance de parole. En effet, les évènements prosodiques sont très variables d'un locuteur à l'autre. Leur réalisation est rarement obligatoire et en grande partie imprévisible. Cependant leur apparition peut fournir des informations utiles et complémentaires aux autres niveaux de traitement. Les indices prosodiques fournissent des informations permettant

- d'extraire la structure générale de la phrase (forme énonciative, interrogative, impérative,...),

- de lever certaines ambiguités dues à la possibilité d'interpréter de plusieurs façons une même chaine phonétique :

par exemple /g R ã t a m i/ peut signifier ''grand ami'' ou ''grand tamis'',

- de trouver des marqueurs tels que les marqueurs de fin de groupes, les pauses , les marqueurs d'accent qui se situent toujours sur la première syllabe de type CV d'un mot.

Ce dernier point est important à deux niveaux de la reconnaissance de parole :

- au niveau syntaxique, lorque les informations prosodiques indiquent une frontière nette, il est possible d'éliminer certaines possibilités syntaxiques. En particulier, lorsque le locuteur lit un texte, il découpe naturellement l'énoncé en groupes syntagmatiques simples s'il lui est demandé d'être intelligible [Perennou 82].

- au niveau du mot où il est possible de limiter la plage phonétique de recherche d'un mot (MYRTILLE II) [Carbonell 82]. Les informations prosodiques permettent ici d'éliminer les mots qui chevaucheraient une frontière prosodique ou qui incluraient un marqueur d'accent en milieu de mot et non sur la première syllabe de type CV.

CHAPITRE 5

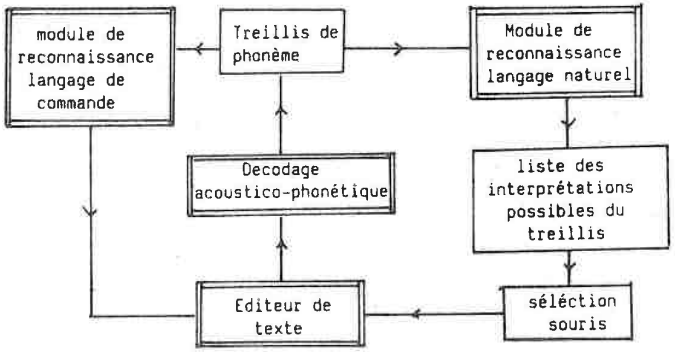
PRESENTATION DE NOTRE PROJET

L'objectif de ce travail est la réalisation d'un prototype de machine à dicter vocale en partant des expériences acquises dans notre équipe au cours de la réalisation des sytèmes Myrtille I [Pierrel 75] et Myrtille II [Pierrel 82]. le système réalisé inclut, en plus du système de dictée vocale lui-même , la possibilité d'activer à la voix des fonctions d'édition et des moyens graphiques pour corriger les erreurs éventuelles. Ce projet a demandé la mise en oeuvre de techniques spécifiques au problème de la dictée automatique. Nous avons développé des algorithmes de programmation dynamique avec contraintes syntaxiques aussi bien pour les fonctions d'édition que pour la saisie proprement dite des textes. Ce système comporte une unité lexicale importante et autorise l'utilisation d'environ 1000 racines de mots dans un français courant sans restrictions syntaxiques majeures. Le modèle syntaxique que nous avons développé est local de manière à couvrir un domaine aussi vaste que possible de la langue.

Le système est composé de quatre modules liés les uns aux autres (cf figure 1) :

- un décodeur acoustico-phonétique.
- un éditeur de texte qui comporte deux modes d'utilisation (parole et clavier).
- un système de reconnaissance permettant la saisie des textes dictés.
- un module de reconnaissance utilisé dans le mode commande de l'éditeur.

Nous aborderons dans cet exposé les deux aspects de notre travail : l'aspect édition vocale que nous développerons dans les parties B et C, et l'aspect manipulation des textes édités. Ce dernier point fera l'objet de la partie D.



architecture de l'éditeur

figure 1

PARTIE B

LES SOURCES D'INFORMATIONS DANS LE MODE EDITION PAROLE

1 Introduction

2 Le lexique

3 Le modèle syntaxique

4 Les règles de phonologie et de stratégie

CHAPITRE 1

INTRODUCTION

Contrairement à la reconnaissance de mots isolés, la reconnaissance de parole continue est un problème qui ne peut pas être entièrement résolu au niveau acoustique. En effet, le double indéterminisme introduit par la localisation des mots et par l'identification de ces derniers conduit à une explosion combinatoire des séquences de mots pouvant interpréter la phrase à reconnaître. Il devient alors nécessaire de faire intervenir un certain nombre d'informations linguistiques et de mettre en oeuvre des stratégies élaborées pour contrôler la recherche.

Dans cette partie, nous allons aborder les différentes sources de connaissances intervenant dans notre système :

- les informations lexicales,
- le modèle syntaxique,
- les règles de phonologie et de stratégie,

CHAPITRE 2

LE LEXIQUE

I. Introduction

Le lexique constitue la connaissance minimale que doit posséder tout système de reconnaissance de parole. Il intègre en effet une description complète des mots utilisables dans le domaine d'application. Dans un système de reconnaissance de mots isolés ou enchaînés, les informations lexicales peuvent se réduire à des formes de référence acoustiques. Par contre, dans un système de reconnaissance de parole continue la description englobe différents aspects liés aux autres niveaux de traitement.

Dans notre projet, le lexique contient trois types d'informations:

- les descriptions phonétiques et phonologiques permettant l'interprétation de la représentation phonétique issue du décodage acoustico-phonétique.
- des informations syntaxiques et sémantiques faisant le lien entre le niveau lexical et le modèle syntaxique.
- la forme orthographique de chaque mot.

II La structure du lexique

La création et la mise au point du lexique sont une phase à laquelle le concepteur doit apporter beaucoup de soin. En effet, toute erreur subsistant à ce niveau influe directement sur les performances de la reconnaissance. Les possibilités de rattrapage par les autres niveaux de traitement seront d'autant plus difficiles que la chaîne phonétique à décoder sera entachée d'erreurs. C'est pourquoi des fonctions de manipulation et d'édition du lexique devaient être prévues.

Pour atteindre cet objectif, nous avons défini un langage de description des informations lexicales. Cette approche permet de contrôler efficacement la nature et la cohérence des informations introduites. De plus,

l'utilisation d'un éditeur de texte, pour la création du lexique, rend aisées la saisie et les modifications nécessaires à la conception de la base de connaissances lexicales.

II. 1. Définition du langage de description

Nous allons décrire dans ce paragraphe la structure générale du lexique et la sémantique associée. Nous donnons la grammaire du langage de description dans une forme étendue de la notation de Backus-Naur (B.N.F).

i) La notation de Backus-Naur étendue

a) les crochets ‹ ›

Les crochets servent à délimiter les éléments non-terminaux de la grammaire.

b) le symbole de réécriture ::=

Ce symbole sert de séparateur entre les parties gauche et droite d'une règle.

c) le symbole /

Cette notation permet d'introduire dans une règle plusieurs productions possibles.

d) le parenthésage ( )

Il permet de factoriser les termes communs à plusieurs productions.

e) Les crochets [ ]

Tout terme entre crochets est optionnel dans la production où il apparait.

f) Les accolades { }

Les accolades indiquent une itération sur le terme qu'elles encadrent.

ii) La grammaire du langage de description

```
‹LEXIQUE› ::= ‹CLASSE›{‹CLASSE›} finlexique ;
‹CLASSE›::=‹NOM-CLASSE›[‹DECLARATION›]
            ‹ENTREE-LEXICALE›;
‹NOM-CLASSE› ::= A / B ... / Z;
```

Le lexique est divisé en classes de mots dans lesquelles apparait la description complète des mots d'une catégorie grammaticale du modèle syntaxique.

Dans chaque classe, trois types d'information sont donnés :

- le nom de la classe grammaticale traitée.
- une zone de déclaration dans laquelle on définit la nature des traits syntaxico-sémantiques qui seront attachés aux mots du lexique.
- une partie entrée lexicale dans laquelle est défini chacun des mots de la catégorie grammaticale.

```
‹DECLARATION›::=TYPE{‹LISTE-TRAITS›=‹TYPE-TRAIT›;};
‹LISTE-TRAITS›::=‹SYMBOLE› { , ‹SYMBOLE› };
‹TYPE-TRAIT›::= BOOLEAN / ‹SCALAIRE›;
‹SCALAIRE›::= [ ‹NOMBRE› .. ‹NOMBRE› ];
‹NOMBRE›::= { 0 / 1 ... / 9 }
‹SYMBOLE›::= { A / B ... / Z };
```

Dans la zone déclaration d'une classe, on définit la nature des traits syntaxico-sémantiques qui seront par la suite utilisés dans la description des mots. Les traits sont de deux types : booléen ou scalaire. Dans le cas d'un type scalaire, on définit les bornes de variations du trait. Les traits de type booléen permettent de définir la présence ou l'absence d'une propriété syntaxico-sémantique. Par contre, les traits de type scalaire sont affectés d'une valeur numérique. Ces derniers ont été introduits essentiellement pour indiquer la personne à laquelle est conjugué un verbe (en vue de l'accord du verbe avec le pronom personnel sujet).

un exemple : définition des traits genre et nombre des noms (masculin, féminin, singulier, pluriel), de la personne et du nombre auxquels sont

conjugués les verbes apparaissant dans la classe VERBE.

```
NOM
  TYPE
    MASC,FEM,SING,PLUR = BOOLEAN;
-------------------------------

VERBE
  TYPE
    SING,PLUR = BOOLEAN;
    PERSONNE = [1..3];
-------------------------------
```

```
<ENTREE-LEXIQUE>::= ENTREE { <UNITE> } FINCLASSE;
<UNITE>::= <GRAPHEME> [ <LISTE_TRAITS> ] <TRANS-PHON>;
<LISTE_TRAITS>= { <TRAIT> [ <VALEUR> ] };
<VALEUR>::= { 0/ 1... /9 };
<GRAPHEME>::= { A / ..... / Z };
<TRAIT>::= { A / .... / Z };
```

Pour chaque unité morphologique d'une catégorie, on indique :

- sa forme orthographique,
- la suite de propriétés syntaxico-sémantiques la caractérisant,
- sa représentation phonétique.

Tout trait utilisé à ce niveau doit avoir été défini préalablement dans la zone de déclaration. Celui-ci s'emploie en juxtaposant à l'identificateur le désignant la valeur qu'il possède. Pour les traits de type BOOLEAN, la valeur est par défaut vrai.

exemple : définition des propriétés du verbe "venir" à la troisième personne du singulier de l'indicatif présent.


```
VERBE
  TYPE
    INDIC_PRES,SING,PLUR = BOOLEAN;
    PERS = [1..3];
  ENTREE
    vient [ INDIC_PRES , PERS3 , SING ]
```

REMARQUE :

Les mots appartenant à plusieurs catégories grammaticales ont plusieurs entrées dans le lexique. Ainsi, par exemple, le mot *"le "* apparait à la fois dans la classe des articles définis et la classe des pronoms personnels.

Par ailleurs, un mot peut posséder plusieurs listes de propriétés en fonction du contexte syntaxique dans lequel il apparait. C'est le cas de l'article possessif *"mon."* Il s'emploie au masculin ou au féminin devant un mot féminin commençant par une voyelle.

mon [MASC , SING] m / on

mon [FEM , SING] m / on : VOY

```
<TRANS-PHON>::=
<SEGMENT>{/<SEGMENT>}[:<CONTRAINTE>]
<SEGMENT>::= [-] <PHON> [ , <PHON> [ , <PHON> ] ]
<PHON>::= ( A..Z ) ( A..Z )
<CONTRAINTE>::= VOY / CONS
```

La représentation phonétique choisie est celle déjà utilisée dans les systèmes MYRTILLE I et II [Pierrel 75],[Pierrel 82]. Elle permet de définir la plupart des variations phonologiques et les erreurs prévisibles du décodeur acoustico-phonétique.

Dans la version actuelle, les différentes variantes sont définies manuellement. Cependant, certaines particularités non spécifiques à un mot sont engendrées automatiquement grâce à un système de règles générales de phonologie (cf chapitre III).

La description phonétique est constituée par une suite de segments séparés par le caractère "/". Trois phonèmes au plus délimités par le symbole "," peuvent être associés à chaque segment, autorisant la description des différentes variantes d'un mot dans les trois catégories: élision,

insertion et substitution. Le premier phonème désigne le phonème le plus probable du segment, le second et le troisième indiquent respectivement un phonème pouvant se substituer au premier et un phonème pouvant s'insérer devant le segment. Dans le cas où le segment décrit peut être omis, celui-ci est précédé du signe ''-''.

En fin de transcription, on exprime une contrainte phonétique locale sur l'enchaînement des mots dans la phrase, afin de prendre en compte les problèmes de coarticulation. La restriction porte sur la nature du premier phonème de chaque mot. Citons, à titre d'exemple, la restriction sur l'emploi de l'article défini ''la'' qui ne peut précéder un mot commençant par une voyelle.

la [FEM , SING] L/a:CONS

UN EXTRAIT DU LEXIQUE

Nous donnons ici quelques unes des entrées lexicales telles qu'elles sont définies dans notre système, dans deux classes grammaticales : les articles définis (ARTD) et les verbes (VERBE). Dans la classe des verbes nous avons donné quelques traits de construction. Ces traits désignent la ou les classes grammaticales pouvant suivre le verbe concerné (A adjectif, I infinitif, Q que, N nom).


```
ARTD
 TYPE
  MASC,FEM,SING,PLUR=BOOLEAN
 ENTREE
  LA (FEM,SING)      : L/a:cons
  LE (MASC,SING)     : L/-&:cons
  ..............................
FINCLASSE

VERBE
 TYPE
  NR=(1..6);              ( personne et nombre du verbe )
  INDPRES= BOOLEAN        ( indicatif présent )
  A,I,Q,N,NQ...= BOOLEAN; ( traits syntaxiques )
 ENTREE
  CONFIRMONS (INDPRES,4NR,Q,N,NAN) : k/on,an/f/i/m,,R/on
  PARTICIPE  (INDPRES,1NR,AN)      : p/a/t,,R/i/s/i/p
  PARTICIPES (INDPRES,2NR,AN)      : p/a/t,,R/i/s/i/p
  PARTICIPE  (INDPRES,3NR,AN)      : p/a/t,,R/i/s/i/p
  ..............................................
FINCLASSE
FINLEXIQUE
```

II. 2. Représentation du lexique

La description du lexique ainsi définie est ensuite compilée vers une représentation machine plus optimisée. Cette représentation comprend deux structures lexicales regroupant l'ensemble des informations linguistiques, phonétiques et phonologiques exprimées :

- un lexique phonétique utilisé pour interpréter le signal,
- un lexique grammatical liant les mots au modèle syntaxique.

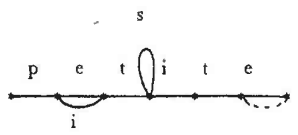
II. 2. 1. le lexique phonétique

La finalité du lexique phonétique est de permettre, au cours de l'analyse, la mise en correspondance du signal vocal avec une suite de

graphèmes. Il est donc nécessaire de disposer pour chaque unité lexicale d'une forme phonétique de référence. Chaque mot pourra ainsi être comparée avec la représentation du signal fournie par le système acoustico-phonétique.

a) représentation d'un mot.

On a vu précédement que la description phonétique d'un mot intégre différentes variantes phonologiques. Chaque mot peut donc se représenter par un graphe dans lequel la transcription phonétique standard est complétée par les différentes élisions, substitutions et insertions prévisibles. L'exemple d'un tel graphe est donné figure 1.



élision possible

substitution

insertion

transcription standard

figure 1: représentation phonétique du mot " petite "

b) représentation générale du lexique.

Le choix de l'organisation du lexique est un problème complexe faisant intervenir diverses contraintes souvent contradictoires telles :

- l'optimisation de la place mémoire occupée,

- la facilité d'accés aux diverses informations : faut-il garder un graphe phonétique pour chaque unité morphologique ou alors faut-il regrouper certains éléments communs?
- l'efficacité de la structure de données vis à vis des algorithmes de reconnaissance.

c) la solution mise en oeuvre.

L'approche choisie est essentiellement dictée par l'utilisation d'une stratégie de reconnaissance fondée sur un algorithme de programmation dynamique. En effet, les algorithmes qui en sont issus optimisent de façon significative la recherche lexicale sur une structure aborescente. De ce fait, on peut sacrifier la facilité d'accés aux éléments du lexique. Par ailleurs, la procédure de prédiction syntaxique mise en oeuvre dans notre système n'agit que sur deux types d'informations :

- les classes grammaticales,
- les traits syntaxico-sémantiques.

La représentation que nous avons adoptée optimise la représentation graphique que nous venons de présenter en factorisant les graphes phonétiques des mots appartenant à une même catégorie grammaticale et reprend l'un des aspects des systèmes MYRTILLE I et II [Laporte 84]. Ainsi deux mots débutant par le même graphe phonétique fusionnent leur partie commune comme indiqué sur la figure 2.

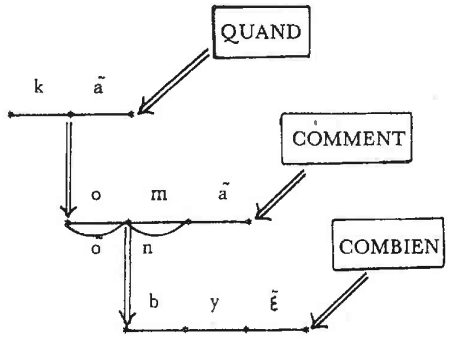


figure 2 : représentation de la sous classe constituée des mots ''quand, comment, combien''.

Chaque feuille de l'arbre ainsi construit correspond à l'aboutissement d'une description phonétique. La liste des mots homophones et de leurs propriétés syntaxico-sémantiques est retrouvée par un pointeur vers lexique grammatical.

d) la structure de donnée :

On accéde à chacune des classes grammaticales du lexique grâce à un tableau de pointeurs désignant la racine de l'arbre correspondant. Chaque classe est définie par un ensemble d'enregistrements de la forme suivante :


```
LEXIQUE = record
            AUTRE=   LEXIQUE
            case FIN: boolean of
            TRUE:( PROCHAIN :   LEXIQUE;
                   PHON1    : integer;
                   PHON2    : integer;
                   INS      : integer;
                   ELIS     : boolean);
            FALSE:(MOT      : integer);
          end;
```

Cette structure définit une liste chaînée à deux niveaux. Chaque noeud de l'arbre ainsi réalisé contient les champs suivants:

- FIN de type booléen marque un noeud terminal du graphe lorsqu'il prend la valeur vrai. Dans ce cas, le noeud pointe vers les informations graphiques et syntaxiques du mot dont la description phonétique aboutit à ce point du graphe.

- PHON1 représente le code du phonème le plus probable du segment.

- PHON2 indique le phonème pouvant se substituer à PHON1.

- INS correspond au phonème pouvant s'insérer devant le segment courant.

- ELIS de type booléen prend la valeur vrai lorsque le segment décrit peut être omis.

- MOT est un pointeur vers le lexique grammatical. Il apparait en fin de transcription et permet de retrouver la liste de mots homophones ainsi que leurs propriétés. La transcription phonétique de ces mots est obtenue en parcourant l'arbre jusqu'au noeud précédent.

Un exemple d'une telle structure est donné figure 5.

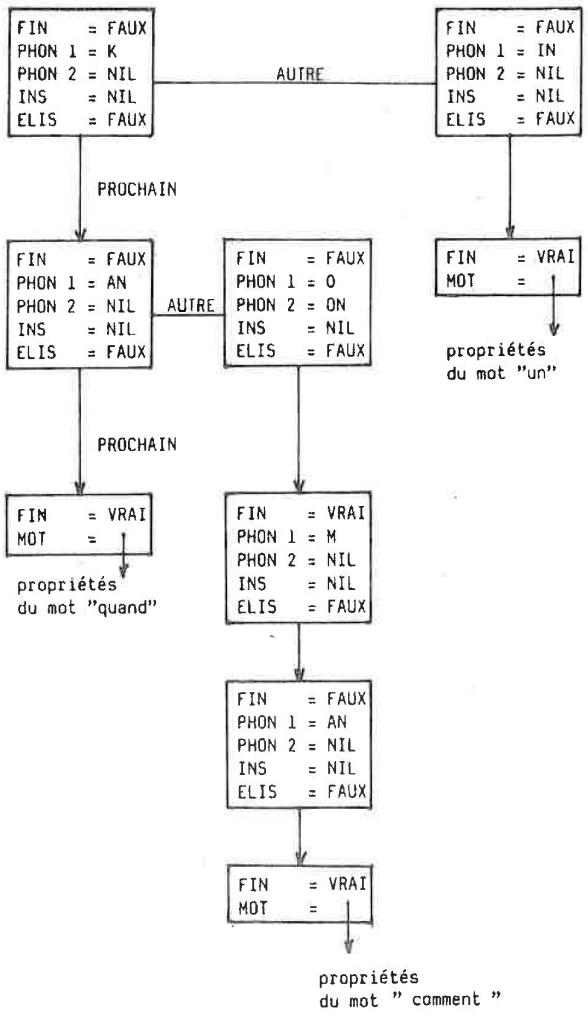


FIGURE 5: codage du lexique phonétique.

II. 2. 2. Le lexique grammatical

Le lexique grammatical contient l'ensemble des informations graphiques, syntaxiques et sémantiques relatives à chacun des mots du lexique. Il contient en outre les contraintes phonologiques sur les frontières de mots.

Implantation mémoire

Chaque mot du lexique est défini dans un tableau dont les éléments constitutifs sont de la forme suivante :

```
MOTLEX = record
            MOT    : string[15];
            CLASSE : integer;
            SUIT   : integer;
            TRAITS : integer;
            AUTRE  : boolean;
            VOY    : boolean;
         end;
```

où

MOT désigne la forme orthographique du mot décrit

CLAS indique le code de la catégorie grammaticale du mot décrit

SUIT correspond aux contraintes locales de phonologie : elle permet de traiter les phénomènes de coarticulation qui apparaissent à la frontière entre deux mots. Il est possible grâce à ce champ de conditionner l'apparition d'un mot en fonction de la nature du premier phonème du mot suivant dans la phrase.

SUIT=0 si il n'y a aucune contrainte sur le mot suivant

SUIT=-1 si le mot suivant doit forcément commencer par une voyelle

SUIT=-2 si le mot suivant doit débuter par une consonne

SUIT= code si le mot suivant doit débuter par le phonème désigné par code.

VOY de type booléen prend la valeur vrai lorsque le mot décrit commence par une voyelle.

AUTRE de type booléen permet de construire des listes regroupant les mots homophones. Lorsque deux mots homophones se suivent dans le

le lexique grammatical, AUTRE prend la valeur vrai.

TRAITS correspond à la forme codifiée de l'ensemble des traits syntaxico-sémantiques que possède le mot traité.

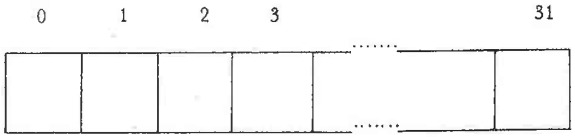
codification des traits syntaxico-sémantiques.

Dans un objectif d'optimisation, la liste de traits syntaxico-sémantiques que possède un mot est codée sous la forme d'une chaîne de bits. Ce codage permet de représenter l'ensemble des informations syntaxico-sémantiques sur un mot unique de 32 bits. Ce mot est partagé entre les différents traits. Les traits de type booléen occupent un bit et les traits de type scalaire occupent le nombre nécessaire de bits à la plus grande valeur qui peut leur être affectée.

Exemple : Codification des traits syntaxiques décrivant les pronoms personnels sujets.

Les traits définissant cette catégorie grammaticale sont :

- les traits masculin et singulier indiquant le genre et le nombre du pronom,
- le trait pers donnant la personne du pronom. Ce trait est de type scalaire et varie entre 1 et 3.



SING PLUR 1<NOMBRE<3

Ainsi, à chaque pronom est associé un mot comme indiqué ci-dessous.

je	1 pers du sing --> 0101
tu	2 pers du sing --> 1001
il,elle	3 pers du sing --> 1101
nous	1 pers du plur --> 0110

vous	2 pers du plur --> 1010
ils,elles	3 pers du plur --> 1110

Remarque :

Le test de la présence d'une propriété se fera par la construction d'un masque dans lequel sera positionné à 1 le bit correspondant. Il suffit alors d'effectuer un ''et'' logique entre le masque et le mot. Si le masque n'est pas modifié par l'opération, alors le test est positif.

III. La conjugaison automatique des verbes

III. 1. Objectif

La saisie et la description des verbes à tous les temps, modes et personnes seraient longues, fastidieuses et source d'erreurs difficilement repérables. C'est pourquoi nous avons choisi, d'engendrer automatiquement ces formes. La définition des verbes se réduit alors à la seule forme infinitive et à un numéro de type pour trouver le mode de conjugaison.

Pour réaliser ce traitement, nous avons écrit un ensemble de règles de construction qui formalisent l'ouvrage ''l'art de conjuguer'' de [Bescherelle 66]. Ces règles engendrent automatiquement toutes les formes conjuguées de la plupart des verbes du français et permettent de construire automatiquement les classes ''verbe '', ''participe-présent'', ''participe-passé'' du lexique. Ce traitement inclut d'une part la construction des formes orthographiques et phonétiques et d'autre part la propagation des informations syntaxico-sémantiques. Ce processus est schématisé sur la figure 6.

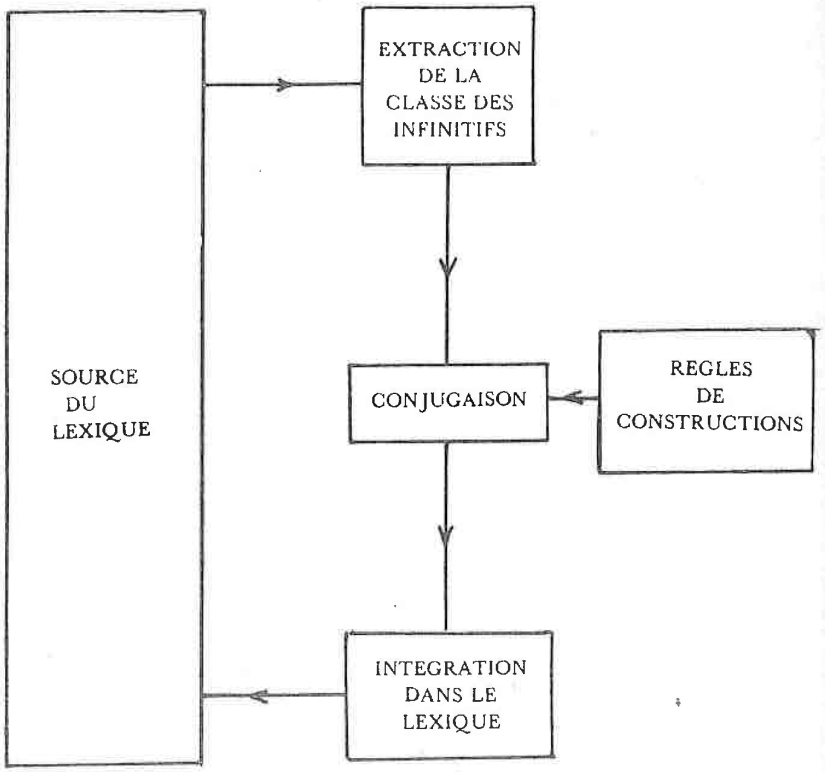


Figure 6 : la conjugaison des verbes.

III. 2 Réalisation

a) introduction

Le Bescherelle est composé de deux grandes parties :

- 82 tableaux de conjugaison. Chaque tableau décrit entièrement un verbe type conjugué à la forme active à tous les temps et modes du français.
- une liste alphabétique comportant huit mille verbes usuels du français. Chaque verbe renvoie à un verbe type qui sert de modèle à

la conjugaison.

Nous avons adopté cette même structure. Ainsi, à chaque verbe infinitif du lexique est associé un numéro renvoyant à l'un des verbes types. Par ailleurs, les 82 descriptions de "l'art de conjuguer" sont traduites sous forme de règles de construction. Afin de décrire certaines variantes et exceptions de la construction d'un verbe par rapport à un verbe type, nous avons ajouté une liste supplémentaire de verbes types tel le verbe dédire qui se conjugue comme le verbe type dire sauf à la deuxième personne du singulier de l'indicatif présent et de l'impératif :

vous dites, vous dédisez.

L'application que nous avons choisie se limitant à la dictée de lettres, il n'était pas nécessaire de définir chaque verbe à tous les temps et modes du français. Nous nous sommes limités à la description des verbes aux sept temps et modes suivants :

- indicatif présent,
- indicatif inparfait,
- indicatif futur,
- subjonctif présent,
- conditionnel présent,
- participe passé au masculin, féminin et pluriel,
- parpicipe présent.

b) Syntaxe et sémantique des règles

```
‹REGLE›::=‹INFINITIF› ‹NREF› ‹DESCRIPTION›
```

Une règle contient trois types d'informations :

- la forme infinitive du verbe,
- le numéro de référence renvoyant à un verbe type,
- la description du verbe aux différents temps et modes.

```
‹DESCRIPTION›::=‹IDENTIFICATION›-->{‹TEMPS_MODE›};
```

La description se compose :

- d'une partie identification à partir de laquelle sont extraits le radical et les différents constituants du verbe intervenant dans la conjugaison. Ce traitement est effectué par comparaison à la forme infinitive du verbe.

- d'une liste de description TEMPS-MODE qui spécifie la construction du verbe aux différent temps et modes désirés.

```
‹IDENTIFICATION›::=‹RADICAL›{''‹CHAINE›''‹VARIABLE›}
                    '' ‹TERM-INF›
```

Plus précisément, la partie IDENTIFICATION permet d'instancier un certain nombre de variables (sous-chaînes) telles que le radical. Ces sous-chaînes sont extraites de la forme infinitive du verbe. Elles seront utilisées par la suite dans la construction des différentes formes verbales.

exemple

Les verbes du type 11 se terminent en *''eter''* ou *''eler.''* Il est donc nécessaire d'extraire, en plus du radical, la partie variable de la terminaison : *''l''* pour les verbes en *''eler''* et *''t''* pour les verbes en *''eter.''* La description de ces deux types de verbes est alors unifiée par la paramétrisation de la conjugaison. Pour ces verbes l'identification s'écrit:

x ''e '' y ''er''

x étant le radical du verbe et y la partie variable ''l'' ou ''t''

```
‹TEMPS-MODE›::=‹SYMBOLE› ( ‹GENRE› [ , ‹GENRE› ] );
‹GENRE›::=‹RAD› ‹TERM› [ ‹VAR› ];
‹RAD›::={ [‹VAR›][" ‹CHAINE› "] }
‹TERM›::=‹VAR›
‹CHAINE›::= { A/.../Z }
‹VAR›::= A/.../Z
```

La spécification d'une forme verbale se compose :

- d'un identificateur qui indique le temps et le mode décrits
- d'une description en deux champs de la conjugaison du verbe aux personnes du singulier puis aux personnes du pluriel. La séparation entre singulier et pluriel permet de regrouper les terminaisons possibles en formes types (17 formes au singulier et 47 formes au pluriel).

Le verbe conjugué se construit par concaténation d'un radical et d'une terminaison type référencée par un identificateur. Le radical est lui même formé par concaténation de chaînes de caractères et de variables instanciées lors de la phase d'identification.

exemple de radical : x ''e '' y

Le système de règles que nous venons de décrire est complété par une table définissant les terminaisons-types. Cette table revêt la forme suivante:

```
‹TABLEC›::={‹VAR›=‹DESCRIPT›,‹DESCRIPT›,‹DESCRIPT›}
‹DESCRIPT›::= {‹CHAINE› [ ? ]} ;
```

Chacune des descriptions correspond à une des trois formes verbales du singulier ou du pluriel. Afin de limiter le nombre de terminaisons types, certaines d'entre elles sont paramétrées par une variable désignée par le symbole ''?''.

exemple : P4 = E?ONS , E?EZ , E?ENT

P4[L] -› ELONS ELEZ ELENT

P4[V] -› EVONS EVEZ EVENT

Exemple complet

```
JETER 11
X''E''Y''ER'' -> INDPRES(X''E''Y-Y+S1,X''E''Y+P6[Y])
                INDIMP(X''E''Y+S8,X''E''Y-Y+P20)
                INDFUT(X''E''Y-Y''ER''+S10,X''E''Y-Y''ER''+P23)
                SUBPRES(X''E''Y-Y+S1,X''E''Y+P27)
                CONDPRES(X''E''Y-Y+S8,X''E''YY''ER''+P20)
                PARTPRES(X''E''Y+P39)
                PARTPAS(X''E''Y+P40)
avec:

S1=E,ES,E
S8=AIS,AIS,AIT
S10=AI,AS,A
P6=ONS,EZ,?ENT
P20=IONS,IEZ,AIENT
P23=ONS,EZ,ONT
P27=IONS,IEZ,?ENT
P39=+,+,ANT
P40=+,',E
```

III. 3. L'interpréteur de règles.

L'appel puis l'interprétation d'une règle suit le processus indiqué sur la figure 7.

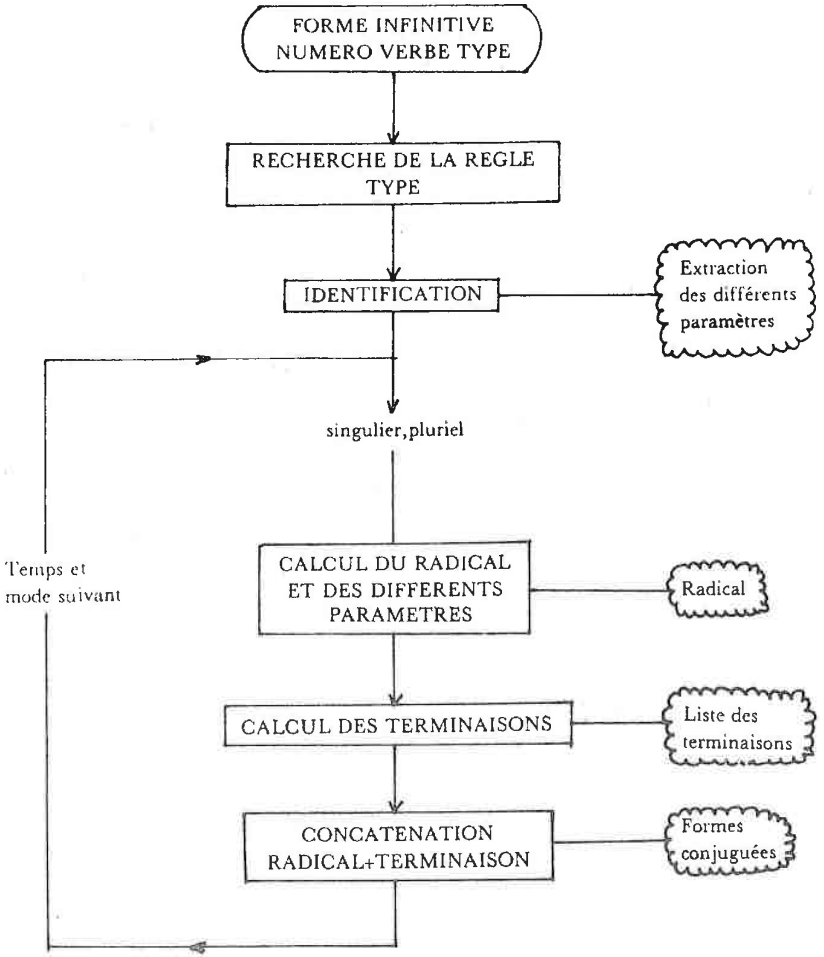
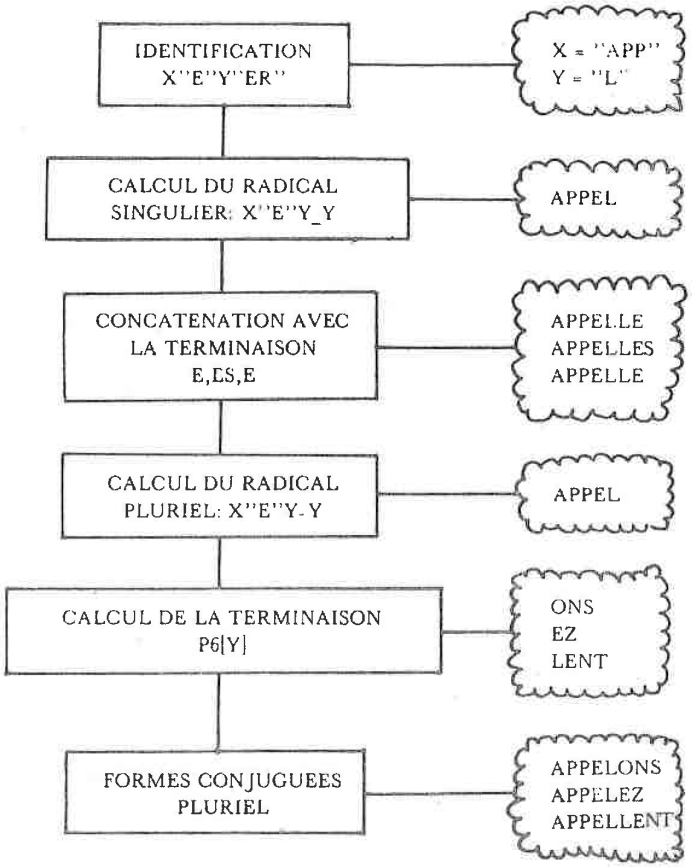


figure 7 : la conjugaison des verbes.

