Département de formation doctorale en informatique **École doctorale IAE + M**
UFR STMIA

Modélisation du langage pour les systèmes de reconnaissance de la parole destinés aux grands vocabulaires : application à MAUD

THÈSE

présentée et soutenue publiquement le 31 mars 2000

pour l'obtention du

Doctorat de l'université Henri Poincaré – Nancy 1
(spécialité informatique)

par

Imed ZITOUNI

Composition du jury

Président :	René Schott Professeur Université Henri Poincaré-Nancy 1
Rapporteurs :	Renato De Mori Professeur Université d'Avignon Michèle Jardino Chargée de recherche LIMSI-CNRS
Examinateurs :	Jean-Paul Haton Professeur Institut Universitaire de France Jean-François Mari Professeur Université de Nancy 2 Kamel Smaïli Maître de conférences Université de Nancy 2

Laboratoire Lorrain de Recherche en Informatique et ses Applications — UMR 7503

Résumé

Le traitement automatique de la parole suscite actuellement un grand intérêt ; il est considéré comme une branche importante de l'interaction homme-machine. En effet, nous éprouvons le besoin de communiquer avec nos ordinateurs, de la façon la plus naturelle et la plus directe qui soit : le langage parlé ; l'interaction et l'échange d'informations s'en trouvent grandement facilités. Le marché des logiciels offre aujourd'hui des produits qui prétendent effectuer une reconnaissance de la parole continue avec un vocabulaire important. En réalité, les performances de ces systèmes sont encore largement inférieures à celles de l'être humain, particulièrement au niveau de la modélisation du langage.

Le travail que nous présentons dans ce manuscrit s'inscrit dans le cadre de la modélisation du langage pour les systèmes de reconnaissance de la parole continue destinés aux grands vocabulaires. Nous proposons de nouveaux modèles fondés sur des séquences de mots de longueur variable. Ces séquences représentent des structures langagières qui s'apparentent à des syntagmes linguistiques. Elles sont détectées automatiquement, à partir d'importants corpus de textes, en utilisant des mesures issues de la théorie de l'information. Nous proposons également une approche hybride combinant les modèles de langage probabilistes, utilisés dans la plupart des systèmes de reconnaissance actuels, avec des connaissances linguistiques explicites supplémentaires.

L'évaluation de l'ensemble de ces modèles est effectuée en terme de perplexité et en terme de prédiction à l'aide du jeu de Shannon. Pour tester leurs performances au niveau de la reconnaissance, nous avons développé un système de reconnaissance vocale nommé MAUD : Machine AUtomatique à Dicter ; il se fonde sur les modèles de Markov cachés de second ordre et utilise un vocabulaire de 20000 mots. Par rapport à la version de base de ce système utilisant un modèle trigrammes, l'intégration de ces modèles de langage a amélioré le taux de reconnaissance d'environ 22%.

Mots-clés: reconnaissance de la parole, modèles de langage, n-grammes, n-classes, grammaire d'unification, modèle hybride, multigrammes, multiclasses, modèle hiérarchique, modèle à base de séquences, MAUD.

Abstract

Automatic speech recognition currently arouses a great interest: it can be considered as a significant field of the man-machine interaction. In fact, we need to communicate with our computers using the most natural and direct way: the spoken language; the interaction and the exchange of information are then largely facilitated. At present, several softwares claim to carry out a continuous speech recognition for a large vocabulary. The performances of these systems are still largely lower than those of the human being, particularly on language modeling.

The work we propose is registered in the framework of language modeling for large vocabularies continuous speech recognition systems. We present in this manuscript new models based on variable length word sequences. These sequences represent linguistic structures or more exactly syntagms. They are automatically detected, on significant text corpora, by using information theory measures. We also propose a hybrid approach combining probabilistic language models, used by the majority of current speech recognition systems, with additional explicit linguistic knowledge.

The evaluation of these models is carried out in terms of perplexity and in terms of word prediction with the Shannon game. To test their performances on recognition, we developed a large vocabulary dictation machine named MAUD; it is based on second order hidden Markov models and uses a vocabulary of 20000 words. Compared to the basic version of MAUD using a trigram model, the integration of these language models improved the recognition rate by 22% approximately.

Keywords: speech recognition, language models, n-gram, n-class, unification grammar, hybrid model, multigram, multiclass, hierarchical model, model based on sequences, MAUD.

Remerciements

Je tiens tout d'abord à manifester ma gratitude et ma reconnaissance envers Jean-Paul HATON qui m'a accueilli, accordé sa confiance et m'a permis d'accomplir ce travail dans les meilleures conditions.

Mes vifs remerciements vont à Kamel SMAILI pour avoir été présent aux instants importants de prise de décision ainsi que pour les discussions riches et vives que nous avons eues durant ces trois dernières années. Je le remercie aussi pour ses remarques et critiques qui ont contribué à l'élaboration de ce manuscrit.

Je tiens également à remercier Jean-François MARI pour ses nombreux et précieux conseils ainsi que pour son aide concernant la mise au point de la nouvelle version du système de dictée vocale MAUD.

Mes remerciements vont ensuite à mes rapporteurs Michèle JARDINO, Renato DE MORI et René SCHOTT qui ont bien voulu accepter d'évaluer le présent travail et ce malgré toutes les responsabilités qu'ils assument.

Un gros merci à tout le groupe de recherche RFIA du LORIA, en particulier l'équipe PAROLE, au sein duquel ce travail fut effectué. Leur compétence et qualité humaine ont grandement facilité mon travail.

Un merci particulier à Alain VOINIER et Armelle BRUN pour avoir accepté de lire et corriger ce document ; leurs conseils et remarques ne furent point perdus.

Je ne peux passer sous silence le soutien moral de Barbara durant ces dernières années.

Finalement, les derniers mais non les moindres, j'exprime ma profonde gratitude à ma grand-mère Zohra, mes parents, mon frère Issam, mes proches et mes amis pour tout ce qu'ils ont fait pour moi.

Je dédie cette thèse à mes grands parents,
à mes parents, à Barbara, à Issam
et à tous ceux qui vont la lire.

Table des matières

Table des figures **xiii**

Liste des tableaux **xv**

Introduction générale **1**

1 La reconnaissance vocale **5**

1.1 La communication orale 6

1.1.1 Le son de la parole 6

1.1.2 Le mot 6

1.1.3 La phrase 7

1.1.4 La chaîne de communication orale 7

1.2 La reconnaissance vocale 7

1.2.1 Les difficultés liées au signal de parole 8

1.2.2 Historique 9

1.3 L'approche probabiliste en RAP 10

1.4 Extraction des paramètres 12

1.5 La modélisation acoustique à base de HMMs 13

1.5.1 Définition des chaînes de Markov 13

1.5.2 Application à la RAP 14

1.6 L'approche phonétique 15

1.7 Les modèles de langage 17

1.8 Les algorithmes de reconnaissance 17

1.8.1 L'algorithme A^* 18

1.8.2 L'algorithme Viterbi synchrone 19

1.8.3 L'algorithme de Viterbi-Bloc 22

1.9 Conclusion 23

2 Quelques notions en théorie de l'information **25**

2.1 Entropie 25

2.1.1 Calcul de l'entropie 26
2.1.2 L'entropie en RAP 27
2.1.3 Autres visions de l'entropie 28
2.2 Information mutuelle 29
2.3 Mesure de la qualité d'un modèle de langage 29
2.3.1 Perplexité 30
2.3.2 Jeu de Shannon 31
2.4 Conclusion 31

3 Les modèles de langage probabilistes 33
3.1 Introduction 33
3.2 Quelques domaines d'application 33
3.3 Avantage d'un modèle de langage probabiliste 34
3.4 Quelques modèles de langage probabilistes 35
3.4.1 Les modèles n-grammes 35
3.4.2 Les modèles n-classes 36
3.4.3 Les modèles POS 37
3.4.4 Les modèles fondés sur la morphologie 38
3.4.5 Les modèles n-grammes dynamiques 39
3.4.6 Les modèles n-grammes distants 39
3.4.7 Les modèles cache et trigger 40
3.4.8 Les modèles utilisant le principe du maximum d'entropie 41
3.4.9 Les modèles à base de grammaires 43
3.4.10 Les modèles à analyse sémantique latente 43
3.4.11 Les modèles à base de séquences 44
3.5 Estimation des données manquantes 45
3.6 La théorie d'estimation d'un modèle optimal 47
3.6.1 Estimation par maximum de vraisemblance 49
3.6.2 Estimation bayésienne 50
3.6.3 Estimation de validation croisée 50
3.7 Conclusion 51

4 Une approche hybride pour la RAP 53
4.1 Introduction 53
4.2 Fonctionnement général 53
4.3 Les classes syntaxiques 54
4.4 Étiquetage du corpus 57

4.5 Le modèle de langage formel 59
4.5.1 Définition des grammaires formelles 60
4.5.2 Les grammaires à traits 63
4.5.3 Les traits utilisés 64
4.5.4 Les grammaires d'unification 65
4.5.5 Les réseaux de transition augmentés 68
4.5.6 Technique d'analyse 71
4.5.7 Fonctionnement du modèle formel 76
4.6 Le corpus 76
4.7 Évaluation 77
4.8 Conclusion 77

5 Modèles de langage multiclasses et hiérarchique 79
5.1 Introduction 79
5.2 Le modèle multigrammes 79
5.2.1 Estimation du modèle 81
5.2.2 Conclusion sur l'approche multigrammes 84
5.3 Le modèle multiclasses 84
5.3.1 Distribution a priori des multiclasses 87
5.3.2 Estimation au moyen de l'algorithme Forward-Backward 87
5.3.3 Estimation au moyen de l'algorithme de Viterbi 88
5.3.4 Conclusion sur les multiclasses 89
5.4 Le modèle hiérarchique 90
5.4.1 Formulation du modèle 91
5.4.2 Estimation des paramètres du modèle hiérarchique 92
5.4.3 Exemple 93
5.4.4 Distribution a priori des multiclasses 94
5.5 Évaluation 94
5.5.1 Le modèle n-classes 95
5.5.2 Le modèle multiclasses 96
5.5.3 Le modèle hiérarchique 98
5.5.4 Bilan comparatif 100
5.6 Conclusion 100

6 Modèles de langage à base de séquences 103
6.1 Introduction 103
6.2 Intérêt des séquences en RAP 103

6.3 Extraction des séquences de mots : l'état de l'art 104
6.4 Une nouvelle méthode d'extraction de séquences à base de classes syntaxiques . 105
6.4.1 Proposition d'un algorithme 106
6.4.2 Estimation du modèle de langage et calcul de la perplexité 108
6.4.3 Accélération du calcul de la perplexité 109
6.4.4 Affinement des séquences 110
6.4.5 Évaluation et résultats 111
6.5 Une méthode d'extraction de séquences sans classes 112
6.6 Les modèles n-SeqGrammes 114
6.6.1 Estimation des paramètres 115
6.6.2 Les modèles construits 118
6.7 Les modèles n-SeqClasses 118
6.8 Le modèle SeqCache 120
6.9 Le modèle SeqTrigger 121
6.10 Évaluation 123
6.11 Conclusion 126

7 Évaluation avec le jeu de Shannon 127

7.1 Introduction 127
7.2 Le jeu de Shannon 127
7.3 Le jeu de hasard 128
7.3.1 Limitation de la liste des hypothèses 128
7.3.2 Une alternative au calcul de la perplexité 129
7.4 Évaluation des modèles de langage 129
7.4.1 Les modèles conventionnels 130
7.4.2 Les modèles à base de séquences 133
7.4.3 L'approche hybride 137
7.5 Conclusion 138

8 Le système MAUD 139

8.1 Introduction 139
8.2 Les systèmes ayant participé à la campagne B1 de l'Aupelf-UREF 139
8.2.1 Le système de l'INRS-Télécommunication de Montréal 139
8.2.2 Le système du LIMSI-Orsay 140
8.2.3 Le système du CRIM pour le français 140
8.3 Le système MAUD 140
8.3.1 Construction d'un treillis de mots 141

8.3.2 Construction des N meilleures phrases 141
8.3.3 Filtrage de phrases 141
8.3.4 Liaison et phonologie 142
8.3.5 Algorithme de Viterbi-Bloc 142
8.3.6 Application de Viterbi-Bloc à la recherche lexicale 144
8.4 Intégration des modèles de langage 145
8.4.1 Les versions conventionnelles 145
8.4.2 Les versions à base de séquences 146
8.4.3 Les versions utilisant l'approche hybride 147
8.4.4 Les versions se servant des modèles multiclasses et hiérarchique 148
8.5 La nouvelle version du système MAUD 150
8.6 Conclusion 151

Conclusion **153**

Perspectives **155**

Annexes **159**

A Les classes syntaxiques **159**

B Les règles grammaticales **165**

C Estimation Forward-Backward des multiclasses **173**

Publications personnelles **175**

Bibliographie **177**

Table des figures

1.1 **La chaîne de communication orale** : psychologie du locuteur, sémantique, règles du discours, syntaxe, lexique, prosodie, phonétique, bruit de l'environnement. 8
1.2 Signal vocal temporel (b) et spectrogramme vocal (c) de la phrase "Voulez vous lui passer le beurre". 9
1.3 L'approche probabiliste de la RAP pour l'hypothèse "*Voulez vous lui passer le beurre*" 11
1.4 Exemple d'un HMM à trois états caractérisé par une distribution de probabilités pour chaque état associé à une observation et par des probabilités de transition entre les états. 14
1.5 HMM d'un mot obtenu par la concaténation de HMMs de phonèmes 16
1.6 HMM d'une phrase obtenu par concaténation de HMMs de mots 17
1.7 L'algorithme de reconnaissance de parole à une-passe (*One-Pass*) 20
1.8 Les chemins possibles de l'algorithme à une-passe (*One-Pass*) 20
1.9 Graphe représentant un mini-lexique. 22

2.1 Représentation de la fonction $f : x \rightarrow -(x.log_2x + (1-x).log_2(1-x))$ 28
2.2 Représentation de la fonction $f : (x,y) \rightarrow -(y.log_2x + (1-y).log_2(1-x))$ 28

4.1 Exemple de construction de graphe de la phrase : "*Elle porte des chaussures de luxe*" 58
4.2 Arbre de dérivation de la phrase : "*il mange une pomme*". 62
4.3 Exemple de réseaux de transition. 69
4.4 Exemple de réseaux de transition simplifié. 69
4.5 Exemple de réseau de transition augmenté. 70
4.6 Exemple de réseau de transition augmenté. 71
4.7 Exemple de réseaux de transition augmentés. 73

5.1 Le modèle hiérarchique appliqué sur la phrase : "*la pomme que j'ai mangée*". . . . 91
5.2 Performance, en terme de perplexité du modèle multiclasses, utilisant les méthodes de réestimation de l'équation 5.21 ($PP(MC)$) et de l'équation 5.22 ($PP^*(MC)$), pour différentes valeurs de n et de C_0, où n est le nombre maximal de classes dans une multiclasse et C_0 le nombre d'occurrences minimal d'une multiclasse. 97
5.3 Performance, en terme de perplexité du modèle multiclasses, utilisant les méthodes de réestimation de l'équation 5.21 ($PP(MC)$) et de l'équation 5.22 ($PP^*(MC)$), pour différentes valeurs de n, où n est le nombre maximal de classes dans une multiclasse. 98

5.4 Performance, en terme de perplexité du modèle hiérarchique, utilisant les méthodes de réestimation de l'équation 5.37 ($PP(MCnv)$) et de l'équation 5.38 ($PP^*(MCnv)$), pour différentes valeurs de n et de ν, où n est la taille maximale d'une multiclasse et ν le niveau maximal de la hiérarchie. 99

6.1 Temps de calcul de l'algorithme d'extraction des séquences de mots pour différentes valeurs de p. 108
6.2 Convergence des algorithmes d'extraction de séquences pour différentes valeurs de q (nombre maximal de mots dans une séquence). 111
6.3 Performances en terme de perplexité pour différents nombres de triggers retenus. 123
6.4 Performances en terme de perplexité, des deux modèles de langage des équations 6.49 et 6.50, pour différentes valeurs de a. 125

8.1 Architecture générale du système MAUD. 141
8.2 Schéma de principe de l'algorithme de Viterbi-Bloc. 143
8.3 Construction du treillis de mots. 144

Liste des tableaux

1.1 Mots d'un mini-lexique. 22

4.1 Exemple d'une grammaire à contexte libre. 61
4.2 Exemple d'une grammaire sensibles au contexte. 62
4.3 Trace d'analyse de la phrase "*un grand garçon mange*" 74
4.4 Trace d'analyse de la phrase "*un sportif travaille*" avec l'algorithme $A2$. 75

5.1 Formule de récurrence de la variable *forward* α. 88
5.2 Formule de récurrence de la variable *backward* β. 88
5.3 Un échantillon de multiclasses avec quelques séquences de mots correspondantes. 89
5.4 Un échantillon de multiclasses non conventionnel. 90
5.5 Les paramètres d'interpolation et la perplexité (PP) des modèles triclasses et biclasses. 96
5.6 Les performances en terme de perplexité des modèles multiclasses (PP_{MC}) et hiérarchique ($PP_{MC_n^4}$) pour différentes valeur de n, où n représente la taille maximale d'une multiclasse. 100