Un exemple de ce processus.

Conjugaison du verbe à l'indicatif présent



On trouvera en annexe l'ensemble des règles pour les conjugaisons orthographiques.

IV. Le logiciel d'aide à la création du lexique

IV. 1 Introduction

Comme nous l'avons vu, la création du lexique une est opération primordiale. Le lexique est la source de connaissances du système qui va devoir le plus évoluer. En effet, la limitation du nombre de mots définissables dans le lexique ne permet pas de couvrir l'ensemble des applications envisageables. Même si l'on peut définir un lexique intégrant les mots du français fondamental, il va falloir adapter celui ci au domaine d'application choisi (les 1000 premiers mots d'une langue permettent la compréhension d'environ 80% d'un texte normal). Même si, pour le moment, le lexique est créé semi-manuellement, nous souhaitons obtenir dans l'avenir un système complet de génération automatique du lexique. En effet, la création de grands vocabulaires nécessite d'approfondir cet aspect, dans le but d'améliorer l'efficacité de cette phase.

IV. 2. Etat du projet

La procédure de création du lexique actuellement opérationnelle est réalisée par:

- le recueil d'un corpus de textes couvrant les différents aspects du domaine d'application.
- l'étiquetage du corpus : à chaque mot du corpus, on asscocie la catégorie grammaticale qui le caractérise dans le contexte où il apparait. Cette phase est semi-automatique grâce à un modèle statistique (voir chapitre III).
- la création automatique d'un dictionnaire intégrant tous les mots du corpus. Ceux-ci sont triés par ordre alphabétique et par catégories grammaticales.
- l'adjonction manuelle des informations syntaxico-sémantiques et phonologiques relatives à chacun des mots du lexique.
- l'extension automatique du lexique aux différentes formes conjuguées des verbes.
- l'extention manuelle des différentes formes des noms et adjectifs (masculin, féminin, singulier, pluriel).

IV. 3. Perspectives

L'extension de l'aide à la création du lexique est envisagée pour deux des étapes citées ci-dessus :

- pour la génération automatique des différentes formes nominales et adjectivales.
- pour la génération automatique de la représentation phonologique de chaque mot. Celle-ci peut être réalisé par un programme de transcription graphème-phonème et un système de règles spécifiant les différentes altérations phonologiques.

La base de donnée lexicale du GRECO repond à ces différents points.

IV. 4. La base de donnée lexicale du GRECO communication parlée

Le GRECO communication parlée développe actuellement une base de donnée lexicale du français (BDLEX). L'objectif de ce projet est la construction d'un dictionnaire adapté au traitement automatique de la parole. Une version préliminaire de la base (BDLEX 0) est actuellement opérationnelle et accessible via Transpac sur DPS8 Multics du CICT (centre inter-universitaire de calcul de Toulouse). BDLEX 0 est organisée en base de donnée relationnelle et comporte trois composantes :

- une composante phonologique.
- un module d'interrogation et de mise à jour.
- une composante morphologique.

La composante phonologique se veut aussi exhaustive que possible. Elle fonctionne grâce à un ensemble de règles fournies par des phonologues. Ces règles s'appliquent aussi bien à l'intérieur d'un mot qu'à la jonction entre deux morphèmes.

Le module d'interrogation et de mise à jour permet l'accès de manière transparente à BDLEX 0. Elle propose actuellement à l'utilisateur les fonctions suivantes :

1) la conjugaison des verbes sous forme graphique ou phonétique.

2) la dérivation en genre et en nombre sous forme graphique et phonétique.

3) l'analyse morphologique pour chercher l'existance d'un mot dans la base.

4) la visualisation d'une unité lexicale.

5) l'ajout d'un mot dans le lexique.

6) la destruction d'un mot dans le lexique.

7) la création d'un verbe sur le modèle d'un autre.

8) la création d'un nom.

9) la sélection d'un code phonétique.

Enfin, la composante morphologique permet d'une part dans un mode génératif de former des mots à partir de morphèmes (conjugaison des verbes, dérivation des genres et nombres des noms et adjectifs), d'autre part dans un mode analytique de segmenter un mot en morphèmes.

BDLEX0 comporte pour le moment un lexique de 6000 mots.

Les applications d'une telle base dépassent le cadre de la reconnaissance de parole et peuvent s'étendre dans les domaines suivants :

- synthèse de la parole.
- linguistique.
- E.A.O.
- T.A.O.
- etc...

L'un des objectifs les plus intéressants de ce projet pour la reconnaissance de parole est l'extraction de sous-lexiques adaptés à une application donnée. En ce sens elle permet d'espérer la génération rapide et fiable de lexiques (en particulier pour notre système de reconnaissance).

CHAPITRE 3

LE MODELE SYNTAXIQUE

I. Introduction

Le niveau syntaxique joue un rôle important dans un système de reconnaissance de parole. Il fait partie de la plupart des systèmes intégrant un vocabulaire important. En effet, l'objectif du modèle syntaxique est d'extraire, parmi l'ensemble des combinaisons possibles de mots, le sous-ensemble des solutions syntaxiquement admissibles.

Dans la stratégie d'analyse de prédiction-vérification que nous avons adoptée, les informations syntaxiques permettent de restreindre de façon significative le nombre de mots émis comme hypothèses à chaque niveau du traitement.

Les textes que nous désirons reconnaître, font partie d'un français écrit courant, sans restrictions syntaxiques majeures, et appartiennent à des domaines variés. De ce fait, le modèle que nous cherchons à définir se doit d'être le moins limitatif possible.

Les modèles syntaxiques issus des travaux en linguistique dans le cadre des langues écrites constituent une base intéressante pour la parole. Citons, parmi les premiers modèles, les grammaires syntagmatiques génératives [Chomsky 57]. Ce type de grammaires se présente comme une suite de règles de réécriture. Le mécanisme de dérivation obtenu en appliquant les unes après les autres ces différentes règles permet d'engendrer l'ensemble des phrases syntaxiquement correctes du langage. Ce type de modèle est couramment utilisé pour les langages artificiels en reconnaissance de parole. Il donne lieu à des représentation sous forme de Backus Naur ou sous forme de réseaux (R.N.P, A.T.N,...). Un certain nombre d'études exhaustives du français écrit a été mené par une telle approche. Citons la grammaire en chaîne de Salkoff [Salkoff 73] qui se situe dans la

théorie des grammaires distributionnelles dont Harris [Harris 71] est l'un des principaux représentant. Dans ce dernier modèle, les catégories grammaticales reposent sur une étude distributionnelle des constituants de la phrase sans aucune considération sémantique. Enfin, notons les grammaires génératives transformationnelles de Chomsky [Chomsky 71], qui ont été introduites pour pallier certaines faiblesses des grammaires syntagmatiques. En effet, celles-ci ne peuvent rendre compte du phénomène d'ambiguité dû au fait qu'une même phrase peut comporter plusieurs significations. Le mécanisme de description du langage comprend plusieurs étapes dans cette théorie:

- une étape qui engendre des structures profondes de sens identique.
- une étape de transformation des structures profondes en structures superficielles (par exemple, les formes actives et passives d'une phrase).

Ces modèles globaux sont cependant peu adaptés au type d'application que nous envisageons. En effet, ils sont très lourds à mettre en oeuvre, leur complexité croissant très fortement dès que l'on cherche à modéliser la syntaxe d'un sous ensemble assez vaste de la langue naturelle.

Une autre façon d'aborder le problème de la formalisation d'un langage consiste à acquérir sur un grand nombre de données, par apprentissage automatique ou semi-automatique, un modèle heuristique de la syntaxe.

C'est cette approche que nous avons adoptée et que nous présentons dans ce chapitre.

Le modèle que nous avons défini est constitué par deux systèmes de description de la structure du langage :

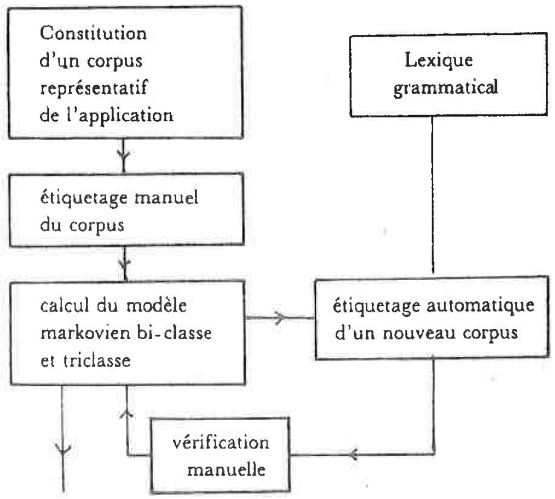
- un ensemble de contraintes positionnelles locales portant sur les catégories grammaticales des mots constitutifs de la phrase.
- un système de règles syntaxiques classiques agissant localement.

La formalisation du langage ainsi obtenue est cependant moins fine que dans l'approche générative. En effet, l'action locale du modèle ne permet pas de rejeter toutes les combinaisons syntaxiques incorrectes. Nous pensons néanmoins que le filtre induit est suffisant dans le cadre de notre application.

II. Le modèle positionnel

II. 1. Introduction

L'ensemble des contraintes positionnelles binaires et ternaires sur les classes grammaticales admissibles constitue l'aspect le plus important du modèle structurel que nous avons développé. La simplicité de cette approche présente l'avantage de permettre un apprentissage automatique sur un corpus de textes préalablement étiquetés. Ce processus peut être schématisée comme indiqué sur la figure 1. Des modèles analogues ont déjà été développés [Andreewsky 73] [Fluhr 77] [Jelinek 82] [Derouault 85].



acquisition du modèle syntaxique.

figure 1

La détermination du modèle positionel est réalisée de façon incrémentale, ce qui permet une amélioration du modèle grâce à un

processus d'apprentissage au fur et à mesure du traitement dont les grandes étapes sont :

1) étiquetage manuel d'un corpus de textes. Ce corpus est constitué d'un ensemble de lettres représentatives de l'application choisie et comprend environ 3000 mots. Cet étiquetage consiste à affecter à chaque mot sa classe grammaticale.

2) construction d'un dictionnaire alphabétique. Chacun des mots apparus dans le corpus est associé à la liste des classes grammaticales qu'il posséde. Cependant, une intervention humaine est nécessaire. En effet la taille du corpus est insuffisante pour que chaque mot apparaisse avec chacune des catégories grammaticales qu'il possède. Les classes manquantes sont insérées manuellement dans le dictionnaire.

3) calcul partiel du modèle sur le corpus étiqueté.

4) étiquetage automatique de nouveaux textes. La résolution des ambiguités syntaxiques est traitée par le modèle partiel de l'étape 3.

5) vérification manuelle et correction de l'etiquetage effectué à l'étape 4.

6) itération des opérations 3,4 et 5 jusqu'à convergence du modèle.

L'acquisition du modèle syntaxique, comme indiqué ci-dessus, est étroitement liée à l'étiquetage de textes. En effet, c'est à partir de cette information que seront extraites les contraintes positionnelles que l'on cherche à définir. Ainsi, l'étiquetage fait partie intégrante de l'acquisition du modèle. Cette phase de l'apprentissage est automatisée petit à petit par la résolution des ambiguités à l'aide du modèle syntaxique engendré par le processus d'apprentissage.

II. 2. L'apprentissage automatique du modèle.

II. 2. 1. Introduction

La difficulté du processus d'apprentissage du modèle syntaxique réside dans la phase d'étiquetage des textes constituant le corpus. En effet, le traitement d'un grand nombre de données exige une automatisation au moins partielle de cette étape du traitement.

La complexité de l'étiquetage provient de la multiplicité des fonctions grammaticales que peut posséder un mot de morphologie donnée. Citons le

cas du mot *''la''* dans la phrase

La pomme, il la mange

où le premier *''la''* fait fonction d'article défini, alors que le deuxième appartient à la catégorie grammaticale des pronoms personnels complément d'objet direct. Dans un autre contexte *''la''* pourrait également désigner la note de musique et appartenir à la catégorie des noms. De telles ambiguités sont fréquentes. C'est le cas entre autres pour :

- certains verbes conjugués à la 3 ième personne du singulier de l'indicatif présent. Ils peuvent, en effet, faire fonction de substantif. Citons

  adresse, demande, juge, porte...

- les articles définis et pronoms personnels complément d'objet direct ''le '',''la'', ''l''', ''les''.

- les conjonctions de coordination et nom ''or'' et ''car''.

Seul le contexte syntaxique et sémantique permet de lever l'ambiguité.

II. 2. 2. Nature des contraintes positionnelles extraites d'un texte

a) formalisation

Nous reprenons en partie la formalisation introduite par Andreewsky [Andreewsky 73].

Soit un texte T. On définit la fonction M par :

$$M : i \dashrightarrow M(i,k)=k$$

où i indique la position, dans le texte T, du mot dont la morphologie est k.

Chaque mot de morphologie k possède une liste de valeurs grammaticales référencées dans un dictionnaire. Soit:

$$V(1,k),V(2,k),\ldots,V(n,k)$$

cette liste.

Nous noterons v(i) la valeur grammaticale du mot de rang i dans le contexte où il apparait.

Nous verrons qu'à partir d'un tel texte, il est possible de construire deux types de contraintes positionnelles constituées soit par des règles de

résolution lexicale, soit par des matrices de précédence.

Dans les deux cas, on définira un modèle d'ordre 2 ou 3 selon la précision recherchée.

remarque

Le choix de l'ordre du modèle doit être un compromis entre deux critères contradictoires: d'une part, la précision du modèle demande un ordre élevé, d'autre part, la rapidité de convergence contraint à limiter l'ordre. De ce fait il est difficile de dépasser l'ordre 3.

b) Les règles de résolution lexicale

Dans cette approche, le modèle est construit par accumulation d'un ensemble de règles qui définissent l'ensemble des enchaînements possibles de mots. Ces règles expriment, en fonction du contexte syntaxique, les différentes réalisations grammaticales possibles d'un mot parmi l'ensemble des classes qui peuvent lui être associées.

Une règle de résolution lexicale prend la forme générale suivante :

```
    Liste    V(M(j,k))-->    Liste    v(j)
j:i--> i+ordre            j:i--> i+ordre
```

où i est l'indice du mot k dans le texte.

Si l'on reprend l'exemple de la phrase:

La pomme, il la mange.

et que l'on cherche à définir un modèle d'ordre 2, on va pouvoir extraire par apprentissage les relations suivantes :

(ARTD,PRON) (NOM) --> (ARTD) (NOM)

(ARTD,PRON) (VERBE) --> (PRON) (VERBE)

Ces règles indiquent qu'un mot de morphologie donnée, possédant les catégories grammaticales *article défini* et *pronom*, prend la valeur :

- article défini s'il est suivi d'un nom.
- pronom s'il précède un verbe.

Une règle de résolution lexicale d'ordre 2 prend donc la forme générale suivante :

V(1,K1)...V(n,K1) V(1,K2)...V(m,K2) --> V(i,K1) V(j,K2)

avec :

n le nombre de catégories grammaticales du mot de morphologie K1

m le nombre de catégories grammaticales du mot de morphologie K2

1< i >n et 1< j >m

Le modèle ainsi défini peut également être complété en associant à chaque règle une fréquence d'apparition (Pour plus de détail sur ce dernier point, consulter [Fluhr 77]).

c) Les matrices de précédence.

L'autre approche que nous proposons est la définition d'une matrice de précédence. Celle-ci est construite par le recensement de toutes les suites grammaticales d'ordre 2 ou 3 apparues dans le corpus d'apprentissage. Elle se définit de la façon suivante :

soit mat cette matrice
soit n l'ordre du modèle

pour i: 1 --> longueur texte faire
  mat(v(i),...,v(i-n+1)= ok;
fin-pour

si mat(v1,...,vn)<>nil alors la suite v1,...,vn est autorisée.

Il est possible également de se ramener à un tel modèle à partir de règles de résolution lexicale. Cette matrice s'obtient alors en supprimant la partie conditionnelle des règles. Par exemple la règle :

(V1,V2,...,Vn) (W1,...,Wn) -> Vi Wj donne
mat(Vi,Wj)=vrai.

De ce fait, le modèle résultant est moins fin.

La matrice est soit de type booleen, soit de type reel. Dans ce dernier cas, on associe à chaque suite grammaticale, une fréquence d'apparition défini pour un modèle d'ordre 2 par :

Fréquence d'apparition du couple (vi,vj) sur l'ensemble des couples possibles

$$f(vi,vj) = \frac{n(vi,vj)}{\Sigma\, n(vi,vj)}$$

ou *Fréquence d'apparition du couple (vi,vj) sur le nombre de mots de la catégorie vi appartenant au corpus.*

$$f(vi,vj) = \frac{n(vi,vj)}{n(vi)}$$

ou *Fréquence d'apparition du couple (vi,vj) en fonction du nombre de mots de la classe vj.*

$$f(vi,vj) = \frac{n(vi,vj)}{n(vj)}$$

Pour une modélisation d'ordre 3, on utilise des fréquences similaires.

L'utilisation de fréquences d'apparition permet de définir un modèle statistique adapté à l'apprentissage automatique [Jelinek 83] [Derouault 85].

II. 2. 3. L'étiquetage automatique.

a) Définition du problème.

Soit un texte :

M(1,a)...M(i,k)...M(l,n)

Il s'agit d'associer à chaque mot M(i,k) la catégorie grammaticale V(j,k) qu'il possède parmi l'ensemble des valeurs possibles. Le problème se ramène alors à la recherche d'un chemin dans un treillis syntaxique de la forme :

V(1,a) V(1,b) ... V(1,k) ...

V(2,a) V(2,b) ... V(2,k) ...

V(3,a) V(3,b) ... V(3,k) ...

V(4,b) ... V(4,k) ...

V(5,k) ...

Le modèle syntaxique intervient pour valider chaque transition du chemin. La description des structures du langage étant locale, toutes les ambiguitées ne peuvent être résolues. Dans le cas de l'approche statistique, l'étiquetage retenu est alors celui qui correspond au chemin obtenant le meilleur score.

b) étiquetage avec les régles de résolution

notations

soient :

- o : l'ordre du modèle
- n(k) : le nombre de catégories grammaticales que possède le mot de morphologie k
- i : le rang d'un mot dans le texte
- m : le numéro de la catégorie grammaticale courante.

algorithme

Il s'agit de rechercher dans le treillis syntaxique défini précédement un chemin continu entre le début et la fin du texte. Nous donnons ici l'algorithme général pour la recherche de ce chemin.

```
RECHERCHE(i,m)
debut
  si REGLE(i+1) alors RECHERCHE(i+1,1);
  si m+1<=n(M(i,k) alors RECHERCHE(i,m+1);
fin

(* test de la validité d'une transition. *)

REGLE(i)
debut
  si V(M(i-o,k))...V(M(i,k)) -> V(m,M(i-o,k))...V(m,M(i-o,k))
  alors REGLE=vrai sinon REGLE=faux;
fin
```

Cet algorithme serait complet si le système de règles définissait complétement la syntaxe du français. En fait, plusieurs cas peuvent se produire :

- la recherche arrive à une impasse car le contexte grammatical rencontré est non encore apparu dans le corpus . Il faut alors ajouter une nouvelle règle pour compléter le modèle. Le traitement peut être automatisé si le contexte grammatical est suffisamment fort.

- un mot du texte est inconnu du dictionnaire. Dans ce cas, si les contraintes contextuelles sont suffisamment fortes, on peut résoudre l'indétermination par des essais successifs en associant au mot inconnu chacune des catégories grammaticales du modèle. Ce processus s'optimise en excluant de l'ensemble des classes possibles celles composées d'un faible nombre de mots d'utilisation fréquente. En effet, ces mots sont complétement définis dans le dictionnaire.

(pour plus de détails se rapporter à [Fluhr 77])

c) étiquetage par les matrices de précédence

Une autre façon de résoudre l'étiquetage automatique est l'approche purement probabiliste. Dans cette optique, les matrices fréquentielles de précédence permettent de calculer, pour chaque interprétation possible d'une phrase, une probabilité. L'interprétation retenue est, dans ce cas, celle qui est affectée de la plus grande probabilité.

Formalisation pour un modèle triclasse.

Soit p(v1/v2,v3) la probabilité d'avoir un mot de la classe v1 après deux mots des catégorie v2 puis v3.

Soient M(1,a)....M(l,k) un texte à analyser et V(i,k) la liste des valeurs grammaticales associées à un mot de morphologie k.

On définit alors la probabilité de réalisation du texte M par :

$$P(M(1,a)\ldots M(l,k)) = \prod_i P(M(i,a)/M(i-1,b)M(i-2,c))$$

$$= \sum_{v(i)} \prod_i P(v(i)/v(i-1)v(i-2))$$

La somme apparaissant dans la probabilité correspond à la prise en compte de toutes les réalisations syntaxiques du texte.

En fait, pour l'étiquetage, ce qui nous intéresse, c'est l'obtention de l'interprétation la plus probable. On cherche alors :

$$\mathcal{P} = \prod_i P(vi)/(v(i-1)v(i-2))$$

L'étiquetage retenu est alors celui qui maximise $\mathcal{P}$.

L'apprentissage.

Si dans le corpus d'apprentissage la séquence syntaxique (v1,v2,v3) apparait pour la première fois, la probabilité P(v3/v2,v1) est nulle. On ne peut donc pas étiqueter le mot de la classe v3. Une façon de tourner la difficulté est d'inclure dans le modèle des contraintes d'ordre deux. En effet, si dans le corpus le couple (v2,v3) est déjà apparu, on peut quand même prédire v3 dans le contexte (v2,v3). On reformule alors P(v3/v2,v1) de la façon suivante :

$$P(v3/v2,v1) = \frac{n(v1,v2,v3)}{n(v1,v2)} + k\,\frac{n(v2,v3)}{n(v2)}$$

d) la solution mise en oeuvre dans notre système.

Nous venons de voir deux modélisations possibles de la syntaxe du français. La solution que nous avons adoptée est celle définie par les

matrices fréquencielle de précédence. Notre choix est motivé par deux points :

- la facilité de mise en oeuvre d'un tel modèle,
- la rapidité de la convergence par rapport à l'approche par régles de résolution lexicale.

III. Les régles de grammaire.

III. 1. Introduction

L'introduction de règles syntaxiques est motivée par plusieurs faiblesses du modèle triclasse; en particulier:

- pour obtenir un modèle efficace, il faut un grand nombre de catégories grammaticales,
- plus le nombre de catégories grammaticales est grand, plus le corpus d'apprentissage doit être grand,
- les contraintes positionnelles du modèle ne dépassent pas l'ordre trois. Des phénomènes bien connus de la linguistique structurelle, telle la construction des verbes [Gross 75], nécessitent un ordre supérieur,
- la description de phénomènes contextuels passe par la multiplication des catégories grammaticales.

Le système de règles que nous proposons permet à la fois de restreindre et de compléter le modèle triclasse. En effet, ces règles ont pour objectif la prédiction et la vérification des traits syntaxico-sémantiques associés à chacun des mots du lexique. Elles interviennent pendant la phase de prédiction lors de la reconnaissance.

III. 2. La structure des règles

Une règle syntaxique est composée de deux éléments distincts:

- une partie identification désignant le contexte d'application de la règle.
- une partie action complétant la connaissance sur les éléments contextuels désignés dans la partie identification.

III. 2. 1. La partie identification

La partie identification d'une règle décrit une structure syntaxique locale et s'exprime sous la forme d'une grammaire hors contexte. L'objectif de cet élément de description est de définir le contexte d'application de la règle.

Pour l'écriture des règles, nous avons adopté un formalisme dérivé de la notation de Backus-Naur (B.N.F) que nous avons étendue aux besoins de notre application. La métagrammaire satisfaisant cette notation est définie par la grammaire:

```
‹REGLE›::=‹SYMBOLE› = ‹EXPRESSION›;
‹EXPRESSION›::= ‹TERME› { / ‹TERME›}
‹TERME›::= ‹FACTEUR› { ‹FACTEUR› }
‹FACTEUR›::=‹TERMINAL› --› ‹ACTION› # /
            ‹NON-TERMINAL› --› ‹ACTION› # /
            [ ‹TERME› ] /
            ( ‹EXPRESSION› )
‹TERMINAL›::={ a/.../b }
‹NON-TERMINAL›::={ A/.../B }
```

Eléments du langage :

= se réécrit

/ sépare les alternatives d'une production

[] encadre un élément facultatif

() factorise l'élément encadré

‹› délimite les terminaux et les non-terminaux

les terminaux sont en minuscule

les non-terminaux sont en majuscule

Quelques exemples

* <REGLE1>=<artd><nom> -> action#

Cette règle est interprétée par l'hypothétiseur à chaque fois qu'une chaîne partiellement reconnue, à un niveau de la reconnaissance, possède le contexte gauche " articles définis" suivi de "nom". La partie action de la règle est dans ce cas interprétée. Elle permettra de prédire les traits que doit posséder le nom et de vérifier les propriétés de l'article défini.

La structure décrite dans la partie identification d'une règle peut être beaucoup plus complexe et correspondre à un groupe syntaxique complet du français. La règle :

```
* <REGLE100> = [<ne>] <verbe> [<pas>] [<adv>] <INF> --> action#
  <INF> = [<ne><pas>] [<adv>] <inf>;
 F;
```

sera déclenchée sur les éléments suivants :

je ne peux pas bien ...

il peut ...

nous pouvons ne pas ...

III. 2. 2. La partie action

a) introduction

La partie action a pour objectif de définir des contraintes positionnelles s'appuyant non plus sur les catégories grammaticales mais sur les traits syntaxico-sémantiques définis dans le lexique. La partie action d'une règle se déclenche lorsque l'une des productions générées par la partie identification correspond au contexte gauche de la phrase à analyser. Les actions sont de deux types : prédictives ou rétroactives. Les premières effectuent le calcul des propriétés du mot de la classe indiquée par le dernier élément d'une production. Les secondes effectuent un contrôle sur les traits des autres éléments de la production.

b) structure des actions

Une action est généralement composée de plusieurs instructions de type prédictif ou rétroactif. Une instruction peut être complexe et se compose le plus souvent d'une suite d'actions élémentaires. Une instruction complexe est délimitée par les symboles "[" et "]" correspondant au "begin" et "end" du langage PASCAL.

```
<ACTION>::= <INSTRUCTION> {; <INSTRUCTION>}
<INSTRUCTION>::= <INSTRUCT-SIMPLE>/
                 [<INSTRUCT-SIMPLE>{;<INSTRUCT-SIMPLE>}]
```

Les objets de base manipulés par une instruction sont les valeurs des traits syntaxico-sémantiques associés aux classes du lexique. Les opérations traitant ces objets sont :

- l'affectation: elle consite à donner une valeur au trait manipulé par l'instruction. L'objet manipulé dans la partie gauche d'une affectation ne peut pas être quelconque. En effet, le caractère prédictif d'une telle opération fait qu'elle ne peut toucher des éléments dont on connait déjà les propriétés syntaxico-sémantiques. Comme nous le verrons dans la partie suivante, seul le dernier élément d'une production engendrée par la partie identification de la règle possède ce critère.

- les opérations booléennes : =, <, >, <=, >=, <>, ou, et. Elles interviennent pour la vérification de la validité de traits déjà connus (règles rétroactives).

```
<INSTRUCT-SIMPLE>::=<AFFECTATION> /
                    <EXPRESSION-BOOLEENNE> /
                    <EXPRESSION-BOOLEENNE>
                    -><INSTRUCTION>;
<AFFECTATION>::=<VARIABLE>:=(<VARIABLE>/(<CONSTANTE>)
<VARIABLE>::=<CLASSE>.<TRAIT>
<CONSTANTE>::= {0/.../9} / vrai / faux
```

III. 2. 3. Exemples d'application

III. 2. 3. 1. Règles prédictives

a) Accord en genre et en nombre du nom avec l'article défini qui précéde.

```
<REGLE0>=<det><nom> --> 
            [nom.masc:=det.masc;
             nom.fem :=det.fem;
             nom.sing:=nom.sing;
             nom.plur:=nom.plur]#
```

Les traits *masc, fém, sing, plur* d'un nom s'identifient aux traits masculin, féminin, singulier et pluriel de l'article qui le détermine.

b) Accord du verbe avec le pronom personnel sujet.

```
<REGLE1>=<point><prps><verbe> --> 
            [verbe.sing.=prps.sing;
             verbe.plur:=prps.pers;
             verbe.pers:=prps.pers] #
```

où

''sing'' et *''plur''* indiquent si le verbe est conjugué aux personnes du singulier ou du pluriel, et

''pers'' définit la personne du verbe.

III. 2. 3. 2. Règles conditionnelles et rétroactives

a)Choix de l'auxiliaire pour la conjugaison d'un verbe à un temps composé.

```
<REGLE 2> = <avoir> [<pas>] [<adv>] <ppas> --> 
              ppas.a := vrai #
```

(chaque verbe du lexique est affecté soit du trait *''et''* indiquant qu'il se

conjugue avec l'auxiliaire ''être '', soit du type *''a''* indiquant qu'il se conjugue avec l'auxiliaire avoir, à la forme active.)

L'emploi de l'auxilliaire ''être '', n'est pas aussi restrictif. En effet, cet auxiliaire s'utilise pour la construction de la forme passive de tous les verbes, par exemple:

''la souris a mangé le fromage '' à la forme active

''le fromage est mangé par la souris '' à la forme passive

Cependant, on peut exprimer certaines restrictions dans son emploi. Par exemple, on sait que pour qu'un verbe puisse s'employer à la forme passive, il faut qu'il soit transitif. Il doit donc posséder le trait *''n''* qui indique la correction de la construction :

verbe + groupe nominal

ce qui s'exprime par :

```
<REGLE 3> = <vetre> [<pas>] [<adv>] <ppas> --> 
              ppas.a = vrai -> ppas.n := vrai #
```

Une règle supplémentaire permet d'améliorer la modélisation des constructions étudiées. Pour qu'un verbe transitif à la forme passive puisse s'employer à toutes les personnes du singulier et du pluriel, il faut que le complément d'agent soit du type humain. Soit :

```
<REGLE4>=<prps><vetre>[<pas>][<adv>]<ppas> --> 
            [[ppas.a=vrai et
              prps.sing<>vrai et
              prps.pers<>3 ] ->
                          [ppas.h=1 ou ppas.anim=vrai]#
```

exemple

le verbe ''obtenir'' est transitif, mais n'est jamais employé avec un complément d'objet direct humain. De ce fait, la forme passive est obligatoirement impersonnelle.

'' il est obtenu ''

est correct.

'' je suis obtenu '' est incorrect.

b) la construction des verbes.

Les règles que nous avons écrites sont tirées de [Gross 75] et de [Le Goffic 75]. Ces deux études répertorient l'ensemble des constructions possibles d'une grande partie des verbes du français. Les constructions envisagées spécifient les catégories syntagmiques des constituants s'articulant autour du verbe. Les catégories syntagmiques qui ont été retenues, dans notre étude, sont les suivantes :

N : le syntagme nominal, simple ou complexe.

Exemple :

la souris, la souris blanche, la belle souris blanche, je, tu...

Dans cette catégorie, nous distinguons les groupes nominaux de type animé et de type inanimé. En effet, ils donnent lieu à deux constructions:

- verbe quelqu'un
- verbe quelque chose

V: le groupe verbal dont nous étudions la structure.

INF: le syntagme verbal infinitif et ses compléments.

exemple : ne pas recevoir, apprendre à conduire, me le dire...

QUE: suivi de IND ou SUBJ introduit les complétives à l'indicatif ou au subjonctif.

exemple : il faut que je le fasse : N V QUE-SUBJ

il pense que je viendrais : N V QUE-IND

ADJ: l'adjectif attribut du sujet.

exemple : il devient *grand*

Les constructions étudiées sont soit directes soit indirectes. Dans ce dernier cas, les seules prépositions *''à''* et *''de''* sont prises en compte dans cette étude. Ces notations étant introduites, la figure 2 schématise l'ensemble des constructions envisagées.

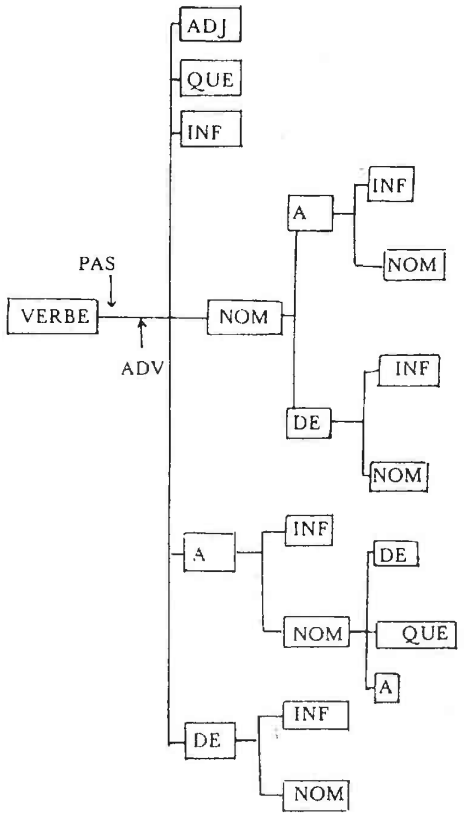


Figure 2: structure du verbe

Si nous avions voulu définir à l'aide du modèle triclasse ces différentes constructions, il aurait fallu regrouper chacun des verbes en catégories. Dans ce cas, deux verbes appartiennent à la même catégorie s'ils admettent des constructions identiques. Or, sur l'ensemble des 3000 entrées verbales de M. Gross (la classification est beaucoup plus complète dans cette étude que celle définie ci-dessus), une classe ne comporte en

moyenne que 1,5 verbe (2000 classes). Une telle approche n'était donc pas envisageable. Par contre, le système de règles que nous venons de décrire permet aisément un tel traitement.

Les règles :

Une façon simple de prendre en compte les différentes constructions verbales est d'affecter à chaque verbe du lexique un ensemble de traits définissant les structures syntaxiques du verbe. Ainsi les verbes :

rester, devenir, ...

sont affectés du trait "ADJ" indiquant la validité de la construction V ADJ. De même, nous définissons les traits suivants :

traits	construction	exemple
I	V I	aller
Q	V QUE	falloir
N	V N	manger
A	V	appartenir
D	V D	provenir
NAI	N A INF	donner
NAN	N A N	preter
NDI	N D INF	prier
NDN	N D N	recevoir

.....

Ces traits étant définis, on peut écrire le système de régles syntaxiques suivant :

```
<REGLE>=<VERBE>[<PAS>][<ADV>]<SUITE>;
<SUITE>=<SUITE1>/<SUITE2>/<SUITE3>/.....;
<SUITE1>=<ADJ> -> VERBE.A=VRAI#;
<SUITE2>=<INF> -> VERBE.I=VRAI#;
<SUITE3>=<QUE> -> VERBE.Q=VRAI#;
<SUITE4>=<NOM> -> VERBE.N=VRAI#;
<NOM>  = <DET>/<NOMP>/<PRON>.....;
```

Le traitement de la pronominalisation

Dans la phrase :

Je demande à Pierre de me rendre un service

la pronominalisation de Pierre donne :

Je lui demande de me rendre un service

Ainsi le verbe *demander* autorise aussi bien la construction V A N DE INF que PRON V DE INF. Ce phénomène est décrit par les deux règles :

```
<REGLE5>=<PRON><VERBE><DE><INF> ->
            VERBE.NDI=vrai#;
<REGLE6>=<PRON>-> RIEN# <VERBE><NOM><DE><INF> ->
            VERBE.NDI=VRAI#
```

Les règles 5 et 6 agissent l'une ou l'autre selon que le verbe est précédé ou non d'un pronom.

remarque

Les constructions s'articulant autour des prépositions "à" et "de " sont plus difficiles à définir. En effet, des ambiguités structurelles sont fréquentes. Citons par exemple la phrase :

il utilise la machine à laver

Le groupe "machine à laver" forme un tout et ne valide pas la construction :

V N A N

Pour le moment, ce type de considération n'est pas encore pris en compte.