6.1 Un échantillon des classes extraites et de séquences correspondantes. 113
6.2 Un échantillon de séquences de mots. 114
6.3 Un échantillon de couple de séquences fortement corrélées ($s_i \rightarrow s_j$). 123
6.4 Evaluation en terme de perplexité de plusieurs variantes de modèles de langage. 124

7.1 Résultats de prédiction du modèle trigrammes. 130
7.2 Résultats de prédiction du modèle triclasses. 132
7.3 Résultats de prédiction du modèle trigrammes avec cache et trigger. 132
7.4 Résultats de prédiction d'une interpolation linéaire entre les modèles trigrammes et triclasses. 133
7.5 Résultats de prédiction du modèle 3-SeqGrammes. 134
7.6 Résultats de prédiction du modèle 3-SeqClasses. 135
7.7 Résultats de prédiction d'une interpolation linéaire entre les modèles 3-SeqGrammes, SeqCache et SeqTrigger. 136
7.8 Résultats de prédiction d'une interpolation linéaire entre les modèles 3-SeqGrammes et 3-SeqClasses. 136
7.9 Résultats de prédiction de l'approche hybride, en utilisant une interpolation linéaire entre les modèles trigrammes, cache et trigger. 137
7.10 Résultats de prédiction de l'approche hybride, en utilisant une interpolation linéaire entre les modèles 3-SeqGrammes, SeqCache et SeqTrigger. 137

8.1 Résultats d'évaluation des versions conventionnelles de MAUD. 146
8.2 Résultats d'évaluation des versions se servant des séquences. 147
8.3 Résultats d'évaluation des versions utilisant l'approche hybride. 148
8.4 Résultats d'évaluation des versions employant les modèles multiclasses ou hiérarchique. 149
8.5 Résultats d'évaluation des nouvelles versions de MAUD. 150

Introduction générale

Le traitement automatique de la parole suscite actuellement un grand intérêt ; il peut être considéré comme une branche importante de l'interaction homme-machine. Nous éprouvons le besoin de communiquer avec nos ordinateurs, de la façon la plus naturelle et la plus directe qui soit : le langage parlé ; l'interaction et l'échange d'informations s'en trouvent grandement facilités. Nous cherchons à rendre ces machines accessibles par la voix au téléphone ou au microphone, pour délivrer et recueillir de l'information sans clavier ni écran de visualisation. Ces techniques d'entrées/sorties vocales sont également d'un intérêt évident pour : les applications d'aide aux handicapés, la reconnaissance de cris de détresse, la prise de commandes avec mains libres dans la conduite de l'automobile, etc.

Comprendre un énoncé, c'est d'abord l'entendre, puis le reconnaître et enfin l'interpréter. L'étape de reconnaissance est la phase que les systèmes de reconnaissance automatique de la parole cherchent à maîtriser. Celle-ci vise à transformer le signal acoustique en une séquence d'unités linguistiques.

Des progrès significatifs ont été réalisés ces vingts dernières années dans ce domaine. Le marché des logiciels offre aujourd'hui des produits qui prétendent effectuer une reconnaissance de la parole continue pour un vocabulaire important. En réalité, les performances de ces systèmes sont encore largement inférieures à celles de l'être humain, particulièrement au niveau de la modélisation du langage.

Un modèle de langage cherche à identifier, adapter et exploiter certaines caractéristiques du langage naturel. Pour communiquer dans une langue, un individu utilise implicitement et inconsciemment un ensemble de connaissances lui permettant de la modéliser. De même, chaque application informatique devant traiter un langage naturel, sans connaissance complète, peut profiter pleinement de cette modélisation.

Le travail que nous présentons s'inscrit dans le cadre de la modélisation du langage pour les systèmes de reconnaissance de la parole continue destinés aux grands vocabulaires. Nous avons développé de nouveaux modèles fondés sur des séquences de mots de longueur variable. Ces séquences représentent des structures langagières qui s'apparentent à des syntagmes linguistiques. Elles sont détectées automatiquement, sur d'importants corpus de textes, à l'aide de mesures issues de la théorie de l'information. Nous proposons également une approche hybride combinant les modèles de langage probabilistes, utilisés par la plupart des systèmes de reconnaissance actuels, avec des connaissances linguistiques explicites supplémentaires.

Ce mémoire se compose de trois parties : la première partie (chapitres 1, 2 et 3) tend à faire l'état de la recherche dans ce domaine. La deuxième partie (chapitres 4, 5 et 6) présente notre contribution à la modélisation du langage. La troisième partie (chapitres 7 et 8) se propose d'évaluer les performances de nos modèles de langage et de les comparer à celles d'autres modèles plus classiques. Nous présentons également dans cette dernière partie notre système de reconnaissance de la parole MAUD : Machine AUtomatique à Dicter.

Dans le premier chapitre, nous décrivons d'abord le moyen le plus naturel de communication, à savoir la parole. Nous analysons ainsi chacune de ses composantes et nous montrons comment établir, à partir de celles-ci, une chaîne de communication parlée entre deux interlocuteurs. Ensuite, sont abordés les difficultés liées à la reconnaissance vocale et les efforts réalisés dans ce sens depuis les années cinquante. Enfin, nous présentons l'architecture générale d'un système probabiliste de reconnaissance de la parole ainsi que les divers éléments qui le composent. La démarche se limite à l'approche la plus utilisée actuellement, celle des modèles de Markov cachés.

L'objet du deuxième chapitre est d'évoquer des notions de base, en théorie de l'information, utiles au développement des modèles de langage. Ainsi, nous abordons certaines propriétés (comme l'entropie et l'information mutuelle) relatives à la quantité d'information délivrée par une source, telle que la langue, et par les modèles de langage qui peuvent la modéliser. Ce chapitre présente également des méthodes permettant d'évaluer les performances des modèles de langage : la perplexité et le jeu de Shannon.

Le troisième chapitre est consacré aux modèles de langage ; nous y évoquons des domaines d'application de ces modèles ainsi que leurs avantages et inconvénients respectifs. Les nombreuses variantes présentes dans la littérature ne permettent pas un énoncé exhaustif. Nous ne livrons ainsi qu'un aperçu des modèles de langage probabilistes fréquemment employés en reconnaissance vocale. Enfin, nous exposons les méthodes les plus utilisées pour l'estimation de ces modèles et pour l'approximation de la vraisemblance des événements non rencontrés.

Le quatrième chapitre rend compte d'une approche hybride de reconnaissance, combinant les modèles de langage probabilistes à des connaissances langagières explicites. Ces connaissances sont modélisées par une grammaire d'unification, appartenant à la famille des grammaires à traits, construisant ainsi le modèle formel. Le formalisme des réseaux de transition augmentés correspond à la notion d'automate à pile ; il est, de ce fait, assez simple à mettre en oeuvre sur le plan informatique. Ces réseaux ont été choisis pour implanter notre grammaire. Préalablement, nous exposons la classification et la méthode d'étiquetage pour lesquelles nous avons opté. En effet, les connaissances linguistiques utilisées reposent sur des classes syntaxiques de la langue française.

Le cinquième chapitre s'intéresse aux modèles multiclasses et hiérarchiques. Ces deux modèles présentent la particularité de pouvoir extraire des séquences de classes, supposées capables de traduire la structure syntaxique de la phrase prononcée. La différence entre ces deux modèles réside dans leurs manières de traiter la dépendance entre ces séquences. Nous définissons alors ces deux modèles de langage. La théorie à l'origine de leur fonctionnement et la méthode utilisée pour leur estimation sont abordées par la suite. La perplexité, présentée dans le chapitre 2, est mise à profit afin de signifier leurs performances et de les situer par rapport à des modèles de langage classiques.

Le sixième chapitre permet de présenter une nouvelle approche de modélisation de langage se fondant sur la notion de séquences. Nous évoquons alors différents modèles de langage se servant de ces séquences, extraites automatiquement selon un critère d'optimisation bien connu en théorie de l'information. La méthode d'extraction de ces séquences est ensuite décrite d'une façon détaillée. Une comparaison de l'ensemble de ces modèles ainsi que d'autres modèles plus classiques, en terme de perplexité, est présentée.

Nous présentons dans le septième chapitre une autre méthode d'évaluation des modèles de langage, inspirée du Jeu de Shannon. Celle-ci est utilisée pour évaluer les performances de nos modèles et les comparer à celles d'autres modèles plus classiques.

Le huitième chapitre décrit le système de reconnaissance de la parole continue, MAUD. Ce système, destiné à la reconnaissance des grands vocabulaires, se fonde sur les modèles de Markov cachés de second ordre. Nous présentons également les résultats d'intégration des différents modèles de langage cités dans cette étude ainsi que leurs apports au système. Nous exposons ensuite la nouvelle version du système MAUD, intégrant les différentes approches proposées.

Au terme d'une conclusion, nous esquissons nos perspectives de recherche.

Chapitre 1

La reconnaissance vocale

La parole est considérée, depuis la nuit des temps, comme étant le moyen le plus simple, le plus naturel et le plus pratique pour communiquer. Ainsi, une des formes de communication les plus naturelles avec une machine est sans doute la parole. D'où les recherches menées depuis longtemps pour reconnaître automatiquement, à l'aide d'une machine, ce mode de communication. Avant d'aborder le problème de la reconnaissance, essayons tout d'abord de définir abstraitement le phénomène de la parole.

La parole est avant tout un signal acoustique plongé dans un espace de communication linguistique. La communication parlée n'est donc possible qu'entre deux locuteurs ayant un même espace de communication linguistique, formés à coder et à décoder un sens au moyen de la même clé. En effet, un message émis d'une source, n'est compris que si le récepteur est en possession de la clé de l'espace de communication, ce qui est compatible avec le modèle de communication de Shannon défini dans [82]. Ce message est bien sûr continu et les effets contextuels d'un son élémentaire sur ses voisins sont considérables. Le découpage du message en unités (phonèmes, mots, contexte) est effectué par le récepteur à la suite d'un processus linguistique complexe.

Le signal vocal est soumis à plusieurs contraintes liées à la source, ce qui le rend porteur d'informations pertinentes : suite d'unités régie par des règles de morphologie et de syntaxe. Ce signal acoustique de parole a également la particularité d'être redondant. En effet, il transporte beaucoup plus d'information que nécessaire, ce qui, par ailleurs, le rend très résistant à quelques bruits parasites. Ces bruits peuvent parfois faire obstacle à la communication, la rendant ainsi difficile ou même impossible. En plus, ce signal est éminemment variable d'un locuteur à l'autre mais également pour un locuteur donné (état émotif, fatigue, etc.). Ainsi, il est nécessaire d'arriver à extraire l'information pertinente sans trop la dégrader.

Nous présentons dans le paragraphe suivant ce moyen de communication, en analysant chacune de ses composantes, et nous décrivons comment s'établit une chaîne de communication parlée entre deux interlocuteurs. Nous exposons ensuite, dans le paragraphe 1.2, les difficultés liées à la reconnaissance vocale et nous abordons les efforts réalisés dans ce domaine depuis les années cinquante. Ensuite, nous montrons l'approche probabiliste et surtout celle à base des modèles de Markov cachés (nous utilisons plutôt l'acronyme anglais HMMs, *Hidden Markov Models*, pour mentionner modèles de Markov cachés), qui est la plus utilisée de nos jours dans le domaine de la reconnaissance automatique de la parole. Enfin, nous décrivons brièvement les différentes composantes utiles à la réalisation d'un système de reconnaissance à base de HMMs : extraction des paramètres, approche phonétique, modélisation de langage et algorithmes de reconnaissance.

1.1 La communication orale

La communication orale est depuis longtemps le centre d'intérêt de plusieurs chercheurs et de plusieurs philosophes. M. Serres a défini la communication comme suit [176] : *la communication est une sorte de jeu pratiqué par deux interlocuteurs, considérés comme associés, contre les phénomènes de brouillage et de confusion, voire contre des individus ayant quelque intérêt à rompre la communication. Ces interlocuteurs sont dans le même camp et luttent ensemble pour surmonter le handicap du bruit et pour faire émerger une vérité sur laquelle le but est de se mettre d'accord, autrement dit, pour réussir la communication.*

Après cette définition abstraite de la communication, nous allons, dans ce qui suit, analyser chacune des composantes de la communication orale. Nous définirons, ainsi, par la suite, la chaîne de communication parlée établie entre deux interlocuteurs.

1.1.1 Le son de la parole

Le son pur n'est rien d'autre que le mouvement d'oscillation d'air le plus simple que l'on puisse imaginer : un va-et-vient régulier autour d'une position moyenne. La parole se distingue des autres sons par des caractéristiques acoustiques qui ont leurs origines dans les mécanismes de production. La principale fonction de ces sons de parole dans une langue est d'établir des distinctions entre des unités de signification [32]. Les phonèmes permettent ainsi de distinguer les mots les uns des autres. En effet, une langue doit pouvoir être décrite en termes d'unités fondamentales ayant la propriété de ne pas pouvoir se substituer à d'autres unités, sous risque de changement du sens de l'énoncé. J.L. Flanagan, dans le cadre de la théorie de la communication [59], va dans ce sens en définissant le son comme suit : *pour être un médium approprié à la transmission de l'information, une langue doit être susceptible d'être décrite par un nombre fini de sons distincts et mutuellement exclusifs.*

Pour comprendre la différence entre la réalisation acoustique d'un phonème et le phonème lui même, prenons comme exemple la lettre *r*. Celle-ci se prononce de plusieurs façons, selon les régions : on distingue notamment l'*r* parisien et l'*r* roulé. Mais ces variations ne jouent aucun rôle distinctif : il n'y a pas un mot "rien" prononcé avec *r* roulé et un autre prononcé avec *r* parisien. Ces deux sons de parole correspondent à un seul phonème. Au contraire, "rien" s'oppose à "bien", "lien", "tien", lesquels s'opposent également entre eux ; chacune des consonnes initiales de ces quatre mots est donc un phonème.

Par conséquent, dans un signal de parole, un même phonème, qui apparaît plusieurs fois dans le signal, peut avoir des prononciations différentes. En effet, les réalisations physiques d'un phonème peuvent varier considérablement en fonction du contexte, de la cadence d'élocution, du dialecte, du style et du locuteur. L'interlocuteur en phase de décodage raisonne dès lors, non sur le phonème concret, mais sur la classe des objets dont les éléments partagent les mêmes traits distinctifs. Cette notion de trait exprime une similarité au niveau articulatoire, acoustique ou perceptif [128].

1.1.2 Le mot

Les sons émis d'une langue font partie d'unités chargées de signification, que la tradition identifie avec les mots. La définition du mot est un sujet bien connu chez les linguistes, nous invitons le lecteur à consulter par exemple [82] pour plus d'informations sur la question.

Dans la communication parlée, il y a peu de pauses entre les mots d'une même phrase. Par ailleurs, il peut exister des intervalles de silence au milieu d'un mot et aucun à la frontière de deux mots successifs [182]. Ceci n'empêche pas deux interlocuteurs de se comprendre. En revanche, si l'un des interlocuteurs ne maîtrise pas bien la langue, il repérera dans le signal quelques mots clés pour pouvoir communiquer. C'est un des cas où la communication s'établira difficilement. Prenons un autre cas de figure, lorsque nous assistons à un dialogue entre deux interlocuteurs parlant une langue que nous ne connaissons pas. Dans ce cas, il nous est difficile de mettre les frontières entre les mots, sauf peut être, lorsque l'interlocuteur reprend son souffle, auquel cas, on peut supposer que la fin d'un mot a été rencontrée. Ces exemples montrent bien les problèmes de la communication parlée homme-homme que nous retrouverons d'ailleurs dans la communication parlée homme-machine. Il nous est donc indispensable de prendre en compte ces contraintes dans la réalisation des machines pouvant dialoguer avec des humains.

1.1.3 La phrase

Dans l'espace de communication, une phrase est une suite de mots respectant un certain nombre de règles linguistiques d'ordre pragmatique, sémantique, syntaxique, lexical, morphologique, phonologique et phonétique. Une phrase n'a, au mieux, qu'une influence indirecte sur la construction de la phrase suivante (cas des anaphores), alors qu'au sein d'une même phrase, la production d'un mot, dépend au moins de l'un des mots précédents, ou sinon d'au moins un des mots suivants. Les premiers mots reconnus, dans une phrase, véhiculent un sens que l'être humain utilise pour anticiper la reconnaissance de la phrase entière. Il s'agit d'un ensemble de connaissances linguistiques que l'être humain utilise pour comprendre et se faire comprendre. Introduire une telle connaissance dans une machine, améliorera et facilitera fortement le dialogue homme-machine. C'est l'orientation que nous allons prendre tout au long de cette thèse, comme nous le verrons ultérieurement.