III. 3. Implantation du sytème de règles

Afin de faciliter la tâche du concepteur et de s'assurer de la fiabilité des informations syntaxico-sémantiques introduites dans la base de connaissances, nous avons développé un logiciel interactif d'aide à la création du système de règles. La structure de ce logiciel est schématisée sur la figure 3. La définition du modéle s'effectue en trois phases :

III. 3. 1. édition des règles.

Le concepteur écrit ces règles en interaction avec le lexique. En effet, c'est dans celui-ci que sont définis les traits syntaxico-sémantiques que traitent les règles. L'édition des règles est facilitée par l'utilisation d'un éditeur de texte (éditeur pleine page).

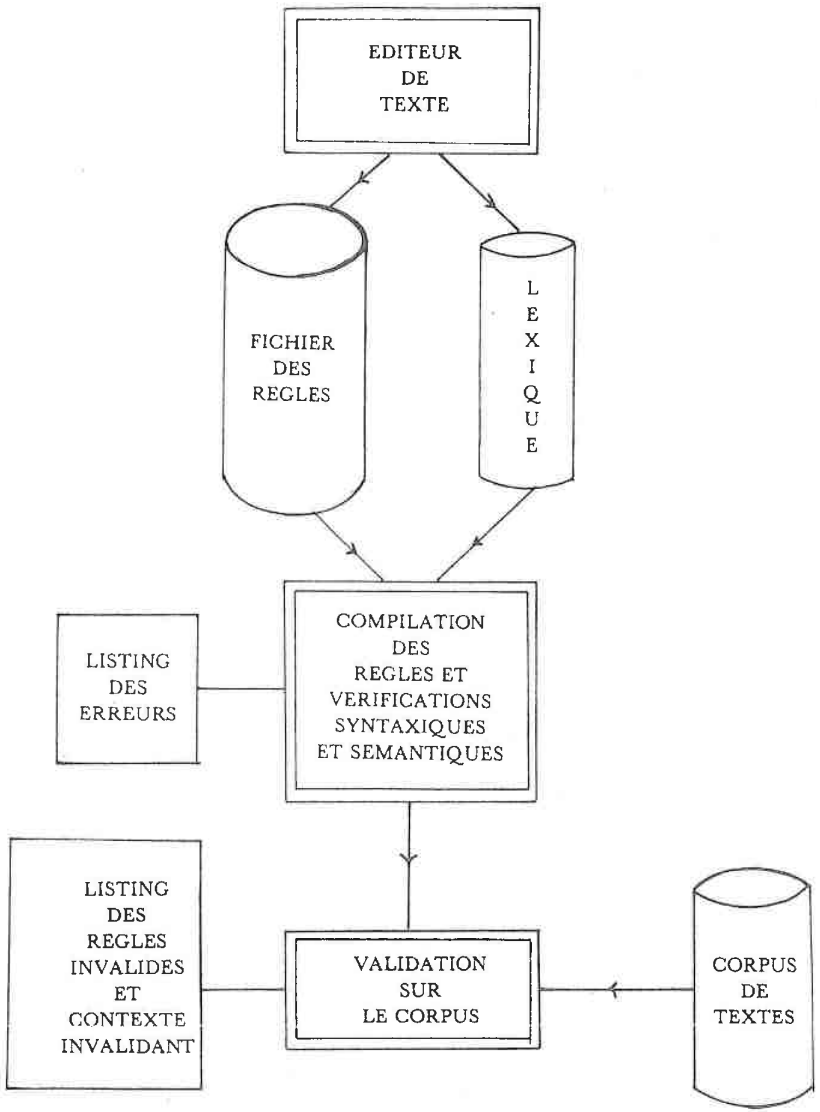


Figure 3 : Le logiciel d'aide à la création du système de règles.

III. 3. 2. compilation des règles.

Une fois le modèle entièrement édité, le système de règles est compilé. Il est alors représenté sous une forme efficace pour les traitements ultérieurs. Au cours de la compilation, un listing des erreurs syntaxiques et sémantiques est produit. Les contrôles sémantiques ont pour objet la vérification de la cohérence des objets manipulés dans les règles. La validation s'effectue en deux étapes

1) vérification de l'existence de l'objet manipulé (en interaction avec le lexique).

2) vérification de la cohérence des opérations. Les parties droite et gauche des affectations et des conditionnelles doivent être de même type booléen ou scalaire.

III. 3. 3. Validation des règles sur le corpus.

Afin de valider le modèle défini par le système de règles, celui-ci est testé sur le corpus de textes ayant servi à l'acquisition du modèle triclasse. Cette procédure apporte une aide importante au concepteur. En effet, elle facilite la mise au point dans la mesure où les règles incorrectes sont repérées. Il suffit alors de compléter ou de rejeter les règles ainsi désignées.

III. 4. Conclusion

Le sytème proposé constitue un outil efficace pour énoncer des restrictions locales sur les structures du langage. Cette approche présente l'avantage d'introduire la description partielle de certains constituants de la phrase, sans aller jusqu'à la définition d'une grammaire complète du français.

CHAPITRE 4

LES REGLES DE PHONOLOGIE ET D'EVALUATION DES ERREURS DU DECODAGE ACOUSTICO-PHONETIQUE

I. Objectif

L'objectif des règles d'évaluation et de phonologie est l'évalution de la variation du nombre d'unités phonétiques d'un mot. Celles-ci relèvent de deux types de contraintes :

- des contraintes générales de phonologie qui complétent la description phonologique intégrée au lexique. Les règles envisagées ici ne peuvent agir directement sur le lexique. En effet, elles font appel à des mesures sur le signal vocal (par exemple, les règles d'allongement).
- des éléments d'évaluation des erreurs de segmentation propres au décodeur acoustico-phonétique.

Ces règles sont interprétées au moment de l'analyse lexicale. Elles effectuent le calcul des pénalisations associées aux insertions et élisions apparaissant lors de la comparaison dynamique entre le treillis phonétique et les formes de référence.

II. Structure des règles

Les règles agissent localement sur trois types d'objets :

- les segments du treillis phonétique. Chaque segment étant défini par plusieurs étiquettes, nous avons introduit, dans le but de pouvoir se référer à un segment sous une désignation unique, les macro-classes suivantes :

  voyelle, consonne, sourde, sonore, nasale, fricative, plosive, silence

- les enregistrements des références phonétiques du lexique. Rappelons que dans un enregistrement, on trouve les différentes

réalisations possibles d'un phonème.

- les mesures fournies par le décodeur acoustico-phonétique. On accède à ces éléments par l'intermédiaire des fonctions :

  TRANS-V-N-V : fonction booléenne à valeur vrai quand la transition entre le segment en cours d'analyse du treillis et le suivant ou le précédent est du type voisé-non voisé ou non voisé-voisé

  LONG-SEGM : fonction donnant la durée en centisecondes d'un segment.

Une règle agit localement, comme indiqué sur la figure 1, sur au plus trois segments consécutifs du treillis et deux segments du graphe phonétique servant de référence à l'analyse lexicale.

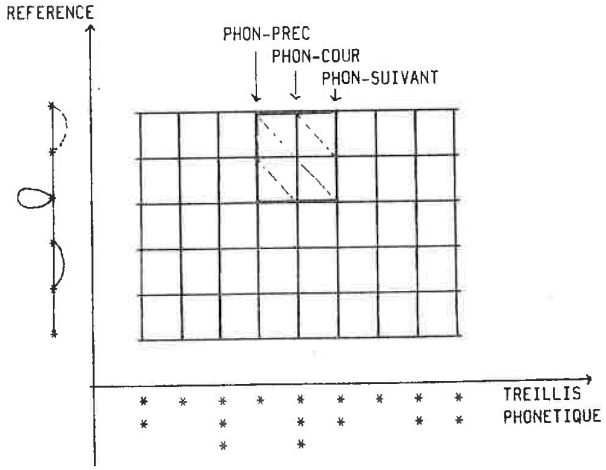


Figure 1 : champ d'action d'une règle

La figure indique le champ d'action d'une règle dans le plan de comparaison entre le treillis phonétique et les références du lexique. PHON-COUR désigne le phonème du treillis qui doit être comparé avec le segment REF d'une forme de référence. Les règles ont comme objectif d'évaluer les distances élémentaires associées à chaque arc de déplacement dans le plan TREILLIS x REFERENCE. Dans les règles, les arcs verticaux sont désignés par la variable "élision" et les arcs verticaux par la variable "insertion".

La métagrammaire du langage d'expression des règles est la suivante:

```
‹REGLE›::= { SI ‹CONDITION›} ALORS ‹CONSEQUENCE›
‹CONDITION›::=‹OBJET› ( = / › / ‹ / ‹= / ›=/eq ) ‹OBJET›
‹OBJET›::=‹VARIABLE›/‹CONSTANTE›/‹FONCTION›
‹CONSEQUENCE›::=(‹ELISION›/‹INSERTION›)=CONSTANTE
```

Une règle est constituée par une liste de conditions suivies par une conclusion. Les conditions opèrent sur des variables, des constantes ou des fonctions. La partie conclusion de la règle exprime la valeur affectée aux pénalisations des élisions et des insertions de phonèmes dans le treillis fourni par le décodeur.

```
‹VARIABLE›::= REF / PHON-COUR / PHON-PREC /
              PHON-SUIVANT / ‹FONCTION›
‹FONCTION›::=( TRANS-VNV / LONG-SEGM ) ( ‹VARIABLE› )
‹CONSTANTE›::=‹MACRO-CLASSE›/TRES-PROBABLE/
              PEU-PROBABLE/ TRES-PEU-PROBABLE
```

Les variables désignent les objets manipulés dans les règles et sont notées: REF, PHON-COUR, PHON-PREC, PHON-SUIVANT (cf figure 1). Les fonctions retournent certains traits calculés par le décodeur. Les constantes définissent soit une des macro-classes du treillis, soit un taux de pénalisation implanté sous forme numérique (TRES-PROBABLE, PEU-PROBABLE...) soit enfin un seuil.

III. Quelques exemples de règles

Exemple 1 : traitement de l'élision d'une plosive sourde après une pause (règle d'évaluation de sous-segmentation du décodage).

L'observation de l'appareil phonatoire montre que la production d'une plosive se caractérise par une rétention, suivie d'une expulsion rapide de l'air insufflé par les poumons dans le conduit vocal. Ce mode articulatoire se traduit dans le signal vocal par la présence d'un silence immédiatement suivi d'une explosion très brève. De ce fait, si une pause précède une plosive sourde, le silence la caractérisant est absorbé et sa localisation devient difficile (cf figure 2). Les erreurs de sous-

segmentation du décodeur acoustico-phonétique sont dans ce cas fréquentes. La règle suivante traite ce cas.

si PHON-PREC=SILENCE
si REF= PLOSIVE SOURDE
alors ELISION= TRES-PROBABLE

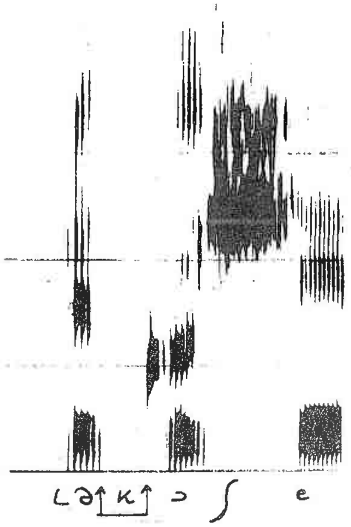


figure 2 : spectogramme d'une plosive sourde.

Exemple 2 : La détection des noyaux vocaliques.

Le décodeur acoustico-phonétique intégré à notre système met en oeuvre un algorithme de détection des noyaux vocaliques (NOCOVA) très performant. Le taux de réussite est de l'ordre de 96% [Carbonell 84].

Il est intéressant, afin d'améliorer les performances de l'analyse lexicale, de tenir compte de cet aspect, et de pénaliser différemment les erreurs dans la comparaison dynamique entre le treillis et un graphe phonétique, selon qu'elles mettent ou non en cause la détection des noyaux vocaliques. Plusieurs cas sont envisagés :

1. Il manque un phonème dans le treillis

Dans le cas où l'élision du phonème manquant n'a pas été prévue dans la description phonétique de la forme de référence, il faut que

l'analyseur lexcical hypothétise une erreur dans la segmentation du décodeur pour pouvoir poursuivre la comparaison dynamique. Si une hypothèse induit un découpage en noyaux vocaliques différent de celui fourni par NOVOCA, alors elle doit être fortement pénalisée. En effet, NOVOCA est suffisamment fiable pour ne pas avoir à mettre en doute ses résultats. Cette pénalisation dépend du contexte phonétique dans lequel l'hypothèse d'élision est émise. Nous allons répertorier ces différents contextes.

les contextes pénalisés pour une hypothèse d'élision :

L'absence d'un phonème invalide la détection des noyaux vocaliques en fonction du contexte d'analyse. Les différentes possibilités sont regroupées dans la table de vérité suivante:

phon-cour \ phon-prec	V	C
V	ref : V | C faux | vrai	ref : V | C faux | faux
C	ref : V | C faux | faux	ref : V | C vrai | faux

les indices du tableau indiquent le type vocalique ou consonantique des phonèmes du contexte d'analyse. La nature du phonème courant et du phonème précédent sont représentées respectivement sur les axes vertical et horizontal. La nature du phonème de la forme de référence (le phonème Lmanquant) est donnée dans les cases du tableau. Les cases du tableau regroupent deux informations :

- la nature du phonème manquant
- un booléen prenant la valeur vrai si l'hypothèse d'élision invalide les résultats de NOVOCA.

L'hypothèse d'élision est pénalisée dans les deux cas suivants :

a) il manque une consonne entre deux voyelles , ce qui s'exprime par la règle ci-dessous :

si PHON-PREC=VOYELLE
si PHON-COUR=VOYELLE
si REF=CONSONNE
alors ELISION= TRES PEU PROBABLE

b) l'absence d'une voyelle dans le contexte consonne-consonne se traduit par la règle :

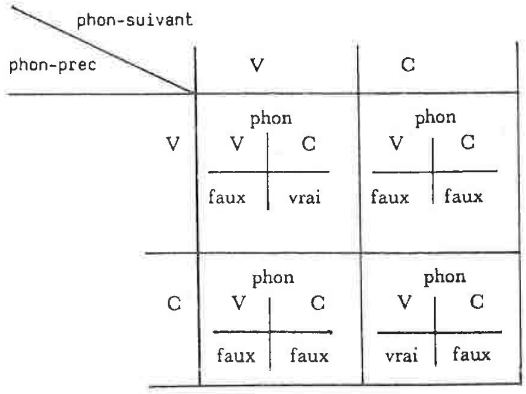
si PHON-PREC= CONSONNE
si PHON-COUR= VOYELLE
si REF= VOYELLE
alors ELISION = TRES PEU PROBABLE

2) Il y a un phonème supplémentaire dans le treillis

Si le phonème inséré n'est pas prévu dans le graphe phonétique du mot en cours d'analyse, il faut remettre en question la segmentation effectuée par le décodeur : il y a sur-segmentation. Comme précédemment, plusieurs cas sont à envisager en fonction du contexte de l'insertion et des noyaux vocaliques.

Les contextes à pénaliser

Les différentes possibilités sont regroupées dans le tableau suivant:



Les indices du tableau indique la nature vocalique ou consonantique du contexte de l'insertion.

Les intersections ligne-colonne donnent selon le type du phonème inséré dans le treillis l'influence sur NOVOCA.

Deux cas sont à envisager:

a) insertion d'une consonne entre deux voyelles

si PHON-PREC = VOYELLE
si PHON-SUIV = VOYELLE
si PHON-COUR = CONSONNE
alors INSERTION = TRES PEU PROBABLE

b) insertion d'une voyelle entre deux consonnes

si PHON-PREC = CONSONNE
si PHON-SUIV = CONSONNE
si PHON-COUR = VOYELLE
alors INSERTION = TRES PEU PROBABLE

Exemple 3 : règle de phonologie générale sur l'allongement d'une voyelle lorsqu'elle est suivie d'un des phonèmes R, Z , z ou v.

Comme on le verra dans le chapitre suivant, l'analyseur lexical utilise entre autres, comme critère d'évaluation, la durée en centisecondes des unités phonétiques proposées par le décodeur. Cette mesure permet de pénaliser les hypothèses de sur et sous segmentation en fonction d'une valeur moyenne nécessaire à la réalisation d'un phonème. Si celle-ci est faible par rapport à la moyenne, il est possible qu'il y ait eu sursegmentation. Par contre, si elle est grande, une sous-segmentation est possible. Avec le formalisme que nous avons introduit, ces différentes considérations s'exprime par:

(1) si REF = VOYELLE
si REF.SUIV = [R, Z, z, v]
si LONG-SEGM (PHON-COUR) > SEUIL
alors ELISION = PEU PROBABLE

où SEUIL est fonction de la durée phonétique moyenne.

(2) si REF<> VOYELLE
si LONG-SEGM(PHON-COUR) > SEUIL
alors ELISION = PROBABLE

(3) si REF.SUIV<>[r,Z,z,v]
si LONG-SEGM(PHON-COUR)>SEUIL
alors ELISION=PROBABLE

Exemple 4 :Les transitions voisé-non voisé.

La production, par l'appareil phonatoire, d'un phonème voisé suivi d'un phonème non-voisé et vice versa, passe par une période transitoire pendant laquelle les cordes vocales restent dans l'état d'excitation ou de non excitation précédent. Ceci conduit fréquemment à l'apparition , dans le treillis, d'un segment parasite de courte durée (quelques centisecondes). Celui-ci appartient généralement, au caractère de voisement près, à la même classe phonétique que l'un des deux phonèmes consciemment émis par le locuteur. Nous avons représenté sur la figure 3, la table des correspondances phonétiques. Ainsi le mot "appât" sera fréquemment décodé par la suite :

a, b, p, a

b phonème parasite

Ces considérations nous amènent à revoir la règle 4 donnée dans l'exemple 3 au paragraphe précédent, et à lui substituer le système suivant:

si LONG(PHON-COUR)< SEUIL
si TRANSITION-V-NV
si PHON-COUR eq REF
alors INSERTION=TRES PROBABLE

si LONG(PHON-COUR)<SEUIL
si TRANSITION V-NV
si PHON-COUR eq REF-PREC
alors INSERTION= TRES PROBABLE

si LONG(PHON-COUR)<SEUIL
alors INSERTION=PROBABLE

REMARQUE :

si deux règles donnent deux évaluations différentes de insertion ou élision, alors l'évaluation la moins pénalisante est choisie.

p	b
t	d
k	G
f	v
s	Z
ch	z
p,t,k	m,n

table de correspondance utilisée quand le caractère de voisement d'un phonème est altéré.

Figure 3

4) CONCLUSION

L'approche par règles constitue une aide importante au concepteur, dans la phase de mise au point. Elle présente plusieurs avantages tels que

- une grande souplesse dans la définition de la stratégie d'analyse
- l'interactivité qui permet de tester efficacement les différentes stratégies envisageables.

Cependant, la recherche lexicale, sur laquelle agissent les règles de stratégie que nous venons de définir, constitue le niveau le plus sollicité de la reconnaissance. C'est pourquoi nous avons implanté de façon procédurale ces règles dans la version définitive du système, en vue d'optimiser, en temps de calcul, l'analyse.

PARTIE C

LA STRATEGIE ET LES ALGORITHMES MIS EN OEUVRE DANS LE MODE EDITION-PAROLE

Chapitres :

1. Présentation générale et architecture

2. L'analyse lexicale

3. Introduction de contraintes syntaxiques dans l'analyse

CHAPITRE 1

PRESENTATION GENERALE

ET

ARCHITECTURE ADOPTEE

I. INTRODUCTION

Dans l'approche analytique de la reconnaissance que nous avons adoptée, le problème que nous nous proposons de résoudre est le suivant :

il s'agit de mettre en oeuvre l'ensemble des connaissances linguistiques connues du système, afin de proposer une ou plusieurs interprétations possibles du treillis phonétique fourni par le système acoustico-phonétique.

Une façon d'atteindre cet objectif est la recherche d'un chemin optimal dans le treillis phonétique de telle manière que l'interprétation résultant du cheminement corresponde à une suite cohérente de mots. Comme indiqué sur la figure 1, les algorithmes mis en oeuvre doivent faire preuve d'une certaine capacité d'adaptation face à deux types de difficultés :

- les erreurs de segmentation et d'identification du décodeur
- les variations inter et intra- locuteur dans la réalisation phonétique acceptable d'un même mot.

On voit que l'interprétation de la phrase :

cher Monsieur,

je vous prie de trouver ci-joint l'attestation

nécessite, pour la recherche d'un chemin continu, de pouvoir selon les cas:

- hypothétiser un phonème manquant. C'est le cas du son / R / de ''cher''.
- substituer aux phonèmes présents dans un segment un autre phonème, comme c'est le cas du /v/ de ''trouver''.

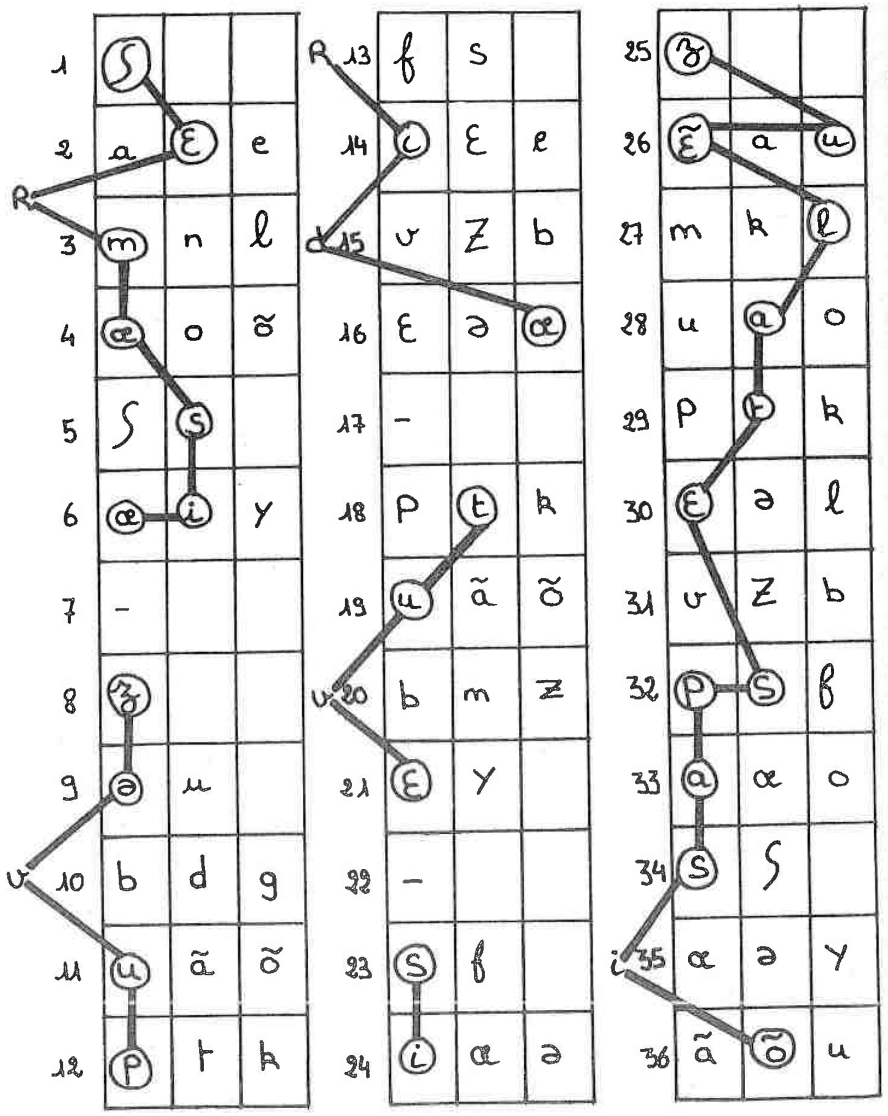


figure 1 . Interprétation du treilli phonétique représentant la phrase: " cher monsieur, je vous prie de trouver ci-joint, l'attestation "

- éliminer un segment parasite (segment 15, 31).

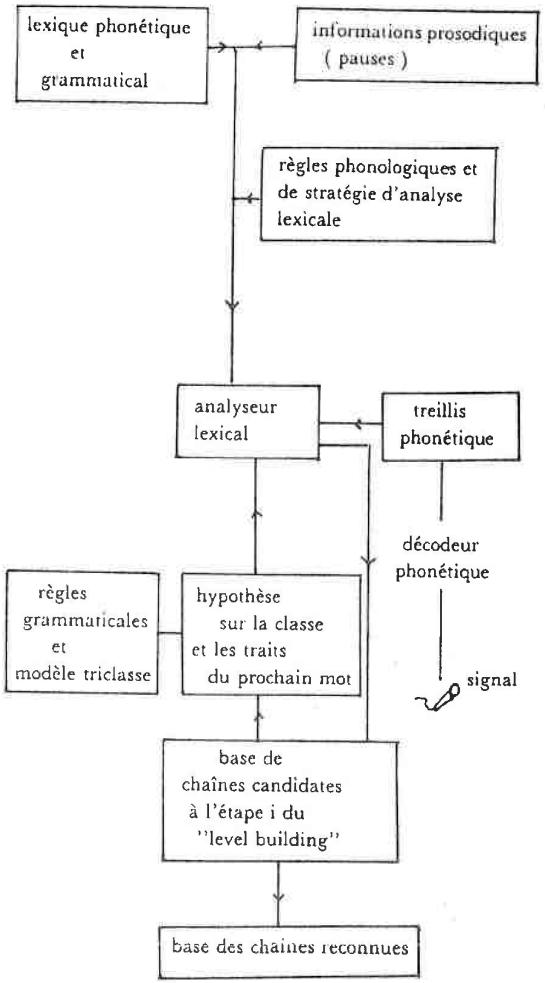
Nous nous proposons de décrire dans cette partie une méthode de résolution issue des techniques de programmation dynamique déjà très largement utilisées dans les domaines de la reconnaissance de mots isolés et enchaînés.

II. Architecture mise en oeuvre.

Le module de reconnaissance que nous avons réalisé est essentiellement construit autour d'un analyseur lexical (cf figure 2). Dans la stratégie générale de prédiction-vérification adoptée, l'objectif de l'analyseur est l'évaluation heuristique d'un taux de dissemblance entre une portion du treillis phonétique et l'ensemble des mots émis en hypothèse par le niveau syntaxique.

Le processus de reconnaissance choisi au niveau de la phrase est une adaptation de l'algorithme de ''level building'' [Myers 81] avec recherche multiple en faisceaux. Le choix de cet algorithme a été dicté par le problème posé du fait de la prédominance des mots courts en reconnaissance de parole continue. En effet, d'un point de vue probabiliste, il est plus facile de décoder sans erreurs un mot court qu'un mot long. De ce fait, parmi l'ensemble des interprétations possibles du treillis, celles qui sont constituées de mots courts sont toujours prédominantes par rapport à celles formées de mots longs.

Cette difficulté est levée dans la stratégie en palier adoptée car les choix, au sens de la programmation dynamique, s'appliquent à des chaînes comportant un nombre identique de mots.



architecture générale du module de reconnaissance

figure 2

La reconnaissance est réalisée par un processus itératif par lequel on construit une base d'hypothèses partielles. Dans cette base, on trouve à l'étape n l'ensemble des chaînes de n mots donnant une interprétation partielle du treillis phonétique. A chaque chaîne, sont associés trois types d'informations :

- le score de reconnaissance.
- le numéro du dernier segment interprété du treillis.
- la classe grammaticale et la liste des traits syntaxico-sémantiques que possède chaque mot de la chaîne, dans le contexte où il a été reconnu.

Dès qu'une chaîne fournit une interprétation complète du treillis, celle-ci est supprimée de la base pour être rangée dans la base des chaînes reconnues.

Le processus de reconnaissance se poursuit jusqu'à ce que la base des chaînes partielles soit vide. Ce processus est schématisé par l'algorithme suivant.

TANT QUE base non vide FAIRE

POUR chaque classe du lexique FAIRE

SELECTIONNER parmi les chaînes reconnues au niveau précédent celles qui peuvent être suivies d'un mot appartenant à la classe courante;

CALCULER en interprétant les règles grammaticales, pour chaque chaîne sélectionnée les traits que doit posséder obligatoirement le mot qui va être identifié dans le treillis et concaténé ensuite à la chaîne traitée;

LANCER l'analyse lexicale sur la classe courante;

CONCATENER les mots reconnus aux chaînes selectionnées précédemment et calculer les scores cumulés;

ELIMINER les chaînes en fonction du score obtenu

FIN POUR

RANGER dans la base des chaînes reconnues les chaînes interprétant complétement le treillis

FIN TANT QUE

CHAPITRE 2

L'ANALYSE LEXICALE

I. Objectif

Dans un système de reconnaissance de parole continue, l'analyse lexicale joue un rôle important. Son objectif est la localisation et/ou la validation de mots, selon que la stratégie d'analyse est descendante, ascendante ou mixte. Dans notre système, de type descendant, c'est l' aspect validation de l'analyse lexicale qui nous a intéressés.

Il s'agit de valider les mots émis en hypothèse par les autres niveaux de traitement en comparant la représentation phonétique de ces mots avec une portion du treillis phonétique issu du décodage acoustico-phonétique. L'objectif du traitement est double et conduit à :

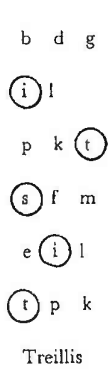
- la représentation phonétique du signal.
- localiser cette forme de référence dans le treillis phonétique.

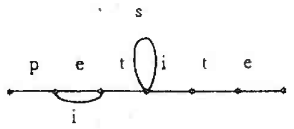
La difficulté provient à la fois de l'imperfection de la description phonétique des formes de référence et des erreurs de décodage du système acoustico-phonétique face à la grande variabilité de la parole.

II. Méthode de résolution dans Myrtille I

a) description générale.

La solution retenue est une adaptation de la méthode développée dans Myrtille I. Ce type de résolution se ramène à la recherche d'un chemin dans le treillis de phonèmes représentant la description phonétique du mot à identifier (cf. figure 1).





Graphe de référence du mot : petite

Figure 1 : recherche d'un chemin

Dans cette approche, lorsqu'un arc du graphe de référence est validé dans le treillis phonétique, un taux de dissemblance local lui est associé. Ce taux est fonction de la nature des altérations pouvant survenir (cf figure 2).

nature de l'arc	taux de dissemblance associé
	0 phonéme standard
	TSUB phonème pouvant se substituer au phonème standard
	TINS phonème pouvant s'insérer
	TEL phonéme optionnel

Figure 2. définition du taux de dissemblance

Le cheminement permet ainsi de définir pour chaque mot testé un score de dissemblance en fonction du chemin parcouru. Si en cours de traitement, une impasse due à l'impossibilité de mettre en correspondance un segment du treillis et un élément du graphe de référence interrompt le cheminement, l'analyseur émet des hypothèses d'élision, de substitution et d'insertion de phonèmes pour synchroniser les deux formes de la manière suivante:

i) Hypothèse d'élision.

l'analyseur suppose l'élision d'un phonème dans le treillis. La synchronisation se réalise alors par la progression d'un pas dans le graphe de référence. Si la poursuite du chemin est possible, il s'ajoute au taux de dissemblance cumulé (TDC) une pénalisation THYPEL. Dans le cas contraire, il y a émission d'une deuxième hypothèse.

ii) Hypothèse de substitution.

Le recalage, lorsque le décodeur a commis une erreur d'étiquetage du segment en cours d'analyse, s'obtient par la progression d'un pas dans le graphe phonétique et dans le treillis. Si l'hypothèse est justifiée, la pénalisation du chemin résultant s'obtient en cumulant à TDC la pénalité

THYPSUB. Par contre, si la poursuite du chemin est toujours impossible, il y a émission d'une troisième hypothèse.

iii) Hypothèse d'insertion.

Dans le cas où les deux hypothèses précédentes ont échoué, la progression se poursuit par l'élimination d'un segment du treillis supposé invalide. Le cumul est déterminé par la formule:

TDC = TDC + THYPINS.

En cas d'échec sur le prochain segment, la recherche est abandonnée.

Dans tous les cas, si le score de dissemblance cumulé dépasse une certaine valeur, le chemin en cours est abandonné et le mot en cours d'analyse est rejeté.

b) Algorithme.

i) notation.

TER(K,J)	le graphe phonétique .
K	le numéro d'un segment du graphe phonétique.
J	varie de 1 à 3 et correspond aux différents types d'arc du graphe.
	TER (K,1)-> phonème standard de la transcription.
	TER (K,2)-> phonème pouvant se substituer à la transcription standard.
	TER (K,3)-> phonème pouvant s'insérer.
	TER (K,1) < 0 si le segment peut être omis.
T (I,L)	le treillis phonétique.
I	le numéro d'un segment du treillis.
L	l'une des trois étiquettes possibles d'un segment.

ii) algorithme.

L'algorithme est schématisé sur la figure 3 donnée ci-après.

Chaque transition dans le cheminement est validée par un choix local unique. En cas d'ambiguité sur la transition, un système de priorités guide la décision :

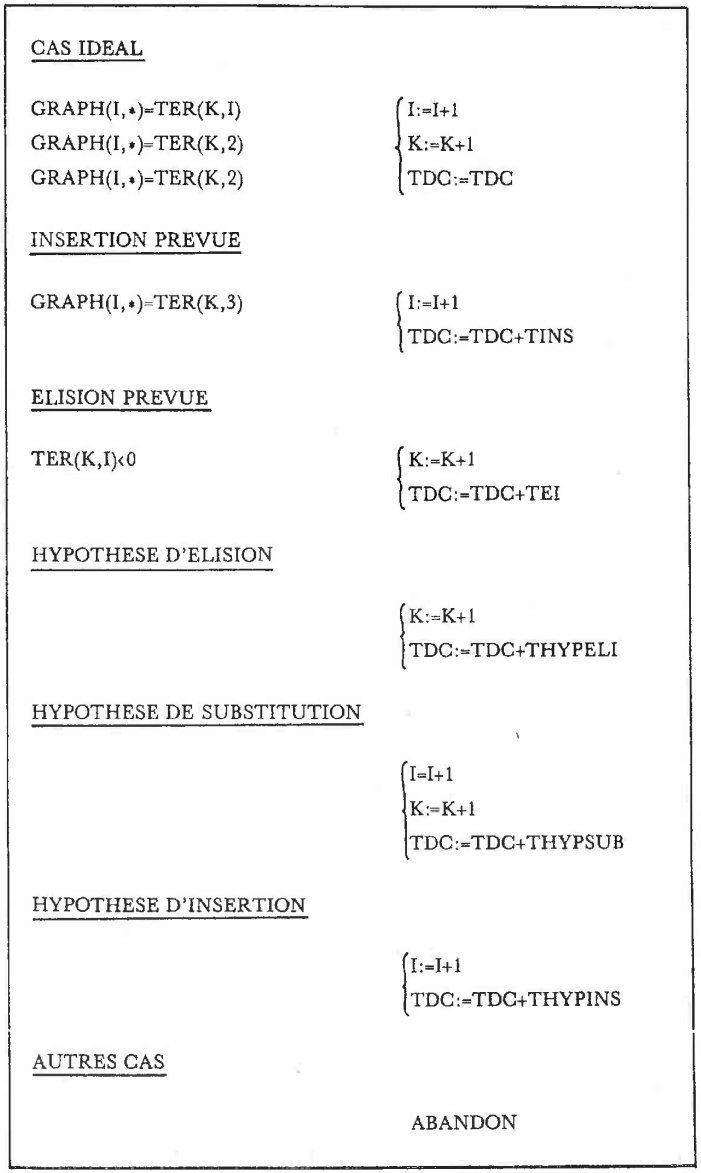


Figure 3: L'analyse lexicale dans Myrtille I

priorité 1 : bon phonème ou substitution prévue

priorité 2 : insertion prévue

priorité 3 : élision prévue

priorité 4 : hypothèse d'élision

priorité 5 : hypothèse de substitution

priorité 6 : hypothèse d'insertion

c) Conclusion.