1.1.4 La chaîne de communication orale

La chaîne de communication orale peut être considérée comme un double processus symétrique qui consiste à produire et percevoir de la parole, comme le montre la figure 1.1 [140, 78]. Lorsqu'un locuteur génère un message oral, il procède à une série d'opérations mettant en jeu tous les niveaux linguistiques : il y a donc, à partir du sens, production d'un ensemble de commandes qui, du système nerveux central jusqu'aux muscles, via le système nerveux périphérique, va permettre de piloter les évolutions du conduit vocal. En revanche, la perception permet, en une première phase, de transformer l'information contenue dans le signal acoustique et de la transmettre au cerveau. Ensuite, une deuxième phase est menée. Cette phase consiste à reconstruire le message linguistique sur la base de cette représentation du signal de parole [32]. Il est donc évident, que la communication n'est possible que si les deux interlocuteurs disposent d'un espace commun de communication.

1.2 La reconnaissance vocale

Avoir une machine qui nous comprend ou même qui transcrit notre parole est un vieux rêve de l'homme que l'intelligence artificielle essaye encore de résoudre. En effet, l'utilisation de la parole dans un processus de communication homme-machine présente des avantages certains, notamment dans des situations d'accès à distance (téléphone, télécommunications), dans des situations où l'utilisation de la main ou de la vue est difficile pour l'utilisateur ou même dans une situation d'utilisation occasionnelle du système (cas d'un utilisateur non spécialiste).

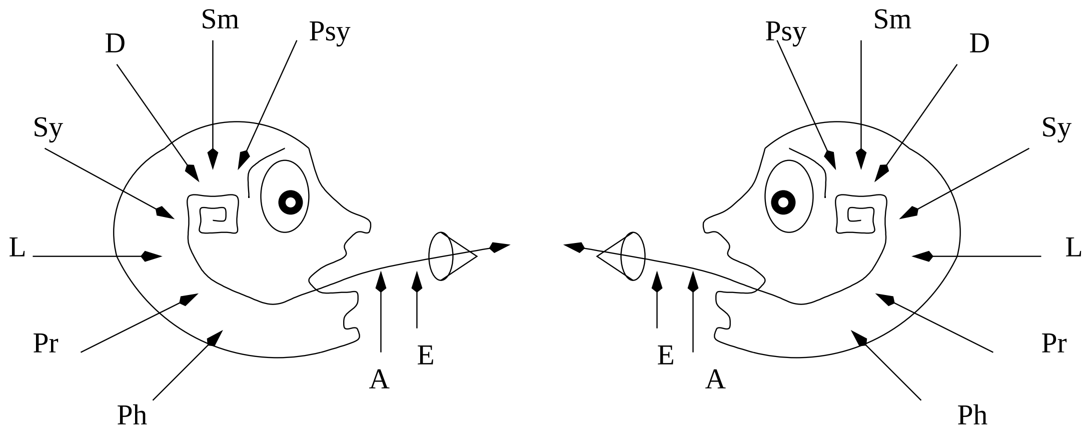


FIG. 1.1 – **La chaîne de communication orale** *: psychologie du locuteur, sémantique, règles du discours, syntaxe, lexique, prosodie, phonétique, bruit de l'environnement.*

Le champ des applications potentielles est donc très vaste. De nombreux progrès ont été réalisés ces dix dernières années dans ce domaine. Il existe des logiciels vendus actuellement dans le commerce permettant d'effectuer une reconnaissance de la parole continue pour un vocabulaire important. Néanmoins, les performances de ces systèmes sont encore largement inférieures à celles des êtres humains et particulièrement au niveau linguistique.

L'évaluation des performances d'un système de reconnaissance (nous utilisons plutôt l'acronyme RAP, pour mentionner Reconnaissance Automatique de la Parole) repose sur plusieurs critères : les conditions d'utilisation (dépendant ou indépendant du locuteur, état du bruit, état des locuteurs, etc.), la taille du vocabulaire utilisé, ainsi que le taux de reconnaissance (le pourcentage de mots reconnus). Plusieurs études ont été menées pour spécifier et implanter des méthodes d'évaluation des systèmes de RAP, nous citons à titre d'exemple ceux présentés dans [64, 54, 35].

1.2.1 Les difficultés liées au signal de parole

En plus des difficultés de la communication orale (cf. §1.1), le problème de la reconnaissance de la parole réside essentiellement dans la spécificité du signal vocal. En effet, en plus de la sensibilité au bruit qui l'entoure, le signal de la parole est éminemment variable d'un locuteur donné à un autre (état émotif, fatigue, etc), ce qui rend plus difficile le problème de la reconnaissance d'un message indépendamment de son locuteur. En parole continue, qui fait l'objet de cette thèse, les effets contextuels d'un son élémentaire sur ses voisins sont considérables [128, 32]. En effet, l'être humain est capable, suivant un processus linguistique complexe, de découper un message en unités plus petites : phonèmes, mots, groupes de mots qui forment un sens ou un groupe de sens. Un système intégrant un tel comportement rend, bien sûr, la reconnaissance plus performante.

Sur la figure 1.2 [79], nous présentons un exemple de signal vocal capté par un microphone, ainsi que l'analyse acoustique correspondante. Cette analyse spectrale fournit une représentation tridimensionnelle de la parole : temps, fréquence, intensité. Cette représentation appelée spectrogramme vocal est largement utilisée par les chercheurs en communication parlée. Cet exemple illustre bien les difficultés qui viennent d'être mentionnées et surtout celles relatives à la parole continue.

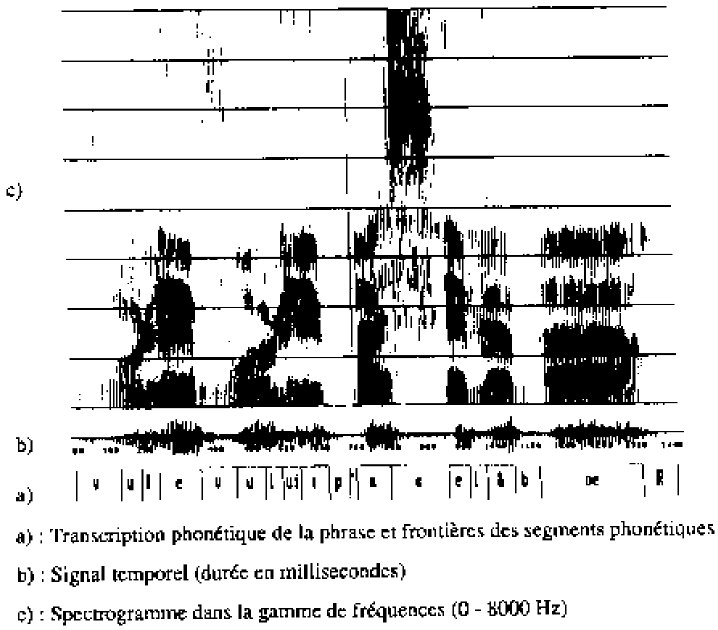


FIG. 1.2 – *Signal vocal temporel (b) et spectrogramme vocal (c) de la phrase "Voulez vous lui passer le beurre".*

1.2.2 Historique

La reconnaissance automatique de la parole a commencé à fasciner les chercheurs depuis les années cinquante. En effet, dès 1949 J. Dreyfus-Graf étudiait la déviation du spot de l'oscilloscope, ce qui lui a permis en 1961 de dicter à un mini système toutes les lettres de l'alphabet, soigneusement prononcées par l'auteur. Cette époque a été marquée, également, par le premier système de reconnaissance de chiffre produit par le laboratoire Bell aux Etats-Unis en 1952 et par la première "machine à écrire phonétique" d'Olson et de Belar en 1956 (machine capable de reconnaître dix syllabes différentes prononcées de façon isolée par un même locuteur). C'est en 1959 que P. Denes, à Londres, utilise pour la première fois l'information "linguistique" dans un système de reconnaissance : utilisation des fréquences des suites de deux phonèmes (diphones) pour améliorer les résultats d'un système de reconnaissance. Toutes ces réalisations utilisaient les moyens de l'électronique analogique de l'époque. La première utilisation de l'ordinateur dans ce domaine a été faite par Forgie au laboratoire Lincoln en 1959.

Dans les années soixante, l'utilisation de l'ordinateur s'imposa progressivement ; provoquant ainsi une nette amélioration dans la reconnaissance de mots isolés et une augmentation progressive de la taille des vocabulaires traités. Cette époque a été marquée également par l'extension de l'utilisation d'informations des niveaux supérieurs. A l'université de Kyoto, en 1965, Doshita utilisait les fréquences de suites de trois phonèmes (triphones) qui, en fait, représentent les contraintes apportées par la structure linguistique du message. Cette période s'acheva sur le petit "Scandale" de J. Pierce, chercheur connu du laboratoire Bell, dans *Journal of the Acoustical Society of America* (en 1969). Il y indiquait que dans l'état des connaissances théoriques d'alors, on était incapable de reconnaître la parole de quelque façon que ce soit. Ce texte fit beaucoup de bruit dans la communauté scientifique, mais eut le mérite de susciter une prise de conscience.

Pour plus de détails sur ces deux périodes de recherche, nous invitons le lecteur à consulter les références suivantes : [87, 58, 161, 126].

Dans les années soixante-dix, comme on ne savait pas bien reconnaître les phonèmes d'une langue, on utilisait les traitements de type intelligence artificielle dans la reconnaissance malgré les erreurs de décodage phonétique, en utilisant les contraintes relatives au langage (celui d'une application bien précise) [32]. A la même époque, les approches probabilistes firent leurs apparition. En effet, l'approche markovienne vient prendre place et constitue un fait saillant des dernières années [10, 94]. Cette approche permet d'attaquer le problème du décodage de la parole continue avec une modélisation stochastique du langage. Les premiers systèmes utilisant une telle approche étaient développés par F. Jelinek et son équipe, au centre de recherche IBM de Yorktown et par J. Baker à CMU.

Depuis, et jusqu'à aujourd'hui, les chaînes de Markov sont de plus en plus utilisées dans les systèmes de reconnaissance [160]. A la fin des années 80, l'utilisation des modèles connexionnistes, fondés sur une modélisation plus ou moins réaliste du cortex humain, s'est largement répandue en reconnaissance vocale [114, 162]. Les résultats obtenus par ces modèles sont comparables, ou un peu inférieurs aux modèles actuels les plus performants. Il faut néanmoins noter que les modèles de Markov bénéficient de plus de dix ans d'efforts continus. D'autres travaux qui consistent à coupler les deux modèles sont apparus ces dernières années et ont donné ainsi naissance aux modèles hybrides [81, 154, 50].

1.3 L'approche probabiliste en RAP

La reconnaissance de la parole, vue comme un problème de la théorie de la communication, a pour but de reconstruire un message m à partir d'une séquence d'observations y. Le message de parole passe de la forme acoustique à la forme électrique par l'emploi d'un microphone, pour être digitalisé et sauvegardé sur un support informatique noté par s. Ensuite, s est subdivisé en unités de temps (typiquement de 10ms chacune). De ces unités de temps, on extrait un certain nombre de paramètres (au moyen de techniques de traitement du signal) caractérisant de manière adéquate les diverses réalisations acoustiques de la voix humaine.

La reconstitution d'un message m inconnu, à partir d'une séquence y, consiste à retrouver, parmi tous les messages possibles, celui, qui selon toute vraisemblance, correspond à y. L'utilisation de la règle de Bayes permet de décomposer la probabilité $P(m/y)$ en deux composantes :

$$m = \arg_m max\ P(m/y) = \arg_m max \frac{P(m)P(y/m)}{P(y)} \tag{1.1}$$

Le dénominateur est constant pour tous les messages possibles, celui-ci peut donc être omis, et m peut alors s'écrire comme suit :

$$m = \arg_m max P(m)P(y/m) \tag{1.2}$$

L'étape de la reconnaissance consiste donc à déterminer la suite de mots m qui maximise le produit des deux termes $P(m)$ et $P(y/m)$. Le premier terme représente la probabilité *a priori* d'observer la suite de mots m indépendamment du signal. Cette probabilité est déterminée par le modèle de langage. Le deuxième terme indique la probabilité d'observer la séquence de vecteurs acoustiques y sachant une séquence de mots spécifiques. Cette probabilité est estimée par le modèle acoustique.

L'outil statistique le plus utilisé et le plus performant, de nos jours, pour la modélisation acoustique est fondé sur les modèles de Markov cachés [12, 94, 159]. Nous allons dans ce qui suit définir les différentes étapes de la reconnaissance, en utilisant ce modèle au niveau acoustique. La figure 1.3 montre les différentes étapes nécessaires à la reconnaissance d'une hypothèse donnée. En effet, en premier le signal de parole prononcé sera subdivisé pour construire une séquence de vecteurs acoustiques. En utilisant ces vecteurs, le modèle acoustique se charge, à partir des HMMs de phonèmes appris sur un corpus d'apprentissage, de construire la suite des phonèmes hypothèses du signal prononcé. Un seul modèle de HMM, représentant l'hypothèse, sera construit par concaténation de l'ensemble des HMMs de phonèmes qui la compose et génère ainsi la probabilité du signal y, ce qui définit la probabilité $P(y/m)$. Ainsi, à partir du dictionnaire des prononciations, la suite des mots hypothèses sera déterminée. Cette suite de mots sera évaluée par le modèle de langage pour estimer la probabilité $P(m)$. En principe, ce processus est répété pour toutes les hypothèses possibles. Le résultat du système est finalement la meilleure hypothèse.

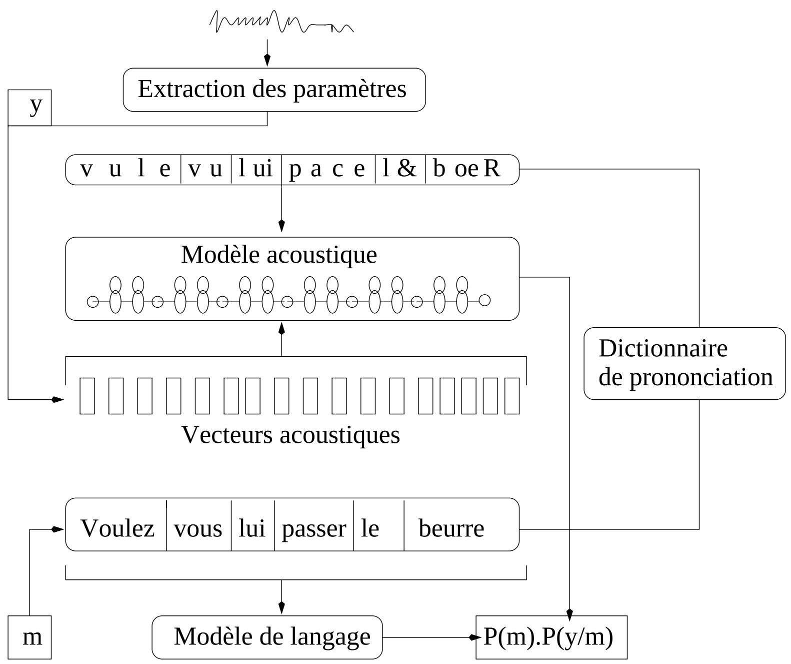


FIG. 1.3 – *L'approche probabiliste de la RAP pour l'hypothèse "Voulez vous lui passer le beurre"*

La construction d'un système de RAP repose donc sur la résolution de plusieurs problèmes [199]. Premièrement, on commence par la paramétrisation du signal de parole. En effet, il faut extraire l'ensemble des informations acoustiques nécessaires, dans un format compact et compatible avec la modélisation acoustique à base de HMM. Deuxièmement, les modèles à base de HMM doivent représenter la distribution de chaque son, dans chaque contexte dans lequel le son apparaît. Les paramètres du modèle à base de HMM sont estimés à partir d'un corpus d'apprentissage, et il n'est jamais possible d'obtenir suffisamment de données pour couvrir tous les contextes possibles. Troisièmement, le modèle de langage doit prédire le mot courant en fonction de son historique. Cependant, comme pour les HMMs, la diversité des données est un problème très présent. Le modèle de langage doit être capable d'estimer la probabilité d'une séquence de mots même si elle n'apparaît pas dans le corpus d'apprentissage. Finalement, énumérer toutes les séquences de mots possibles pour déterminer la meilleure hypothèse n'est pas possible avec les machines actuelles. Par conséquent, il est indispensable d'écarter les hypothèses aussitôt qu'elles deviennent improbables. Construire un bon algorithme de reconnaissance est cependant très important, pour la réalisation d'un système de RAP précis et rapide sur les plates-formes de nos jours [201].

1.4 Extraction des paramètres

L'objectif de cette phase de reconnaissance est de subdiviser le signal de parole en trames (blocs) et d'extraire pour chaque trame un ensemble de coefficients qui représente au mieux le signal modélisé [32]. L'espace entre les trames est de l'ordre de 10 ms et les trames se chevauchent car la fenêtre d'analyse est plus longue (de l'ordre de 25 ou 30 ms : durée pendant laquelle la parole peut être considérée comme stationnaire). Ce traitement repose sur une hypothèse importante qui consiste à supposer que le signal de parole est pseudo-stationnaire (les caractéristiques spectrales sont constantes dans un intervalle de quelques millisecondes).

Après la mise en forme du signal (comme la plupart des méthodes d'analyse de la parole) [50], des coefficients représentant au mieux le signal seront engendrés, ce qui constitue les vecteurs acoustiques finaux. Parmi les coefficients les plus utilisés en reconnaissance de la parole, surtout dans un milieu non bruité, nous trouvons les coefficients cepstraux, appelés également cepstres. Les deux méthodes les plus connues pour l'extraction de ces cepstres sont : l'analyse spectrale et l'analyse paramétrique. Pour l'analyse spectrale (exemple : *Mel-Scale Frequency Cepstral Coeficients* (MFCC)) comme pour l'analyse paramétrique (exemple : le codage prédictif linéaire (LPC)), le signal de parole sera transformé en une série de vecteurs cepstraux calculés pour chaque trame [36, 8].

D'autres types de coefficients existent également, comme les coefficients PLP (*Perceptual Linear Predictive*) par exemple. Ces coefficients (PLP) sont surtout utilisés dans les milieux bruités où ils permettent d'estimer les paramètres d'un filtre auto-régressif, modélisant au mieux le spectre auditif [65, 80]. Il existe plusieurs techniques permettant l'amélioration de la qualité de ces coefficients, nous citons à titre d'exemple : CMS (*Cepstral Mean Substraction*), LDA (Analyse discriminative linéaire), NLDA (Analyse discriminative non linéaire), etc. Pour plus d'informations sur les différentes méthodes d'extractions de ces coefficients, nous invitons le lecteur à consulter [9, 63, 76, 181].

L'énergie du signal plus 12 coefficients sont utilisés par la plupart des systèmes de RAP actuels pour former les 13 éléments de base du vecteur acoustique. Comme il sera mentionné dans le paragraphe 1.5, la modélisation acoustique suppose que chaque vecteur acoustique n'est pas corrélé avec ses voisins. Cette hypothèse n'est pas correcte surtout si on sait que les contraintes physiques de l'appareil vocal humain assurent qu'il existe une continuité entre les estimations spectrales successives. Pour pallier ce problème, on ajoute les dérivées premières et secondes aux coefficients cepstraux de base [7, 66]. Le vecteur acoustique final aura donc 39 coefficients (13 + 13 (dérivée première) + 13 (dérivée seconde)).

Ces coefficients jouent un rôle capital dans les approches utilisées pour reconnaître la parole. En effet, ces paramètres qui modélisent le signal seront fournis aux couches supérieures du système de RAP pour l'estimation de la probabilité $P(y/m)$ (cf. §1.3). En ce qui concerne notre travail, étant donné que nous ne nous intéressons qu'au milieu non bruité, nous nous sommes limités à l'utilisation des coefficients MFCC. La justification de l'utilisation de ces coefficients est leur bonne représentation des aspects perceptuels du spectre de parole [45]. De plus, des tests, dans un milieu non bruité, effectués avec d'autres méthodes (log-RASTA, PLP) n'ont pas montré leurs supériorités par rapport au modèle MFCC.

1.5 La modélisation acoustique à base de HMMs

Les modèles de Markov cachés (ou HMMs) sont les plus communément utilisés en RAP ces dernières années, pour estimer la probabilité $P(y/m)$ (cf. §1.3). Ces modèles se sont avérés les mieux adaptés aux problèmes de la RAP. La quasi-totalité des systèmes de reconnaissance de la parole disponibles actuellement sur le marché sont fondés sur ces modèles. Un modèle de Markov caché est un automate stochastique particulier capable, après avoir été appris, d'estimer la probabilité qu'une séquence d'observations ait été générée par ce modèle. L'objectif de ces HMMs en RAP est donc de modéliser au mieux les unités représentatives du signal de parole. En effet, le modèle acoustique à base de HMMs nous permet de calculer la vraisemblance de n'importe quelle séquence de vecteurs acoustiques Y sachant un mot w. La distribution de probabilité doit être apprise à partir de plusieurs échantillons du même mot w et pour chacun de ces mots, on extrait les paramètres acoustiques qui le composent. Cependant, ceci est irréalisable pour les systèmes de RAP à grands vocabulaires, du fait qu'il est presque impossible de trouver le nombre d'échantillons nécessaires pour chaque mot. Par conséquent, afin de réduire le nombre de paramètres à apprendre et de trouver suffisamment d'échantillons pour chaque unité, on décompose la séquence de mots en un ensemble d'unités plus petites, comme la syllabe ou le phonème. Le modèle de HMM associé à une phrase est ainsi constitué d'une concaténation de HMMs des sous-unités lexicales (syllabe ou phonème) qui la composent (cf. §1.6).

Avant de montrer comment on applique ces modèles à la reconnaissance de la parole, nous allons définir tout d'abord les principes de base des chaînes de Markov.

1.5.1 Définition des chaînes de Markov

Une chaîne de Markov peut être vue comme un ensemble discret de noeuds ou d'états (avec au moins un noeud initial et un noeud final) et de transitions ou arcs reliant ces états. Elle peut être donc définie par l'ensemble des paramètres [158] :

$$\Omega = (N, A, B, \Pi)$$

- N est le nombre de noeuds ou d'états $S = \{s_1, s_2, \ldots, s_N\}$,
- $A = \{a_{ij}\} = \{p(q_j/q_i)\}$: l'ensemble des probabilités de transition entre les états q_i et q_j. La probabilité de transition est la probabilité de choisir la transition a_{ij} pour accéder à l'état q_j, étant donné un processus à l'état q_i. Cette probabilité ne dépend que de l'état précédent pour un HMM d'ordre un

  $$P(q_t = s_k/q_{t-1} = s_j, q_{t-2} = s_i, \ldots) = P(q_t = s_k/q_{t-1} = s_j)$$

  ou des deux états précédents dans le cas d'un HMM d'ordre deux

  $$P(q_t = s_k/q_{t-1} = s_j, q_{t-2} = s_i, \ldots) = P(q_t = s_k/q_{t-1} = s_j, q_{t-2} = s_i).$$

  En d'autres termes, l'évolution du système entre deux instants t et $t+1$ ne dépend que de l'état de ce système au temps $t-1$ (ordre un) ou aux instants précédents $t-1$ et $t-2$ (ordre deux),
- $B = \{b_j(y_n)\} = p(y_n/q_j)$: l'ensemble des probabilités d'émission de l'observation y_n dans l'état q_j. La forme que prend cette distribution détermine le type de HMM. C'est ainsi qu'on parle de HMMs discrets, semi-continus, continus, à livre de codes (*codebooks*) partagés, etc. Pour plus d'informations sur les différents types de HMMs, se référer à [169],

– Π : la distribution initiale des états. L'ensemble des $p(q_j/q_I)$, $\forall j \in [1,N]$. q_I représente l'état initial du modèle HMM, il ne peut émettre de vecteurs acoustiques.

Une chaîne est dite homogène si les probabilités de transition ne dépendent pas du temps. On peut alors poser : $a_{jk} = P(q_t = s_k/q_{t-1} = s_j)$ et $a_{ijk} = P(q_t = s_k/q_{t-1} = s_j, q_{t-2} = s_i)$.

Les contraintes qu'on impose à une chaîne de Markov de N états sont :

– chaîne du premier ordre

$$\sum_{k=1}^{N} a_{jk} = 1, \quad 1 \leq j \leq N$$

– chaîne du second ordre

$$\sum_{k=1}^{N} a_{ijk} = 1 \quad avec \ \ (i,j) \in [1,N] \times [1,N]$$

1.5.2 Application à la RAP

Les modèles de Markov cachés utilisés en RAP constituent un cas très particulier des modèles de Markov. La figure 1.4 illustre un HMM à 3 états typique tel qu'utilisé en RAP pour la modélisation d'un phonème. Les états d'entrée et de sortie sont fournis pour faciliter la concaténation des modèles entre eux. L'état de sortie d'un modèle de phonème peut être fusionné avec l'état d'entrée d'un autre modèle de Markov caché pour former un modèle composite. Ceci permet aux modèles de phonème d'être concaténés ensemble pour former les mots et ainsi les phrases. On remarque que seules sont permises des transitions de type gauche-droite et ce, dans le but de mieux modéliser la contrainte temporelle de la parole[1].

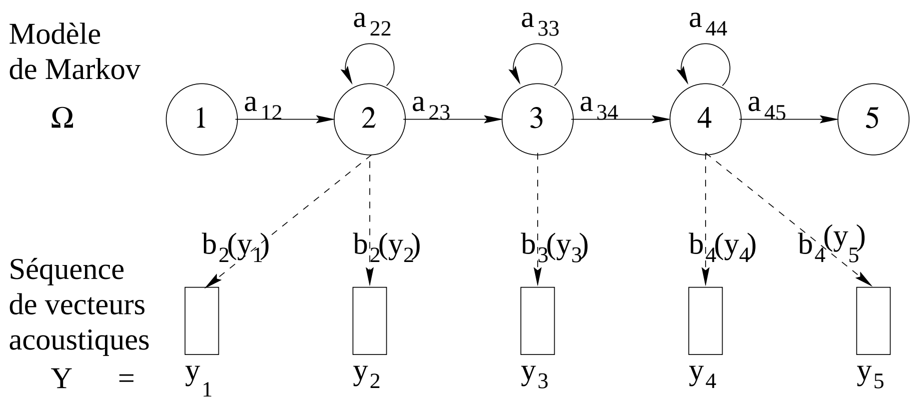


FIG. 1.4 – *Exemple d'un HMM à trois états caractérisé par une distribution de probabilités pour chaque état associé à une observation et par des probabilités de transition entre les états.*

Un HMM est considéré comme un générateur de vecteurs acoustiques. C'est une machine à états finis qui change d'état à chaque unité de temps. Pour chaque unité de temps t, une fois arrivé à l'état q_j, un vecteur acoustique y_t est généré avec une densité de probabilité $b_j(y_t)$. De plus, la transition de l'état q_i à l'état q_j est probabiliste et elle est estimée par la probabilité de

1. Plusieurs systèmes de RAP modélisent d'une façon explicite la durée des phonèmes, dans le but d'améliorer la reconnaissance au niveau du mot [133, 50].

transition a_{ij}. La figure 1.4 montre un exemple où la séquence d'états $X = 1{,}2{,}2{,}3{,}4{,}4{,}5$ génère la séquence $Y = y_1 \dots y_5$. La probabilité de la séquence de vecteurs acoustiques Y et la séquence d'états X sachant un modèle Ω est définie par le produit des probabilités de transitions et les probabilités d'émissions. Pour la séquence X de la figure 1.4, elle est estimée par :

$$P(Y{,}X/\Omega) = a_{12}b_2(y_1)a_{22}b_2(y_2)a_{23}b_3(y_3)\dots \tag{1.3}$$

Généralement, la probabilité d'une séquence de vecteurs acoustiques Y et d'une séquence d'états $X = q_1{,}q_2{,}q_3{,}\dots{,}q_N$ est définie par :

$$P(Y{,}X/\Omega) = a_{q_0q_1}\prod_{n=1}^{N} b_{q_n}(y_n)a_{q_nq_{n+1}} \tag{1.4}$$

où q_0 est l'état d'entrée du modèle et q_{N+1} est l'état de sortie.

En pratique, c'est seulement la séquence d'observations Y qui est connue. La séquence d'états X est cachée, d'où le nom de modèle de Markov caché.

Le problème de l'estimation des probabilités peut être énoncé de la façon suivante : étant donné un modèle de Markov Ω, comment calculer la probabilité $P(Y/\Omega)$ qui génère la séquence de vecteurs acoustiques Y?

Il existe deux procédures récurrentes de calcul de la probabilité $P(Y/\Omega)$ [130] :

- l'algorithme "*Forward-Backward*" qui fournit une solution exacte à ce problème faisant intervenir tous les chemins dans le modèle HMM ;
- l'algorithme Viterbi qui fournit une solution approchée faisant intervenir uniquement le meilleur chemin dans le modèle HMM.

1.6 L'approche phonétique

Un HMM peut représenter n'importe quel ensemble d'unités acoustiques : mots, phonèmes, etc. Or, dans le cas des grands vocabulaires, le fait d'associer à chaque mot un HMM distinct pose de sérieux problèmes au niveau de l'apprentissage et du stockage. En effet, pour effectuer cette tâche correctement, il faudrait que le corpus d'apprentissage contienne plusieurs occurrences de chaque mot. Ceci est pratiquement impossible. De plus, à l'ajout d'un nouveau mot au lexique, correspondrait la collecte de nouvelles occurrences acoustiques et un nouvel apprentissage, ce qui n'est guère souple. Dans les faits, la reconnaissance de grands vocabulaires est donc toujours effectuée au moyen de HMMs de mots élaborés à partir de HMMs représentant des sous-unités de mots.

Il est facile de combiner plusieurs HMMs en un seul. Un exemple de ce type de construction, où un HMM correspondant au mot *débit* est obtenu par la concaténation des HMMs correspondant aux phonèmes *d,e,b* et *i* est illustré par la figure 1.5. On montre, en A, que la construction peut être obtenue par la coïncidence du dernier état du HMM d'un phonème, avec le premier état du HMM du phonème suivant. Un HMM un peu plus élaboré en B, pour le même mot, illustre l'ajout d'une transition vide permettant l'omission d'un phonème.

Les HMMs obtenus par concaténation d'autres HMMs donnent en général des résultats de reconnaissance inférieurs à ceux obtenus directement au moyen de HMMs appris sur les unités qu'ils sont censés représenter. Cependant, l'utilisation de ce type de construction permet de réduire le nombre de HMMs nécessaires pour une tâche donnée (par exemple, pour un vocabulaire de 20000 entrées, 35 HMMs de phonèmes pourraient suffire plutôt que 20000 HMMs de mots).

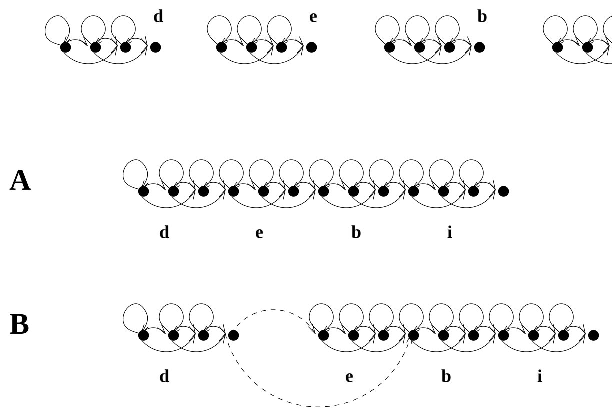


FIG. 1.5 – *HMM d'un mot obtenu par la concaténation de HMMs de phonèmes*

Bien que d'un point de vue linguistique, il soit naturel de définir un mot au moyen d'une séquence de phonèmes, ce dernier type de sous-unités n'est pas celui le plus couramment utilisé pour construire des HMMs de mots. En effet, la réalisation acoustique d'un phonème est fortement influencée par l'identité des phonèmes qui précèdent et qui suivent immédiatement un phonème donné. C'est ce que l'on appelle le phénomène de co-articulation. C'est pourquoi en RAP, les unités servant à la construction des HMMs de mots, bien qu'en général dérivées de phonèmes, seront plus nombreuses et mieux définies d'un point de vue acoustique. Les unités de construction de phonème les plus simples à utiliser, pour faire face à ce phénomène de co-articulation, sont les diphones et les triphones. Le diphone permet de fixer le contexte, seulement au phonème précédent ou au phonème suivant immédiatement le phonème à modéliser. Tandis que le triphone tient compte des deux phonèmes précédent et suivant pour fixer le contexte. Néanmoins, le problème de ces deux approches est le nombre potentiellement trop grand de HMMs de triphones ou de diphones à modéliser [169]. Pour contourner ce problème, tout en conservant le potentiel accru de modélisation qu'il procurent, d'autres techniques ont été proposées. Elles reposent surtout sur le regroupement de modèles de triphones ou diphones similaires, notamment à l'aide d'arbres de décision [11, 127, 200].

Cette approche, qui par ailleurs est connue sous le nom d'approche phonétique, permet d'éviter la collecte d'énormes corpus d'apprentissage qui auraient été nécessaires lors de l'utilisation de HMMs de mots. Ceci permet ainsi de rendre réalisable la reconnaissance automatique de la parole sur de grands vocabulaires. De plus, l'impact au niveau flexibilité est également très important. En effet, l'ajout de nouveaux mots dans le lexique ne nécessite plus aucune collecte ni apprentissage supplémentaire ; il suffit simplement de déterminer la ou les séquences de sous-unités correspondant aux nouveaux mots et de construire selon ces séquences et à partir des HMMs des sous-unités correspondantes, les HMMs modélisant ces nouveaux mots. En prime, l'approche phonétique permet une économie au niveau de l'espace mémoire requis tant lors de la reconnaissance que de l'apprentissage. En effet, le nombre de distributions devant être conservé en mémoire est, de beaucoup, inférieur à celui qu'aurait nécessité l'utilisation d'un HMM par mot de lexique.