Nous allons proposer dans la suite de ce chapitre une généralisation de cet algorithme avec deux objectifs:

1) améliorer la stratégie d'analyse : la stratégie de type ''meilleur d'abord'' mise en oeuvre dans cet algorithme ne permet pas l'obtention d'une solution optimale.

2) donner la possibilité de rechercher un mot dans le treillis, avec un flou sur le point d'ancrage.

III. Recherche lexicale et programmation dynamique

III. 1. Introduction

Les techniques de programmation dynamique ont connu un succès certain dans les domaines de la reconnaissance des mots isolés ou enchaînés. Dans ces systèmes, les formes de référence décrivant le vocabulaire de l'application sont codées sous la forme d'une suite de vecteurs, dont les composantes sont des paramètres acoustiques extraits du signal (par exemple, sorties d'un banc de filtres ou coefficients cepstraux). Les algorithmes de programmation dynamique ont pour objet, dans une telle approche, le recalage temporel entre le signal vocal et les formes de référence. Ce recalage est rendu nécessaire par les variations dans la vitesse et le rythme d'élocution entre les phases d'apprentissage et de reconnaissance.

Nous allons voir que ces techniques sont adaptables au traitement de la parole continue. Dans ce cas, l'objectif est différent. Il s'agit de comparer un graphe de référence, incluant les diverses variations phonologiques d'un mot, avec la représentation de type phonétique ou syllabique qui a été extraite du signal vocal. Dans cette approche, le recalage a pour objectif de synchroniser la forme de référence et la représentation du signal

lorsque une erreur de segmentation ou d'étiquetage apparait lors du décodage acoustico-phonétique. De tels algorithmes ont déjà été implantés efficacement dans différents systèmes [Sauter 84], [Kobayashi 85], [Levinson 83], [Mari 79], [Vives 85].

Dans ce chapitre, nous présentons plusieurs algorithmes de comparaison dynamique entre deux formes phonétiques. Nous verrons d'abord un algorithme d'analyse sur des mots isolés. Nous généraliserons ensuite cet algorithme à la reconnaissance de la parole continue. Notre approche est originale par la différenciation du traitement entre les trois types de recalage possibles (élision, substitution, insertion), et par l'introduction de contraintes temporelles.

III.2. Quelques éléments utiles sur la programmation dynamique.

III.2.1 Le principe d'optimalité de Bellman.

La stratégie de recherche, que nous avons développée dans nos algorithmes, est issue du principe de Bellman :

> quels que soient l'état initial et la décision initiale, les décisions qui restent à prendre doivent constituer une stratégie optimale compte tenu de l'état résultant de la première décision.''

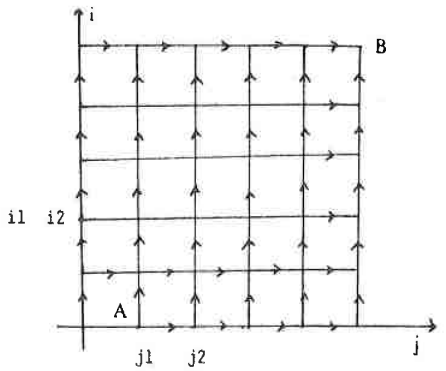
Bellman '' la programmation dynamique et ses applications ''

[Bellman 57]

III.2.2. Application à la recherche dans un graphe.

Ce principe, lorsqu'il est appliqué à la recherche d'un chemin optimal dans un graphe, optimise de façon significative le parcours.

Soit le graphe orienté suivant :



où chaque arc reliant deux points du graphe est affecté d'une pondération d([i1,j1]. [i2,j2])

Appliquons le principe d'optimalité à la recherche d'un chemin optimal entre les points A et B. Pour ce faire, il suffit de calculer pour chaque noeud [i,j] du graphe une distance optimisant le parcours depuis A de la manière suivante :

$$D([i,j]) = \text{optimum} \begin{cases} D([i-1,j]) + d([i-1,j][i,j]) \\ D([i,j-1]) + d([i,j-1][i,j]) \end{cases}$$

$$D([0,0]) = D(A) = 0$$

Le chemin résultant est optimal compte-tenu de l'état précédent (et donc, par récurence, de la première décision) et des choix qui restent à prendre.

Il n'est donc pas nécessaire de parcourir l'ensemble des chemins possibles pour aboutir à la solution, puisqu'à chaque étape du parcours une décision a été prise. Dans le cas d'un graphe binaire, l'algorithme de recherche passe d'une complexité

$$\sum_{i=1}^{I} i^2$$

en évaluant tous les chemins à un algorithme de complexité

$$N = \sum_{i=1}^{I} i$$

en appliquant le principe d'optimalité de Bellman

où i repésente le nombre d'étapes nécessaires au parcours et N le nombre de noeuds du graphe.

III.2.3 Application à la reconnaissance de mots isolés.

III.2.3.1 analyse mot par mot.

a) introduction

Nous supposerons, dans un premier temps, que chaque mot du dictionnaire dispose d'une représentation phonétique indépendante et que les frontières des mots à comparer sont connues sans ambiguités.

b) la fonction de recalage

Il s'agit alors de comparer deux formes phonétiques :

- GRAPH [i,m] i : 1 --> I
  m : 1 --> 3

forme de référence d'un mot, où I représente la longueur en nombre de segments du graphe phonétique et où m correspond à la désignation des trois variantes phonologiques de chaque segment:

m = 1 ---> phonème le plus probable
m = 2 ---> phonème pouvant se substituer à GRAPH [i,1]
m = 3 ---> phonème pouvant s'insérer devant le segment i

et

- T[j,n] j: 1 ---> J
  n: 1 ---> 3

treillis phonétique fourni par le système acoustico-phonétique.

J représente la longueur en nombre de segments du treillis.

n permet l'accès aux trois étiquettes (ou moins) associées à chaque segment.

La comparaison dynamique entre les deux formes phonétiques consiste à chercher dans le graph GRAPH x T un chemin optimum. Plus formellement, il s'agit de trouver parmi l'ensemble des fonctions de recalage possibles celle qui est optimale au sens d'une certaine métrique. Les différentes fonctions de recalage W sont définies de la manière suivante :

$$[1,\ldots,I+J+1] \xrightarrow{W} [1,\ldots,I]x[1,\ldots J]u[I+1,J+1]$$

$$k \longrightarrow W(k)= [I(k),J(k)]$$

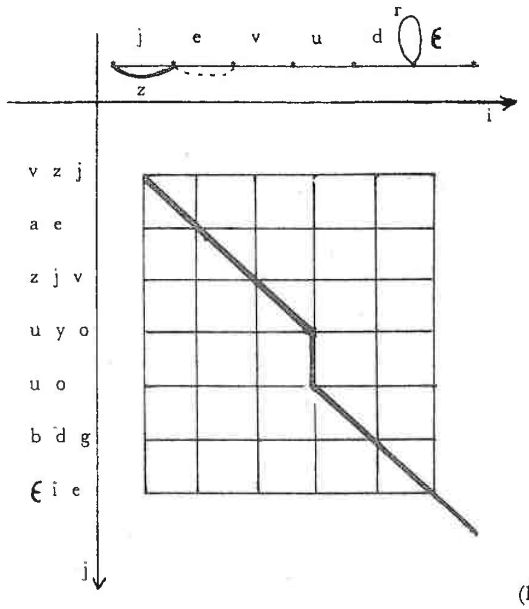
où :

W(k) indique que la fonction de recalage passe par les points de coordonnées I(k),J(k).

REMARQUE

Les points I(k),J(k) ne correspondent pas à la comparaison des segments I(k) du graphe phonétique et J(k) du treillis. Ce point est essentiel et constitue l'originalité de notre méthode par rapport aux algorithmes de reconnaissance globale. Il ne s'agit plus d'effectuer uniquement un recalage temporel par le calcul de distances inter-segment, mais d'évaluer un chemin dynamique tenant compte des diverses insertions, élisions et substitutions intervenant dans le treillis.

La figure 1 donne un exemple d'une fonction de recalage. Le point particulier [I+1,J+1] correspond à la mise en coincidence des segments GRAPH[I] et T[j].



recherche d'un chemin optimum
dans treillis x ref
figure 1

La métrique associée à la fonction de recalage s'exprime par :

$$D(W) = D([i(k),j(k)])$$

$$= \sum_{k=1}^{K} d([I(k-1),J(k-1)],[I(k),J(k)])$$

où :

d([I(k-1),J(k-1)] ,[I(k), J(k)]) représente une pondération associée à chaque arc élémentaire [W(k),W(k+1)] de déplacement dans le graphe et

K la longueur du chemin.

Le but de la recherche est alors de trouver le chemin optimal $\hat{W}$ défini par:

$$\hat{W} = \text{ARG MIN} \sum_{I(k),J(k)} d([I(k-1), J(k-1)],[I(k), J(k)])$$

$$k$$

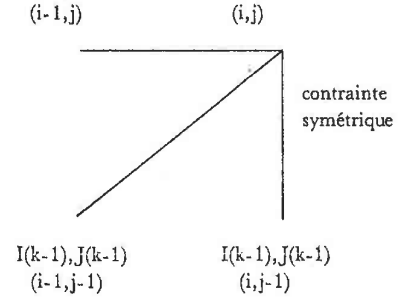
c) Les contraintes de déplacement

Afin de respecter la réalité physique de l'évolution temporelle de la parole, la fonction de recalage est soumise à une condition de monotonie qui s'exprime par les relations :

$$\begin{cases} I(k) <= I(k+1) \\ J(k) <= J(k+1) \end{cases}$$

D'autre part, dans le but d'éviter dans le cheminement des déplacements incohérents, les fonctions I et J doivent respecter les contraintes locales suivantes:

$$\begin{cases} I(k) = I(k-1) + (1,0) \\ J(k) = J(k-1) + (1,0) \end{cases}$$



De plus, les mots à reconnaitre étant isolés, les conditions aux limites sont connues :

Conditions initiales.

La recherche débute sur les premiers segments du treillis et du graphe de référence, soit:

$$\begin{cases} I(1) = 1 \\ J(1) = 1 \end{cases}$$

Conditions finales.

Le treillis et la référence doivent coincider totalement, ce qui conduit à:

$$\begin{cases} I(k) = I+1 \\ J(k) = J+1 \end{cases}$$

d) Les formules de programmation dynamique

En appliquant le principe d'optimalité de Bellman, il est possible d' écrire la définition de la fonction de recalage optimale de la façon suivante :

$$\hat{W} = \underset{I(k),J(k)}{\arg} D(I(k),J(k))$$

avec D défini récursivement par :

$$\begin{cases} D(I(k),J(k)) = \underset{I(k),J(k)}{\min} \begin{cases} D([I(k-1),J(k-1)])+ \\ d([I(k-1),J(k-1)],[I(k),J(k)]) \end{cases} \\ D(I(1),J(1)) = 0 \end{cases}$$

En intégrant les contraintes locales dans la formulation, les relations précédentes deviennent:

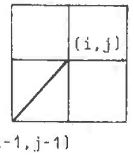
$$\begin{cases} D(i,j) = \min \begin{cases} D(i-1,j)+d[(i-1,j),(i,j)] \\ D(i,j-1)+d[(i,j-1),(i,j)] \\ D(i-1,j-1)+d[(i-1,j-1),(i,j)] \end{cases} \\ D(1,1) = 0 \end{cases}$$

Une méthode analogue de comparaison dynamique de chaînes de symboles de longueurs différentes a déjà été proposée par [Haton 74].

e) Evaluation des pénalisations associées aux arcs

Les pénalisations sont calculées en traitant différemment les cas possibles d'altération du treillis T par rapport à la forme de référence GRAPH. L'évaluation se fait par le calcul d'une distance interphonème et par utilisation des règles de phonologie et d'évaluation des erreurs de segmentation (cf partie B).

1) évaluation d'un arc diagonal.

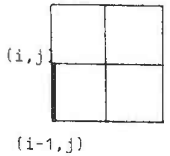


Si le chemin optimum passe par le point [(i-1),(j-1)] du graphe GRAPH x T , la poursuite au point [i,j] n'a de sens que si les segments i-1 et j-1 coincident. La pénalisation associée à l'arc doit donc être fonction de la distance interphonème entre les segments (i-1) et (j-1) :

D(i-1,j-i,i,j) = min ph (GRAPH [i-1,m], T [j-1, n])
m: 1--->2
n: 1--->3

où ph donne la distance interphonème.

2) évaluation d'un arc vertical.



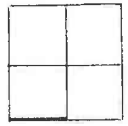
Si W(k) = [i-1,j] est optimum, la poursuite sur [i,j] de la fonction de recalage indique une synchronisation du graphe de référence par rapport au treillis. Elle n'a de sens que si les segments i-1 et j ne coincident pas. La continuation du chemin par le déplacement [i-1,j],[i,j] correspond dans ce cas à la résolution de deux cas possibles:

- il y a eu une erreur de sous-segmentation lors du décodage acoustico-phonétique. La pénalisation qui s'ensuit est alors fonction du contexte phonétique et de la durée du segment j du treillis. Le calcul s'effectue par interprétation des règles de phonologie et de stratégie.

- le segment i du graphe phonétique est optionnel. Le cas était prévisible et la distance d s'exprime par:

d([i-1,j],[i,j]) = ELIS

3) évaluation d'un arc horizontal.



Le choix d'avancer d'un pas dans le treillis phonétique après (i,j-1) a pour objectif la comparaison entre les segments GRAPH(i) et T(j). Le segment i du graphe de référence ayant déjà été comparé précédemment avec T(j-1), le déplacement n'a de sens physique que dans les trois cas suivants:

- les segments i et j-1 ne coincident pas.Le décodeur acoustico-phonétique a commis une erreur de sur-segmentation. L'évaluation est alors calculée par application des régles de phonologie et de stratégie.

- GRAPH [i,3] et T [j-1] coincident. Une insertion est donc prévue dans le graphe phonétique et les segments i-1 et i . Vu que GRAPH [i] contient les descriptions du prochain segment, la poursuite normale de W est:

W(k) = [i,j]. Dans ce cas, la pondération de l'arc s'exprime par :

d(i,j-1,i,j) = Min PH (GRAPH[i,3],T[j-1,n])
n=1->3

- Les segments i et j-1 coincident et il y a une répétition de phonèmes dans le treillis. Ce cas est traité pendant le décodage acoustico-phonétique et n'est donc pas envisagé dans l'algorithme que nous proposons.

f) contraintes supplémentaires sur la fonction de recalage

Pour optimiser la comparaison dynamique, nous imposons des contraintes supplémentaires sur la fonction de recalage. Nous n'admettons pas en effet que deux erreurs consécutives de segmentation se produisent. L'ensemble des configurations possibles est regroupé dans le tableau de la figure 2. Les transitions autorisées sont marquées par la valeur 1.

	elp	elnp	ip	inp
elp	1	1	1	1
elnp	1	0	1	0
ip	1	1	1	1
inp	1	0	1	0

elp : élision prévue
elnp : élision non prévue
ip : insertion prévue
inp : insertion non prévue

Figure 2 : contraintes sur la fonction de recalage

g) l'algorithme

étape 1 : initialisation

Pour i de 1 à I faire
Pour j de 1 à J faire D(i,j) = infini
D(1,1) = 0

étape 2 : évaluation de la fonction
de recalage

pour j de 1 à J faire
pour i de 1 à I faire
étape 3
étape 4
Fin pour
Fin pour

étape 3 : évaluation des déplacements élémentaires

d(i,j,i+1,j+1)= infini
d(i,j,i+1,j) = interprétation-règles (i,j)
d(i,j,i,j+1) = interprétation-règles (i,j)

Pour n de 1 à 3 faire
  pour m de 1 à 2 faire

arc diagonal

dist = PH(GRAPH[i,m],T[j,n])
si dist < d(i,j,i+1,j+1) alors d(i,j,i+1,j+1) =dist

fin pour

arc horizontal

dist = PH (GRAPH [i,3], T[j,n])
si dist <d(i,j,i,j+1) alors d (i,j,i,j+1)= dist

fin pour

arc vertical

si GRAPH [i,1] < 0 alors d(i,j,i+1,j) = ELIS

étape 4: calcul des distances cumulées

$$D(i+1,j) = \text{MIN} \begin{cases} D(i+1,J) \\ D(i,j) + d(i,j,i+1,j) \end{cases}$$

$$D(i,j+1) = \text{MIN} \begin{cases} D(i,j+1) \\ D(i,j) + d(i,j,i,j+1) \end{cases}$$

$$D(i+1,j+1) = \text{MIN} \begin{cases} D(i+1,j+1) \\ D(i,j) + d(i,j,i+1,j+1) \end{cases}$$

étape 5: résultat de l'évaluation

$$D(W) = D(I+1,J+1)$$

REMARQUES

- La procédure interprétation-régles active l'interpréteur de règles de phonologie et de stratégie et elle permet le calcul de certaines distances locales en fonction du contexte d'analyse donné en paramètre.

- La reconnaissance d'un mot s'obtient par itération de l'algorithme ci-dessus sur l'ensemble du vocabulaire. Le mot reconnu est alors celui qui possède la meilleure fonction de recalage.

- Les distances élémentaires sont calculées à l'étape 3 et cumulées dans le domaine de définition de W.

h) Introduction d'un flou sur les conditions initiales et finales

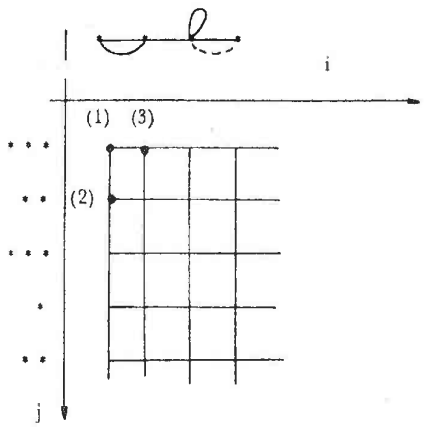
Les algorithmes de détection parole-non parole ne permettent pas toujours d'extraire correctement le signal vocal du continuum sonore

ambiant. Il en résulte un certain flou sur le point d'ancrage et le point final de la comparaison dynamique.

Les conditions aux limites données ci-après permettent de prendre en compte ce phénomène. Elles s'expriment par :

condition initiales :

I(1)=1	ou	I(1)=1	ou	I(1)=2
J(1)=1		J(1)=2		J(1)=1
(1)		(2)		(3)

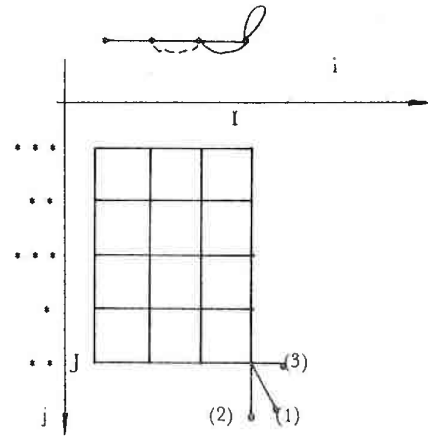


(1) cas normal (2) insertion (3) élision

condition finales

de même :

I(K)=I+1	ou	I(K)=I	ou	I(K)=I+1
J(K)=J+1		J(K)=J+1		J(K)=J
(1)		(2)		(3)



(1) cas normal (2) insertion (3) élision

Les étapes 1 et 5 de l'algorithme se réécrivent alors ainsi :

étape 1 : initialisation

```
Pour i de 1 à I faire
Pour j de 1 à J faire D(i,j)=infini
D(1,1)=0;
D(1,2)=0;
D(2,1)=0;
```

étape 5 : décision

$$D(W) = MIN \begin{cases} D(I+1,J+1) \\ D(I,J+1) \\ D(I+1,J) \end{cases}$$

Un important travail a été effectué sur le relachement des contraintes aux frontières dans les algorithmes de reconnaissance de mots isolés et enchaînés [Di Martino 84].

III.2.3.1 analyse sur une structure arborescente

a) introduction

Comme nous l'avons vu précédemment, la représentation des formes phonétiques est optimisée par la factorisation des mots appartenant à une même classe grammaticale. Nous proposons ici une généralisation de l'algorithme précédent pour analyser une telle structure.

b) notation

Nous nous ramenons à une forme linéaire du graphe phonétique en numérotant chaque noeud de l'arbre par ordre préfixé (cf. figure 3).

Définissons :

GRAPH	G(1,m) G(2,m)____ G(I,m) le graphe phonétique factorisé.
i	le numéro d'un noeud de l'arbre. Il est numéroté dans l'ordre préfixé.
I	la longueur du graphe.
G(i,m)	un segment phonétique de la description.
m	l'indice permettant d'accéder aux différents champs de la description.
	m=1 -> phonème le plus probable.
	m=2 -> phonème pouvant se substituer.
	m=3 -> phonème pouvant s'insérer.
	G (i,m) <0 si le segment peut s'omettre.

T[j,n]	le treillis phonétique.
j	le numéro de segment du treillis.
n	
fils(i)	indique le fils de noeud. Sur la figure 3, fils(2) = 3
	fils(i) = Nil si le noeud i est une feuille.
frère(i)	indique le frère du noeud i. frère(2) = 5
	frère(i) = Nil si le noeud n'a pas de frères.
feuille	l'ensemble des feuilles de l'arbre. Les feuilles correspondent au dernier segment de la forme phonétique de chaque mot factorisé dans l'arbre. Sur l'exemple donné ci-dessous feuille=[4,7,11]
mot	donne la forme orthographique associée à chaque feuille de l'arbre.

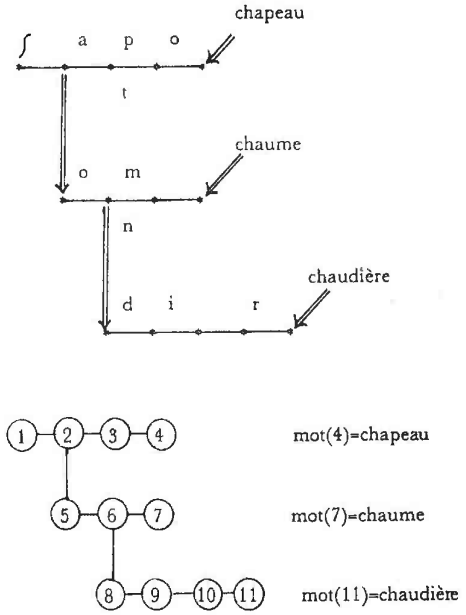


Figure 3 : numérotation de l'arbre phonétique

c) domaine de définition de W

W
N --------> [1..I]x[1..J]u feuillesx[J+1]

De la même manière que précédemment le chemin de recalage peut évoluer dans l'espace bi-dimentionnel: treillis x états du graphe phonétique.

d) algorithme

parcours de la structure arborescente et comparaison dynamique

```
Recherche (i)

  Pour j de 1 à J faire
  début

    si fils(i) <> Nil alors
                        étape 1
                        étape 2
                        recherche (fils(i))

                    sinon étape 3

    si frère(i) <> nil alors
                        étape 4
                        Recherche (frère(i))

  fin
```


étape 1 : calcul des distances locales

Les distances locales sont calculées de la même manière qu'à l'étape 3 de l'algorithme précédent.

étape 2 : calcul des distances cumulées

$$D(fils(i),j) = \text{Min} \begin{cases} D(fils(i),j) \\ D(i,j) + d(i,j,fils(i),j) \end{cases}$$

$$D(i,j+1) = \text{Min} \begin{cases} D(i,j+1) \\ D(i,j) + d(i,j,fils(i),j) \end{cases}$$

$$D(fils(i),(j+1)) = \text{Min} \begin{cases} D(fils(i),j+1) \\ D(i,j) + d(i,j,fils(i),j+1) \end{cases}$$

étape 3 : décision

distance cumulée optimale pour un mot

D (W) = D(i,j) + d(i,j,fils(i),j+1)

comparaison avec les autres mots

si D(W) < D(W) alors
début
D(W) = D(W)
Meilleur-mot = mot(i)
fin

étape 4 : propagation des distances cumulées

pour j de 1 à J faire
D(frère(i),j)=D(i,j)

Cet algorithme présente une nette amélioration par rapport au précédent. En effet, la struture de donnée utilisée amoindrit de façon significative l'espace de recherche. Le nombre d'états de l'espace passe de

(nombre de mots x longueur moyenne d'un mot x longueur du treillis)

à

(nombre d'états de l'arbre phonétique x longueur du treillis)

III.2.4 Application à la reconnaissance de parole continue.

III.2.4.1. Introduction générale

Le problème à résoudre est beaucoup plus complexe que pour la reconnaissance de mots isolés. En effet, à l'indéterminisme introduit par l'identification des mots s'ajoute l'imprécision de la localisation des mots dans le continuum de parole (cf figure 4). Il s'agit en effet de comparer le signal avec l'ensemble des combinaisons possibles des mots du vocabulaire. La combinaison qui réalise la meilleure coincidence est retenue.

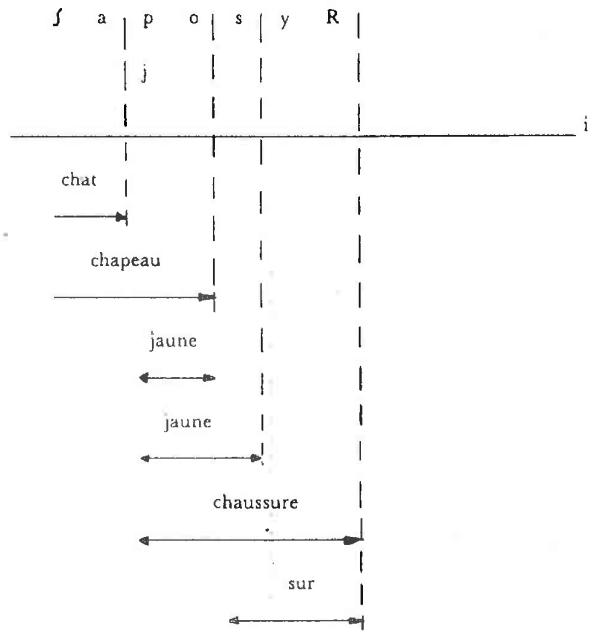


figure 4 : l'indéterminisme de la localisation des mots en parole continue.

Plusieurs équipes ont résolu le problème de la reconnaissance de mots enchaînés par l'approche de la programmation dynamique. Citons Sakoe [Sakoe 79] qui propose un algorithme fonctionnant en deux étapes :

- d'abord il effectue une comparaison dynamique de toutes les formes de référence par balayage de la totalité du signal.

- ensuite il cherche au niveau phrase la meilleure combinaison de mots par le cumul des scores de l'étape précédente.

Un autre algorithme plus performant a été proposé par Myers et Rabiner [Myers 81]. C'est cet algorithme que nous avons adapté et que nous décrirons plus loin. Citons enfin Bridle [Bridle 82] et Nakagawa [Nakagawa 83] qui proposent un algorithme en une passe fonctionnant en temps réel. Dans cet algorithme, la comparaison dynamique s'effectue en progressant dans le signal sans retours arrière. Il évalue pas à pas la meilleure combinaison de mots aboutissant en chaque segment du signal (combinaisons comportant un nombre quelconque de mots).

Le choix d'adopter l'algorithme de Myers et Rabiner est motivé par le fait qu'il est particulièrement adapté à notre problème. En effet, il offre la possibilité de n'effectuer des choix qu'entre des chaînes de même longueur en nombre de mots, sans modification majeure de l'algorithme. Pour ce type de stratégie, l'algorithme de Bridle et Nakagawa demanderait l'adjonction d'une dimension supplémentaire à l'espace de recherche (le nombre de mots des chaînes reconnues). De plus, ce dernier se prête mal à une stratégie d'analyse en faisceaux que nous avons pu facilement intégrer dans l'algorithme de ''Level Building''.

III.2.4.2. Formalisation.

a) Notations.

Soient:

N	= le nombre de formes de référence.
GRAPH(i,m,l)	= le ième segment de la lième forme de référence.
T (j,n)	= le treillis phonétique.
I (l)	= la longueur en segments de la lième référence.
J	= la longueur du treillis.

Le problème est de trouver parmi l'ensemble des combinaisons possibles de mots celle qui correspond le mieux au treillis phonétique. Il s'agit, comme pour le cas des mots isolés, de chercher dans un espace multidimensionnel une fonction de recalage W optimale au sens d'une certaine métrique.

L'ensemble des fonctions de recalage possibles est défini de la façon suivante :

$$W$$
$$N \xrightarrow{\quad} [1,\ldots,I(l)] \times [1,\ldots,J] \times [1,\ldots,N] \cup [I(l)+1,J+1] \times [1,\ldots,N]$$

$$k \xrightarrow{\quad} W(k) = [i(k),j(k),l(k)]$$

La figure 5 donne l'exemple d'une fonction de recalage. On cherche alors à optimiser la fonction D (W) qui s'exprime par:

$$(1)\ D(W) = d([i(k-1),j(k-1),l(k-1)],[i(k),j(k),l(k)])$$

où

K indique la longueur du chemin et

d la pénalisation associée à chaque transition de W.

b) Les contraintes.

Les contraintes à exprimer sont plus compliquées que celles définies dans le cas des mots isolés. Deux types de contraintes s'imposent :

- des contraintes internes
- des contraintes externes

Les premières agissent à l'intérieur d'une même forme de référence. En reprenant le même type de contraintes que précédemment (contraintes symétriques), W(k) s'exprime alors en fonction de W(k-1) par :

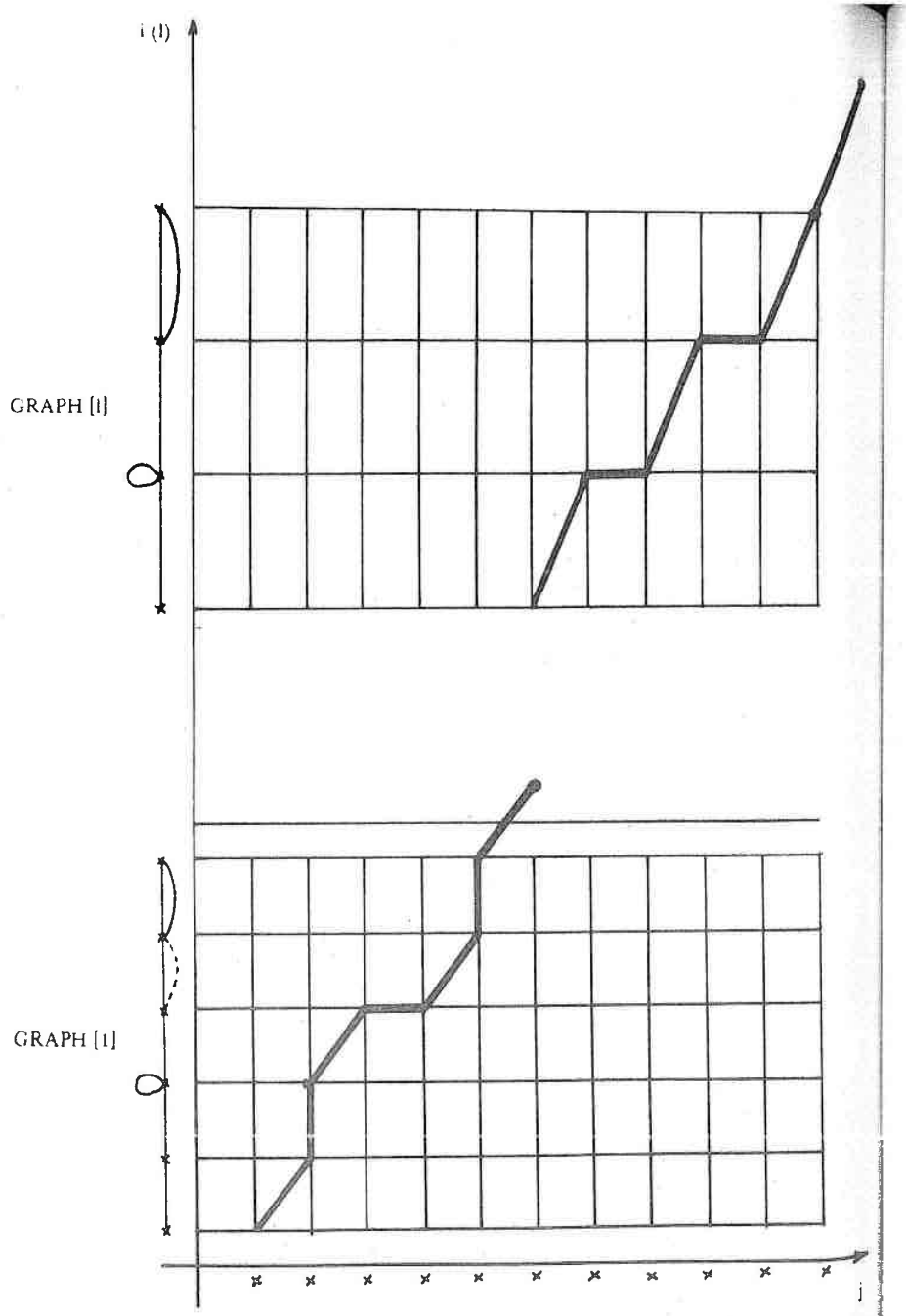
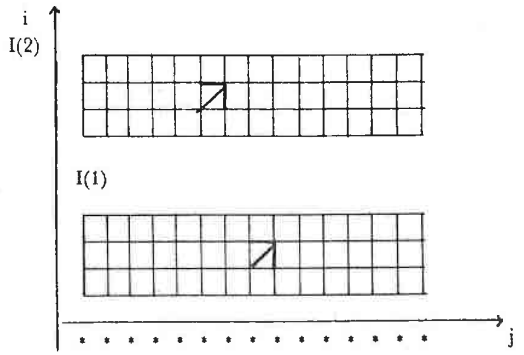


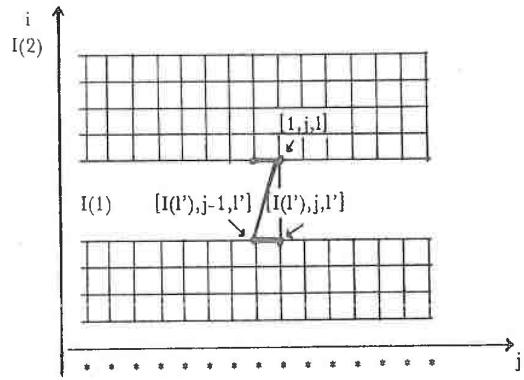
figure 5. Le chemin de recalage en parole continue.

(i(k),j(k),l(k)) = (i(k-1)+[0,1],j(k-1)+[0,1],l(k))
quel que soit i(k)<>1



Les secondes agissent aux frontières entre deux formes de référence :

soit :
(1,j(k+1),l(k+1)) = (I(l),j(k)+(0,1),l(k)) ou
(1,j(k)+1,l(k+1))
l(k) :1--> N



c) Les relations de programmation dynamique.

Dans ces conditions, la relation (1) se reformule par :

$$D(i,j,l) = \min \begin{cases} D(i-1,j,l)+d[(i-1,j,l),(i,j,l)] \\ D(i,j-1,l)+d[(i,j-1,l),(i,j,l)] \\ D(i-1,j-1,l)+d[(i-1,j-1,l),(i,j,l)] \end{cases}$$

si i<>1

$$D(1,j,l) = \min_{l' \in [1..N]} \begin{cases} D(1,i-1,l) \\ D(I(l'),i-1,l') \\ D(I(l'),i,l') \end{cases}$$

et l'optimisation de D(I(l),J,l) est recherchée dans le domaine de variation de l.

III. 2. 4. 2 Algorithme

Parmi les méthodes déjà mises en oeuvre dans la reconnaissance globale, nous avons adapté à notre problème la solution proposée par Myers et Rabiner. La comparaison dynamique est effectuée par la concaténation successive des formes de référence. L'algorithme résultant fonctionne par niveaux. A chaque niveau n de la reconnaissance, l'algorithme évalue la distance cumulée associée à la meilleure chaîne de n mots coincidant avec les i premiers prélèvements de la forme test.

La phrase reconnue est celle qui donne au dernier élément de la forme inconnue la plus petite distance cumulée. Un exemple de ce processus est donné figure 6.

notations

Dist(j,niv)	$\text{MIN} \begin{cases} D(I(l),j,l)+d([I(l),j,l],[I(l)+1,j,l]) \\ D(I(l),j,l)+d([I(l),j,l],[I(l)+1,j-1,l]) \end{cases}$ l 1->N la distance cumulée associée à la coincidence entre les segments 1 et j du treillis et d'une superforme de référence optimale au niveau niv.
Nivmax	le nombre maximum autorisé de niveaux dans l'analyse.
niv	le niveau d'analyse
DEB(i,j,l)	le dernier segment du treillis auquel aboutit la superforme optimale calculée à l'étape précédente. La concaténation de la superforme avec les j premiers segments de la forme de référence l est optimale jusqu'au point [i,j,l].

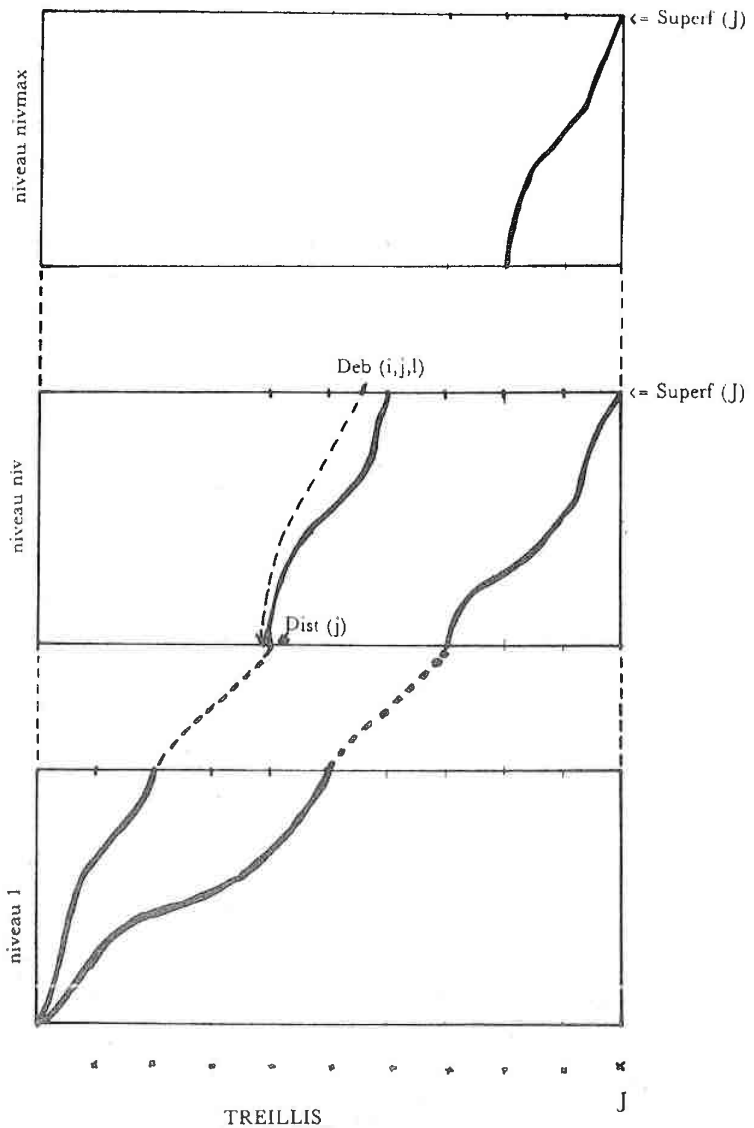


figure 6 . Algorithme de programmation dynamique par niveaux.

SUPERF(j,niv) la superforme calculée à l'étape niv. Le dernier segment de SUPERF a été mis en correspondance jusqu'au segment j du treillis.

algorithme

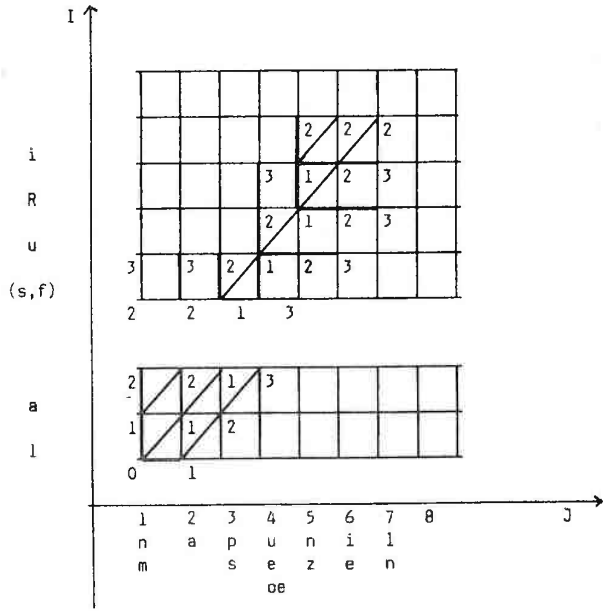
Le processus de minimisation par niveaux est illustré figure 7.

```
étape 0 : initialisation

pour j de 2 à J faire Dist(j,0)=infini;
Dist(1,0)=0;
```

```
étape 1 : reconnaissance d'une phrase

pour niv de 1 à nivmax faire
   étape 2
   étape 3
fin
étape 6
```

Dist[1,1]=2; Dist[2,1]=2; Dist[3,1]=1;
Superf[1,1]= la
Superf[6,2]= la souris

Figure 7 : Recherche dynamique par Level Building

étape 2 : au niveau niv
pour tous les mots

pour l de 1 à N faire
étape 4
ètape 5
fin

étape 4 : initialisation d'un niveau

pour j de 1 à J faire
D(1,j,l)=Dist(j,niv-1)
DEB(1,j,l)=j
fin

étape 5 : évaluation des distances
cumulées et de la fonction deb

pour i de 1 à I(l) faire
pour j de 1 à J faire
distances cumulées

$$D(i,j+1,l) = \text{MIN}\begin{cases} D(i,j+1,l) \\ D(i,j,l)+d([i,j,l],[i,j+1,l]) \end{cases}$$

$$D(i+1,j,l) = \text{MIN}\begin{cases} D(i+1,j,l) \\ D(i,j,l)+d([i,j,l],[i+1,j,l]) \end{cases}$$

$$D(i+1,j+1,l) = \text{MIN}\begin{cases} D(i+1,j+1,l) \\ D(i,j,l)+d([i,j,l],[i+1,j+1,l]) \end{cases}$$

calcul de la fonction DEB

si D(i,j+1,l)=D(i,j,l)+d([i,j,l],[i,j+1,l])alors
DEB[i,j+1,l]=DEB[i,j,l]

si D(i+1,j,l)=D(i,j,l)+d([i,j,l],[i+1,j,l])alors
DEB[i+1,j,l]=DEB[i,j,l]

si D(i+1,j+1,l)=D(i,j,l)+d([i,j,l],[i+1,j+1,l])alors
DEB[i+1,j+1,l]=DEB[i,j,l]

fin

étape 3 : contraintes externes
passage à un niveau supérieur

pour j de 1 à J faire

Dist(j,niv)=MIN D(I(l)+1,J+1,l)
l

l= ARG MIN D(I(l)+1,J+1,l)
l

SUPERF(j,niv)=concat(SUPERF(Deb(I(l)+1,j,l)),MOT(l))
fin

étape 6 : résultat de la reconnaissance

pour niv=1 à nivmax faire
sortir(SUPERF(J+1,niv))

REMARQUES

1) A l'étape 5, toutes les superformes aboutissant au dernier segment du treillis sont retenues. Chacune d'elles étant composée d'un nombre de mots différents, nous résolvons ainsi le problème de la prédominance des mots courts.

2) Comme pour l'algorithme sur les mots isolés, nous tenons compte du flou sur les conditions aux frontières. Les étapes 0 et 5 se réécrivent dans ce cas de la manière suivante:

étape 0 : initialisation

pour j de 2 à J faire Dist(j,0)=infini;
Dist(1,0)=0;
Dist(2,0)=0;

étape 5 : résultat

pour niv=1 à nivmax faire
sortir(SUPERF(J+1,niv))
sortir(SUPERF(J,niv))
fin

CHAPITRE 3

INTRODUCTION DE CONTRAINTES SYNTAXIQUES.

I. Introduction

L'introduction de contraintes syntaxiques est fondamentale dans un système de parole continue utilisant un vocabulaire important. En effet, elles permettent de limiter à chaque étape du processus de comparaison le nombre d'hypothèses de mots à analyser [Boyer 84,85].

Dans les systèmes de reconnaissance globale traitant des langages artificiels, le problème est résolu par la modélisation syntaxique sous forme de graphe. La résolution par programmation dynamique nécessite l'adjonction d'une dimension à l'espace de définition de la fonction de recalage. Celle-ci permet de tenir compte des différents états du graphe syntaxique. La reconnaissance consiste à comparer la représentation du signal vocal à l'ensemble des superformes syntaxiquement correctes obtenues par concaténation d'un nombre quelconque de mots du vocabulaire de l'application. Les minimisations n'interviennent alors plus qu'entre des superformes aboutissant au même état syntaxique.

Une telle approche n'est pas envisageable dans notre application. En effet, la recherche d'une solution optimale serait beaucoup trop coûteuse en raison de la complexité du modèle syntaxique.

Nous proposons, dans ce chapitre, un algorithme dérivé du ''level building'' fonctionnant avec une stratégie d'analyse en faisceau. Ce nouvel algorithme n'est cependant pas optimal au sens de la programmation dynamique.

II. Nécessité d'une stratégie en faisceau

II.1 Introduction

L'algorithme de recherche lexicale par niveaux que nous avons présenté dans le chapitre précédent résoud un des problèmes de la reconnaissance de parole continue : la prédominance des mots courts. Cependant, l'introduction de contraintes syntaxiques dans cet algorithme nous oblige à une adaptation supplémentaire face à deux difficultés :

- le nombre d'états du modèle syntaxique est grand,
- le modèle syntaxique est non déterministe.

II. 2. nombre élevé d'états

Pour chercher une solution optimale, il faut ajouter à la formulation de la programmation dynamique donnée dans le chapitre précédent une dimension supplémentaire. Cette dimension doit définir le contexte syntaxique d'une superforme. Un choix est alors possible entre deux superformes si elles ont même avenir syntaxique. La fonction de recalage recherchée appartient dans ce cas à l'ensemble des fonctions W:

$$N \xrightarrow{W} [1,\ldots,I(l)]x[1,\ldots,J]x[1..N]x[1..C]$$

$$k \dashrightarrow W(k)=[i(k),j(k),l(k),c(k)]$$

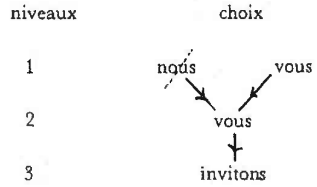
avec c(k) un contexte syntaxique possible.

Cette solution n'est pas envisageable. En effet, en supposant que seul le modèle triclasse agisse, le contexte syntaxique c(k) serait constitué par les catégories grammaticales des deux derniers mots d'une superforme. c(k) représenterait alors l'une des bi-classes possibles du français. Face à cette difficulté, deux adaptations heuristiques sont possibles :

- réduire le contexte syntaxique à la classe grammaticale du dernier mot d'une superforme.
- ne plus tenir compte du contexte syntaxique au niveau des choix. Cette stratégie présente cependant l'inconvénient de rejeter des chaînes susceptibles de constituer une solution. Cette perte peut être cependant compensée par une analyse en faisceau.

II. 3. Indéterminisme du modèle

Les contraintes syntaxiques introduites par les règles grammaticales ont un effet rétroactif et remettent en cause des choix pris à des niveaux précédents. De ce fait, le principe d'optimalité de Bellman ne peut plus s'appliquer entre deux niveaux. En effet, les décisions qui restent alors à prendre ne constituent plus une stratégie optimale compte tenu uniquement de l'état résultant de la première décision. La figure 1 donne un exemple où l'indéterminisme du modèle conduit au rejet de la phrase prononcée.



l'action de la règle :

‹PRPS›‹PPOi›‹VERBE› -› VERBE.NR:=PRPS.NR (accord du verbe avec le pronom personnel qui précéde)

remet en cause le choix effectué au niveau 1 (le mot ''nous'' ayant

Nécessité d' une stratégie
en faisceau

figure 1

Pour résoudre ce problème, trois solutions se présentent :

- se ramener à un modèle entièrement déterministe.
- effectuer des retours arrière.
- utiliser une recherche en faiseaux.

La première solution est assez peu réaliste. En effet, pour se ramener à un modèle déterministe, il faudrait augmenter le nombre de classes grammaticales du modèle. On se retrouve alors face au problème précédent.

La deuxième solution, avec retours arrière, est coûteuse. En effet, les règles grammaticales peuvent remettre en cause un choix sur 3,4 ou même 5 niveaux antérieurs.

C'est la troisième solution que nous avons retenue. Elle consiste à retarder les choix effectués à un niveau donné de l'analyse, lorsque deux superformes de scores voisins sont en concurrence. Pour chaque segment j du treillis, on cherche alors les n meilleures superformes qui coincident entre les indices 1 et j du treillis. Le nombre de superformes retenues est paramétrable. Il est fonction de l'écart avec la superforme de meilleur score.

III. solution mise en oeuvre

Face aux deux difficultés que nous venons d'évoquer, nous avons choisi une stratégie d'analyse en faisceau. Le niveau syntaxique agit alors comme un serveur qui alimente chaque niveau de l'algorithme (''level building'') décrit précedemment. Le modèle syntaxique joue dans ce cas le rôle d'un filtre, limitant le nombre de superformes à tester. Ce processus est schématisé figure 1.

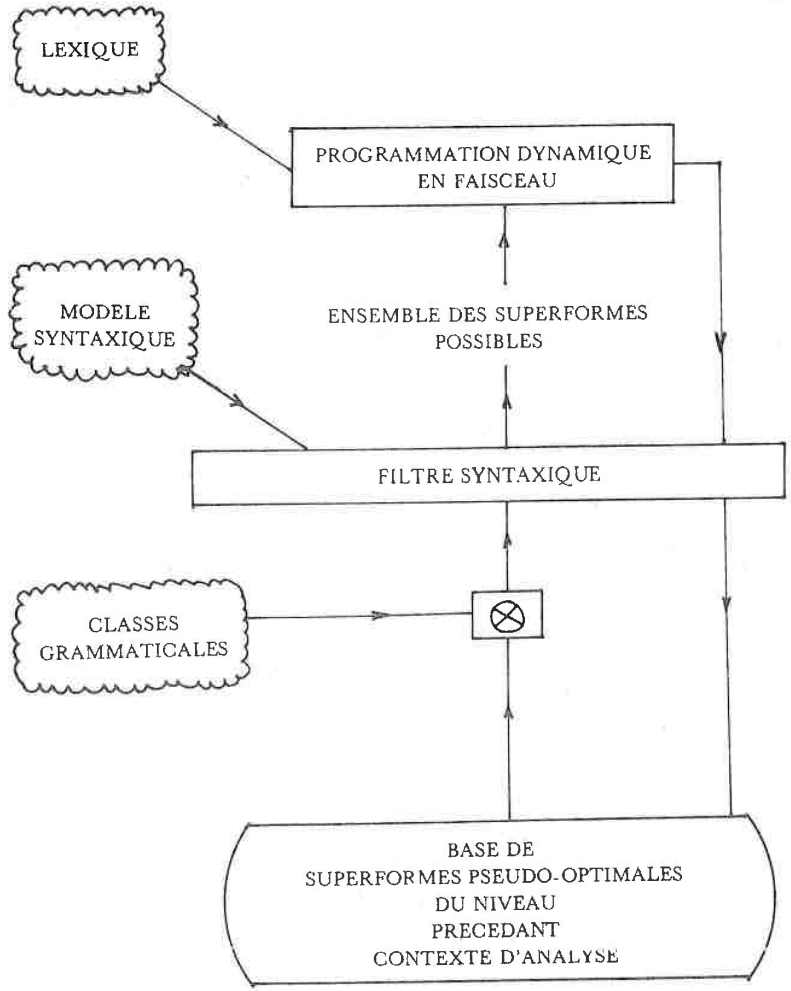


Figure 1. processus de reconnaissance

III. 1. programmation dynamique en faisceau.

A chaque étape de la reconnaissance, on cherche les n superformes aboutissant en chaque segment j du treillis. L'algorithme donné au chapitre 2 (paragraphe III.2.4.2) se réécrit comme suit :

notations

flou-max	la largeur du faisceau.
I(l)	segment d'une forme de référence.
l	une forme de référence.
j	un segment du treillis.
N	le nombre de formes de référence.
Dist(j,niv,flou)	les distances cumulées associées à la coincidence entre les segments 1 et j du treillis et les meilleures superformes de référence au niveau niv.
Nivmax	le nombre maximum de niveaux autorisés dans l'analyse
niv	le niveau d'analyse
DEB(i,j,l)	le dernier segment du treillis auquel aboutit la superforme optimale calculée à l'étape précédente. La concaténation de cette superforme avec les i premiers segments de la forme de référence l est optimale jusqu'au point j du treillis.
SUPERF(j,niv,flou)	les superformes calculées à l'étape niv. Le dernier segment de ces superformes a été mis en correspondance avec le segment j du treillis.

ALGORITHME


```
étape 0 : initialisation

pour flou de 1 à flou-max
début
  pour j de 2 à J faire Dist(j,0,flou)=infini;
  Dist(1,0,flou)=0;
fin
```

```
étape 1 : reconnaissance d'une phrase

pour niv de 1 à nivmax faire
pour flou de 1 à flou-max faire
début
  étape 2
  étape 3
fin
ètape 5
```

étape 2 : au niveau niv
pour tous les mots
sélectionnés par
le filtre syntaxique

pour l de 1 à N faire
étape 4
étape 5
fin

étape 4 : initialisation d'un niveau

pour j de 1 à J faire
si SUPERF(j,niv-1,flou) suivi de MOT(l) est admissible alors
D(1,j,l)=Dist(j,niv-1,flou)
DEB(1,j,l)=j
fin

étape 5 : évaluation des distances
cumulées et de la fonction deb

pour i de 1 à I(l) faire
pour j de 1 à J faire
distances cumulées

$$D(i,j+1,l) = \text{MIN}\begin{cases} D(i,j+1,l) \\ D(i,j,l)+d([i,j,l],[i,j+1,l]) \end{cases}$$

$$D(i+1,j,l) = \text{MIN}\begin{cases} D(i+1,j,l) \\ D(i,j,l)+d([i,j,l],[i+1,j,l]) \end{cases}$$

$$D(i+1,j+1,l) = \text{MIN}\begin{cases} D(i+1,j+1,l) \\ D(i,j,l)+d([i,j,l],[i+1,j+1,l]) \end{cases}$$

calcul de la fonction DEB

si D(i,j+1,l)=D(i,j,l)+d([i,j,l],[i,j+1,l])alors
DEB[i,j+1,l]=DEB[i,j,l]

si D(i+1,j,l)=D(i,j,l)+d([i,j,l],[i+1,j,l])alors
DEB[i+1,j,l]=DEB[i,j,l]

si D(i+1,j+1,l)=D(i,j,l)+d([i,j,l],[i+1,j+1,l]alors
DEB[i+1,j+1,l]=DEB[i,j,l]

fin

étape 3 : contraintes externes
passage à un niveau supérieur

pour j de 1 à J faire
pour flou de 1 à flou-max faire

Dist(j,niv,flou)=MIN[flou, D(I(l)+1,J+1,l)]
l

l= ARG MIN[flou,D(I(l)+1, J+1,l)]
l

si la concaténation est admissible alors
SUPERF(j,niv,flou)=concat(SUPERF(Deb(I(l)+1,j,l)),MOT(l)
fin

où la fonction MIN(n,•) rend la n ième plus petite valeur de •.

étape 5 : résultat de la reconnaissance

pour niv=1 à nivmax faire
pour flou de 1 à flou-max faire
sortir(SUPERF(J+1,niv,flou))

A chaque niveau de l'algorithme, la programmation dynamique s'effectue plusieurs fois sur chaque mot (dimension flou) pour obtenir les n meilleures solutions optimales. Afin d'optimiser les temps de calcul, nous avons utilisé l'heuristique consistant à considérer la fonction de recalage indépendante des conditions initiales. Dans ce cas, seule la fonction de recalage optimale est calculée (au lieu des trois meilleures).

III. 2. La base des superformes

La base des superformes contient l'ensemble des chaînes pseudo-optimales calculées à l'étape précédente. Les informations suivantes sont associées à chaque superforme :

- j, le segment du treillis coincidant avec le dernier segment de la superforme de référence.
- D, la pénalité associée à la fonction de recalage.

A chaque mot de la superforme est associé un pointeur permettant de retrouver les informations syntaxiques de ce mot. Soient:

- V la classe grammaticale qu'il possède dans le contexte où il se trouve.
- M la liste de traits syntaxico-sémantiques qu'il possède.
- voy qui indique s'il commence par une voyelle.
- S qui indique les contraintes phonétiques imposées au prochain mot.

Cette structure est schématisée figure 2.

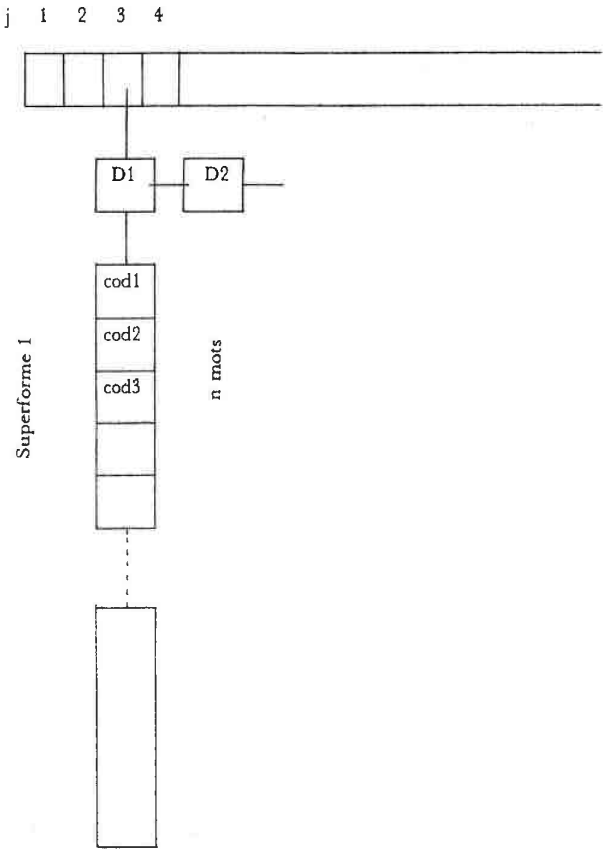


Figure 2 : structure de la base des superformes

III. 3. Le filtrage syntaxique

a) introduction

A chaque niveau n de la reconnaissance, le modèle syntaxique filtre, parmi l'ensemble des concaténations possibles entre chaque mot du lexique et chaque superforme S(j,n-2) celles qui sont admissibles.

Le filtrage agit dans deux sens:

- d'abord, il joue le rôle d'un hypothétiseur de mots. La prédiction concerne la classe et les traits syntaxico-sémantiques du mot à concaténer.

- ensuite, il vérifie la validité des traits syntaxico-sémantiques de chaque mot reconnu, en fonction du contexte syntaxique de la superforme à laquelle il va être rattaché.

b) l'hypothétiseur

- hypothèse sur la classe du prochain mot.

Parmi l'ensemble des superformes du niveau précédent et pour chaque catégorie grammaticale, le filtre sélectionne celles qui peuvent précéder un mot de la classe grammaticale testée.

Algorithme:

j	un segment du treillis.
SuperF(j,d,n,m)	une des superformes coincidant avec le segment j.
d	le score de reconnaissance cumulé de la superforme.
n	indice permétant d'accéder à chaque superforme reconnue de score d.
m	permet d'accéder à chacun des mots de la superforme.
cl	une fonction qui rend la valeur grammaticale d'un mot (dans le contexte où il apparaît).
N(j,d)	le nombre de superformes aboutissant au segment j du treillis avec le score d.
select(j,d,n,classe)	fonction retournant la valeur vrai si la superforme SUPERF(j,d,n) peut être concaténée à un mot de classe.
classe	une des classes grammaticales du modèle syntaxique.
mat	la matrice d'adjacence triclasse.

L'algorithme de sélection s'écrit alors :

```
n: = niveau
POUR chaque classe grammaticale du modèle FAIRE
POUR d appartenant à son domaine de définition FAIRE
POUR n de 1 à N(j,d) FAIRE
  SI    Mat    [cl(SuperF(j,d,n,m-1)),cl(SuperF(j,d,n,m)),classe]
ALORS
        Select (j,d,n,m,classe) = vrai
```

- **hypothèse sur les traits syntaxico-sémantiques**

Les traits syntaxico-sémantiques sont calculés par application des régles grammaticales décrites précédemment.

Chaque règle est préidentifiée avec chacun des triplets du modèle tri-classe à l'initialisation de la reconnaissance. On connait donc pour chaque triplet c1, c2, c3 la liste des règles pouvant agir sur une chaîne dont les trois derniers mots appartiennent aux classes c1, c2 et c3.

Soit

REGLE(c1,c2,c3,x)	une régle activable sur (c1,c2,c3).
NR (c1,c2,c3)	le nombre de règles activables sur (c1,c2,c3).
traits(j,d,n)	la liste de traits syntaxico-sémantiques que devra posséder le mot concaténé à superf(j,d,n).

Le processus de prédiction se schématise alors de la manière suivante:

```
POUR j de 1 à J FAIRE
POUR tout d FAIRE
POUR toute classe FAIRE
POUR n de 1 à N(j,d) FAIRE
SI Select (j,d,n,classe) ALORS interpréter-règle(j,d,n,classe)
```



```
Interpréter-règle

POUR x DE 1 A NR(cl(superf(j,d,n,niveau-2),
                cl(superf(j,d,n,niveau-1),
                classe)                FAIRE
DEBUT
  SI unification(règle) ALORS interpréter-action(Rejet,traits)
  SI rejet ALORS select(j,d,n,classe) = False.
FIN
```

La procédure interpréter-règle prédit pour chaque superforme la liste de traits syntaxico-sémantiques que doit posséder le mot qui lui sera concaténé. La prédiction s'effectue en deux étapes : d'abord, la procédure ''unification'' tente d'unifier la partie identification de chaque règle avec la superforme analysée. En cas de réussite, la partie action de la règle est interprétée pour le calcul des traits. Si la partie action conduit au rejet de la superforme, celle-ci est rayée de la liste select.

c) Filtrage après la reconnaissance.

Le calcul de la nouvelle base s'effectue en deux étapes: concaténation des mots reconnus avec les superformes de la base précédente puis élimination des interprétations donnant des scores trop mauvais.

La concaténation s'effectue classe par classe de la manière suivante :

1) tout mot reconnu est concaténé avec les superformes sélectionnées par l'hypothétiseur si les traits syntaxico-sémantiques sont compatibles dans le nouveau contexte (superforme + mot).

2) les superformes dont le score est insuffisant sont éliminées: le critère de rejet est fonction du score obtenu par la meilleur superforme.

IV. Conclusion

La technique de programmation dynamique employée ici s'apparente aux algorithmes de recherche en faisceau. Elle présente par rapport à ces derniers l'avantage d'optimiser la recherche. Cependant, l'algorithme de type ''Level Building'' que nous avons développé présente l'inconvénient d'être assez peu orienté temps réel. En effet, il nécessite d'effectuer des retours arrière sur le treillis à chaque changement de niveau. Une perspective intéressante pour pallier cet inconvénient serait d'adapter à notre problème l'algorithme de Bridle et Nakagawa.

PARTIE D

LE MODE COMMANDE-PAROLE

DE L'EDITEUR

1. Introduction

2. Les sources de connaissance

3. Technique de reconnaissance employée

CHAPITRE 1

INTRODUCTION

Comme nous l'avons vu précédement, l'éditeur de texte fonctionne selon deux modes d'utilisation : le mode insertion pour l'édition de textes et le mode commande pour la correction ou la manipulation des textes édités. C'est ce second mode que nous présentons dans cette partie.

Là encore, l'utilisation d'une entrée vocale apporte, en complément des systèmes de saisie classiques de type clavier ou souris, une certaine souplesse dans la manipulation des documents édités. Elle permet en effet l'appel de commandes partielles ou complètes et de menus en interaction avec les autres modes de désignation.

Les techniques de reconnaissance à mettre en oeuvre dans ce type d'application sont plus simples que celles présentées pour le mode parole èdition. En effet, les phrases ou les mots à reconnaitre font partie d'un domaine restreint du langage. La simplification apparait à deux niveaux du traitement :

- au niveau lexical.
- au niveau syntaxique.

D'abord au niveau lexical, la reconnaissance de mots est plus efficace. En effet, le nombre de mots susceptibles d'apparaitre dans une commande est réduit (quelques dizaines de mots). De plus, ceux-ci peuvent être choisis en évitant l'emploi de mots proches phonétiquement, de façon à éviter les ambiguités lors de la reconnaissance.

Ensuite au niveau syntaxique, la complexité du langage traité est moindre, le langage de l'application étant artificiel. De ce fait la description des contraintes syntaxiques se réduit à l'écriture d'une grammaire hors contexte.

La résolution que nous proposons est guidée par une analyse syntaxique descendante. Les techniques de programmation dynamique s'avèrent ici particulierement efficaces et permettent d'atteindre une solution optimale avec des temps de calcul raisonnables. Nous proposons un algorithme entièrement paramétrable par la grammaire décrivant le langage. Cette approche est similaire à celle adoptée pour la reconnaissance globale de la parole.

CHAPITRE 2

LES SOURCES DE CONNAISSANCES

I. Introduction

Toutes les sources de connaissances nécessaires au fonctionnement de la reconnaissance sont des paramètres du système, de façon à lui garder un caractère de généralité suffisant pour traiter d'autres applications. Le système que nous présentons ici pourrait tout aussi bien servir au pilotage de robots ou à la commande numérique de machines-outils.

Les données du sytème se composent de trois types de description :

- la définition syntaxique du langage de l'application,
- la définition sémantique associée au langage,
- le lexique phonétique.

II. La définition syntaxique du langage

Le langage de l'application étant simple, celui-ci est décrit par une grammaire hors-contexte. Celle-ci est décrite sous une forme proche de la notation de Backus-Naur.

La structure générale d'une règle s'exprime dans le symbolisme que nous avons utilisé jusqu'à présent par :

```
REGLE::= SYMBOLE = EXPRESSION ;
EXPRESSION::= TERME {/ TERME}
TERME::= FACTEUR {FACTEUR}
FACTEUR::= TERMINAL / NON-TERMINAL / [ TERME ] /
           (EXPRESSION)
TERMINAL::= {a/.../z}
NON-TERMINAL::={A/.../Z}
```

où

- les crochets "[" et "]" encadrent une partie facultative d'une production

- les parenthèses "(" et ")" délimitent un élément commun d'une production à plusieurs alternatives. L'introduction des parenthèses permet d'optimiser la grammaire et, par conséquent, l'analyse syntaxique du système de reconnaissance. En effet, la factorisation qui en résulte réduit de façon significative le nombre d'étapes nécessaires à la reconnaissance.

- les noms terminaux de la grammaire font référence à l'une des classes du lexique phonétique.

Nous donnons figure 1 un extrait de cette grammaire.


```
‹COMMANDE›::=‹RECHERCHE›/‹EFFACEMENT›/‹ECHANGE›;
‹RECHERCHE›::= chercher [‹PARTITION-TEXTE›]‹MOT›;
‹EFFACER›::=‹effacer›[‹PARTITION-TEXTE›];
‹ECHANGE›::=‹remplacer›[‹PARTITION-TEXTE›]‹MOT›
            ‹par›‹MOT›;
‹PARTITION-TEXTE›::=‹a-la-ligne›‹numero›/
                    ‹de-la-ligne›‹numero›
                    ‹a-la-ligne›‹numero›/
                    ‹les-lignes›‹numero›‹a›‹numero›;
‹MOT›::=‹le-mot›‹WORD›;
```

où :
- numéro renvoie à la classe des chiffres.
- WORD renvoie au lexique général.

FIGURE 1. extrait de la grammaire du langage de requête

La définition du langage est facilitée par un logiciel d'aide à la mise au point. Le processus de création de la grammaire se compose de quatre étapes :

- édition sous un éditeur pleine page des règles de réécriture.
- vérification syntaxique de la grammaire et génération d'un listing des erreurs.
- grammaticales désignées par les terminaux de la grammaire.
- compilation des règles vers la représentation interne.

Ce mode de définition présente l'avantage d'offrir la possibilité de modifier à tout moment la grammaire avec le maximum de contrôles.

La représentation interne de la grammaire est assez proche de sa forme initiale. En effet, chaque règle est ramenée à une liste d'éléments de trois types :

- des caractères de contrôle : ce sont les méta-symboles de la grammaire.

- des non-terminaux.
- des terminaux.

Chacun de ces éléments est codé sous forme numérique. Les non-terminaux renvoient aux numéros de la production qu'ils désignent.

III. Définition des actions associées au langage de commande

III. I. Introduction

Le langage de commande dont nous venons de donner la structure permet la dictée des requêtes d'édition en langage quasi-naturel. Citons par exemple la requête :

effacer les lignes 1 à 10

La structure des énoncés n'étant pas directement interprétable par l'éditeur, il fallait définir une interface langage naturel afin de construire les commandes à partir des phrases reconnues. En reprenant la requête ci-dessus, il faut alors se ramener à :

DEL 1-10

Plutôt que de figer par programme le passage d'un langage à l'autre, nous avons préféré définir de façon déclarative le mécanisme de traduction. Ainsi le langage de requête pourra évoluer en fonction des besoins de chaque utilisateur. La méthode utilisée est proche des A.T.N et R.T.N de Woods [Woods 70] et des R.N.P du système MYRTILLE II [Pierrel 82].

III. 2. Les A.T.N et R.T.N

Les A.T.N sont apparus pour simplifier l'implantation machine des grammaires transformationnelles. L'idée de base est de définir ces grammaires par un automate d'états finis dont les arcs sont étiquetés par des symboles et des actions. Les symboles représentent soit des mots, soit des classes complètes de mots appartenant au langage décrit et conditionnent les transitions entre deux états. Les actions associées à chaque arc du graphe sont activées chaque fois que la transition correspondante est validée. Ces actions ont pour objet la prise en compte de phénomènes contextuels. Elles permettent également la construction des structures syntaxiques des phrases analysées.

De plus, les R.T.N introduisent par rapport aux A.T.N la notion de récursivité en ajoutant la possibilité d'étiqueter un arc par un symbole

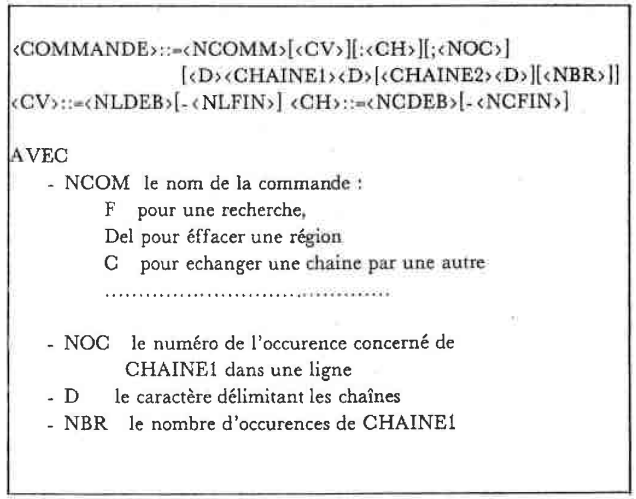
référençant un autre automate.

III. 3. Mise en oeuvre dans la traductition de requêtes

Le mécanisme que nous venons de décrire est particuliérement adapté au problème de la traduction. La résolution s'effectue par association d'un automate d'états finis à chaque règle B.N.F de la grammaire du langage de requête. Les actions du R.T.N ainsi construites auront pour but la traduction du langage de requêtes en langage de commande.

a) le langage de commande

Nous avons repris, à quelques simplifications près, le langage de commande de l'éditeur de texte implanté sur le système de développement Motorola Exormacs. La structure d'une commande est donnée figure 2.