Finalement, de la même façon qu'on peut construire des HMMs de mots à partir de HMMs de phonèmes, on peut construire des HMMs de phrases à partir de HMMs de mots. C'est ce qui est illustré sur la figure 1.6. On soulignera que *"pause"* n'est pas ici le mot *pause* mais bien la réalisation acoustique d'une pause entre deux mots. Cette pause est optionnelle (on a rajouté une transition vide en parallèle avec le HMM correspondant) de la même façon qu'on l'avait fait pour le phonème e sur la figure 1.5B.

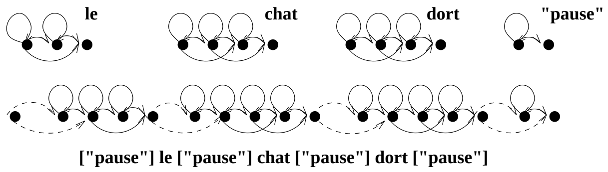


FIG. 1.6 – *HMM d'une phrase obtenu par concaténation de HMMs de mots*

1.7 Les modèles de langage

La probabilité du modèle de langage, $P(m)$ de la formule 1.2, est censée représenter les probabilités relatives *a priori* des messages possibles. Par exemple, pour une séquence de mots $m = m_1 m_2 \ldots m_{L_m}$, $P(m)$ peut être estimée par le produit des probabilités associées à chacun des mots de la séquence, soit :

$$P(m) \;=\; P(m_1)P(m_2)\ldots P(m_{Lm})$$

où $P(m_i)$ est une probabilité d'occurrence hors de tout contexte[2].

Ce modèle tout à fait valable mathématiquement est le modèle unigrammes. Cependant, celui-ci est d'une aide toute relative quand il s'agit de guider le choix de la reconnaissance face à plusieurs possibilités valables d'un point de vue acoustique. La principale raison de cette faiblesse est justement qu'il aide au choix sans tenir compte du contexte. Or, une langue n'est pas seulement formée d'un ensemble de mots, mais également d'une série de règles syntaxiques régissant l'ordre selon lequel ces divers mots peuvent apparaître. Mais plus encore, les mots eux-mêmes, selon la façon dont ils sont utilisés, ont une influence sur la réalité de ce qu'ils décrivent ainsi que sur le sens. On parle alors de sémantique et de pragmatique. Une bonne valeur de $P(m)$, et par extension un bon modèle de langage, devra donc, autant que possible, tenir compte de ces diverses contraintes. Par conséquent, l'intégration de ces modèles dans les systèmes de RAP augmentera le taux de reconnaissance et réduira la complexité de la procédure de recherche [148].

On distingue actuellement trois grandes familles de modèles de langage : deux, fondées sur des approches totalement différentes, qui sont les modèles à base de grammaires[3] et les modèles probabilistes ; une dernière, la famille des modèles à grammaire probabiliste, qui, comme son nom l'indique, est issue du mariage des deux premières. Les modèles probabilistes sont ceux qui dominent les systèmes de RAP actuellement et ceci grâce à la performance qu'ils apportent aux systèmes de RAP. D'autres modèles, dits hybrides, sont nés ces dernières années et sont issus de la combinaison de plusieurs modèles appartenant à ces trois grandes familles (cf. chapitre 3).

1.8 Les algorithmes de reconnaissance

Il existe deux principales approches, en RAP continue, permettant de trouver la suite de mots W qui maximise la formule 1.2 : **en profondeur d'abord** et **en largeur d'abord**. Pour

2. En pratique, cette probabilité est estimée par la fréquence de chaque mot dans un corpus d'apprentissage.

3. des modèles directement calqués sur la syntaxe propre à une langue, souvent présentés sous forme de séries de règles et souvent traduits sous forme de graphe

la première approche, l'hypothèse la plus prometteuse est parcourue jusqu'à la fin de l'échantillon de parole. Nous citons à titre d'exemple l'algorithme à pile[4] [73] et l'algorithme A^* [155, 152]. En revanche, dans l'approche en largeur d'abord, toutes les hypothèses sont parcourues en parallèle. Cette approche, qui exploite le principe d'optimalité de Bellman, utilise les méthodes dérivées de la programmation dynamique [142]. De nombreux algorithmes, utilisant cette approche, ont été proposés. Nous citons à titre d'exemple : le "*Level Building*" [138], le "*Two Level DTW*" [171] et l'algorithme de type Viterbi synchrone [27]. L'algorithme de Viterbi est parmi les plus utilisés en RAP.

Le nombre de combinaisons possibles des mots est très important. Ainsi, des techniques d'élagage (*pruning*) ont été développées pour permettre l'utilisation de ces algorithmes [77, 188]. Ces techniques visent à limiter l'espace de recherche afin que l'algorithme puisse tourner en temps réel.

Les algorithmes de reconnaissance permettent soit de générer la phrase qui a la plus grande probabilité d'avoir été émise soit de générer les N meilleures séquences de mots. Les N meilleures séquences obtenues doivent être syntaxiquement différentes et non pas des alignements différents d'une même solution. De récents travaux ont montré l'utilité de générer une liste des N meilleures séquences de mots reconnues afin de trouver, avec l'aide d'informations supplémentaires, la phrase effectivement prononcée (par exemple, la phrase satisfaisant à des restrictions supplémentaires comme l'accord en genre et en nombre, ou encore la meilleure phrase après une réévaluation des scores respectifs de chaque élément de la liste à l'aide des modèles de langage plus performants).

Nous présentons brièvement dans les sous paragraphes suivants, les algorithmes les plus utilisés dans les systèmes de RAP actuels : l'algorithme A^* et l'algorithme de Viterbi synchrone. Nous présentons également une variante de l'algorithme de Viterbi (Viterbi-Bloc) qui permet une réponse en temps réel au fur et à mesure de la prononciation de la phrase sans attendre la fin de celle-ci.

Pour plus d'informations sur les algorithmes de reconnaissance, le lecteur pourra se référer à [50, 159, 201].

1.8.1 L'algorithme A^*

Les méthodes de recherche dans un graphe ont été utilisées depuis longtemps pour générer les mots reconnus dans un signal de parole. La plus adaptée à la RAP est l'algorithme A^*. Nous décrivons dans ce qui suit cet algorithme (inspiré des algorithmes de programmation dynamique) ainsi que son adaptation à la reconnaissance.

Soit un graphe G, défini par un couple (S,A) où S représente l'ensemble de noeuds et A représente l'ensemble des arcs (couples de noeuds de S). Un chemin dans G est donc une suite de noeuds liés entre eux par des arcs. Dans le cas de la RAP à base de HMMs, le graphe G représente l'ensemble des chemins possibles dans le réseau acoustique. Un noeud est défini par une position fixée par la paire (s_i,t), où s_i est un état du réseau acoustique et t un instant. En revanche, les arcs sont définis par une règle de progression à l'intérieur de ce même réseau acoustique. L'objectif des algorithmes de recherche dans un graphe est de déterminer le chemin entre le noeud initial et le noeud final, avec un effort de génération peu coûteux.

Dans le cas de l'algorithme A^*, une information partielle d'ordre heuristique, sur le domaine du problème et sur la nature du but à atteindre, est utilisée pour aider le guidage de la recherche vers une direction prometteuse. En effet, lors de la recherche, une pile d'hypothèses contenant des séquences distinctes de mots se terminant à différents instants est gardée. A chacune de ces

4. stack-decoders

hypothèses, s'achevant à un noeud n, est associée un score $f^*(n)$ qui correspond à une estimation du coût minimal du chemin passant par le noeud en cours n et arrivant au noeud final. Ce score $f^*(n)$ se décompose en deux parties qui dépendent de la "distance" du noeud n à la situation initiale $g^*(n)$ et de sa distance au noeud final $h^*(n)$, soit : $f^*(n) = g^*(n) + h^*(n)$. La fonction $g^*(n)$ n'est en fait que le score du meilleur chemin (mesuré en : $-logProb()$) déjà rencontré et conduisant de l'état initial au noeud courant n. En revanche, l'estimation de la fonction $h^*(n)$ est plus difficile puisqu'on ne connaît pas, au moment où l'on traite le noeud n, de chemin conduisant au noeud final. Il est alors nécessaire de se référer à des informations heuristiques permettant de faire un choix dit "non déterministe". Pour estimer les fonctions $g^*(n)$ et $h^*(n)$, une solution consiste à leur affecter respectivement les probabilités $\alpha(n)$ et $\beta(n)$ de l'algorithme forward-backward [104]. Ces probabilités sont les résultats d'une combinaison entre le score acoustique et celui du modèle de langage. Le chemin possédant le score estimé le plus élevé est étendu. Lorsque l'un de ces chemins atteint le noeud final, si le score correspondant est encore le plus élevé, il est l'hypothèse d'une phrase reconnue. Pour avoir les N meilleures hypothèses, il suffit de garder les N meilleurs chemins arrivant au noeud final. L'algorithme A^* n'a donc pas besoin de parcourir l'espace complet pour trouver une solution [84, 153, 186].

1.8.2 L'algorithme Viterbi synchrone

L'algorithme de type Viterbi synchrone, inspiré des algorithmes de programmation dynamique, comme l'algorithme A^*, est le plus utilisé en RAP ; mais il ne faut pas oublier que de nombreuses adaptations de cet algorithme ont du être réalisées, pour la mise en oeuvre d'algorithmes capables de reconnaître la parole continue pour un grand vocabulaire. Le plus utilisé de nos jours en RAP est l'algorithme *à une passe* (*One Pass*). Cet algorithme a été proposé par Vintsyuk en 1971 et a été "redécouvert" plusieurs fois dans les deux dernières décennies. L'idée de base derrière l'algorithme est illustrée sur la figure 1.7 [27]. Cet algorithme choisit un axe pour la phrase prononcée, Γ représenté par l'axe horizontal, et un autre axe pour l'ensemble des mots de référence (vocabulaire), $\{\Re_1, \Re_2, \ldots, \Re_V\}$ représenté par l'axe vertical.

Utilisons la notation standard de m $(1 \leq m \leq M)$ pour représenter l'indice de symbole (phonème ou trame) de la phrase prononcée, v $(1 \leq v \leq V)$ pour représenter l'indice de mot de référence $(\Re_v)$, n $(1 \leq n \leq N_v)$ pour représenter l'indice du symbole (phonème ou trame) du mot de référence $\Re_v$, et $p(j,n,v)$ pour représenter la pénalité de passer de la forme de référence $\Re_v(j)$ à la forme de référence $\Re_v(n)$. La distance (en $log(Prb()))$ cumulée de chaque symbole de la phrase prononcée, notée $d_A(m,n,v)$, sera calculée comme suit :

$$d_A(m,n,v) \;=\; d(m,n,v) \;+\; \min_{n-1 \leq j \leq n} (d_A(m-1,j,v) + p(j,n,v)) \tag{1.5}$$

Nous calculons la formule 1.5 pour $2 \leq n \leq N_v$, $1 \leq v \leq V$, où $d(m,n,v)$ est la distance locale entre le symbole de la phrase prononcée $t(m)$ et la forme de référence $\Re_v(n)$.

La récurrence de la formule 1.5 est menée sur tous les symboles internes de chaque mot de référence $(n \geq 2)$. A la frontière du mot de référence, $n = 1$, on a la simple récursivité suivante :

$$\begin{aligned} d_A(m,1,v) \;=\; d(m,1,v) \;+\; \min_{1 \leq r \leq V} \; & [(d_A(m-1,\, N_r,\, r) + p(N_r,1,v)), \\ & (d_A(m-1,\, 1,\, v) + p(1,1,v))] \end{aligned} \tag{1.6}$$

La figure 1.8.a montre que pour chaque symbole $\Re_v(n)$ interne $(n \geq 2)$, l'algorithme choisit le meilleur chemin, interne au mot de référence $\Re_v$, ramenant à $\Re_v(n)$. Ce chemin est choisi suivant le principe de la programmation dynamique, en effectuant l'hypothèse du premier ordre

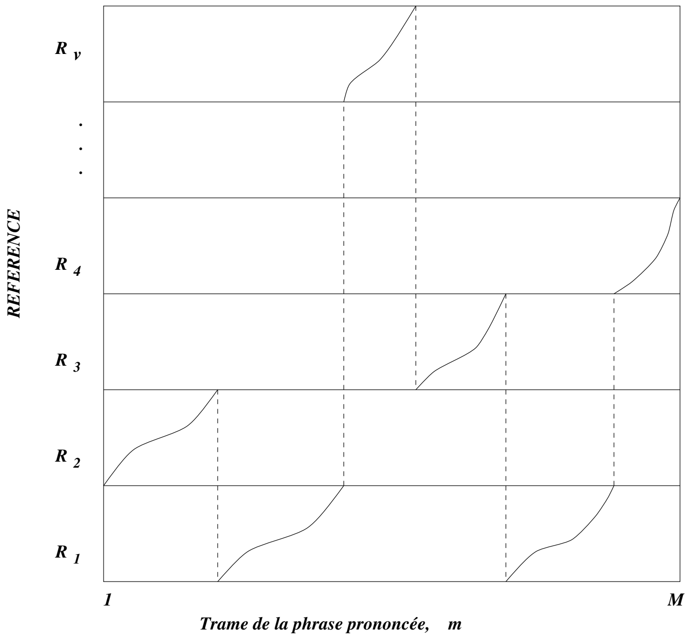


Fig. 1.7 – *L'algorithme de reconnaissance de parole à une-passe (One-Pass)*

à l'intérieur d'une unité $\Re_v$: pour calculer le score à l'instant m, seul le score du même état $\Re_v(n)$ ou de l'état précédent $\Re_v(n-1)$ à l'instant $m-1$ est pris en compte [201]. En revanche, pour définir le meilleur chemin arrivant au premier symbole $\Re_v(1)$ de chaque mot de réference (cf. figure 1.8.b), l'algorithme tient compte :

- du chemin provenant du même état $\Re_v(1)$ à l'instant d'avant,
- des chemins provenant des états finaux de tous les mots de référence ($\Re_r(N_r)$, $1 \leq r \leq V$). La valeur de $p(N_r,1,v)$ est obtenue en combinant le score acoustique et le score du modèle de langage.

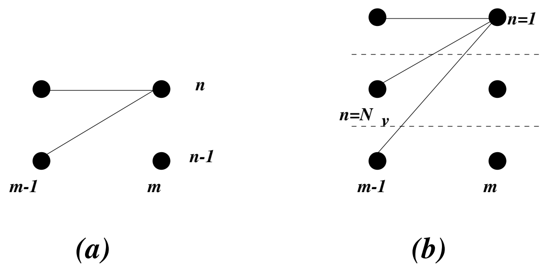


Fig. 1.8 – *Les chemins possibles de l'algorithme à une-passe (One-Pass)*

La solution finale pour le meilleur chemin (correspondant à la meilleure chaîne de mots) est :

$$D^* = \min_{1 \leq v \leq V} [d_A(M, N_v,v)]. \tag{1.7}$$

La séquence de mots correspondant à la solution optimale est ensuite récupérée par une procédure de recherche arrière (back track). Sachant que pour obtenir la séquence de mots correspondant au chemin optimal, seules les transitions entre les mots doivent être gardées. Il suffit de mémoriser pour chaque symbole prononcé $m(1 \leq m \leq M)$, le mot $\Re_v(1 \leq v \leq V)$ pour lequel la distance cumulée $d_A(m,n,v)$ est minimale et l'indice de symbole où le mot a commencé. Par conséquent, un simple algorithme de parcours de ces informations permet d'obtenir la séquence de mots optimale. En gardant la trace des N meilleurs scores (équation : 1.7), on récupérera les N meilleures séquences de mots.

La principale différence entre cette approche et les autres est formalisée par la formule 1.6. Cette approche permet à un chemin arrivant au dernier symbole d'un mot au niveau $(l-1)$ de devenir le chemin du symbole de début de mot au niveau l. Le principal avantage de l'algorithme à une-passe est que l'estimation des distances cumulées pour un symbole donné m ne dépend que des distances cumulées calculées au symbole $m-1$ [141].

Nous présentons dans ce qui suit deux extensions portées à cet algorithme, les méthodes d'élagage et la représentation en arbre, lui permettant de réduire le temps de calcul.

Les méthodes d'élagage

Pour chaque symbole (phonème ou trame) de début de mot et pour chaque symbole t de la phrase prononcée, l'algorithme à une-passe parcourt tous les mots de référence pour choisir le meilleur symbole final de mot à l'instant $t-1$. Ceci correspond à $V^2 \times M$ traitements, où M est le nombre de symboles t de la phrase prononcée et V est le nombre de mots de référence, ce qui rend cet algorithme impraticable dans le cas de grands vocabulaires.

Puisque l'algorithme à une-passe ne conserve, pour chaque noeud et unité de temps, que la meilleure des hypothèses, on gagne, lors de son emploi, à utiliser une méthode permettant d'éliminer rapidement les hypothèses les moins prometteuses[5]. En effet, ces dernières risquent éventuellement d'entrer en compétition avec une autre hypothèse plus prometteuse et donc d'être abandonnées à son profit.

On remarque également que l'algorithme à une passe est de nature synchrone, ce qui veut dire que toutes les hypothèses actives le sont à une même unité de temps (ou colonne de matrice). La décision d'abandonner une hypothèse ne peut donc se faire, que sur la base de sa probabilité cumulée ou score courant.

Le principe de recherche en faisceau consiste donc à déterminer tout d'abord un seuil S. Ce seuil contrôle la fraction de mots actifs à chaque instant. Cette valeur est calculée statistiquement, de façon à obtenir un rapport acceptable, entre le taux de reconnaissance et le temps de calcul. Par conséquent, au niveau du symbole (phonème ou trame) suivant, $i+1$, de la phrase prononcée et pour chaque symbole (phonème ou trame) de début de mot, nous choisissons le meilleur symbole (phonème ou trame) final de mot parmi l'ensemble des mots actifs [77, 188].

La représentation en arbre

La manière dont le lexique (les mots du vocabulaire) est représenté, permet de déterminer l'espace mémoire requis ainsi que le temps de calcul nécessaire. La façon la plus simple de représenter le lexique est la représentation matricielle (comme le montre la figure 1.7). Avec cette représentation, chaque mot du lexique est parcouru séparément même s'il y a préfixe en commun avec d'autres mots du lexique.

Dans le cas d'une représentation en arbre, on n'alloue que l'espace mémoire exactement requis pour loger le lexique, ce qui permet de gagner en espace mémoire et également en temps de calcul. En effet, un préfixe commun à plusieurs mots n'est représenté qu'une seule fois et donc il n'est

5. Cette méthode est appelée recherche en faisceaux ou *pruning*.

parcouru qu'une seule fois dans la recherche. Ainsi, le lexique du tableau 1.1 est représenté en arbre sur la figure 1.9.

Indice du mot	**Représentation graphique**	**Représentation phonétique**
1	Papa	/p A p A/
2	Patate	/p A t A t @/
3	Patte	/p A t @/
4	Parent	/p A R a∼ /
5	Rampa	/R a∼ p A/
6	Rampant	/R a∼ p a∼ /

TAB. 1.1 – *Mots d'un mini-lexique.*

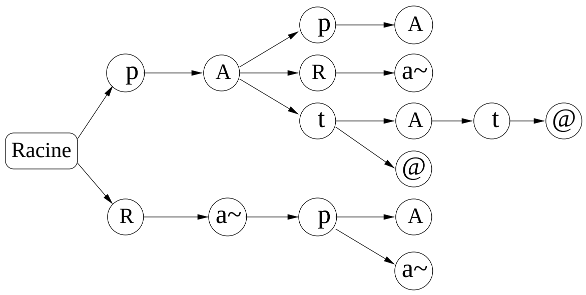


FIG. 1.9 – *Graphe représentant un mini-lexique.*

Si dans la représentation matricielle classique (algorithme de Viterbi de base) on connaît un mot dès son début, la représentation en cours (représentation en arbre) n'a pas cet avantage. En effet, la connaissance du mot en cours d'exploitation n'étant possible qu'à la rencontre de la feuille correspondante. L'inconvénient de ne pas connaître le mot à son début réside surtout au niveau de l'application du modèle de langage. En effet, dans une représentation matricielle à la frontière entre les mots, on connaît le mot à parcourir ainsi que le mot qui précède, ce qui n'est pas le cas dans une représentation en arbre. On est donc obligé de garder la trace des mots précédents pour introduire le score du modèle de langage une fois que la feuille de l'arbre est atteinte. Ceci nécessite de l'espace mémoire et du temps de calcul supplémentaire, qui reste largement inférieur à ce que nécessite la représentation matricielle classique.

La représentation arborescente du lexique permet de gagner un espace mémoire considérable. Toutefois, le lexique n'est pas toujours représenté d'une manière optimale. Par conséquent, plusieurs algorithmes de compression ont été proposés [147, 135]. L'idée de base de ces algorithmes est de factoriser les préfixes ainsi que les suffixes communs des mots sans répétition.

1.8.3 L'algorithme de Viterbi-Bloc

L'idée principale de cet algorithme est d'utiliser, pendant la reconnaissance, des critères fondés sur une notion d'optimalité locale plutôt que globale [78]. En procédant localement, les algorithmes mettent en oeuvre des structures moins volumineuses [130]. Rappelons que l'algorithme de Viterbi (*One-Pass*), construit et préserve à chaque instant, tous les chemins partiels s'achevant sur chaque état du HMM alors qu'un seul chemin, issu de l'état final, sera utilisé pour retrouver la suite des états empruntés.

L'algorithme de Viterbi-Bloc effectue une segmentation simultanément avec la reconnaissance et donne une réponse en temps réel au fur et à mesure de la prononciation de la phrase sans attendre la fin de celle-ci. Le principe général consiste à utiliser une fenêtre glissante de N symboles (trames ou phonèmes) de parole dans laquelle une décision est prise localement [26, 116]. Il n'est plus nécessaire de stocker des informations relatives à la totalité de la phrase. La place mémoire nécessaire à l'exécution ne dépend que de la taille de la fenêtre de construction, ce qui rend cet algorithme intéressant pour la reconnaissance de longues phrases. Nous décrivons en détail, dans le chapitre 8, cet algorithme ainsi que l'adaptation que nous lui avons apportée pour l'utiliser dans notre système de RAP.

1.9 Conclusion

Dans ce chapitre nous avons décrit brièvement l'intérêt, l'historique, ainsi que les différentes composantes nécessaires pour la réalisation des systèmes de reconnaissance vocale. L'approche probabiliste à base de modèles de Markov cachés (HMMs) est la plus utilisée, de nos jours, et donne ainsi de bons résultats au niveau acoustique. Néanmoins, les systèmes de RAP sont encore pauvres au niveau langagier. En effet, une bonne partie des erreurs de ces systèmes est due à un non-respect de l'ensemble des contraintes syntaxiques ou sémantiques de la langue parlée. L'intégration d'autres contraintes langagières permet donc l'amélioration de leurs performances.

De nouveaux modèles de langage originaux, intégrés à notre système de RAP MAUD (cf. chapitre 8), sont proposés dans les chapitres 4, 5 et 6. Avant tout cela, nous décrivons dans le chapitre 2 quelques notions de la théorie de l'information, utile pour l'estimation et l'évaluation de ces modèles. Ensuite, nous présentons dans le chapitre 3 certains modèles de langage, parmi les plus utilisés en RAP.

Chapitre 2

Quelques notions en théorie de l'information

La théorie de l'information a vu le jour avec le développement des télécommunications, où un des problèmes était de transmettre des données le plus rapidement possible, compte tenu des propriétés de la ligne. Plusieurs notions fondamentales ont été ainsi introduites en particulier par Hartley et par Shannon [178]. Cette théorie a montré également son utilité dans d'autres disciplines comme la cybernétique et la thermodynamique : on trouvera dans les ouvrages de L. Brillouin [28] et de O. Costa de Beauregard [40], une analyse de ce dernier aspect.

Une des plus grandes vertus de la théorie de l'information du point de vue de Shannon est de fournir une méthode permettant d'étudier intrinsèquement les propriétés générales des messages émis par une source. Le message que nous utilisons ici, qui est à la base de cette théorie, est considéré comme un ensemble d'éléments de perception qui déterminent la réaction et le comportement de la source qui l'a émis. Ces éléments de perception peuvent être, soit des signes naturels, soit des symboles artificiels, ou signaux traduisant des symboles. Ce qu'il importe de connaître, ce n'est pas comment un message particulier peut être créé par une source, c'est-à-dire l'origine de l'information, mais comment cette source peut agir. Ce n'est pas la signification d'un certain message qui nous intéresse, mais l'ensemble des messages qui peuvent être exprimés. Les différentes notions, présentées dans ce chapitre, sont introduites dans cet esprit et les résultats ne doivent être utilisés que dans ce contexte.

Dans les deux paragraphes qui suivent, nous exposons quelques propriétés (entropie et information mutuelle) relatives à la quantité d'information apportée par une source (par exemple, la langue) et par les modèles de langage qui peuvent la modéliser. Nous donnons ensuite quelques méthodes, autres que les systèmes de RAP, d'évaluation des performances des modèles de langage.

2.1 Entropie

En théorie de l'information, la langue est considérée comme une source L [1], qui émet une séquence de symboles (mots) w_1,w_2,$\ldots w_N$ à partir d'un ensemble fini d'éléments (vocabulaire). A chaque émission d'un nouveau symbole (mot) w_i, la source apporte une quantité d'information mesurable par l'incertitude que la connaissance du symbole w_i retire au destinataire. On note $H(L)$ la valeur moyenne de cette quantité d'information, encore appelée entropie de la source par analogie avec la mesure de désordre en thermodynamique.

Une source L délivre une quantité d'information importante, lorsque l'incertitude à propos du prochain symbole à émettre w_i est grande. Cette incertitude et l'information apportée par celle-ci sont maximales lorsque tous les symboles à émettre w_i sont équiprobables (source uniforme). Dans ce cas, l'imprédictibilité du message émis par la source ne peut pas être réduite.

Faire l'hypothèse que la séquence de symboles, $w_1,w_2,\ldots w_N$, émis par la source L s'articule autour d'un nombre de motifs revient à supposer que les symboles émis n'ont pas une distribution uniforme, et qu'il existe une dépendance entre eux. Dans le cas où la source L est gouvernée par un processus déterministe, la séquence $w_1,w_2,\ldots w_N$ devient parfaitement prévisible : les émissions de la source n'apportent plus aucune information. Par conséquent, l'entropie de cette source L devient nulle : $H(L) = 0$. En revanche, si le processus gouvernant la source est aléatoire, alors ses émissions apportent une quantité d'information non nulle : $H(L) > 0$ [105]. C'est d'ailleurs le cas des langues naturelles comme le français, où la suite de mots émise est soumise à des contraintes syntaxiques et sémantiques. Ainsi, l'émission d'un nouveau mot, qui est fortement dépendante de l'identité des mots émis précédemment, apporte une quantité d'information non nulle.

Exemple : Soit une séquence (suite de symboles), émise par une source L, ne contenant que les symboles "a" et "b". Si nous supposons que la source L est gouvernée par un processus déterministe, qui n'émet qu'une seule et même séquence $[aba]$, la seule suite de symboles qui peut être émise par L est "...abaabaabaabaaba...". Par conséquent, l'entropie de la source L régie par ce processus déterministe est nulle.

En revanche, si nous supposons que le processus gouvernant la source L attribue des probabilités d'émission à "a" et à "b" égales respectivement à $\frac{2}{3}$ et à $\frac{1}{3}$, on ne pourra pas prédire exactement les suites de symboles émises. Ainsi, le processus n'est pas déterministe et l'entropie de la source L régie par ce processus est non nulle.

2.1.1 Calcul de l'entropie

Dans ce paragraphe, nous présentons directement le calcul de l'entropie tel qu'il est défini dans le cadre des modèles de langage probabilistes. En théorie de l'information, l'imprédictibilité d'un événement se mesure comme l'opposé du logarithme de sa probabilité. Ainsi, plus un événement est improbable, plus il est imprédictible, conformément à notre intuition. L'imprédictibilité, ou encore la quantité d'information contenue dans une suite de mots $w_1,w_2\ldots w_N$, émise par une langue, vaut :

$$-log_b P(w_1,w_2\ldots w_N)$$

où $P(.)$ est la vraisemblance de la suite de mots en argument.

En se fondant sur la mesure de désordre en thermodynamique, l'entropie d'une source qui émet des symboles indépendamment de leur historique est :

$$H_b(L) = -\sum_{w_i} P(w_i) log_b P(w_i) \tag{2.1}$$

où w_i est le symbole émis avec une probabilité $P(w_i)$. Le choix de la base du logarithme (log_b) définit l'unité d'information. Pour une valeur de b équivalant à 2, l'unité est le *bit*. Dans la suite, les logarithmes seront exprimés en base 2, qui est la base habituellement utilisée en transmission de l'information. En effet, d'après la théorème de Shannon [177], le codage d'une source L nécessite au moins $H_2(L)$ bits par symbole (mot). On notera ainsi $logx$ à la place de $log_2 x$, et $H(L)$ pour $H_2(L)$.

Pour évaluer la quantité d'information d'une langue, il faudrait observer ses réalisations pendant un temps infini, de sorte que son entropie se déduise naturellement en sommant sur l'ensemble des N-uplets de mots susceptibles d'être observés et en faisant tendre N vers l'infini :

$$H(L) = -\lim_{N\to\infty} \frac{1}{N}(\sum P(w_1,w_2 \dots w_N) logP(w_1,w_2 \dots w_N)). \tag{2.2}$$

En pratique, il n'est pas possible d'observer toutes les réalisations $w_1,w_2 \dots w_N$ pouvant résulter de l'activité de la langue[6]. C'est pour cela que l'on suppose que toute suite suffisamment longue de ces émissions est représentative de son comportement, et permet d'en étudier la structure statistique (source ergodique). L'entropie de la langue est par conséquent approximée par la quantité d'information apportée par la seule réalisation $w_1,w_2 \dots w_N$:

$$H(L) \simeq -\frac{1}{N} logP(w_1,w_2 \dots w_N) \tag{2.3}$$

2.1.2 L'entropie en RAP

Dans un système de RAP, la quantité d'information contenue dans l'ensemble des mots déjà reconnus permet de réduire l'incertitude sur la phrase prononcée. Par conséquent, on peut prédire les mots qui peuvent plus vraisemblablement suivre un historique donné (les mots déjà reconnus par le système), ce qui permet de réduire le facteur de branchement. En effet, trouver le modèle qui simule le comportement de la langue permet d'apporter un aspect prédictif à la reconnaissance : il lui permet de prédire quels sont les mots qui ont une chance d'apparaître dans les instants qui suivent. La recherche de la phrase prononcée se fera alors, en priorité, sur les mots les plus vraisemblables ; d'où un gain en temps et en puissance de reconnaissance.

L'entropie est une bonne mesure de la réduction du facteur de branchement fourni par un modèle de langage donné. En effet, examinons la représentation de la fonction $f : x \to -(x.log_2 x + (1-x).log_2(1-x))$ (cf. figure 2.1), qui donne la valeur de l'entropie pour deux valeurs de probabilité complémentaires (dans le cas par exemple, d'un vocabulaire contenant deux mots). On réduit cette étude à l'intervalle]0,1] car, dans le cas de l'entropie, f prend en argument des valeurs correspondant à celles de probabilités. Comme on le voit sur la figure 2.1, l'entropie donne sa plus petite valeur, 0, quand l'une des deux probabilités est égale à 1 (et donc l'autre est égale à 0). Inversement, elle donne sa plus grande valeur, 1, quand les deux probabilités sont de valeurs identiques. Cette constatation se généralise pour k probabilités (dont la somme est égale à 1). En effet, si une probabilité est égale à 1, l'entropie est nulle, et si toutes les probabilités sont égales à $\frac{1}{k}$, l'entropie est de valeur maximale : $log_2 k$, où k est la taille du vocabulaire.

D'après la formule 2.2, une faible entropie engendre de nombreuses probabilités proche de 0 ou 1. Le modèle gouvernant le comportement de la langue peut alors être qualifié de déterministe. En effet, dans un système de RAP, en fonction d'un historique h déjà reconnu, le modèle pourrait éliminer des hypothèses (mots de probabilité conditionnelle $P(w_i/h)$ proche de 0), et retenir les quelques unes qui sont proches de 1. Une bonne entropie (une valeur faible) revient donc à un abaissement du facteur de branchement dans un système de RAP [195].

Si le modèle gouvernant le comportement de la langue est capable de décider pour de nombreuses chaînes si oui ou non ces chaînes appartiennent au langage, la formule 2.2 de ce modèle donnera des petites valeurs. Si au contraire les probabilités de ces chaînes sont semblables, la

6. Dans le cas d'une langue morte, quand le temps tend vers l'infini, la production d'une langue morte se tarit. Par conséquent, tous les écrits depuis le commencement peuvent être considérés comme une seule grande phrase, qui est l'unique phrase W de la langue. $P(W)$ serait alors égal à 1 et les autres phrases seraient de probabilité nulle. L'entropie est donc égale à 0.

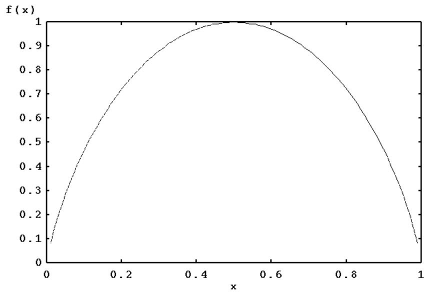


FIG. 2.1 – *Représentation de la fonction* $f : x \rightarrow -(x.log_2 x + (1-x).log_2(1-x))$

valeur de l'entropie est proche du maximum, cela signifie que le modèle n'a pas assez de connaissance sur la langue. Il est donc inadapté (ou du moins pas assez adapté) à ce langage. Ainsi, plus l'entropie d'un modèle de langage est faible, plus le modèle apporte des informations pertinentes sur le langage.