NLDEB,NLFIN,NCDEB,NCFIN définissent la région du texte sur laquelle agit la commande (numéros de ligne, numéros de colonne des 4 extrémités de la zone)

Figure 2 : les commandes de l'éditeur

b) la traduction

Nous présentons ici un extrait du R.T.N auquel nous sommes parvenu pour construire une commande éditeur à partir d'une requête en langage naturel. L'objectif des actions du R.N.P va être de remplir chacun des champs de la commande. Ces actions sont de deux types :

- affectation d'une constante à un champ.
- affectation d'une variable à un champ.

Les premières permettent d'affecter directement une valeur à un champ en fonction du contexte d'analyse. Par exemple, on affecte la valeur ''F'' à ''NCOM'' lorsque la requête est une recherche de sous-chaîne dans le texte.

Les secondes ont pour objet l'extraction de la requête d'une variable. Par exemple, sur la requête :

effacer les lignes 1 à 10

les valeurs 1 et 10 sont extraites du message et affectées aux registres NLDEB et NLFIN.

c) un exemple d'application

Nous donnons ici le traitement complet pour la traduction des requêtes utilisées pour la recherche d'une chaîne dans le texte (cf figures 3, 4 et 5)

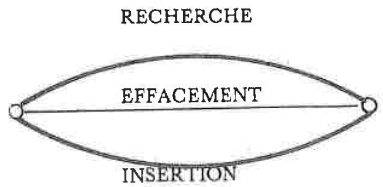


figure 3: automate principal

L'automate principal indique qu'une commande est soit une recherche de sous chaîne, soit un effacement d'une région du texte, soit

une insertion d'un élément.

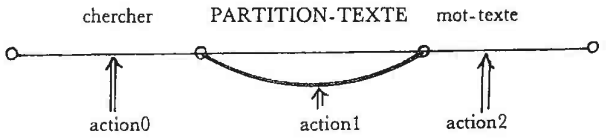


Figure 4 : automate RECHERCHE

Une recherche est déclenchée par le mot clé ''chercher''. PARTITION-TEXTE indique la région du texte dans laquelle la recherche doit s'effectuer, et mot-texte désigne la sous-chaîne à rechercher.

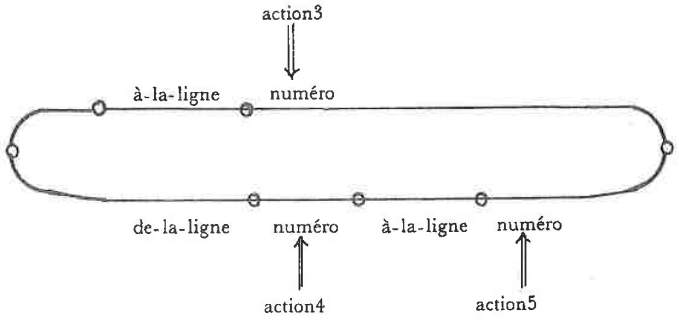


Figure 5: automate PARTITION-TEXTE

Une partition est constituée soit d'un numéro de ligne, soit de deux numéros de ligne délimitant la région d'application d'une commande, où:

action0 la commande analysée est une recherche. On affecte le registre NCOM à F.

action1 dans le cas où la région de la commande n'est pas spécifiée la commande s'applique à tout le texte. On affecte alors au

registre NLDEB la valeur 1 et au registre NLFIN la longueur totale du texte.

action2 mot-texte désigne la chaîne recherchée. Elle est affectée à CHAINE2.

action3,4,5 numéro définit la région concernée par la recherche. Il remplit les champs NLDEB et NLFIN.

d) Définition déclarative du R.T.N

Les actions du R.T.N étant très simples, il est facile d'écrire un interpréteur permettant la définition déclarative du réseau. Chaque sous-automate du réseau est alors décrit par une règle de grammaire de type B.N.F . Les actions s'insèrent dans cette optique entre chacun des éléments formant une règle.

Le R.T.N cité en exemple ci-dessus se réécrit alors dans ce formalisme en :


```
‹COMMANDE›=‹RECHERCHE›/
                ‹EFFACER›/
                ‹ECHANGE›;

‹RECHERCHE›=‹chercher›
                    # NCOM:=F #
                (‹PARTITION-TEXTE›/
                ‹›
                    # NLDEB:=1;
                      NLFIN:=FIN-TEXTE  #)
                ‹MOT-TEXTE›
                    # CHAINE1:=  MOT-TEXTE#;

‹PARTITION-TEXTE›=‹a-la-ligne›‹numero›
                            # NLDEB:=  NUMERO;
                              NLFIN:=  NUMERO # /
                        ‹de-la-ligne›‹numero›
                            # NLDEB:=  NUMERO #
                        ‹a-la-ligne›‹numero›
                            # NLFIN:=  NUMERO # /
                        ‹les-lignes›‹numero›
                            # NLDEB:=  NUMERO #
                        ‹a›‹numero›
                            # NLFIN:=  NUMERO #;
```

III. 4. Le lexique

Dans une requête, deux types de mots peuvent intervenir :

- les mots du langage de commande comme "chercher", " effacer", "remplacer", "à la ligne ".
- les mots manipulés par une requête. Parmi ceux-ci, on trouve des mots déjà édités comme, par exemple, le mot "monsieur" dans la phrase :

chercher dans la page courante le mot monsieur.

Dans ce cas la reconnaissance s'effectue sur un sous lexique constitué par les mots du texte édité. Des mots appartenant à un lexique plus vaste du français peuvent également être désignés par une commande. Citons par

exemple le mot ''madame '' dans la requête :

remplacer à la ligne courante le mot monsieur par le mot madame

Selon les cas, le module de reconnaissance fera appel soit au lexique du langage de commande, soit au lexique général que nous avons décrit précédemment. Dans ce dernier cas, lorsque la requête fait appel à un mot déjà édité, on aura recours à un sous-ensemble du lexique.

La structure du lexique du langage de commande est assez proche de celle que nous avons définie précédement pour le mode parole-édition de l'éditeur. Elle est cependant simplifiée par la suppression des traits syntaxico-sémantiques qui étaient alors associés à chaque unité lexicale.

Nous donnons figure 6 un extrait de ce lexique dont les classes correspondent aux noms terminaux de la grammaire.

```
CHERCHER
  CHERCHER: ch/e/ch,,R/e
FINCLASSE

EFFACER
  EFFACER: e/f/a/s/e
FINCLASSE

REMPLACER
  REMPLACER : R/an/p/L/a/s/e
  SUBSTITUER: s/y/b/s/t/i/t/u/e
FINCLASSE

PAR
  PAR : p/a/R
FINCLASSE


FINLEXIQUE
```

Figure 6 : le lexique du langage de commande

CHAPITRE 3

TECHNIQUES DE RECONNAISSANCE

EMPLOYEES

I. Introduction

La méthode de reconnaissance que nous avons adoptée est réalisée par une analyse syntaxique descendante travaillant de la gauche vers la droite. L'analyseur syntaxique permet de restreindre, en fonction de la partie de la phrase déjà reconnue, le nombre de mots à reconnaitre à chaque étape de l'analyse.

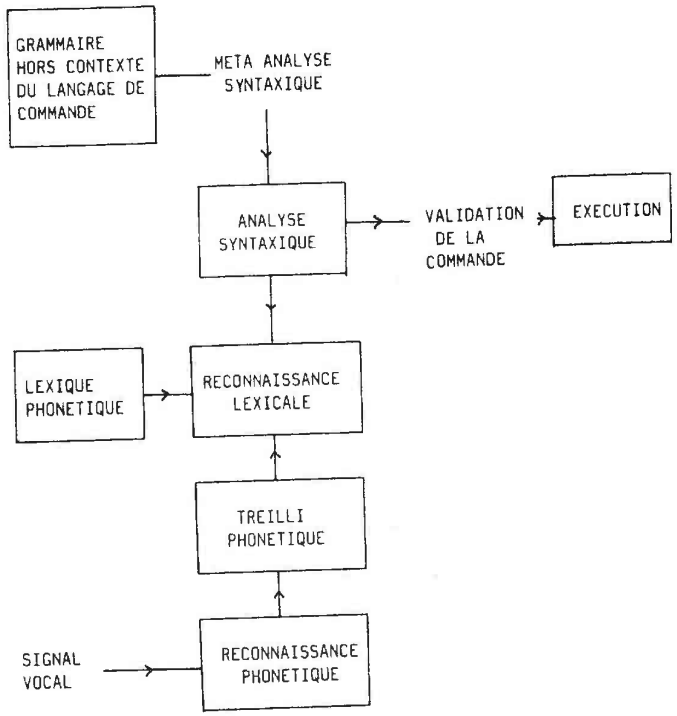
Nous proposons dans ce chapitre, après une présentation générale du système de reconnaissance, un analyseur syntaxique entièrement paramétré par la grammaire du langage. Nous verrons que l'indéterminisme de l'analyse dû aux ambiguités apparaissant dans la localisation des mots est résolu par programmation dynamique, ce qui permet l'obtention d'une solution optimale.

II- Architecture du système

L'objectif de ce sytème est la reconnaissance puis l'exécution des requêtes orales de l'utilisateur. La figure 1 donne le schéma général de ce processus. Le traitement des demandes vocales s'effectue en cinq modules principaux :

- un décodeur acoustico-phonétique qui fournit un treillis de phonèmes.
- un analyseur syntaxique descendant.
- un analyseur lexical asservi par l'analyse syntaxique.

- une procédure rudimentaire de dialogue qui permet de répondre aux impératifs de sécurité nécessaires à la bonne marche du système. Son objectif est la validation des phrases reconnues ou éventuellement la résolution d'ambiguités non levées dans la phase de reconnaissance.
- un interpréteur pour la traduction des commandes reconnues puis validées.



Principe de la reconnaissance d'une commande

Figure 1

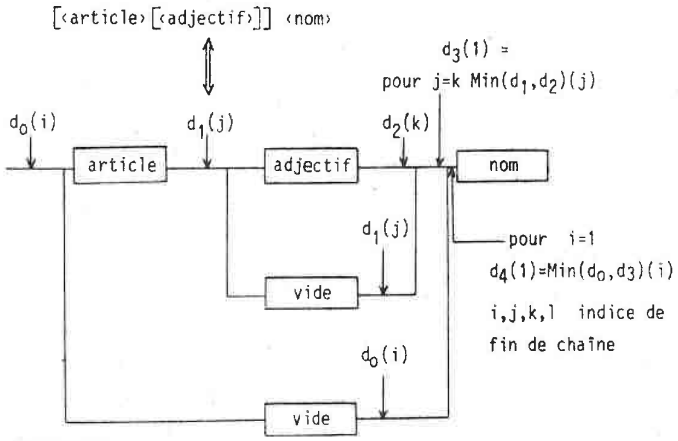
III. Programmation dynamique et syntaxe

III. 1. Introduction

La principale difficulté rencontrée lors de la mise en oeuvre d'un analyseur syntaxique provient de l'indéterminisme de la localisation des terminaux de la grammaire dans le continuum vocal. Ce n'est plus l'absence ou la présence d'un terminal mais la plausibilité de présence de celui-ci qui est détectée.

Une méthode pour résoudre cet indéterminisme est d'évaluer parmi l'ensemble des chemins syntaxiques générés par la grammaire celui qui optimise le critère de comparaison avec le treillis phonétique. Les distances locales associées à chaque pas du cheminement sont calculées par l'analyseur lexical. Celui-ci effectue une comparaison dynamique des mots désignés par les terminaux de la grammaire avec le treillis phonétique.

La grammaire étant hors contexte, le principe d'optimalité de Bellman peut s'appliquer à chaque étape de l'analyse, et on peut choisir des minima locaux sans affecter la globalité du résultat (cf figure 2). Nous désignons sur la figure 2 les différentes étapes du processus de programmation dynamique. Les distances d0, d1 et d2 indiquent pour chaque indice i du treillis phonétique le taux de similitude de la meilleure superforme mise en correspondance jusqu'au segment i. A un noeud du graphe, lorsque plusieurs superformes aboutissant au même point du treillis phonétique ont été reconnues, seule la meilleure est conservée. En effet, l'avenir syntaxique de ces différentes superformes étant le même, elles n'ont aucune possibilité de rattraper la meilleure d'entre elles.



Remarque :

"article" ou "adjectif" ou "nom" peuvent être eux-mêmes des sous-graphes.

(un élément entre crochets [ ] peut être omis).

Figure 2 : cheminement par programmation dynamique dans un graphe syntaxique

III. 2. Description formelle

III. 2. 1. Sur un automate d'états finis.

Il est simple de passer d'une grammaire à un automate d'états finis. Dans ce cas, on peut directement adapter les algorithmes de programmation dynamique traitant de la parole globale [Boyer 84].

Rappelons qu'un automate d'états finis est constitué par :

- un ensemble fini d'états S
- un alphabet fini A
- un ensemble de transitions T, définis dans SxAxS

si t=(x,a,y) x $\in$S, Y $\in$S, a $\in$ A

on dit que t va de l'état x à l'état y en produisant le symbole a . Ici les éléments de A appartiennent à l'ensemble des classes grammaticales du langage.

Il s'agit alors de trouver un chemin optimal continu t(1)...t(i)...t(n) dans l'ensemble des transitions de l'automate d'états finis. Le critère de continuité s'exprime par :

quel que soit i

si ti=(xi,ai,yi)

et ti+1=(xi+1,ai+1,yi+1) alors yi=xi+1

la distance D se calcule alors par

$$D(ti) = \sum_{t(1)}^{t(n)} d(xi,ai,yi)$$

d étant la distance élémentaire associée à chaque transition. Elle correspond à la valeur de la meilleure comparaison dynamique entre chaque mot de la classe ai et la portion analysée du treillis.

On cherche

$$D(\hat{T})=MIN\ D(t(1...n))$$

Formulation en termes de programmation dynamique.

Le problème est de trouver parmi l'ensemble des combinaisons de mots possibles celles qui interprétent le mieux le treillis phonétique. Pour celà, on cherche la fonction de recalage W optimale par rapport au critère de comparaison. En reprenant les notations utilisées dans la partie précédente, l'ensemble des fonctions de recalage possible est défini dans l'espace de comparaison suivant:

$$n \xrightarrow{W} [1,...,1(l)][1,...,j(l)][1,...,N][1,s]u\ [i(l)+1,j+1] \times [1,...,N] \times [1...S]$$

$$k \longrightarrow W(k) = [i(k), j(k), l(k), s(k)]$$

la dimension supplémentaire [1...s] permet de tenir compte des différents états que peut prendre une forme de référence au cours du processus de comparaison. Les minimisations n'interviennent alors qu'entre des superformes aboutissant au même état et au même point du treillis phonétique.

les contraintes.

a) les contraintes internes.

les contraintes internes sont directement généralisées à partir de celles que nous avons définies précédemment (cf partie C chapitre 2) en ajoutant la dimension supplémentaire, soit :

$$(i(k),j(k),l(k),s(k))=(i(k-1)+[o,1],j[k-1]+[0,1],l(k),s(k))$$

b) les contraintes externes.

Les nouvelles contraintes externes doivent tenir compte des transitions définies par l'automate d'états finis : en effet, on ne peut passer d'un état s(k) à un état s(k+1) si la transition (s(k),l,s(k+1)) n'existe pas, ce qui s'exprime par:

$$(1,j(k+1),l(k+1),s(k+1)) = (I(l),j(k)+(0,1),l(k),s(k)) \text{ ou } (1,j(k)+1,l(k+1),s(k+1))$$

si s(k),l(k),s(k+1) est une transition de l'automate.

Formulation.

En intégrant dans la formulation les diverses contraintes, D s'écrit :

$$D(i,j,l,s) = \min \begin{cases} D(i-1,j,l,s)+d[(i-1,j,l,s),(i,j,l,s)] \\ D(i,j-1,l,s)+d[(i,j-1,l,s),(i,j,l,s)] \\ D(i-1,j-1,l,s)+d[(i-1,j-1,l,s),(i,j,l,s)] \end{cases}$$

si i<>1

$$D(1,j,l,s) = \min \begin{cases} D(1,i-1,l,s) \\ D(I(l'),i-1,l',s') \\ D(I(l'),i,l',s') \end{cases}$$

pour tout l' tel que :
(s,l',s) appartient à l'ensemble des transitions de l'automate.

III. 2. 2. Sur une grammaire hors-contexte.

a) introduction

Nous avons généralisé la formulation précédente au cas où la syntaxe est décrite non plus par un automate d'états finis, mais par une grammaire hors contexte. L'algorithme que nous proposons est paramétré directement par la forme linéaire de la grammaire dans la notation de Backus-Naur.

b) principe général.

La méthode consiste à comparer des partitions du treillis phonétique à des superformes de référence (suites de mots). Ces dernières sont construites par concaténation successive des formes désignées par les terminaux, en parcourant de gauche à droite chaque règle de grammaire. Les minimisations, en accord avec le principe de la programmation dynamique, interviennent en cours d'analyse entre des superformes générées par les termes d'une même production, au sens du méta-langage (voir la métagrammaire donnée chapitre 2 paragraphe II). La phrase reconnue est parmi l'ensemble des productions possibles d'une règle celle qui donne la meilleure mise en correspondance avec le treillis. Ce processus est illustré figure 3.

c) formalisation.

Notations.

REGLE	REGLE (1) REGLE (2) --------REGLE (n) une des règles de grammaire.
REGLE (K)	un des trois types possibles de symboles composant une règle: - soit un nom terminal désignant une classe de mot du lexique. - soit un mot terminal renvoyant à une autre règle. - soit un caractère de contrôle: ( ) [ ] /.
début, fin	indique le début et la fin d'une règle. Le paramètre de ces deux fonctions est un non terminal ou l'axiome de la grammaire.
expression(k1,k2,i1,i2)	la superforme construite à partir d'une expression : REGLE (k1)-REGLE (k2) mise en correspondance entre les segments i1 et i2 du treillis phonétique.

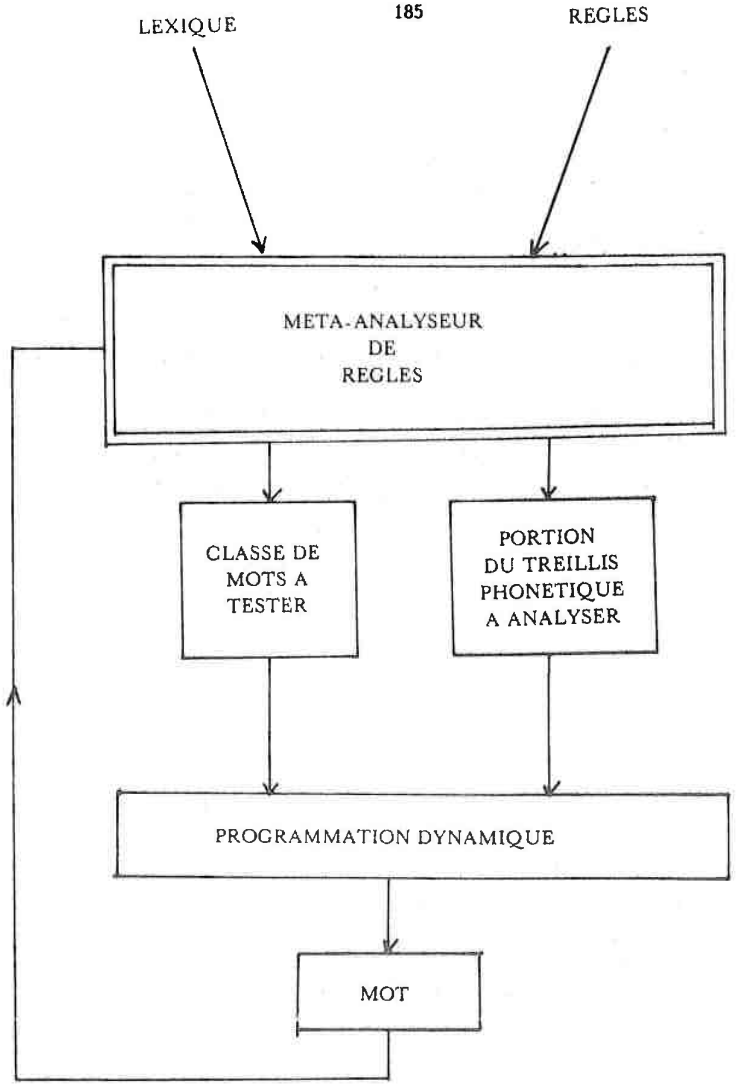


figure 3. processus de comparaison.

TERME(k1,k2,i1,i2) la superforme construite à partir d'un terme.

FACTEUR(k1,k2,i1,i2) la superforme construite à partir d'un facteur.

Pour alléger les notations, nous confondrons la superforme avec la distance qui lui est associée.

La figure 4 illustre ces notations.

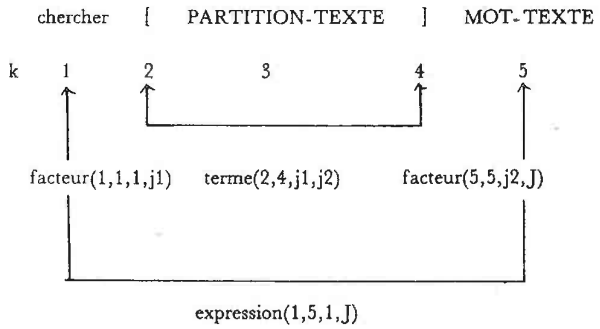


Figure 4. décomposition d'une règle

La règle donnée figure 4 se compose de facteurs qui se regroupent en termes puis en expression. Les deux premiers indices localisent les élément constitutifs de la règle. Les deux indices suivants indiquent la portion du treillis qui a été mise en correspondance avec la superforme engendrée par les éléments de la règle désignés par les deux premiers indices.

d) Relations de programmation dynamique

La formulation est calquée sur la définition méta-syntaxique d'une règle de grammaire (cf chapitre 2 paragraphe II). C'est le processus d'analyse d'une règle qui guide la reconnaissance. Par exemple la méta-règle :

Expression ::= Terme | Terme / Expression

exprime le fait qu'une règle peut se scinder en termes séparés par le symbole ''/''. Les termes permettent chacun d'engendrer des productions dont les avenirs syntaxiques sont identiques. De ce fait une minimisation au sens de la programmation dynamique peut avoir lieu entre ces différentes productions ce qui est exprimé par (1) b.

(1) expression (k1,k2,i1,i2) =

a) Term (k1 k2,i1,i2)

b) |_|_|_| / |_|_|_|
k'

si Règle(k')= /
Min expression (k1,k'-1,i1,i')
i' k' Term (k',k2,i',i2)

(2) term (k1,k2,i1,i2) =

a) Facteur (k1,k2,i1,i2)

b) Term (k1,k',i1,i') + Facteur (k',k2,i',i2)

c) |_|_| [|_|_|_|]
k' k2

Min Term (k1,k'-1,i1,i')
Term(k'+1,k2-1,i',i2)

(3) Facteur (k1, k2, i1,i2) =

a) terminal (k1,k2,i1,i2)

b) NON TERMINAL (K1,k2,i1,i2)

c) |_|_| (|_|_|_|)
k1 k2

Expression (k1+1,k2-1,i1,i2)

(4) Terminal (k1,k2,i1,i2) = Min D(I(l),i2,l)
l appartient à classe (k1,k2)

où :

$$D(i,j,l) = \min \begin{cases} D(i-1,j,l)+d[(i-1,j,l),(i,j,l)] \\ D(i,j-1,l)+d[(i,j-1,l),(i,j,l)] \\ D(i-1,j-1,l)+d[(i-1,j-1,l),(i,j,l) \end{cases}$$

(5) Non- terminal (k1,k2,i1,12) =

a) Expression(début(k1,k2),fin(k1,k2),i1,i2)

Algorithme.

Notations.

i	segment d'une forme de référence.
j	indice courant du treillis.
l	mot courant.
k	indice courant de règle.
DEB(i,j,l)	le dernier segment du treillis auquel aboutit la superforme optimale calculée à l'étape précédente. La concaténation de la superforme avec les j premiers segments de la forme de référence l est optimale jusqu'au point [i,j,l].
SUPERF(j)	la superforme aboutissant en j.
Dist(j)	la distance cumulée optimale d'une superforme aboutissant au point j.

algorithme

```
    initialisation et
    lancement du processus
    de reconnaissance

pour j de 1 à J faire
  dist(j)=infini
  superf(j)=nil
fin
dist(1)=0
REGLE(axiome,dist,superf)
```

```
REGLE(production,dist,superf)
  expression(début(production),fin(production),dist,superf
fin
```

```
Expression(k,dist,superf)
début
  Term(k,dist,superf)
  tant-que Règle(k)=/ faire term(k,dist,superf)
fin
```



```
Term(k,dist,superf)
début
  facteur(k,dist,superf)
  tant-que règle(k)<>['/','[',']','(',')'] faire facteur(k,dist,superf)
fin
```

```
facteur(k,dist,superf)
début
  cas règle(k)
      - terminal      : évaluer(dist,superf)
      - non terminal : REGLE(règle(k),dist,superf)
      - '('           : expression(k+1); k:=k+1;
      - '['           : term(k+1); k:=k+1;
  k:=k+1;
fin
```

```
évaluer(dist,superf)
début
  pour tout l appartenant à la classe règle(k) faire
  début
    initialisation
    distance-cumulée
  fin
  résultat-reconnaissance
fin
```

```
initialisation
début
  pour j de 1 à J faire
    D(1,j,l):=dist(j)
    deb(1,j,l):=j
  fin
fin
```



```
distance-cumulée

   pour i de 1 à I(l) faire
   pour j de 1 à  J   faire
            distances cumulées

      D(i,j+1,l)=   MIN {D(i,j+1,l)
                        {D(i,j,l)+d([i,j,l],[i,j+1,l])

      D(i+1,j,l)=   MIN {D(i+1,j,l)
                        { D(i,j,l)+d([i,j,l],[i+1,j,l])

      D(i+1,j+1,l)= MIN {D(i+1,j+1,l)
                        { D(i,j,l)+d([i,j,l],[i+1,j+1,l])

            calcul de la fonction DEB

      si D(i,j+1,l)=D(i,j,l)+d([i,j,l],[i,j+1,l])alors
           DEB[i,j+1,l]=DEB[i,j,l]

      si D(i+1,j,l)=D(i,j,l)+d([i,j,l],[i+1,j,l])alors
           DEB[i+1,j,l]=DEB[i,j,l]

      si D(i+1,j+1,l)=D(i,j,l)+d([i,j,l],[i+1,j+1,l])alors
           DEB[i+1,j+1,l]=DEB[i,j,l]

fin
```

```
résultat-reconnaissance

pour j de 1 à J faire
début
  Dist(j)=MIN D(I(l)+1,J+1,l)
          l
  l= ARG MIN D(I(l)+1,J+1,l)
     l
  SUPERF(j,niv)=concat(SUPERF(Deb(I(l)+1,j,l)),MOT(l))
fin
```

IV. Resultats

Les performances du sytème ont été évaluées en paramétrant l'analyseur par le langage artificiel utilisé dans Myrtille I. Il s'agit d'un langage de communication avec un standard téléphonique, dont la partie ''demandé' peut être décrite par:

<PHRASE>=[<allo>][<bonjour>][<type>][(<INT>/<AFFIR>)][<svp>][<merci>];
<INT>=<int><DEM>;
<AFFIR>=<je voudrais><DEM>/[<passez-moi>]<objet>;
<DEM>=[<parler à>]<type><nom>/<avoir><OBJET>;
<OBJET>=<le>(<QUALIF><de><type><nom>/<num>/<qual><nom>)/<type><nom>;
<QUALIF>=<bureau>/<quel>;

type désignant la classe : monsieur, madame, mademoiselle.
nom - - - : Albert, Dupont, Durant, Pierre, Pierrel.
num _ _ _ : 221, 22, 223, 229, 240, 241.
qual - - - : poste, numéro.

Essai de déroulement de l'algorithme :

Nous allons voir, sur un exemple simple, le déroulement dans le temps de l'algorithme sur l'axiome de la grammaire. Afin de simplifier la présentation des résuktats, nous utiliserons le graphe syntaxique la forme de Backus-Naur des règles).

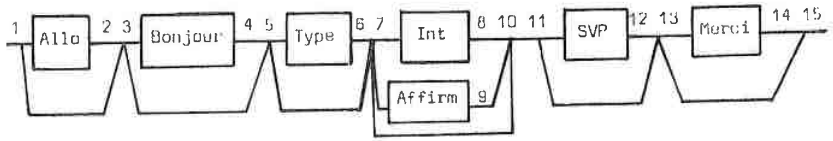
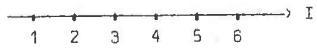


Figure 8: graphe syntaxique équivalent à l'axiome de la grammaire.

En introduisant la chaîne phonétique simulée suivante :

A I B AI J ON
AU AN V ON B OU
D AN



on aura, si D représente le score cumulé de reconnaissance d'une chaîne et I l'indice dans le treillis de phonèmes de la fin de la chaîne reconnue:

Noeud	I	TD	CHAINE RECONNUE	COMMENTAIRES
1	1	0	''	Point de départ
2	2	3	ALLO	
	3	1	ALLO	
	4	4	ALLO	Reconnaissance
	5	3	ALLO	de ALLO
	6	5	ALLO	
	8	4	ALLO	

Noeud	I	TD	CHAINE RECONNUE	COMMENTAIRES
3	1	0	''	
	2	3	ALLO	$\tilde{M}$in entre 1 et 2
	3	1	ALLO	
	4	4	ALLO	
	5	3	ALLO	
	6	5	ALLO	
	8	4	ALLO	
4	3	3	Bonjour	
	4	4	Bonjour	Reconnaissance
	5	3	Bonjour	de Bonjour
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	
5	1	0	''	
	2	3	ALLO	$\tilde{M}$in entre 4 et 3
	3	1	ALLO	
	4	4	Bonjour	
	5	3	Bonjour	
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	
6	5	4	Madame	Reconnaissance de
	6	4	Madame	la classe Type
	8	4	Allo Madame	
7	1	0	''	
	2	3	ALLO	$\tilde{M}$in entre 5 et 6
	3	1	ALLO	
	4	4	Bonjour	
	5	3	Bonjour	
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	

Noeud	I	TD	CHAINE RECONNUE	COMMENTAIRES
8	ECHEC			
9	ECHEC			
10	ECHEC			$\tilde{M}$in entre 8 et 9
11	1	0	''	
	2	3	ALLO	$\tilde{M}$in entre 10 et 7
	3	1	ALLO	
	4	4	Bonjour	
	5	3	Bonjour	
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	
12	ECHEC			Reconnaissance de SVP
13	1	0	''	
	2	3	ALLO	$\tilde{M}$in entre 11 et 12
	3	1	ALLO	
	4	4	Bonjour	
	5	3	Bonjour	
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	
14	3	3	Merci	
	5	5	Allo Merci	Reconnaissance de
	6	5	Allo Merci	Merci
	8	4	Allo Merci	
15	1	0	''	
	3	3	ALLO	$\tilde{M}$in entre 13 et 14
	4	4	Bonjour	
	5	3	Bonjour	
	6	3	Allo Bonjour	
	8	1	Allo Bonjour	

CONCLUSION

CONCLUSION

Dans cette étude, nous avons abordé le problème de la dictée vocale de textes. L' objectif était la réalisation d'un système de reconnaissance de parole continue autorisant un vocabulaire de l' ordre de mille racines de mots. Les traitements devant être mis en oeuvre dans un tel système sont spécifiques à la dictée automatique. Il est notamment difficile de recourir à des informations linguistiques car le langage autorisé se veut par définition le plus général possible. C'est pourquoi l'indéterminisme propre au traitement de la parole est plus délicat à résoudre. La reconnaissance ne peut utiliser aucune connaissance pragmatique. Un autre aspect particulier à un système de dictée vocale est la procédure de dialogue dont l' objectif consiste à valider les résultats de la reconnaissance qui peuvent être multiples. Cette procédure de dialogue peut être réalisée, dans notre système à l' aide d'un système de désignation de type souris ou bien par un dialogue oral utilisant un langage de commande artificiel limité. Nous avons donc dû nous intéresser à la fois au problème de la reconnaissance vocale dans le cadre d' un langage naturel (le texte lu) et d' un langage artificiel (le langage de commande). Nous avons été confronté à divers problèmes liés au langage traité comme:

- la création et la gestion d'un vocabulaire important,
- la définition d'un modèle syntaxique suffisamment général pour accepter un français courant,
- le choix de stratégies performantes.

Une grande partie de notre étude a été consacrée à l'extention d'algorithmes de programmation dynamique utilisés par ailleurs avec succès pour la reconnaissance de mots enchaînés. Les algorithmes proposés dans ce travail mettent à profit des contraintes syntaxiques. Elles sont soit locales pour le langage de saisie de texte lu, soit globales pour le langage de commande.

Pour le mode édition de texte lu nous avons élaboré une extension de l'algorithme "Level Building". Une analyse en faisceaux résoud plusieurs problèmes propres à la reconnaissance analytique comme la prédominance des mots courts et l'intégration de contraintes syntaxiques.

Le travail que nous avons présenté dans ce rapport ne constitue qu'une étape. Il reste encore beaucoup à faire pour parvenir à la réalisation d'une machine à dicter vocale opérationnelle notamment dans les domaines du décodage acoustico-phonétique et des modèles linguistiques.

ANNEXES

1. Résultats du système de reconnaissance

2. Les règles de conjugaison

3. Les règles grammaticales

ANNEXE 1

RESULTATS DU SYSTEME

DE RECONNAISSANCE

Dans cette annexe, nous donnons une première évaluation du sytème de reconnaissance du mode édition parole. Les algorithmes que nous avons présentés ont été implantés sur un système de développement Motorola Exormacs (microprocesseur 68000). Le sytème n'ayant pas encore été connecté avec un décodeur acoustico-phonétique, il n'a pu être évalué dans des conditions réelles d'utilisation.

Les résultats que nous proposons ici sont donnés à titre indicatif, le sytème n'ayant pas été testé sur un corpus suffisant. L'évaluation a été effectuée à partir de 30 spectrogrammes de phrases. Trois locuteurs masculins ont enregistré chacun 10 phrases phonétiquement équilibrées du corpus de Combescure [Combescure 81]. Les spectrogrammes ont été décodés en unités phonétiques par un expert humain en lecture de spectrogrammes (François Lonchamp). Les performances du décodage phonétique atteignent 80% de réussite pour la reconnaissance des consonnes et 73% pour les voyelles [Carbonell 84]. Chaque spectrogramme est décrit par une liste de segments pour lesquels le phonétitien propose jusqu'à six étiquettes phonétiques. Nous donnons figure 1 un exemple d'un tel treillis phonétique. Les phrases testées comportent en moyenne un peu plus de six mots prononcés sans pause artificielle entre les mots.

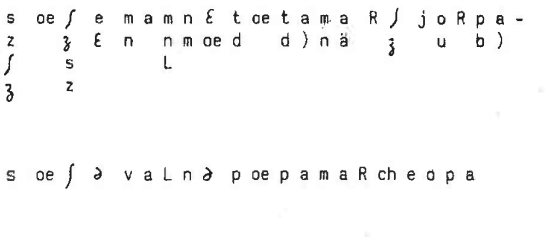


Figure 1 treillis phonétique de la phrase :
"ce cheval ne peut pas marcher au pas "

Le tableau donné figure 2 indique les résultats de notre système. Il comporte pour chaque phrase du corpus :

- le nombre de mots reconnus,
- le nombre de mots prononcés,
- la phrase effectivement prononcée,
- la phrase reconnue.

REMARQUE

La phrase reconnue que nous donnons dans le tableau figure 2 est choisie manuellement par l'utilisateur parmis un ensemble de phrases proposées par le système de reconnaissance.

Les résultats obtenus sont encourageant puisqu'ils atteignent près de 77% de mots reconnus. Si nous ne tenons pas compte des erreurs commises sur les articles ou les prépositions, le taux de reconnaissance est de 82%. Cependant, ces résultats sont encore insuffisants pour une exploitation commerciale. Toutefois, si chaque mot est prononcé isolément, les taux de réussite dépassent 95% de réussite.

En ce qui concerne la parole continue, plusieurs voies de recherche sont envisageables pour améliorer les résultats que nous avons obtenus.

D'abord, une plus grande interactivité avec le niveau phonétique semble une voie intéressante à explorer. Notamment, dans le cas de l'utilisation d'un sytème expert comme décodeur acoustico-phonétique, il serait intéressant de valider par un système de question réponse les hypothèses d'insertion, d'élision et de substitution émises lors de l'analyse lexicale.