2.1.3 Autres visions de l'entropie

Dans [129], M. Fédérico et R. de Mori donnent une définition de l'entropie croisée (*cross-entropy*) :

$$H(P,\widehat{P}) = -\frac{1}{N}\sum_{W} P(W)log(\widehat{P}(W)) \tag{2.4}$$

où W représente une phrase du langage, $P(W)$ est la probabilité d'apparition de W dans la langue (sa fréquence d'apparition dans un corpus est supposée être assez importante), et $\widehat{P}(W)$ est la vraisemblance de cette même phrase W donnée par un modèle de langage. N est la taille du corpus sur lequel est calculé l'entropie croisée. On somme sur toutes les chaînes du corpus.

Cette formule donne une mesure plus fine que l'équation 2.2 pour l'adéquation du modèle au langage. En effet, la figure 2.2 qui définit l'entropie croisée pour deux chaînes de probabilités complémentaires (cet exemple d'étude a déjà été utilisé pour l'entropie en 2.1.2 (cf. figure 2.1)), montre que cette entropie est toujours supérieure à $H(P,P)$. Plus l'entropie croisée $H(P,\widehat{P})$ se rapproche de $H(P,P)$, meilleure sont les capacités prédictives du modèle de langage.

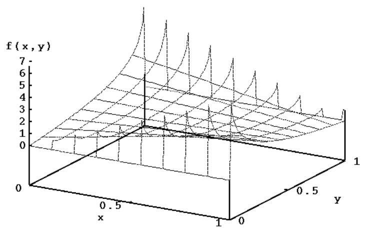


FIG. 2.2 – *Représentation de la fonction* $f : (x,y) \rightarrow -(y.log_2 x + (1-y).log_2(1-x))$

2.2 Information mutuelle

L'information mutuelle est considérée comme une mesure quantitative de l'information fournie par une variable aléatoire X sur une autre variable aléatoire Y [1]. L'information mutuelle est par conséquent définie comme étant la réduction apportée à l'entropie de Y sachant que X est connue [7]. En utilisant la formule 2.1, on obtient :

$$\begin{aligned} I(X:Y) &\stackrel{def}{=} H(Y) - H(Y/X) \\ &= H(Y) - \sum_x P(x)H(Y/x) \\ &= -\sum_y P(y)logP(y) + \sum_x P(x)\sum_y P(y/x)logP(y/x) \\ &= \sum_{x,y} P(x,y)log\frac{P(x,y)}{P(x)P(y)} \end{aligned} \tag{2.5}$$

Plusieurs propriétés peuvent être déduites :

- l'information mutuelle est symétrique : $I(X:Y) = I(Y:X)$,
- si les variables aléatoires X,Y sont indépendantes ($P(x,y) = P(x)P(y)$), l'information mutuelle est nulle $I(X:Y) = 0$.

Dans une langue, la quantité d'information apportée par un mot w_i, sachant un historique h connu, est définie par l'information mutuelle $I(h:w_i)$. Si on se limite au dernier mot w_{i-1} dans l'historique, et si on suppose que le corpus utilisé pour l'estimation des vraisemblances est représentatif du comportement de la langue, l'information mutuelle, estimée à partir de l'entropie définie par la formule 2.3, vaut :

$$\begin{aligned} I(w_{i-1}:w_i) &\stackrel{def}{=} log\frac{P(w_{i-1},w_i)}{P(w_{i-1})P(w_i)} \\ &= log\frac{C(w_{i-1},w_i)N}{C(w_{i-1})C(w_i)} \end{aligned} \tag{2.6}$$

où $C(.)$ est l'occurrence de la suite de mots en argument et N est la taille du corpus.

L'information mutuelle $I(h:w_i)$ mesure la quantité d'information existante dans l'historique h pour la prédiction du mot w_i. Elle ne mentionne pas comment l'information peut être extraite entièrement, encore moins comment l'extraire. Elle ne donne qu'une estimation de sa limite supérieure. L'information mutuelle permet également de mesurer le degré d'association entre deux mots w_j et w_i, $I(w_j:w_i)$. Elle reflète ainsi l'importance du lien qui existe entre w_j et w_i. R.L. Mercer mentionne que deux mots w_1,w_2 qui se suivent et qui ont une information mutuelle assez grande peuvent être considérés comme une unité (séquence de deux mots) de la langue [93]. Par conséquent, la langue peut être considérée comme une suite de séquences de mots plutôt qu'une suite de mots.

2.3 Mesure de la qualité d'un modèle de langage

Mesurer la qualité d'un modèle de langage $p(m)$ revient à examiner son impact sur les performances de l'application pour laquelle il est dédié. En RAP, on se fonde sur son impact sur le taux d'erreur de reconnaissance. En pratique, il est souvent très difficile d'utiliser cette mesure :

7. Quelques auteurs considèrent $I(X:Y)$ comme étant l'information mutuelle *moyenne*.

mesurer le taux d'erreur nécessite le traitement d'une grande quantité de données, qui prend énormément de temps de calcul. De plus, le taux d'erreur est le résultat d'un ensemble d'interactions complexes et souvent non linéaires entre plusieurs composantes. Il est généralement impossible de trouver une expression analytique déterminant le rapport entre le taux d'erreur et les différents paramètres du modèle de langage. Par conséquent, un apprentissage automatique qui minimise directement le taux d'erreur est généralement très difficile [165].

Malgré ce qui précède, le paradigme des N meilleures hypothèses produites par un système de reconnaissance [174], donne une optimisation directe du taux d'erreur, au moins partiellement faisable. Une fois que le système produit les N meilleures hypothèses, un post-traitement est appliqué pour les réévaluer et ainsi fournir l'hypothèse qui possède le meilleur score. Sous ces conditions, plusieurs paramètres du modèle de langage peuvent être rapidement accordés pour optimiser le traitement [85].

2.3.1 Perplexité

Une autre méthode pour évaluer les performances d'un modèle de langage M est d'étudier son pouvoir prédictif sur un certain texte T jusqu'ici invisible. Ceci peut être mesuré par l'entropie de M générant T. Cette entropie $H(M)$ peut également être considérée comme la quantité de surprise de voir T, quand on utilise M pour anticiper les événements. Souvent, c'est la perplexité qui est utilisée pour évaluer les performances d'un modèle de langage [97] :

$$PP(M) = 2^{H(M)} \tag{2.7}$$

Dans la langue, les modèles sont généralement appris à partir d'un corpus de mots suffisamment grand, représentatif de son comportement. Par conséquent, la perplexité sera déduite de l'entropie calculée par la formule 2.3 [93] :

$$PP(M) = 2^{-\frac{1}{N} log P(w_1, w_2 ... w_N)} \tag{2.8}$$

où $P(w_1, w_2 \ldots w_N)$ est la vraisemblance de la chaîne de mots $w_1, w_2 \ldots w_N$ et N est la taille du corpus.

La perplexité peut être interprétée comme l'inverse de la moyenne géométrique de la vraisemblance de la suite de mots qui constituent le corpus. Cette valeur de perplexité est également interprétée comme le facteur de branchement d'un système de RAP. Ainsi, un système de RAP, utilisant un modèle de langage de perplexité X, peut se limiter à parcourir (en moyenne), à chaque instant t, les X meilleurs mots prédits par le modèle de langage (cf. §1.8.2).

D'après la formule 2.8, la perplexité est une fonction qui dépend du modèle et du corpus. Ce fait doit être pris en compte, lors de la comparaison des perplexités de différents modèles avec des corpus différents. Une comparaison significative peut être faite entre les différents modèles, en gardant les mêmes corpus et le même vocabulaire. Pour un même vocabulaire et un même corpus, le modèle qui a la plus petite perplexité est par conséquent le plus performant. En pratique, une petite perplexité n'engendre pas toujours une diminution du taux d'erreur dans un système de reconnaissance, bien que ce soit souvent le cas, surtout quand la réduction de la perplexité est significative. La perplexité ne donne donc qu'une appréciation moyenne de la qualité d'un modèle de langage.

La perplexité ne tient pas compte des problèmes de confusion au niveau acoustique. P. De-Souza suggère ainsi la perplexité acoustique. M. Feretti *et al.* dans [57], suggèrent plutôt l'utilisation de l'entropie du système de RAP (*Speech Decoder Entropy*), qui combine l'information des modèles acoustiques et de langage. Cette information est typiquement inférieure à la somme de leurs contributions individuelles, puisqu'une partie de l'information se superpose. En utilisant un corpus de test, nous avons constaté que les sources acoustiques et linguistiques sont plutôt complémentaires.

2.3.2 Jeu de Shannon

La quantité d'information apportée par une source (langue) M peut être évaluée en faisant tester cette source par un humain. Pour ce faire, la personne (experte ou non) doit utiliser la source M, optionnellement d'autres sources, et de nombreuses règles pour manipuler les données. La tâche de cette personne est d'essayer de prédire le prochain mot w_i sachant un historique h déjà émis par la source M. Cette même opération sera répétée dans les mêmes conditions à l'exception de l'historique h. La différence entre les taux de prédiction avec et sans historique donne une idée sur la quantité d'information existante dans la source M une fois l'historique h émis. Ces deux opérations doivent être répétées avec des données différentes (historiques différents), et/ou des personnes différentes. Ceci est une généralisation du jeu de Shannon, utilisé par C.E. Shannon pour estimer l'entropie de l'anglais [178].

Si l'expérience est correctement réalisée, la différence entre les deux essais (avec et sans historique) peut seulement être attribuée à la source M. D'autre part, il est possible que plusieurs informations existantes dans l'historique h émis par la source M n'aient pas été correctement exploitées. Cependant, cette expérience ne donne qu'une limite inférieure de l'information existante dans la source M. Les techniques utilisées dans les modèles de langage actuels, qui eux également peuvent évaluer la quantité d'information apportée par une source M, sont souvent moins performantes que les humains au niveau de la prédiction ; les performances humaines sont rarement atteintes. Par conséquent, les performances de l'humain dans un tel jeu sont souvent considérées comme la limite supérieure de la quantité d'information apportée par une langue.

Le jeu de Shannon a été implanté et testé par IBM. Les résultats ont été utilisés pour estimer la quantité d'information apportée par la phrase courante par rapport à la phrase précédente. Ces mêmes résultats ont permis également d'encourager les recherches concernant le traitement d'un historique de longue distance, émis par une source comme la langue, lors de la prédiction [165].

D'autres variantes de ce jeu de Shannon ont été décrites dans la littérature (voir chapitre 7). Une d'entre elles a été choisie comme une méthode supplémentaire pour l'évaluation de nos modèles de langage.

2.4 Conclusion

Dans ce chapitre, nous avons décrit brièvement quelques notions fondamentales en théorie de l'information. Tout d'abord, nous avons décrit l'entropie et l'information mutuelle, qui sont nécessaires à l'étude des propriétés d'une source d'information comme la langue. Ensuite, nous avons présenté d'autres notions, telles que la perplexité et le jeu de Shannon, permettant d'estimer les performances des modèles de langage indépendamment des systèmes de RAP. Nous décrivons brièvement dans le prochain chapitre certains modèles de langage, parmi les plus connues et utilisées dans les systèmes de RAP.

Chapitre 3

Les modèles de langage probabilistes

3.1 Introduction

Toute application informatique traitant du langage naturel a besoin d'un modèle lui permettant de caractériser, de capturer et d'exploiter les régularités de la langue traitée : le modèle de langage. L'être humain utilise implicitement et inconsciemment un ensemble de connaissances lui permettant de capturer, de caractériser et d'exploiter les régularités de la langue traitée.

Nous exposons dans le paragraphe suivant quelques domaines d'application des modèles de langage, en particulier la reconnaissance vocale. Nous présentons brièvement dans le paragraphe 3.3 l'avantage d'utiliser un modèle probabiliste par rapport à un modèle structurel à base de connaissances. Ensuite, dans les paragraphes suivants, nous décrivons les modèles de langage les plus utilisés en RAP ainsi que la manière de les évaluer. L'objectif des modèles de langage probabilistes est d'estimer la probabilité $P(W_1^T)$ d'une séquence de mots $W_1^T = w_1,w_2,\ldots w_T$. Ces modèles doivent également être capables d'estimer les probabilités des suites de mots non rencontrées dans le corpus d'apprentissage.

3.2 Quelques domaines d'application

Une utilisation des modèles de langage se trouve dans la reconnaissance automatique de la parole. Cette appréciation du rôle de la connaissance linguistique a mené au développement de plusieurs modèles, qui sont la plupart du temps de nature probabiliste. Comme mentionné auparavant dans le chapitre 1, ces modèles probabilistes attribuent à une phrase une valeur de vraisemblance, qui est interprétée comme la probabilité *a priori* pour qu'un locuteur l'énonce. La prise en compte de cette valeur lors du processus de reconnaissance permet d'écarter les hypothèses créditées d'un bon score acoustique, mais linguistiquement peu vraisemblables. Compte-tenu de la grande variabilité du signal de parole et compte-tenu de la difficulté actuelle d'obtenir une modélisation acoustique fiable, les modèles de langage ont un rôle déterminant dans la reconnaissance automatique de la parole. Il est à noter, que même dans le cas d'une modélisation acoustique fiable, l'utilisation des modèles de langage reste toujours primordiale pour respecter au maximum le comportement de la langue.

Une situation similaire existe dans le domaine de la traduction automatique. La traduction d'un langage naturel à l'autre implique beaucoup d'incertitudes dues à des phénomènes de détail de langage. En effet, les langages incluent des mots de sens multiples, des expressions idiomatiques, des contraintes d'ordre des mots, et d'autres. Les efforts réalisés dans les années cinquante au niveau de la traduction des langues naturelles, n'ont pas abouti à des résultats convaincants.

Des techniques d'étiquetage, d'analyse (qui peuvent être considérées comme une sorte de modélisation de langage) et d'autres ont été développées afin de traiter l'ambiguïté, la variabilité et l'idiosyncrasie des langues source et cible. Au début des années quatre-vingt-dix, des modèles probabilistes ont été utilisés et ont donné de bons résultats [29].

Il existe d'autres applications qui peuvent bénéficier des avantages des modèles de langage, nous citons, à titre d'exemple, la reconnaissance du texte écrit, la correction orthographique et grammaticale d'un texte, ainsi que l'identification automatique de thèmes. Dans le cas de la reconnaissance du texte écrit, le texte initial doit être récupéré à partir d'une image potentiellement tordue. Cette image est encore plus tordue si le texte est écrit à la main. Dans le cas de la correction orthographique et grammaticale d'un texte, on se fonde sur des facteurs psycholinguistiques, moteurs (fautes de frappe) et d'accord pour corriger les erreurs [119]. Pour la troisième application, on se fonde sur des analyses sémantiques pour identifier le thème d'un texte donné. Récemment des méthodes probabilistes ont été proposées et ont donné de très bons résultats dans le domaine d'identification de thèmes [21]. Dans ces trois applications, l'exploitation de la connaissance linguistique améliorera d'une façon remarquable leurs performances.

3.3 Avantage d'un modèle de langage probabiliste

Dans les modèles de langage structurels à base de connaissances, les connaissances linguistiques sont codées "à la main" par des experts suivant le domaine d'application. Ces modèles fournissent souvent des réponses booléennes de type vrai/faux.

Les modèles probabilistes possèdent des avantages par rapport aux modèles structurels :

- Dans les systèmes de RAP, au niveau de la prédiction, il est plus intéressant d'utiliser les probabilités produites par les modèles probabilistes plutôt que d'utiliser une réponse booléenne du genre "accepter"/ "refuser". En effet, les probabilités peuvent être combinées et manipulées autrement durant la reconnaissance. Ces probabilités permettent une meilleure prise en compte des hypothèses fournies par le système. De plus, il est quasi impossible de nos jours de construire un modèle déterministe structurel qui traite toutes les contraintes de la langue naturelle.
- La réalisation d'un modèle probabiliste ne nécessite pas l'intervention d'experts. En effet, à la différence des modèles probabilistes qui se construisent d'une manière automatique à partir d'un corpus, les modèles structurels ont toujours besoin d'un expert définissant leurs comportements.
- Une fois le modèle probabiliste développé et la procédure d'apprentissage programmée sur ordinateur, il peut être lancé automatiquement sur de nouvelles données. La création d'un modèle pour un nouveau domaine d'application peut donc être réalisée rapidement.
- Généralement, les modèles structurels à base de connaissances (par exemple, les analyseurs syntaxiques) sont coûteux en temps de calcul. Les modèles probabilistes ont tendance à fonctionner plus rapidement.

Les modèles probabilistes présentent également des inconvénients :

- Ils ne capturent pas le sens d'un texte. Par conséquent, des phrases absurdes peuvent être considérées comme probables par ces modèles (elles peuvent avoir des probabilités élevées).
- Les modèles probabilistes nécessitent une grande quantité de données pour l'apprentissage (corpus). Dans notre cas, pour l'estimation des modèles de langage avec un vocabulaire de 20000 mots, nous utilisons un corpus d'environ 50 millions de mots.
- Les modèles probabilistes ne se servent pas souvent, d'une façon explicite, des connaissances linguistiques et du domaine d'application. En effet, pour estimer la vraisemblance, quelques

connaissances utiles peuvent et doivent parfois être obtenues par des linguistes ou par des experts du domaine d'application. Ceci n'est pas le cas pour un modèle probabiliste qui se fonde surtout sur des connaissances statistiques.