Ensuite, au niveau linguistique, la définition d'une grammaire probabiliste telle que celle mise en oeuvre dans le système de transcription d'IBM [Derouault 85], permettrait d'affiner le filtrage syntaxique.

7	7	ici il fait toujours froid en hiver ici il fait toujours froid en hiver
3	4	mes gants sont usés les gants sont usés
6	6	j'aime Sylvie quand elle est j'aime Sylvie quand elle est
5	6	Diane ne reviendra pas avant lundi Diane ne reviendra pas par lundi
4	4	aimez vous ce dessin aimez vous ce dessin
6	6	j'ai déjà lu la réponse j'ai déjà lu la réponse
8	8	ce cheval ne peut pas marcher au pas ce cheval ne peut pas marcher au pas
7	7	voilà toujours deux choux pour le repas voilà toujours deux choux pour le repas
2	4	le repas de midi le repas veut lui dire
5	7	la barque du pecheur a été emportée la barque des pecheurs a été pourrait
2	3	par la tempête par leur tempête
7	7	le mal s'envenime faute de soin le mal s'envenime faute de soin
3	4	on entend le gazouilli on entend les gazouillis
3	5	un oiseau dans le jardin leur oiseau le jardin
7	7	il s'arretait tout l'été ici il s'arretait tout l'été ici
6	7	les manches de son manteau sont décousues les manches dans son manteau sont décousues
8	9	ce vaisseau parcourt les mers à travers le monde ce vaisseau parcourt les mers à travers un monde
4	8	je suis sur que vous connaissez ce nom le chasseur vous connaissez ce nom
3	6	ce livre provient de la bibliothèque ce livre bibliothèque
7	7	Jean quant à lui est très grand Jean quant à lui est très grand
3	3	pour son âge pour son âge
2	9	L'été tout le monde se mettait aux fenêtres aux fenêtres
4	6	le cocher a fouetté sa jument le cocher heureux dès sa jument
5	7	le forçat s'est évadé du bagne le forçat se met par aide du bagne
1	5	il a broyé du noir du
5	6	depuis la perte de son ami depuis la perte pour son ami

7	7	je rend souvent visite à mon oncle je rend souvent visite à mon oncle
3	6	ma soirée se passera sans incidents passera sans incidents
4	7	la police veut les papiers du chauffeur leur police avec papiers du chauffeur

Figure 2 : évaluation des performances des algorithmes de reconnaissance

ANNEXE 2

LES REGLES DE CONJUGAISON

```
AIMER 6
X"ER"++>INDPRES(X+S1,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"ER"+S10,X"ER"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"ER"+S8,X"ER"+P20)
        PARTPRES(X+P39)
        PARTPASS(X+P40);
PLACER 7
X"CER"++>INDPRES(X"C"+S1,X+P2)
         INDIMP(X"@"+S8,X+P21)
         INDFUT(X"CER"+S10,X"CER"+P23)
         SUBJPRES(X"C"+S1,X"C"+P24)
         CONDPRES(X"CER"+S8,X"CER"+P20)
         PARTPRES(X"@"+P39)
         PARTPASS(X"C"+P40);
MANGER 8
X"ER"++>INDPRES(X+S1,X+P3)
        INDIMP(X"E"+S8,X+P22)
        INDFUT(X"ER"+S10,X"ER"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"ER"+S8,X"ER"+P20)
        PARTPRES(X"E"+P39)
        PARTPASS(X+P40);
PESER 9
MIC"E"MAC"ER"++>INDPRES(MIC"`"MAC+S1,MIC+P4[MAC])
                INDIMP(MIC"E"MAC+S8,MIC"E"MAC+P20)
                INDFUT(MIC"`"MAC"ER"+S10,MIC"`"MAC"ER"+P23)
                SUBJPRES(MIC"`"MAC+S1,MIC+P25[MAC])
                CONDPRES(MIC"`"MAC"ER"+S8,MIC"`"MAC"ER"+P20)
                PARTPRES(MIC"E"MAC+P39)
                PARTPAS(MIC"E"MAC+P40);
C'DER 10
X"'"Y"ER"++>INDPRES(X"`"Y+S1,X+P5[Y])
            INDIMP(X"'"Y+S8,X"'"Y+P20)
            INDFUT(X"'"Y"ER"+S10,X"'"Y"ER"+P23)
            SUBJPRES(X"`"Y+S1,X+P26[Y])
            CONDPRES(X"'"Y"ER"+S8,X"'"Y"ER"+P20)
```

```
            PARTPRES(X"'"Y+P39)
            PARTPAS(X"'"Y+P40);
JETER 11
X"E"Y"ER"++>INDPRES(X"E"Y-Y+S1,X"E"Y+P6[Y])
            INDIMP(X"E"Y+S8,X"E"Y+P20)
            INDFUT(X"E"Y-Y"ER"+S10,X"E"Y-Y"ER"+P23)
            SUBJPRES(X"E"Y-Y+S1,X"E"Y+P27[Y])
            CONDPRES(X"E"Y-Y"ER"+S8,X"E"Y-Y"ER"+P20)
            PARTPRES(X"E"Y+P39)
            PARTPAS(X"E"Y+P40);
MODELER 12
X"E"Y"ER"++>INDPRES(X"`"Y+S1,X+P4[Y])
            INDIMP(X"E"Y+S8,X"E"Y+P20)
            INDFUT(X"`"Y"ER"+S10,X"`"Y"ER"+P23)
            SUBJPRES(X"`"Y+S1,X+P25[Y])
            CONDPRES(X"`"Y"ER"+S8,X"`"Y"ER"+P20)
            PARTPRES(X"E"Y+P39)
            PARTPAS(X"E"Y+P40);
CR'ER 13
X"ER"++>INDPRES(X+S1,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"ER"+S10,X"ER"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"ER"+S8,X"ER"+P23)
        PARTPRES(X+P39)
        PARTPASS(X+P40);
ASSI'GER 14
X"'"Y"ER"++>INDPRES(X"`"Y+S1,X+P7)
            INDIMP(X"'GE"+S8,X"'G"+P22)
            INDFUT(X"'GER"+S10,X"'GER"+P23)
            SUBJPRES(X"`G"+S1,X+P26[Y])
            CONDPRES(X"'GER"+S8,X"'GER"+P20)
            PARTPRES(X"'GE"+P39)
            PARTPAS(X"'G"+P40);
APPR'CIER 15
X"ER"++>INDPRES(X+S1,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"ER"+S10,X"ER"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"ER"+S8,X"ER"+P23)
        PARTPRES(X+P39)
        PARTPASS(X+P40);
PAYER 16
X"YER"++>INDPRES(X"I"+S1,X+P8)
         INDIMP(X"Y"+S8,X"Y"+P20)
         INDFUT(X"IER"+S10,X"IER"+P23)
         SUBJPRES(X"Y"+S1,X+P28)
         CONDPRES(X"IER"+S8,X"IER"+P20)
         PARTPRES(X"Y"+P39)
         PARTPAS(X"Y"+P40);
BROYER 17
X"YER"++>INDPRES(X"I"+S1,X+P8)
         INDIMP(X"Y"+S8,X"Y"+P20)
         INDFUT(X"IER"+S10,X"IER"+P23)
         SUBJPRES(X"I"+S1,X+P28)
         CONDPRES(X"IER"+S8,X"IER"+P20)
         PARTPRES(X"Y"+P39)
         PARTPAS(X"Y"+P40);
ENVOYER 18
X"OYER"++>INDPRES(X"OI"+S1,X"O"+P8)
          INDIMP(X"OY"+S8,X"OY"+P20)
          INDFUT(X"ERR"+S10,X"ERR"+P23)
          SUBJPRES(X"OI"+S1,X"O"+P28)
          CONDPRES(X"ERR"+S8,X"ERR"+P20)
          PARTPRES(X"OY"+P39)
          PARTPAS(X"OY"+P40);
FINIR 19
X"IR"++>INDPRES(X"I"+S4,X"ISS"+P1)
          INDIMP(X"ISS"+S8,X"ISS"+P20)
          INDFUT(X"IR"+S10,X"IR"+P23)
          SUBJPRES(X"ISS"+S1,X"ISS"+P24)
          CONDPRES(X"IR"+S8,X"IR"+P20)
          PARTPRES(X"ISS"+P39)
          PARTPAS(X+P41);
HAIR 20
X"IR"++>INDPRES(X"I"+S4,X"ISS"+P1)
          INDIMP(X"ISS"+S8,X"ISS"+P20)
          INDFUT(X"IR"+S10,X"IR"+P23)
          SUBJPRES(X"ISS"+S1,X"ISS"+P24)
          CONDPRES(X"IR"+S8,X"IR"+P20)
          PARTPRES(X"ISS"+P39)
          PARTPAS(X+P41);
BREAKDANCER 21
TYPE BIDON++>;
ALLER 22
X"ALLER"++>INDPRES("VA"+S3,""+P10)
          INDIMP("ALL"+S8,"ALL"+P20)
          INDFUT("IR"+S10,"IR"+P23)
          SUBJPRES("AILL"+S1,""+P30)
          CONDPRES("IR"+S8,"IR"+P20)
          PARTPRES("ALL"+P39)
          PARTPAS("ALL"+P40);
TENIR 23
X"ENIR"++>INDPRES(X"IEN"+S4,X+P11)
          INDIMP(X"EN"+S8,X"EN"+P20)
          INDFUT(X"IENDR"+S10,X"IENDR"+P23)
          SUBJPRES(X"IENN"+S1,X+P31)
          CONDPRES(X"IENDR"+S8,X"IENDR"+P20)
          PARTPRES(X"EN"+P39)
          PARTPAS(X"EN"+P42);
ACQU'RIR 24
X"'RIR"++>INDPRES(X"IER"+S4,X+P9)
          INDIMP(X"'R"+S8,X"'R"+P20)
          INDFUT(X"ERR"+S10,X"ERR"+P23)
```

```
          SUBJPRES(X"I`R"+S1,X+P29)
          CONDPRES(X"ERR"+S8,X"ERR"+P20)
          PARTPRES(X"'R"+P39)
          PARTPAS(X"I"+P44);
SENTIR 25
X"TIR"++>INDPRES(X+S4,X"T"+P1)
         INDIMP(X"T"+S8,X"T"+P20)
         INDFUT(X"TIR"+S10,X"TIR"+P23)
         SUBJPRES(X"T"+S1,X"T"+P24)
         CONDPRES(X"TIR"+S8,X"TIR"+P20)
         PARTPRES(X"T"+P39)
         PARTPAS(X"T"+P41);
V^ETIR 26
X"IR"++>INDPRES(X+S7,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"IR"+S10,X"IR"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"IR"+S8,X"IR"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+P42);
COUVRIR 27
X"RIR"++>INDPRES(X"R"+S1,X"R"+P1)
         INDIMP(X"R"+S8,X"R"+P20)
         INDFUT(X"RIR"+S10,X"RIR"+P23)
         SUBJPRES(X"R"+S1,X"R"+P24)
         CONDPRES(X"RIR"+S8,X"RIR"+P20)
         PARTPRES(X"R"+P39)
         PARTPAS(X"ER"+P43);
CUEILLIR 28
X"IR"++>INDPRES(X+S1,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"ER"+S10,X"ER"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"ER"+S8,X"ER"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+P41);
ASSAILLIR 29
X"IR"++>INDPRES(X+S1,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"IR"+S10,X"IR"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"IR"+S8,X"IR"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+P41);
FAILLIR 30
X"AILLIR"++>INDPRES(X"AU"+S5,X"AILL"+P1)
            INDIMP(X"AILL"+S8,X"AILL"+P20)
            INDFUT(X"AILLIR"+S10,X"AILLIR"+P23)
            SUBJPRES(X"AILL"+S1,X"AILL"+P24)
            CONDPRES(X"AILLIR"+S8,X"AILLIR"+P20)
            PARTPRES(X"AILL"+P39)
            PARTPAS(X"AILL"+S15);
```

```
BOUILLIR 31
X"ILLIR"++>INDPRES(X+S4,X"ILL"+P1)
           INDIMP(X"ILL"+S8,X"ILL"+P20)
           INDFUT(X"ILLIR"+S10,X"ILLIR"+P23)
           SUBJPRES(X"ILL"+S1,X"ILL"+P24)
           CONDPRES(X"ILLIR"+S8,X"ILLIR"+P20)
           PARTPRES(X"ILL"+P39)
           PARTPAS(X"ILL"+P41);
DORMIR 32
X"MIR"++>INDPRES(X+S4,X"M"+P1)
         INDIMP(X"M"+S8,X"M"+P20)
         INDFUT(X"MIR"+S10,X"MIR"+P23)
         SUBJPRES(X"M"+S1,X"M"+P24)
         CONDPRES(X"MIR"+S8,X"MIR"+P20)
         PARTPRES(X"M"+P39)
         PARTPAS(X"M"+S15);
COURIR 33
X"IR"++>INDPRES(X+S4,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+P42);
MOURIR 34
X"OU"Y"IR"++>INDPRES(X"EUR"+S4,X+P12[Y])
             INDIMP(X"OUR"+S8,X"OUR"+P20)
             INDFUT(X"OURR"+S10,X"OURR"+P23)
             SUBJPRES(X"EUR"+S1,X+P32[Y])
             CONDPRES(X"OURR"+S8,X"OURR"+P20)
             PARTPRES(X"OU"Y+P39)
             PARTPRES(X"OR"+P43);
SERVIR 35
X"VIR"++>INDPRES(X+S4,X"V"+P1)
         INDIMP(X"V"+S8,X"V"+P20)
         INDFUT(X"VIR"+S10,X"VIR"+P23)
         SUBJPRES(X"V"+S1,X"V"+P24)
         CONDPRES(X"VIR"+S8,X"VIR"+P20)
         PARTPRES(X"V"+P39)
         PARTPAS(X"V"+P41);
FUIR 36
X"IR"++>INDPRES(X"I"+S4,X+P8)
        INDIMP(X"Y"+S8,X"Y"+P20)
        INDFUT(X"IR"+S10,X"IR"+P23)
        SUBJPRES(X"I"+S1,X+P28)
        CONDPRES(X"IR"+S8,X"IR"+P20)
        PARTPRES(X"Y"+P39)
        PARTPAS(X+P41);
OUIR 37
X"UIR"++>INDPRES(X"I"+S4,X"Y"+P1)
         INDIMP(X"Y"+S8,X"Y"+P20)
         INDFUT(X"UIR"+S10,X"UIR"+P23)
```

```
        SUBJPRES(X"I"+S1,X+P28)
        CONDPRES(X"UIR"+S10,X"UIR"+P20)
        PARTPRES(X"Y"+P39)
        PARTPAS(X"U"+P41);
RECEVOIR 38
X"CEVOIR"++>INDPRES(X"@OI"+S4,X+P14)
            INDIMP(X"CEV"+S8,X"CEV"+P20)
            INDFUT(X"CEVR"+S10,X"CEVR"+P23)
            SUBJPRES(X"@OIV"+S1,X+P33)
            CONDPRES(X"CEVR"+S8,X"CEVR"+P20)
            PARTPRES(X"CEV"+P39)
            PARTPAS(X"@"+P42);
VOIR 39
X"OIR"++>INDPRES(X"OI"+S4,X"O"+P8)
         INDIMP(X"OY"+S8,X"OY"+P20)
         INDFUT(X"ERR"+S10,X"ERR"+P23)
         SUBJPRES(X"OI"+S1,X"O"+P28)
         CONDPRES(X"ERR"+S8,X"ERR"+P20)
         PARTPRES(X"OY"+P39)
         PARTPAS(X+P42);
POURVOIR 40
X"OIR"++>INDPRES(X"OI"+S4,X"O"+P8)
         INDIMP(X"OY"+S8,X"OY"+P20)
         INDFUT(X"OIR"+S10,X"OIR"+P23)
         SUBJPRES(X"OI"+S1,X"O"+P28)
         CONDPRES(X"OIR"+S8,X"OIR"+P20)
         PARTPRES(X"OY"+P39)
         PARTPAS(X+P42);
SAVOIR 41
X"AVOIR"++>INDPRES(X"AI"+S4,X"AV"+P1)
           INDIMP(X"AV"+S8,X"AV"+P20)
           INDFUT(X"AUR"+S10,X"AUR"+P23)
           SUBJPRES(X"ACH"+S1,X"ACH"+P24)
           CONDPRES(X"AUR"+S8,X"AUR"+P20)
           PARTPRES(X"ACH"+P39)
           PARTPAS(X+P42);
DEVOIR 42
X"EVOIR"++>INDPRES(X"OI"+S4,X+P15)
           INDIMP(X"EV"+S8,X"EV"+P20)
           INDFUT(X"EVR"+S10,X"EVR"+P23)
           SUBJPRES(X"OIV"+S1,X+P34)
           PARTPRES(X"EV"+P39)
           PARTPAS(X+P46);
POUVOIR 43
X"OU"Y"OIR"++>INDPRES(X"EU"+S5,X+P12[Y])
              INDIMP(X"OUV"+S8,X"OUV"+P20)
              INDFUT(X"OURR"+S10,X"OURR"+P23)
              SUBJPRES(X"UISS"+S1,X"UISS"+P24)
              CONDPRES(X"OURR"+S8,X"OURR"+P20)
              PARTPRES(X"OUV"+P39)
              PARTPAS(X+S13);
MOUVOIR 44
X"OU"Y"OIR"++>INDPRES(X"EU"+S4,X+P12[Y])
              INDIMP(X"OUV"+S8,X"OUV"+P20)
              INDFUT(X"OUVR"+S10,X"OUVR"+P23)
              SUBJPRES(X"EUV"+S1,X+P32[Y])
              CONDPRES(X"OUVR"+S8,X"OUVR"+P20)
              PARTPRES(X"OUV"+P39)
              PARTPAS(X+P46);
PLEUVOIR 45
X"EUVOIR"-->INDPRES(X"EU"+S6,X"EUV"+P16)
          INDIMP(X"EUV"+S9,X"EUVAI"+P16)
          INDFUT(X"EUVR"+S11,X"EUVR"+P17)
          SUBJPRES(X"EUV"+S2,X"EUV"+P16)
          CONDPRES(X"EUVR"+S9,X"EUVRAI"+P16)
          PARTPRES(X"EUV"+P39)
          PARTPAS(X+S13);
FALLOIR 46
X"LLOIR"-->INDPRES(X"U"+S6)
           INDIMP(X"LL"+S9)
           INDFUT(X"UDR"+S11)
           SUBJPRES(X"ILL"+S2)
           CONDPRES(X"UDR"+S9)
           PARTPAS(X"LL"+S13);
VALOIR 47
X"LOIR"-->INDPRES(X"U"+S5,X"L"+P1)
          INDIMP(X"L"+S8,X"L"+P20)
          INDFUT(X"UDR"+S10,X"UDR"+P23)
          SUBJPRES(X"ILL"+S1,X+P35)
          CONDPRES(X"UDR"+S8,X"UDR"+P20)
          PARTPRES(X"L"+P39)
          PARTPAS(X"L"+P42);
VOULOIR 48
X"OU"Y"OIR"-->INDPRES(X"EU"+S5,X+P12[Y])
              INDIMP(X"OU"Y+S8,X"OU"Y+P20)
              INDFUT(X"OUDR"+S10,X"OUDR"+P23)
              SUBJPRES(X"EUILL"+S1,X+P36)
              CONDPRES(X"OUDR"+S8,X"OUDR"+P20)
              PARTPRES(X"OUL"+P39)
              PARTPAS(X"OUL"+P42);
ASSEOIR 49
X"EOIR"-->INDPRES(X"IED"+S7,X"EY"+P1)
          INDIMP(X"EY"+S8,X"EY"+P20)
          INDFUT(X"IER"+S10,X"IER"+P23)
          SUBJPRES(X"EY"+S1,X"EY"+P24)
          CONDPRES(X"I'R"+S8,X"I'R"+P20)
          PARTPRES(X"EY"+P39)
          PARTPAS(X"I"+P44);
SEOIR 50
X"EOIR"-->INDPRES(X"IE"+S12,X"I'"+P16)
          INDIMP(X"EY"+S9,X"EYAI"+P16)
          INDFUT(X"I'R"+S11,X"I'R"+P17)
          SUBJPRES(X"I'"+S2,X"I'"+P16)
          CONDPRES(X"I'R"+S9,X"I'RAI"+P16)
```

```
          PARTPRES(X"'"+P39);
SURSEOIR 51
X"EOIR"-->INDPRES(X"OI"+S4,X"O"+P8)
          INDIMP(X"OY"+S8,X"OY"+P20)
          INDFUT(X"EOIR"+S10,X"EOIR"+P23)
          SUBJPRES(X"OI"+S1,X"O"+P28)
          CONDPRES(X"EOIR"+S8,X"EOIR"+P20)
          PARTPRES(X"OY"+P39)
          PARTPAS(X"I"+P44);
CHOIR 52
X"OIR"==>INDPRES(X"OI"+S4,X"OI"+P16)
         INDFUT(X"OIR"+S10)
         CONDPRES(X"OIR"+S8)
         PARTPAS(X+P42);
RENDRE 53
X"RE"==>INDPRES(X+S7,X+P1)
         INDIMP(X+S8,X+P20)
         INDFUT(X"R"+S10,X"R"+P23)
         SUBJPRES(X+S1,X+P24)
         CONDPRES(X"R"+S8,X"R"+P20)
         PARTPRES(X+P39)
         PARTPAS(X+P42);
PRENDRE 54
X"E"Y"DRE"==>INDPRES(X"END"+S7,X"EN"+P6[Y])
           INDIMP(X"EN"+S8,X"EN"+P20)
           INDFUT(X"ENDR"+S10,X"ENDR"+P23)
           SUBJPRES(X"ENN"+S1,X"EN"+P27[Y])
           CONDPRES(X"ENDR"+S8,X"ENDR"+P20)
           PARTPRES(X"EN"+P39)
           PARTPAS(X"I"+P44);
BATTRE 55
X"TRE"==>INDPRES(X+S7,X"T"+P1)
         INDIMP(X"T"+S8,X"T"+P20)
         INDFUT(X"TR"+S10,X"TR"+P23)
         SUBJPRES(X"T"+S1,X"T"+P24)
         CONDPRES(X"TR"+S8,X"TR"+P20)
         PARTPRES(X"T"+P39)
         PARTPAS(X"T"+P42);
METTRE 56
X"ETTRE"==>INDPRES(X"ET"+S7,X"ETT"+P1)
           INDIMP(X"ETT"+S8,X"ETT"+P20)
           INDFUT(X"ETTR"+S10,X"ETTR"+P23)
           SUBJPRES(X"ETT"+S1,X"ETT"+P24)
           CONDPRES(X"ETTR"+S8,X"ETTR"+P20)
           PARTPRES(X"ETT"+P39)
           PARTPAS(X"I"+P44);
PEINDRE 57
X"NDRE"==>INDPRES(X"N"+S4,X"GN"+P1)
          INDIMP(X"GN"+S8,X"GN"+P20)
          INDFUT(X"NDR"+S10,X"NDR"+P23)
          SUBJPRES(X"GN"+S1,X"GN"+P24)
          CONDPRES(X"NDR"+S8,X"NDR"+P20)
          PARTPRES(X"GN"+P39)
          PARTPAS(X"N"+P43);
JOINDRE 58
X"NDRE"==>INDPRES(X"N"+S4,X"GN"+P1)
          INDIMP(X"GN"+S8,X"GN"+P20)
          INDFUT(X"NDR"+S10,X"NDR"+P23)
          SUBJPRES(X"GN"+S1,X"GN"+P24)
          CONDPRES(X"NDR"+S8,X"NDR"+P20)
          PARTPRES(X"GN"+P39)
          PARTPAS(X"N"+P43);
CRAINDRE 59
X"NDRE"==>INDPRES(X"N"+S4,X"GN"+P1)
          INDIMP(X"GN"+S8,X"GN"+P20)
          INDFUT(X"NDR"+S10,X"NDR"+P23)
          SUBJPRES(X"GN"+S1,X"GN"+P24)
          CONDPRES(X"NDR"+S8,X"NDR"+P20)
          PARTPRES(X"GN"+P39)
          PARTPAS(X"N"+P43);
VAINCRE 60
X"CRE"==>INDPRES(X"C"+S7,X"QU"+P1)
         INDIMP(X"QU"+S8,X"QU"+P20)
         INDFUT(X"CR"+S10,X"CR"+P23)
         SUBJPRES(X"QU"+S1,X"QU"+P24)
         CONDPRES(X"CR"+S8,X"CR"+P20)
         PARTPRES(X"QU"+P39)
         PARTPAS(X"C"+P42);
TRAIRE 61
X"IRE"==>INDPRES(X"I"+S4,X+P8)
         INDIMP(X"Y"+S8,X"Y"+P20)
         INDFUT(X"IR"+S10,X"IR"+P23)
         SUBJPRES(X"I"+S1,X+P28)
         CONDPRES(X"IR"+S8,X"IR"+P20)
         PARTPRES(X"Y"+P39)
         PARTPAS(X"I"+P43);
FAIRE 62
X"AIRE"==>INDPRES(X"AI"+S4,X+P18)
          INDIMP(X"AIS"+S8,X"AIS"+P20)
          INDFUT(X"ER"+S10,X"ER"+P23)
          SUBJPRES(X"ASS"+S1,X"ASS"+P24)
          CONDPRES(X"ER"+S8,X"ER"+P20)
          PARTPRES(X"AIS"+P39)
          PARTPAS("FAI"+P43);
PLAIRE 63
X"AIRE"==>INDPRES(X"A"+S14,X"AIS"+P1)
          INDIMP(X"AIS"+S8,X"AIS"+P20)
          INDFUT(X"AIR"+S10,X"AIR"+P23)
          SUBJPRES(X"AIS"+S1,X"AIS"+P24)
          PARTPRES(X"AIS"+P39)
          PARTPAS(X+S13);
CONNAITRE 64
X"A^ITRE"==>INDPRES(X"A"+S14,X"AISS"+P1)
            INDIMP(X"AISS"+S8,X"AISS"+P20)
```

```
          INDFUT(X"A^ITR"+S10,X"A^ITR"+P23)
          SUBJPRES(X"AISS"+S1,X"AISS"+P24)
          CONDPRES(X"A^ITR"+S8,X"A^TR"+P20)
          PARTPRES(X"AISS"+P39)
          PARTPAS(X+P42);
NAITRE 65
X"A^ITRE"==>INDPRES(X"A"+S14,X"AISS"+P1)
          INDIMP(X"AISS"+S8,X"AISS"+P20)
          INDFUT(X"A^ITR"+S10,X"A^ITR"+P23)
          SUBJPRES(X"A^ITR"+S8,X"A^ITR"+P20)
          PARTPRES(X"AISS"+P39)
          PARTPAS(X+P40);
PAITRE 66
X"A^ITRE"==>INDPRES(X"A"+S14,X"AISS"+P1)
           INDIMP(X"AISS"+S8,X"AISS"+P20)
           INDFUT(X"A^ITR"+S10,X"A^ITR"+P23)
           SUBJPRES(X"AISS"+S1,X"AISS"+P24)
           CONDPRES(X"A^ITR"+S8,X"A^ITR"+P20)
           PARTPRES(X"AISS"+P39)
           PARTPAS(X+S13);
CROITRE 67
X"O^ITRE"==>INDPRES(X"O^I"+S4,X"OISS"+P1)
           INDIMP(X"OISS"+S8,X"OISS"+P20)
           INDFUT(X"O^ITR"+S10,X"O^ITR"+P23)
           SUBJPRES(X"OISS"+S1,X"OISS"+P24)
           CONDPRES(X"O^ITR"+S8,X"O^ITR"+P20)
           PARTPRES(X"OISS"+P39)
           PARTPAS(X+P46);
CROIRE 68
X"OIRE"==>INDPRES(X"OI"+S4,X"O"+P8)
          INDIMP(X"OY"+S8,X"OY"+P20)
          INDFUT(X"OIR"+S10,X"OIR"+P23)
          SUBJPRES(X"OI"+S1,X"O"+P28)
          CONDPRES(X"OIR"+S8,X"OIR"+P20)
          PARTPRES(X"OY"+P39)
          PARTPAS(X+P42);
BOIRE 69
X"OIRE"==>INDPRES(X"OI"+S4,X+P13)
          INDIMP(X"UV"+S8,X"UV"+P20)
          INDFUT(X"OIR"+S10,X"OIR"+P23)
          SUBJPRES(X"OIV"+S1,X+P38)
          CONDPRES(X"OIR"+S8,X"OIR"+P20)
          PARTPRES(X"UV"+P39)
          PARTPAS(X+P42);
CLORE 70
X"RE"==>INDPRES(X+S4,X"S"+P1)
        INDIMP(X"S"+S8,X"S"+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X"S"+S1,X"S"+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X"S"+P39)
        PARTPAS(X+P44);
```

```
CONCLURE 71
X"RE"==>INDPRES(X+S4,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+S16);
ABSOUDRE 72
X"UDRE"==>INDPRES(X"U"+S4,X"LV"+P1)
          INDIMP(X"LV"+S8,X"LV"+P20)
          INDFUT(X"UDR"+S10,X"UDR"+P23)
          SUBJPRES(X"LV"+S1,X"LV"+P24)
          CONDPRES(X"UDR"+S8,X"UDR"+P20)
          PARTPRES(X"LV"+P39)
          PARTPAS(X"U"+P47);
COUDRE 73
X"DRE"==>INDPRES(X"D"+S7,X"S"+P1)
         INDIMP(X"S"+S8,X"S"+P20)
         INDFUT(X"DR"+S10,X"DR"+P23)
         SUBJPRES(X"S"+S1,X"S"+P24)
         CONDPRES(X"DR"+S8,X"DR"+P20)
         PARTPRES(X"S"+P39)
         PARTPAS(X"S"+P42);
MOUDRE 74
X"DRE"==>INDPRES(X"D"+S7,X"L"+P1)
         INDIMP(X"L"+S8,X"L"+P20)
         INDFUT(X"DR"+S10,X"DR"+P23)
         SUBJPRES(X"L"+S1,X"L"+P24)
         CONDPRES(X"DR"+S8,X"DR"+P20)
         PARTPRES(X"L"+P39)
         PARTPAS(X"L"+P42);
SUIVRE 75
X"VRE"==>INDPRES(X+S4,X"V"+P1)
         INDIMP(X"V"+S8,X"V"+P20)
         INDFUT(X"VR"+S10,X"VR"+P23)
         SUBJPRES(X"V"+S1,X"V"+P24)
         CONDPRES(X"VR"+S8,X"VR"+P20)
         PARTPRES(X"V"+P39)
         PARTPAS(X"V"+P41);
VIVRE 76
X"IVRE"==>INDPRES(X"I"+S4,X"IV"+P1)
          INDIMP(X"IV"+S8,X"IV"+P20)
          INDFUT(X"IVR"+S10,X"IVR"+P23)
          SUBJPRES(X"IV"+S1,X"IV"+P24)
          CONDPRES(X"IVR"+S8,X"IVR"+P20)
          PARTPRES(X"IV"+P39)
          PARTPAS(X+P45);
LIRE 77
X"IRE"==>INDPRES(X"I"+S4,X"IS"+P1)
         INDIMP(X"IS"+S8,X"IS"+P20)
         INDFUT(X"IR"+S10,X"IR"+P23)
```

```
         SUBJPRES(X"IS"+S1,X"IS"+P24)
         CONDPRES(X"IR"+S8,X"IR"+P20)
         PARTPRES(X"IS"+P39)
         PARTPAS(X+P42);
DIRE 78
X"RE"==>INDPRES(X+S4,X+P19)
         INDIMP(X"S"+S8,X"S"+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X"S"+S1,X"S"+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X"S"+P39)
        PARTPAS(X+P43);
RIRE 79
X"RE"==>INDPRES(X+S4,X+P1)
        INDIMP(X+S8,X+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X+S1,X+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X+P39)
        PARTPAS(X+S17);
ECRIRE 80
X"RE"==>INDPRES(X+S4,X"V"+P1)
        INDIMP(X"V"+S8,X"V"+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X"V"+S1,X"V"+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X"V"+P39)
        PARTPAS(X+P43);
CONFIRE 81
X"RE"==>INDPRES(X+S4,X"S"+P1)
        INDIMP(X"S"+S8,X"S"+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X"S"+S1,X"S"+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X"S"+P39)
        PARTPAS(X+P43);
CUIRE 82
X"RE"==>INDPRES(X+S4,X"S"+P1)
        INDIMP(X"S"+S8,X"S"+P20)
        INDFUT(X"R"+S10,X"R"+P23)
        SUBJPRES(X"S"+S1,X"S"+P24)
        CONDPRES(X"R"+S8,X"R"+P20)
        PARTPRES(X"S"+P39)
        PARTPAS(X+P43);
DERNIER 1000
dernier==>;
```

```
S1 :          E;        ES;         E;
S2 :          +;         +;         E;
S3 :         IS;         S;         ;
S4 :          S;         S;         T;
S5 :          X;         X;         T;
S6 :          +;         +;         T;
S7 :          S;         S;         ;
S8 :        AIS;       AIS;       AIT;
S9 :          +;         +;       AIT;
S10:         AI;        AS;         A;
S11:          +;         +;         A;
S12:          +;         +;         D;
S13:          +;         +;         U;
S14:         IS;        IS;        ^IT;
S15:          +;         +;         I;
S16:          +;         ;          E;
S17:          +;         +;         ;
P1 :        ONS;        EZ;       ENT;
P2 :       @ONS;       CEZ;      CENT;
P3 :       EONS;        EZ;       ENT;
P4 :      E?ONS;      E?EZ;      `?ENT;
P5 :      '?ONS;      '?EZ;      `?ENT;
P6 :        ONS;        EZ;       ?ENT;
P7 :     'GEONS;     'GEZ;      `GENT;
P8 :       YONS;       YEZ;       IENT;
P9 :      'RONS;      'REZ;     I`RENT;
P10:     ALLONS;     ALLEZ;       VONT;
P11:      ENONS;      ENEZ;    IENNENT;
P12:     OU?ONS;     OU?EZ;     EU?ENT;
P13:      UVONS;      UVEZ;      OIVENT;
P14:     CEVONS;     CEVEZ;     @OIVENT;
P15:      EVONS;      EVEZ;      OIVENT;
P16:          +;          +;        ENT;
P17:          +;          +;        ONT;
P18:     AISONS;     AITES;       ONT;
P19:       SONS;       TES;       SENT;
P20:       IONS;       IEZ;      AIENT;
P21:      CIONS;      CIEZ;     @AIENT;
P22:       IONS;       IEZ;     EAIENT;
P23:        ONS;        EZ;       ONT;
P24:       IONS;       IEZ;       ENT;
P25:      E?IONS;     E?IEZ;     `?ENT;
P26:      '?IONS;     '?IEZ;     `?ENT;
P27:       IONS;       IEZ;      ?ENT;
P28:      YIONS;      YIEZ;      IENT;
P29:      'RIONS;     'RIEZ;    I`RENT;
P30:     ALLIONS;    AILLIEZ;   AILLENT;
P31:      ENIONS;     ENIEZ;    IENNENT;
P32:     OU?IONS;    OU?IEZ;    EU?ENT;
P33:     CEVIONS;    CEVIEZ;    @OIVENT;
P34:      EVIONS;     EVIEZ;     OIVENT;
P35:       LIONS;      LIEZ;     ILLENT;
```

```
P36:     OULIONS;     OULIEZ;   EUILLENT;
P37:          +;          +;          +;
P38:     UVIONS;      UVIEZ;     OIVENT;
P39:          +;          +;        ANT;
P40:          +;          ';         'E;
P41:          +;          I;         IE;
P42:          +;          U;         UE;
P43:          +;          T;         TE;
P44:          +;          S;         SE;
P45:          +;        'CU;       'CUE;
P46:          +;         ^U;         UE;
P47:          +;          S;         TE;
fin:
```

ANNEXE 3

LES REGLES DE

GRAMMAIRE

```
(* accord du nom ou de l'adjectif avec l' article qui precede *)
(* ------------------------------------------------------------ *)
  REGLE10
    <R10>=<artd><nom>
    #ARTD.NEU<>1 ->[NOM.MASC:=ARTD.MASC;NOM.FEM:=ARTD.FEM];
     NOM.SING:=ARTD.SING;NOM.PLUR:=ARTD.PLUR#;
  F
  REGLE11
    <R11>=<artd><adja>
    #ADJA.MASC:=ARTD.MASC;ADJA.FEM:=ARTD.FEM;
     ADJA.SING:=ARTD.SING;ADJA.PLUR:=ARTD.PLUR#;
  F
  REGLE12
    <R12>=<arti><nom>
    #NOM.MASC:=ARTI.MASC;NOM.FEM:=ARTI.FEM;
     NOM.SING:=ARTI.SING;NOM.PLUR:=ARTI.PLUR#;
  F
  REGLE13
    <R13>=<arti><adja>
    #ADJA.MASC:=ARTI.MASC;ADJA.FEM:=ARTI.FEM;
     ADJA.SING:=ARTI.SING;ADJA.PLUR:=ARTI.PLUR#;
  F
  REGLE14
    <R14>=<artp><nom>
    #NOM.MASC:=ARTP.MASC;NOM.FEM:=ARTP.FEM;
     NOM.SING:=ARTP.SING;NOM.PLUR:=ARTP.PLUR#;
  F
  REGLE15
    <R15>=<artp><adja>
    #ADJA.MASC:=ARTP.MASC;ADJA.FEM:=ARTP.FEM;
     ADJA.SING:=ARTP.SING;ADJA.PLUR:=ARTP.PLUR#;
  F
  REGLE16
    <R16>=<dem><nom>
    #NOM.MASC:=DEM.MASC;NOM.FEM:=DEM.FEM;
     NOM.SING:=DEM.SING;NOM.PLUR:=DEM.PLUR#;
  F
  REGLE17
```

```
    <R17>=<dem><adja>
    #ADJA.MASC:=DEM.MASC;ADJA.FEM:=DEM.FEM;
     ADJA.SING:=DEM.SING;ADJA.PLUR:=DEM.PLUR#;
  F
  REGLE18
    <R18>=<num><nom>
    #NOM.MASC:=NUM.MASC;NOM.FEM:=NUM.FEM;
     NOM.SING:=NUM.SING;NOM.PLUR:=NUM.PLUR#;
  F
  REGLE19
    <R19>=<num><adja>
    #ADJA.MASC:=NUM.MASC;ADJA.FEM:=NUM.FEM;
     ADJA.SING:=NUM.SING;ADJA.PLUR:=NUM.PLUR#;
  F
  REGLE200
    <R200>=<qua><nom>
    #NOM.MASC:=QUA.MASC;NOM.FEM:=QUA.FEM;
     NOM.SING:=QUA.SING;NOM.PLUR:=QUA.PLUR#;
  F
  REGLE210
    <R210>=<qua><adja>
    #ADJA.MASC:=QUA.MASC;ADJA.FEM:=QUA.FEM;
     ADJA.SING:=QUA.SING;ADJA.PLUR:=QUA.PLUR#;
  F
(*-------------------------------------------------------------*)
(* accord du nom avec l'adjectif qui precede                  *)
(*-------------------------------------------------------------*)
  REGLE20
    <R20>=<adja><nom>
    #NOM.MASC:=ADJA.MASC;NOM.FEM:=ADJA.FEM;
    NOM.SING:=ADJA.SING;NOM.PLUR:=ADJA.PLUR#;
  F
  REGLE21
    <R21>=<adja><nomp>
    #NOMP.MASC:=ADJA.MASC;NOMP.FEM:=ADJA.FEM;
    NOMP.SING:=ADJA.SING;NOMP.PLUR:=ADJA.PLUR#;
  F
(*-------------------------------------------------------------*)
(* accord de l'adjectif avec le nom qui precede               *)
(*-------------------------------------------------------------*)
  REGLE30
    <R30>=<nom><adjp>
    #ADJP.MASC:=NOM.MASC;ADJP.FEM:=NOM.FEM;
    ADJP.SING:=NOM.SING;ADJP.PLUR:=NOM.PLUR#;
  F
  REGLE31
    <R31>=<nomp><adjp>
    #ADJP.MASC:=NOMP.MASC;ADJP.FEM:=NOMP.FEM;
    ADJP.SING:=NOMP.SING;ADJP.PLUR:=NOMP.PLUR#;
  F
(*-------------------------------------------------------------*)
(* ACCORD DU NOM ET DE L'ADJECTIF AVEC LE PARTITIF QUI PRECEDE *)
```

```
(*-------------------------------------------------------------*)
  REGLE40
    <R40>=<part><nom>
    #NOM.MASC:=PART.MASC;NOM.SING:=PART.SING;NOM.PLUR:=PART.PLUR#;
  F
  REGLE41
    <R41>=<part><nomp>
    #NOMP.MASC:=PART.MASC;NOMP.SING:=PART.SING;NOMP.PLUR:=PART.PLUR#;
  F
  REGLE42
    <R42>=<part><adja>
    #ADJA.MASC:=PART.MASC;ADJA.SING:=PART.SING;ADJA.PLUR:=PART.PLUR#;
  F
(*-------------------------------------------------------------*)
(* ACCORD EN NOMBRE AVEC UN CARDINAL                          *)
(*-------------------------------------------------------------*)
  REGLE50
    <R50>=<card><nom>
    #NOM.PLUR:=1#;
  F
  REGLE51
    <R51>=<card><adja>
    #ADJA.PLUR:=1#;
  F
(*-------------------------------------------------------------*)
(* CONJUGAISON DU VERBE AVEC LE PRONOM PERSONNEL SUJET ET ACCORD *)
(*-------------------------------------------------------------*)
  REGLE60
    <R60>=<prps>[(<ppo1>/<ppo2>/<ppo3>)]<verbe>
    #VERBE.NR:=PRPS.NR#;
  F
  REGLE61
    <R61>=<prps>[(<ppo1>/<ppo2>/<ppo3>)]<vetre>
    #VETRE.NR:=PRPS.NR#;
  F
  REGLE62
    <R62>=<prps>[(<ppo1>/<ppo2>/<ppo3>)]<avoir>
    #AVOIR.NR:=PRPS.NR#;
  F
  REGLE63
    <R63>=<prps>(<avoir>/<vetre>)<ppas>
    #PRPS.FEM=1 -> PPAS.FEM:=1;PRPS.MASC=1 -> PPAS.MASC:=1#;
  F
  REGLE64
    <R64>=<point><verbe><prps>
    #PRPS.NR:=VERBE.NR#;
  F
  REGLE65
    <R65>=<point><vetre><prps>
    #PRPS.NR:=VETRE.NR#;
  F
  REGLE66
```

```
    <R66>=<point><avoir><prps>
    #PRPS.NR:=AVOIR.NR#;
  F
(*-------------------------------------------------------------*)
(* CHOIX DE L'AUXILLIAIRE ACCOMPAGNANT LE PARTICIPE PASSE     *)
(*-------------------------------------------------------------*)
  REGLE70
    <R70>=<avoir>[<*>]<ppas>
    #PPAS.A:=1#;
  F
  REGLE71
    <R71>=<prps><vetre><ppas>
    #PPAS.A=1 -> PRPS.NR <>3 ->PPAS.N:=1#;
  F
(*-------------------------------------------------------------*)
(* REGLES DE CONSTRUCTION DU VERBE                            *)
(*-------------------------------------------------------------*)
  REGLE80
    <R80>=<avoir><nom>
    #NOM.OPA:=1#;
  F
  REGLE81
    <R81>=<avoir>(<advp>/<pas>)<nom>
    #NOM.OPA:=1#;
  F
  REGLE82
    <R82>=<verbe>(<advp>/<pas>)<nom>
    #NOM.OPV:=1#;
  F
  REGLE83
    <R83>=<vetre>[<advp>]<nom>
    #NOM.A:=1#;
  F
  REGLE84
    <R84>=<ppo3><verbe>
    #VERBE.AN:=1#;
  F
  REGLE85
    <R85>=<ppo1><verbe><*>
    #VERBE.AN=1 OU VERBE.N=1#;
  F
  REGLE86
    <R86>=<ppo2><verbe>
    #VERBE.N:=1#;
  F
  REGLE87
    <R87>=<ppo1><ppo2><verbe>
    #VERBE.NAN:=1#;
  F
  REGLE88
    <R88>=<ppo2><ppo3><verbe>
    #VERBE.NAN:=1#;
F
REGLE89
  <R89>=<ppo2><verbe><a>
  #VERBE.NAI=1 OU VERBE.NAN=1#;
F
REGLE90
  <R90>=<ppo2><verbe><a><INF>
  #VERBE.NAI=1#;
  <INF>=<inf>/<ne>;
F
REGLE91
  <R91>=<ppo2><verbe><a><NOM>
  #VERBE.NAN=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>/<adja>;
F
REGLE92
  <R92>=<ppo2><verbe><au>
  #VERBE.NAN=1#;
F
REGLE93
  <R93>=<ppo2><verbe><de>
  #VERBE.NDI=1 OU VERBE.NDN=1#;
F
REGLE94
  <R94>=<ppo2><verbe><du>
  #VERBE.NDN=1#;
F
REGLE95
  <R95>=<ppo2><verbe><de><INF>
  #VERBE.NDI=1#;
  <INF>=<inf>/<ne>;
F
REGLE96
  <R96>=<ppo2><verbe><de><NOM>
  #VERBE.NDN=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>;
F
REGLE97
  <R97>=<ppo3><verbe><de>
  #VERBE.ANDI=1 OU VERBE.ANDN=1#;
F
REGLE98
  <R98>=<ppo3><verbe><du>
  #VERBE.ANDN=1#;
F
REGLE99
  <R99>=<ppo3><verbe><de><INF>
  #VERBE.ANDI=1#;
  <INF>=<inf>/<ne>;
F
```

```
REGLE100
  <R100>=<ppo3><verbe><de><NOM>
  #VERBE.ANDN=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>;
F
REGLE101
  <R101>=<ppo3><verbe><a>
  #VERBE.ANAI=1#;
F
REGLE102
  <R102>=(<ppo3>/<ppo1>)<verbe><que>
  #VERBE.ANQ=1#;
F
REGLE103
  <R103>=<ppo1><verbe><a>
  #VERBE.NAI=1 OU VERBE.NAN=1 OU VERBE.ANAI=1#;
F
REGLE104
  <R104>=<ppo1><verbe><au>
  #VERBE.NAN=1#;
F
REGLE105
  <R105>=<ppo1><verbe><a><NOM>
  #VERBE.NAN=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>;
F
REGLE106
  <R106>=<ppo1><verbe><a><INF>
  #VERBE.NAI=1 OU VERBE.ANAI=1#;
  <INF>=<inf>/<ne>;
F
REGLE107
  <R107>=<ppo1><verbe><de>
  #VERBE.NDI=1 OU VERBE.NDN=1 OU VERBE.ANDI=1 OU VERBE.ANDN=1#;
F
REGLE108
  <R108>=<ppo1><verbe><du>
  #VERBE.NDN=1 OU VERBE.ANDN=1#;
F
REGLE109
  <R109>=<ppo1><verbe><de><NOM>
  #VERBE.NDN=1 OU VERBE.ANDN=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>;
F
REGLE1100
  <R1100>=<ppo1><verbe><de><INF>
  #VERBE.NDI=1 OU VERBE.ANDI=1#;
  <INF>=<inf>/<ne>;
F
```

```
REGLE1101
  <R1101>=<verbe>[<pas>][<advi>]<adj>
  #VERBE.A=1#;
F
REGLE1102
  <R1102>=[<PPOI>#RIEN#]<verbe>[<pas>][<advp>]<inf>
  #VERBE.I=1#;
  <PPOI>=<ppo1>/<ppo2>/<ppo3>;
F
REGLE1103
  <R1103>=<verbe>[<pas>][<advp>]<que>
  #VERBE.Q=1 OU VERBE.ANQ=1#;
F
REGLE1104
  <R1104>=[<PPOI>#RIEN#]<verbe>[<pas>][<advp>]<de>(<SUITE41>/<SUITE42>);
  <SUITE41>=<inf>
       #VERBE.DI=1#;
  <SUITE42>=<NOM>
       #VERBE.DN#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>/<adja>;
  <PPOI>=<ppo1>/<ppo2>/<ppo3>;
F
REGLE1105
  <R1105>=[<PPOI>#RIEN#]<verbe>[<pas>][<advp>]<du>
       #VERBE.DN#;
  <PPOI>=<ppo1>/<ppo2>/<ppo3>;
F
REGLE1106
  <R1106>=[<PPOI>#RIEN#]<verbe>[<pas>][<advp>]<a>(<SUITE51>/<SUITE52>);
  <SUITE51>=<inf>
       #VERBE.AI=1#;
  <SUITE52>=<NOM>
       #VERBE.AN=1#;
  <SUITE6>=<NOM>
       #VERBE.N=1#;
  <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
        <nom>/<adja>;
  <PPOI>=<ppo1>/<ppo2>/<ppo3>;
F
REGLE1107
  <R1107>=[<PPOI>#VERBE.NAN=1#]<verbe>[<pas>][<advp>]<au>
       #VERBE.AN=1#;
  <PPOI>=<ppo1>/<ppo2>/<ppo3>;
F
(*-------------------------------------------------------------------------*)
(* CONSTRUCTION DE L'INFINITIF                                             *)
(*-------------------------------------------------------------------------*)
  REGLE110
    <R110>=<inf><inf>
    #INF(2).I=1#;
  F
```

```
    REGLE111
      <R111>=<inf>[<advp>]<nom>
      #NOM.OPV:=1#;
    F
(*-------------------------------------------------------------------*)
(* REGLES DE CONSTRUCTION INDUIT PAR LES PRONOMS RELATIFS            *)
(*-------------------------------------------------------------------*)
    REGLE120
      <R120>=<verbe><prre><verbe>/<*><prre><verbe>
      #PRRE.Q=1 -> VERBE.N:=1#;
    F
    REGLE121
      <R121>=[<pind>#PRRE.QI=1 OU PRRE.Q=1 OU PRRE.D=1#]<prre><SUITE>
      #PRRE.QI=1 OU PRRE.D=1#;
      <SUITE>=<ppo1>/<ppo2>/<ppo3>;
    F
    REGLE122
      <R122>=<a><prre><verbe>
      #PRRE.QI=1;VERBE.AN:=1#;
    F
    REGLE123
      <R123>=<de><prre><verbe>
      #PRRE.QI=1;VERBE.DN:=1#;
    F
    REGLE124
      <R124>=<PRON><prre><verbe>
      #VERBE.N:=1#;
      <PRON>=<pdem>/<ppos>/<pind>;
    F
    REGLE125
      <R125>=<prep><prre>
      #PREP.D=1 OU PREP.N=1 OU PREP.NP=1#;
    F
    REGLE126
      <R126>=<prep><prre><*>
      #PRRE.QI=1 OU PRRE.LQ=1#;
    F
(*-------------------------------------------------------------------*)
(* CONSTRUCTION VERBALE A LA FORME COMPOSEE                          *)
(*-------------------------------------------------------------------*)
    REGLE130
      <R130>=<ppas><de><SUITE>
      #PPAS.DI=1#;
      <SUITE>=<ppo1>/<ppo2>/<ppo3>/<inf>/<ne>;
    F
    REGLE131
      <R131>=<ppas><adj><nom>
      #NOM.OPV:=1#;
    F
    REGLE132
      <R132>=<ppas><nom>
      #NOM.OPV:=1#;
    F
    REGLE133
      <R133>=<ppas><ppas>
      #PPAS(2).ET=1 OU PPAS(2).A=1#;
    F
(*-------------------------------------------------------------------*)
(* CONSTRUCTION DE L'ADJECTIF                                        *)
(*-------------------------------------------------------------------*)
    REGLE140
      <R140>=(<ppo1>/<ppo2>)<vetre><adj>
      #ADJ.AN:=1#;
    F
    REGLE141
      <R141>=<adjp><de><INF>
      #ADJP.DI=1#;
      <INF>=<inf>/<ne>;
    F
    REGLE142
      <R142>=<adj><de><INF>
      #ADJ.DI=1#;
      <INF>=<inf>/<ne>;
    F
    REGLE145
      <R145>=<adj><de><NOM>
      #ADJ.DN=1#;
      <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
            <nom>;
    F
    REGLE146
      <R146>=<adj><du>
      #ADJ.DN=1#;
    F
    REGLE147
      <R147>=<adjp><a><INF>
      #ADJP.AI=1#;
      <INF>=<inf>/<ne>;
    F
    REGLE148
      <R148>=<adj><a><INF>
      #ADJ.AI=1#;
      <INF>=<inf>/<ne>;
    F
    REGLE151
      <R151>=<adj><au>
      #ADJ.AN=1#;
    F
    REGLE152
      <R152>=<adj><a><NOM>
      #ADJ.AN=1#;
      <NOM>=<artd>/<arti>/<artp>/<dem>/<card><den>/<num>/<qua>/<qur>/<nomp>/
            <nom>;
    F
```

```
   REGLE154
     <R154>=<adj><que>
     #ADJ.Q=1#;
   F
   REGLE155
     <R155>=<prps>[<*>]<*><adj><*>
     #ADJ.IMP=1 ->PRPS.NR=3#;
   F
(*-------------------------------------------------------------------*)
(* CONSTRUCTION DE LA PREPOSITION                                    *)
(*-------------------------------------------------------------------*)
   REGLE170
     <R170>=<prep><nom>
     #PREP.N=1#;
   F
   REGLE171
     <R171>=<prep><adja>
     #PREP.N=1#;
   F
   REGLE172
     <R172>=<prep><inf>
     #PREP.I=1#;
   F
   REGLE173
     <R173>=<prep><nom>
     #PREP.MASC=1->NOM.MASC:=1;
      PREP.SING=1->NOM.SING:=1;
      PREP.PLUR=1->NOM.PLUR:=1#;
   F
   REGLE174
     <R174>=<prep><adja>
     #PREP.MASC=1->ADJA.MASC:=1;PREP.SING=1->ADJA.SING:=1;
      PREP.PLUR=1->ADJA.PLUR:=1#;
   F
   REGLE175
     <R175>=<de><arti>
     #ARTI.SING:=1#;
   F
   REGLE176
     <R176>=<prep><nomp>
     #PREP.NP=1 OU PREP.NPH=1;NOMP.NPH:=PREP.NPH#;
   F
   REGLE177
     <R177>=(<de>/<a>)<artd><*>
     #ARTD.FEM=1 OU ARTD.NEU=1#;
   F
   REGLE178
     <R178>=<prep><arti>
     #PREP.NARI<>1#;
   F
   REGLE179
     <R179>=<prep><DET>
     #PREP.D=1#;
     <DET>=<arti>/<artd>/<artp>/<dem>/<den>;
   F
   REGLE1701
     <R177>=<prep><artd><*>
     #PREP.DE=1 OU PREP.A=1 -> ARTD.FEM=1 OU ARTD.NEU=1#;
   F
   REGLE1702
     <R175>=<prep><arti>
     #PREP.DE=1 -> ARTI.SING:=1#;
   F
(*-------------------------------------------------------------------*)
(* CONSTRUCTION NOMINALE                                             *)
(*-------------------------------------------------------------------*)
   REGLE180
     <R180>=<nom><nom>
     #NOM(1).A:=1#;
   F
   REGLE181
     <R181>=<nom><de><inf>
     #NOM.DI=1#;
   F
(*-------------------------------------------------------------------*)
   REGLE190
     <R190>=<point><prps>(<ppo1>/<ppo2>/<ppo3>)<inf>
     #REJET#;
   F
   REGLE194
     <R194>=(<nom>/<adjp>)<prre>[(<ppo1>/<ppo2>/<ppo3>)]<verbe><*>
     #PRRE.QI=1 -> VERBE.NR=3 OU VERBE.NR=6#;
   F
   REGLE195
     <R195>=(<nom>/<adjp>)<prre>[(<ppo1>/<ppo2>/<ppo3>)]<vetre><*>
     #PRRE.QI=1 -> VETRE.NR=3 OU VETRE.NR=6#;
   F
   REGLE196
     <R196>=(<nom>/<adjp>)<prre>[(<ppo1>/<ppo2>/<ppo3>)]<avoir><*>
     #PRRE.QI=1 -> AVOIR.NR=3 OU AVOIR.NR=6#;
   F
   REGLE197
     <R197>=<vetre><ppas><par>
     #PPAS.A=1 -> PPAS.N=1#;
   F
   REGLE198
     <R198>=<prre><prps>
     #PRRE.QI=0#;
   F
(*-------------------------------------------------------------------*)
(* ACCORD DU NOM ET DE L'ADJECTIF AVEC LE PARTICLE QUI PRECEDE       *)
(*-------------------------------------------------------------------*)
   REGLE200
     <R200>=<du><nom>
```

```
    #NOM.MASC:=DU.MASC;NOM.SING:=DU.SING;NOM.PLUR:=DU.PLUR#;
  F
  REGLE201
    <R201>=<du><nomp>
    #NOMP.MASC:=DU.MASC;NOMP.SING:=DU.SING;NOMP.PLUR:=DU.PLUR#;
  F
  REGLE202
    <R202>=<du><adja>
    #ADJA.MASC:=DU.MASC;ADJA.SING:=DU.SING;ADJA.PLUR:=DU.PLUR#;
  F
(*-------------------------------------------------------*)
(* ACCORD EN GENRE DE L'ADJECTIF ATTRIBUT                *)
(*-------------------------------------------------------*)
REGLE210
  <R210>=<prps>[<ne>](<vetre>/<verbe>)<adj>
  #PRPS.FEM=1 -> ADJ.FEM:=1;
   PRPS.MASC=1 -> ADJ.MASC:=1#;
F
(*-------------------------------------------------------*)
(* POSSIBILITE DE LA SUCCESSION NOM-NOMP                 *)
(*-------------------------------------------------------*)
REGLE220
  <R220>=<nom><nomp>
  #NOM.NP=1#;
F
(*-------------------------------------------------------*)
(* ACCORD DU VERBE EN DEBUT DE PHRASE                    *)
(*-------------------------------------------------------*)
REGLE300
  <R300>=<point><*>[<adja>]<nom>[<adjp>]<verbe>
  #NOM.SING=1 -> VERBE.NR:=3;
   NOM.PLUR=1 -> VERBE.NR:=6#;
F
REGLE300
  <R300>=<point><*>[<adja>]<nom>[<adjp>]<avoir>
  #NOM.SING=1 -> AVOIR.NR:=3;
   NOM.PLUR=1 -> AVOIR.NR:=6#;
F
REGLE300
  <R300>=<point><*>[<adja>]<nom>[<adjp>]<vetre>
  #NOM.SING=1 -> VETRE.NR:=3;
   NOM.PLUR=1 -> VETRE.NR:=6#;
F
FINREGLES.
```

BIBLIOGRAPHIE

[Andrewsky 73] A. Andreewsky : 'apprentissage, analyse automatique du langage, application à la documentation', Document de linguistique quantitative, 1973.

[Bescherelle 66] 'L'art de conjuguer ', Hatier, 1966

[Bellity 85] D. Bellity, A. Lund, J. Mariani, G. Adda, F. Neel, C. Fluhr : 'Dictée vocale par mots isolés' 14 ième Journées d'Etudes sur la Parole, Paris, juin 85, pp 231

[Bellman 57] R. Bellman : 'Dynamic Programming', Princeton, N.F. :Princeton, Univ. Press, 1957.

[Boyer 84] A. Boyer, 'Etude d'algorithmes de reconnaissance de mots enchainés', Rapport de D.E.A., Université Nancy I, 1984.

[Bridle 82] J.S. Bridle, MD. Brown, R.M. Chamberlain : 'An algorithm for connected word recognition' PROC 1982 IEEE International Conference on Acoustics, Speech and Signal Processing, Paris, France, pp. 899-902, May 1982.

[Carbonell 82] N. Carbonell, J.P. Haton, F. Lonchamp, J.M. Pierrel : 'Elaboration experimentale d'indices prosodiques pour la reconnaissance; Application a l'analyse syntaxico-semantique dans le système Myrtille II', Seminaire 'prosodie et reconnaissance automatique de la parole', Aix, 1982.

[Carbonell 84] N. Carbonell, M.O. Cordier, D. Fohr, J.P. Haton, F. Lonchamp, J.M. Pierrel : 'Acquisition et formalisation du raisonnement dans un système-expert de lecture de spectrogrammes vocaux', Colloque ARC, Orsay 1984.

[Carbonell 84] N. Carbonell, D. Fohr, J.P. Haton, F. Lonchamp, J.M. Pierrel : 'An Expert System for the Automatic Reading of French Spectrograms|', IEEE ICASSP-84, San Diego, March 84.

[Charpillet 85] F. CHARPILLET, J.P HATON, J.M PIERREL : "Un système de reconnaissance de parole continue pour la saisie de textes lus" à paraître au 5ème congrès AFCET R.F.I.A., Grenoble, novembre 1985.

[Charpillet 84] F. CHARPILLET, J.P. HATON, J.M. PIERREL : "Apport de la programmation dynamique en reconnaissance automatique de la parole continue " Congrés AFCET Reconnaissance des Formes et Intelligence Artificielle, Paris, janvier 1984, pp 65.

[Combescure 81] P. Combescure : 'Vingt listes de dix phrases phonétiquement équilibrées', revue d'acoustique, 14, n 56, 1981.

[Chomsky 59] N. Chomsky : 'On certain Formal Properties of Grammars' Information and Control, vol. 2, 1959.

[Chomsky 71] N. Chomsky, 'Aspect de la théorie syntaxique'. Edition du Seuil, Paris, 1971.

[Colmerauer 79] A. Colmerauer : 'Un sous-ensemble intéressant du français', R.A.I.R.O. série rose, vol. 13 n 4, 1979.

[Damestoy 85] J.P. Damestoy, J.P. Haton : 'A Frame Language for the Control of Phonetic Decoding in Continus Speech Recognition', IEEE ICASSP-85, Tampa, Florida, March 85.

[Demori 83] R. De Mori : 'Extraction of Acoustic Cues Using a Grammar of Frames', Speech Communication, n 2-3, 1983.

[Derouault 85] A.M. Derouault: 'Modélisation d'une langue naturelle pour la désambiguation des chaînes phonétiques', Thèse de doctorat es sciences, université de Paris VII, 1985.

[Derouault 83] A.M. DEROUAULT, B. MERIALDO, J.L. STEHLE : 'Estimation des paramètres d'un modèle de Markov du français', T.S.I., volume 2, n[5, septembre-octobre 1983.

[Di Martino 84] J. Di Matino : 'Contribution à la reconnaissance globale de la parole : Mots isolés et mots enchaînés', Thèse de docteur ingénieur en informatique, université de Nancy I, avril 1984.

[Divoux 85] P. Divoux : 'Utilisation de la classification ascendante hiérarchique en reconnaissance de mots isolés multi-locuteurs', à paraître au 5 ième congrès AFCET R.F.I.A., Grenoble, novembre 1985.

[Fillmore 67] J. Fillmore : 'The Case for Case' in 'Universals in Linguistic Theory'. E. Bach, R.T. Harms Editors, Rinehard and Winston INC, April 1967.

[Fluhr 77] 'Algorithmes à apprentissage et traitement automatique des langues' Doctorat es-sciences, université de Paris-sud, 1977.

[Fohr 85] N. Carbonell, J.P. Damestoy, D. Fohr, J.P Haton, F. Lonchamp, J.M. Pierrel : 'Techniques d'intelligence artificielle en décodage acoustico-phonétique' 14 ième Journées d'Etudes sur la Parole, Paris, juin 85, pp 299

[Gross 75] M. Gross, 'Méthode en syntaxe', Hermann,1975.

[Harris 71] Z.R. Harris : 'Structures mathématiques du langage', Dunod, Paris, 1971.

[Haton 74] J.P. Haton : 'Une méthode dynamique de comparaison de chaîne de symboles de longueurs différentes : application à la recherche lexicale', CRAS série A 278, pp 1527-1530.

[Haton 85] J.P. Haton : 'Techniques d'intelligence artificielle en compréhension de la parole et vision : état des recherches' A paraitre dans T.S.I.

[Itakura 75] F. Itakura : 'Minimum Production Residual Principle Applied to Speech Recognition', IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-23, pp. 67-72, February 1975.

[Jelinek 82] F. JELINEK, R.L. MERCER, L.R. BAHL : 'Continuous Speech Recognition: statistical methods' C.S.R. group, IBM T.J. Watson Research Center, Yorktown Heigths NY 10598

[Jelinek 85] Speech Recognition Group (Headed by F. Jelinek) : 'A real time isolated word speech recognition system for dictation transcription' IEEE ICASSP-85, Tampa, Florida, March 85

[Kobayashi 85] Y. Kobayashi, Y.Niimi : 'Matching Algorithms Between a Phonetic Lattice and Two Types of Templates_Lattice and a Graph', IEEE ICASSP-85, Tampa, Florida, March 85

[Laporte 84] M. Laporte, J.M. Pierrel, 'Représentation phonétique et vérification de mots dans le système MYRTILLE II : Etude comparative de deux solutions' Congrès AFCET Reconnaissance Des Formes Et Intelligence Artificielle,Paris, janvier 1984, pp 77.

[Lazrek 83] M. Lazrek : 'Decodage acoustico-phonétique en compréhension automatique de la parole continue', Thèse de 3 ième cycle, Université de Nancy I, 1983.

[Lazrek 84] M. Lazrek, J.P Haton : 'Segmentation et identification des phonèmes dans un système de reconnaissance automatique de la parole continue', Congrès AFCET Reconnaissance Des Formes Et Intelligence Artificielle,Paris, janvier 1984, pp 5.

[Le Goffic 75] P. Le Goffic, N. Combe McBride, 'Les constructions fondamentales du français' Librairies Hachette et Larousse, 1975.

[Levinson 83] S.E. Levinson, L. R. Rabiner, M. M. Sondhi : 'Speaker Independent Isolated Digit Recognition Using Hidden Markov Models', Proc 1983, IEEE International Conference on Acoustics, Speech and Signal Processing, Boston, pp. 1049-1052, April 1983.

[Mari 79] J.F. Mari, 'Contribution à l'analyse syntaxique et à la recherche lexicale en reconnaissance du discours continu', Thèse de 3ème cycle, Universitè de Nancy I, 1979.

[Mariani 81] J. Mariani : 'ESOPE : Un Système de comphréhension de la parole continue' Thèse de doctorai ès Sciences

[Myers 81] C. MYERS, L.R. RABINER : 'A Level Building Dynamic Time Warping Algorithm for Connected Word Recognition' IEEE Transactions on Acoustics, Speech, and Signal Processing, vol. ASSP_29, n[2, April 1981.

[Minsky 75] M. Minsky, 'A Framework for Representing Knowledge', in 'The Psychology of Computer Vision' (P.H. Winston,ed.) pp 211-277, McGraw Hill,1975.

[Nakagawa 83] S. Nakagawa : 'A Connected spoken word Recognition Method by o(n) Dynamic Programming Pattern Matching Algorithm', Proc 1983, IEEE International Conference on Acoustics, Speech and Signal Processing, Boston, pp. 296-299, April 1983.

[Perennou 80] G. Perennou, A. Fatholahzadeh, M. de CALMES : 'Le filtrage syntaxique pour l'entrée vocale de texte' Syntax and Semantics in speech understanding, Paimpont, sept 80

[Perennou 82] G. Perennou, G. Caelen : 'Utilisation de la prosodie pour la reconnaissance de la parole continue dictée', Seminaire 'prosodie et reconnaissance automatique de la parole', Aix, 1982.

[Pierrel 75] J.M. Pierrel : 'Contribution à la compréhension automatique du discours continu'. Thèse de 3ème cycle, Université de Nancy I, 1975.

[Pierrel 81] J.M. Pierrel : 'Etude et mise en oeuvre de contraintes liguistiques en compréhension automatique du discours continu'. Thèse d'état, Université de Nancy I, 1981.

[Pister 84] C. Pister : ' Adaptation au locuteur par apprentissage automatique : application à un système de reconnaissance automatique de la parole', Doctorat de 3 ième cycle en informatique, Université de Nancy I, 1984.

[Ruske 82] G. Ruske, T. Schotola : ' The Efficiency of Demisyllable Segmentation in the Recognition of Spoken Words', Automatic Speech Analysis and Recognition, NATO advanced study institutes series, Bonas, France, June 1981.

[Sakoe 71] H. Sakoe, S. Chiba : 'A Dynamic Programming Approach to Continuous Speech Recognition' in Proc.Int.Congr.Acoust., Budapest, Hungary, 1971.

[Woods 70] W.A Woods : 'Transition Network Grammars for Natural Language Analysis', C.A.C.M., vol. 13, n 10, october 1970, pp 591-602.

[Sakoe 78] H. Sakoe, S. Chiba : 'Dynamic Programming Algorithm Optimisation for Spoken Word Recognition', IEEE Transactions on Acoustics, Speech, and Signal Processing, February 1978.

[Salkov 73] M. Salkoff : 'Une grammaire en chaîne du français' Dunod, éditeur. Paris, monographies de linguistique mathématique, 1973.

[Sauter 84] L. Sauter : 'RAPACE : un système de reconnaissance analytique de parole continue', Congrès AFCET Reconnaissance Des Formes Et Intelligence Artificielle, Paris, janvier 1984, pp 89.

[Schank 75] C. Schank : 'Conceptuel Information Processing' The Bell System Technical Journal, vol. 54, n 1, january, 1975.

[Schank 77] C. Schank, P. Abelson, 'Scripts, Plans, Goals and Understanding', Lawrence Erlbaum Ass., 1977.

[Vives 85] R. Vives, 'Mise en correspondance temporelle de descriptions phonologique et prosodique de mots dans le sytème de reconnaissance de la parole KEAL', 14 ième Journées d'Etudes sur la Parole, Paris, juin 85, pp 253.

[Zue 81] V.W. Zue : 'Acoustic-phonetic Knowledge Representation : Implications from Spectrogram Reading Experiments', Automatic Speech Analysis and Recognition, NATO advanced study institutes series, Bonas, France, June 1981.

NOM DE L'ETUDIANT : CHARPILLET François

NATURE DE LA THESE : Doctorat de l'Université de Nancy I en Informatique.

VU, APPROUVE ET PERMIS D'IMPRIMER

NANCY, le 29 AOUT 1985

LE PRESIDENT DE L'UNIVERSITE DE NANCY I

RESUME

Dans le cadre de recherches plus générales sur le traitement automatique de la parole, nous étudions une application au domaine de la saisie orale de textes par ordinateur.

Ce travail comporte à la fois une partie fondamentale et une partie d'implantation pratique d'un système de dictée. La réalisation inclut, en plus du système de dictée vocale lui-même, la possibilité d'activer à la voix les fonctions d'édition et des moyens graphiques pour corriger les erreurs éventuelles. Le système réalisé autorise l'utilisation d'un vocabulaire de l'ordre de 1000 racines de mots.

Les traitements que nous avons mis en oeuvre sont spécifiques à la dictée automatique. Il est notamment difficile de recourir à des informations linguistiques car le langage permis pour la dictée se veut par définition le plus général possible. C'est pourquoi l'indéterminisme propre au traitement de la parole est plus délicat à résoudre. La reconnaissance ne peut utiliser aucune connaissance pragmatique. Un autre aspect particulier à un système de dictée vocale est la procédure de dialogue dont l'objectif consiste à valider les résultats de la reconnaissance qui peuvent être multiples et entachés d'erreurs. Cette procédure de dialogue peut être réalisée dans notre système à l'aide d'un système de désignation de type souris ou par un dialogue oral utilisant un langage de commande artificiel limité. Nous avons donc dû nous intéresser à la fois au problème de la reconnaissance vocale dans le cadre d'un langage naturel (le texte lu) et d'un langage artificiel (le langage de commande). Nous avons été confrontés à divers problèmes liés au langage traité comme:

- la création et la gestion d'un vocabulaire important,
- la définition d'un modèle syntaxique suffisamment général pour accepter un français courant,
- le choix de stratégies performantes.

Une grande partie de notre étude a été consacrée à l'extention d'algorithmes de programmation dynamique utilisés par ailleurs avec succès pour la reconnaissance de mots enchainés. Les algorithmes proposés dans ce travail mettent à profit des contraintes syntaxiques. Elles sont soit locales pour le langage de saisie de textes lus (triplets syntaxiques), soit globales pour le langage de commande (grammaire hors contexte).

Pour le mode édition de textes lus nous avons élaboré une extension de l'algorithme ''Level Building'' de Myers et Rabiner. Une analyse en faisceaux résoud plusieurs problèmes propres à la reconnaissance analytique comme la prédominance des mots courts et l'intégration de contraintes syntaxiques.