Par conséquent, la modélisation du langage naturel peut bénéficier grandement de la complémentarité de deux approches qui sont : l'approche probabiliste prédictive et l'approche grammaticale, soucieuse de cerner les structures du langage naturel. Un modèle hybride bénéficiant de la complémentarité de l'approche probabiliste et de l'approche grammaticale est présenté dans le chapitre 4 [184].

3.4 Quelques modèles de langage probabilistes

L'objectif de ce paragraphe est de présenter les modèles de langage les plus utilisés dans le domaine de la reconnaissance de la parole. Nous ne cherchons pas à être exhaustif, vu le nombre élevé de variantes qui existent dans la littérature.

3.4.1 Les modèles n-grammes

Les modèles de langage probabilistes, qui reposent sur l'application de la règle des probabilités conditionnelles, définissent la vraisemblance d'une suite de mots $W_1^N = w_1,w_2,\dots w_N$ comme suit :

$$P(w_1,w_2,\dots w_N) = \prod_{i=1}^{N} p(w_i/w_1,\dots w_{i-1}) \tag{3.1}$$

où $p(w_i/w_1,\dots w_{i-1})$ est la probabilité du $i^{ème}$ mot de la suite W_1^N, sachant tous les mots précédemment émis.

En pratique, une estimation fiable des valeurs des probabilités conditionnelles des mots, $p(w_i/w_1,\dots w_{i-1})$, est impossible. Aucun corpus d'apprentissage, aussi long et représentatif soit il, ne peut permettre d'observer un nombre suffisant de fois toutes les chaînes de mots possibles $w_1,w_2,\dots w_N$. Dans l'approche n-grammes, on y remédie en considérant comme équivalents tous les passés se terminant par les mêmes $(n-1)$ mots. En effet, les modèles n-grammes reposent sur l'hypothèse que deux suites $M_1^k = m_1,m_2,\dots m_k$ et $W_1^t = w_1,w_2,\dots w_t$ sont équivalentes si les n derniers mots sont identiques pour chacune des deux suites de mots. Ainsi, seules les probabilités $p(w_i/w_{i-1},\dots w_{i-n+1})$ doivent être estimées. On parle alors de modèle bigrammes si $n=2$, trigrammes si $n=3$ et unigrammes dans le cas où $n=1$ (sans historique $p(w_i)$). La vraisemblance de toute suite de mots, $W_1^N = w_1,w_2,\dots w_N$, est donc approximée par [93, 95] :

$$P(w_1,w_2,\dots w_N) = \prod_{i=1}^{N} p(w_i/w_{i-n+1},\dots w_{i-1}) \tag{3.2}$$

Dans la version la plus simple du modèle, la probabilité du mot w_i sachant son contexte $w_{i-n+1},\dots w_{i-1}$ est estimée suivant le principe du maximum de vraisemblance sur un corpus d'apprentissage W représentant la langue :

$$p(w_i/w_{i-n+1},\dots w_{i-1}) = \frac{N(w_{i-n+1},\dots w_{i-1},w_i)}{\sum_{j\in W} N(w_{i-n+1},\dots w_{i-1},w_j)} \tag{3.3}$$

où $N(.)$ est le nombre d'occurrences de la suite de mots en argument dans le corpus W [93, 198]. D'autres méthodes d'estimation seront présentées dans le paragraphe 3.6.

En pratique, on ne peut pas rencontrer toutes les séquences de mots possibles dans un ensemble de phrases servant à l'apprentissage (corpus), même si ces séquences sont correctes linguistiquement. La formule 3.3 conduit donc à attribuer à ces suites de mots non observées une probabilité nulle. Par conséquent, les séquences correctes, mais non rencontrées dans le corpus d'apprentissage, seront systématiquement écartées lors de la reconnaissance. Le problème de la fiabilité de l'estimation des probabilités n-grammes se pose également pour les combinaisons de mots dont le nombre d'occurrences est faible. En effet, pour réduire l'espace de paramètres requis, plusieurs méthodes d'estimation forcent à zéro les occurrences des séquences de mots peu rencontrées. Ceci nous met dans une situation semblable à celle où les séquences ne sont pas rencontrées dans le corpus. De nombreuses techniques visant à améliorer la fiabilité des probabilités estimées pour les n-grammes rares ou inconnus ont été mises au point. Les principales d'entre elles seront présentées dans le paragraphe 3.5.

Plusieurs études ont montré que le modèle n-grammes atteint ses limites pour des valeurs de n équivalentes à 4 ou 5. Ce modèle se contente ainsi de tenir compte des contraintes locales imposées par les 3 ou 4 derniers mots de l'historique, ce qui nous semble insuffisant. En effet, chacun sait que l'apparition d'un mot peut dépendre de mots plus ou moins distants ainsi que d'autres paramètres comme par exemple le thème du texte prononcé.

3.4.2 Les modèles n-classes

L'espace de paramètres requis par les modèles n-grammes peut être sensiblement réduit, et par conséquent la fiabilité des évaluations peut être améliorée, en regroupant les mots dans des classes. Ce regroupement a le notable avantage de réduire le problème de manque de données que l'on risque de trouver dans un modèle n-grammes. En effet, puisque le nombre de classes est généralement inférieur au nombre de mots, il est alors plus réaliste de rencontrer beaucoup plus souvent les séquences de classes que les séquences de mots correspondantes.

La probabilité d'apparition d'un mot w_i à la suite d'un historique dépend de la classe $C(w_i)$ à laquelle appartient ce mot w_i, et de la probabilité d'apparition de cette classe $C(w_i)$ à la suite de l'historique de mots (exprimé par la suite des classes auxquelles appartiennent les mots de l'historique) [91, 196]. Si on suppose qu'un mot ne peut appartenir qu'à une seule classe, la probabilité d'un mot sachant son contexte est définie comme suit :

$$p(w_i/w_{i-n+1}, \ldots w_{i-1}) = p(w_i/C(w_i))p(C(w_i)/C(w_{i-n+1}), \ldots C(w_{i-1})) \tag{3.4}$$

où $C(.)$ est la classe à laquelle appartient le mot en argument.

La probabilité d'appartenance d'un mot à une classe, $p(w_i/C(w_i))$, est calculée comme suit :

$$p(w_i/C(w_i)) = \frac{N(w_i)}{N(C(w_i))} \tag{3.5}$$

où $N(.)$ est le nombre d'occurrences de l'argument dans le corpus. Bien sûr, le corpus doit être étiqueté (chaque mot du corpus doit être accompagné de sa classe). Il existe plusieurs méthodes pour l'étiquetage automatique d'un texte qui sont également fondées sur des méthodes probabilistes. Ces méthodes d'étiquetage sont utiles surtout dans le cas où un mot peut appartenir à plusieurs classes [183, 38].

En ce qui concerne la probabilité de successions de classes,

$$p(C(w_i)/C(w_{i-n+1}), \ldots C(w_{i-1})),$$

elle peut être estimée de la même façon que la probabilité de succession de mots,

$$p(w_i/w_{i-n+1}, \ldots w_{i-1}),$$

définie dans le paragraphe 3.4.1.

Il existe plusieurs méthodes de classification : manuelle ou automatique. La classification manuelle repose surtout sur des connaissances du domaine d'application mais également sur des connaissances linguistiques. Une des méthodes en classification manuelle est de regrouper les mots en classes syntaxiques : *ARTICLE*, *VERBE*, etc. Ainsi, on suppose que les mots appartenant à une même classe syntaxique ont le même comportement dans la langue. Une phrase sera donc décrite par sa structure syntaxique [182, 67]. La connaissance du domaine d'application auquel le modèle est destiné permet également, avec l'aide d'experts, de classer les mots. Par exemple, dans le cas d'une machine vocale pour la réservation de billets d'avion, il est plus raisonnable de regrouper les noms des compagnies aériennes, les noms des aéroports, les noms des villes, etc, puisqu'ils ont un comportement similaire, et non identique, vis-à-vis de leurs contextes droit et gauche [194].

En ce qui concerne la classification automatique, plusieurs algorithmes ont été proposés permettant de rassembler les mots d'un corpus d'apprentissage suivant un critère d'optimisation bien défini [93, 110, 30]. Généralement, ces algorithmes sont sous-optimaux et le résultat de la classification dépend de la partition initiale des mots dans les classes candidates. Les principaux critères d'optimisation applicables sont le maximum de vraisemblance, la validation croisée (*Cross-validation*) et l'information mutuelle entre les classes de mots adjacentes. D'autres algorithmes de classification, fondées sur le recuit simulé, ont été proposés dans [90]. Une comparaison entre toutes ces techniques de classification a été proposée dans [136]. Sur des corpus d'apprentissage assez grands et assez représentatifs du domaine d'application, un modèle n-grammes donne en général une perplexité inférieure à un modèle n-classes, pour la même valeur de n.

Dans la littérature, plusieurs variantes des modèles n-classes ont été présentées, nous citons dans les deux sous-paragraphes ci-dessous les plus connues. Néanmoins, ces modèles (n-classes, POS et morphologiques) ont le même inconvénient que celui des modèles n-grammes, en ce qui concerne la taille de l'historique pris en compte. En effet, la majorité des modèles présentés dans la littérature se limite à un historique restreint de 3 ou 4 classes, ce qui reste insuffisant pour une bonne modélisation.

3.4.3 Les modèles POS

Ces modèles, qui se fondent sur une partition de parole ("*Part of Speech*"), ont été introduits pour les langages fortement flexionnels, comme le français, l'allemand ou l'italien. Les modèles POS [8] sont inspirés de la connaissance syntaxique [33]. Ainsi, un mot donné peut appartenir à différentes classes (POS) à des instants différents : par exemple, le mot "part" peut être un nom ou un verbe. Par définition, chaque occurrence d'un mot doit avoir une seule classe à un instant donné. En pratique, il n'est pas aisé d'extraire la vraie classe d'un mot dans un contexte, parmi l'ensemble des classes associées à ce mot. Ainsi, pour estimer la probabilité d'un mot w_t dans un contexte h, le modèle POS calcule la somme des probabilités n-classes (la formule 3.4) sur toutes les classes associées au mot à prédire w_t. Un modèle POS trigrammes (3-grammes), estimant la probabilité d'un mot w_t sachant un historique de classes, est généralement défini comme suit :

$$p(w_t/g_{t-2},g_{t-1}) \approx \sum_{g_t \in G_{w_t}} p(w_t/g_t)p(g_t/g_{t-2},g_{t-1}) \tag{3.6}$$

où $g(w_t) = g_t$ est la classe (POS) du mot w_t au temps t et G_{w_t} est l'ensemble de toutes les classes associées au mot w_t, $p(w_t/g_t)$ est la probabilité d'appartenance du mot w_t à la classe g_t

8. Part of Speech

(peut être estimée par la formule 3.5) et $p(g_t/g_{t-2},g_{t-1})$ est la probabilité de la suite de classes g_{t-2},g_{t-1},g_t qui peut être estimée par la formule 3.3.

Une variante de ce modèle présentée dans [151] suppose que l'historique de classes est non fixe. Par conséquent, le modèle fait la somme sur les probabilités n-classes et sur tous les historiques de classes possibles. En gardant les mêmes notations utilisées dans la formule 3.6, un modèle POS bigrammes (2-grammes) estime donc la probabilité d'un mot w_t sachant un historique w_{t-1} comme suit :

$$\begin{aligned} p(w_t/w_{t-1}) &= \sum_{g_t \in G_{w_t}} p(w_t/g_t) p(g_t/w_{t-1}) \\ &= \sum_{g_t \in G_{w_t}} p(w_t/g_t) \sum_{g_{t-1} \in G_{w_{t-1}}} p(g_t/g_{t-1}) p(g_{t-1}/w_{t-1}) \end{aligned} \quad (3.7)$$

avec

$$p(g_{t-1}/w_{t-1}) = \frac{N(w_{t-1},g_{t-1})}{N(w_{t-1})},$$

où $N(.)$ est l'occurrence de la suite de mots en argument.

Estimer un modèle POS nécessite l'étiquetage de chaque mot du corpus par sa vraie classe parmi l'ensemble des classes associées à ce mot. D'où la nécessité d'utiliser des algorithmes d'étiquetage ou *tagger* [42].

3.4.4 Les modèles fondés sur la morphologie

les modèles morphologiques [56] sont considérés comme une extension des modèles POS. L'inconvénient des modèles POS provient du fait qu'ils n'utilisent pas de connaissances sémantiques entre les mots. Ainsi, une composante, supposée être de nature sémantique a été ajoutée en plus au niveau du lemme. Des mots distincts peuvent en effet correspondre à différentes flexions d'un seul lemme, par exemple les mots "avoir" et "ont". Le nombre de flexions par lemme dépend généralement de la classe syntaxique (POS). En effet, les noms ont seulement 4 variations (genre et nombre) pour chaque lemme, tandis que les verbes en possèdent beaucoup plus (temps, mode et personne).

Pour les modèles trigrammes morphologiques, la prédiction d'un mot est fondée sur les classes des deux derniers mots de l'historique ainsi que sur les lemmes de ces deux derniers mots. Si on note G l'ensemble des classes (POS) et L l'ensemble des lemmes, la formule est définie comme suit :

$$P(w_t/h_t) = \sum_{g_t \in G} p(g_t/g_{t-2},g_{t-1})(\lambda(g_t)p(w_t/g_t) + (1-\lambda(g_t))p_M(w_t/g_t)) \quad (3.8)$$

où la composante morphologique p_M est définie par :

$$p_M(w_t/g_t) = \sum_{l_t \in L} p(l_t/l_{t-k},l_{t-j})p(w_t/g_t,l_t) \quad (3.9)$$

où l_t est un lemme possible de w_t et (l_{t-k},l_{t-j}) sont les lemmes des deux derniers mots dans l'historique h_t. Les paramètres d'interpolation dépendent de la classe du mot à prédire : la somme des paramètres d'interpolation λ est égale à 1.

Pour l'apprentissage du modèle il faut un corpus contenant à la fois les classes (POS) et les lemmes. Néanmoins, on peut simplifier ceci, en supposant que chaque couple (mot, classe) identifie un seul lemme. De cette façon, après une phase d'étiquetage des classes (POS) qui peut être effectuée automatiquement, l'étiquetage des lemmes peut être réalisé de manière déterministe. D'autres améliorations apportées à ce modèle ont été présentées dans [33, 192, 193].

3.4.5 Les modèles n-grammes dynamiques

Les modèles n-grammes se sont avérés performants dans la prédiction, ce qui les a rendus les plus utilisés dans les systèmes de RAP. Néanmoins, pour des valeurs assez grandes de n ($n > 3$), le nombre de paramètres (probabilité de vraisemblance de toutes les combinaisons possibles) à estimer de ces modèles devient très élevé. De plus, le nombre de suites de mots, peu ou non rencontrés, augmente fortement.

Ayant pour objectif de réduire le nombre de paramètres à estimer, un modèle nommé "*scalable n-gram*" a été proposé [31]. Cette approche supprime tous les paramètres qui n'affectent pas beaucoup les performances du modèle. Par conséquent, plusieurs suites de mots, même si elles apparaissent suffisamment de fois dans le corpus, sont estimées par des modèles m-grammes plus petits, $m \leq n$ (cf. §3.5). En réduisant les paramètres des modèles 4-grammes et 5-grammes, R. Kneser a obtenu des performances meilleures qu'un modèle trigrammes (3-grammes) interpolé [109].

Dans le but d'extraire le maximum possible d'informations dans les n derniers mots de l'historique, plusieurs modèles (variantes des n-grammes) ont été proposés ces dernières années. Sans vouloir être exhaustif, nous présentons brièvement, ci-dessous, ceux qui nous ont paru originaux.

Le modèle x-grammes [25] suppose que le nombre de mots utiles à la prédiction est variable en fonction de l'historique h. Par conséquent, dans les n derniers mots de h, fixés comme taille maximale (n-grammes), le modèle extrait une séquence de m mots ($m < n$) utiles pour la prédiction. Une extension de cette approche utilisant des classes de type POS (cf. §3.4.3) a été présentée [144].

Un autre modèle, nommé *permugrammes*, a été proposé [173]. Ce modèle, supposé être une généralisation des n-grammes, est une interpolation (cf. §3.5) linéaire d'un grand nombre de modèles n-grammes (bigrammes, trigrammes, ...) agissant sur des permutations spécifiques et différentes de l'historique h du mot à prédire. Cette approche permet de tenir compte des dépendances entre les mots non adjacents, sans avoir recours à l'utilisation des modèles n-grammes d'ordres élevés.

En partant de l'hypothèse qui suppose que la connaissance du thème en cours améliorera la prédiction, le modèle n-grammes à mixture ("*Mixture n-gram*") a été proposé [111, 88]. Ce modèle qui est une interpolation de plusieurs modèles n-grammes spécifiques à des thèmes, a pour objectif de donner un poids plus élevé aux thèmes considérés être en cours.

Ces modèles ont amélioré considérablement les performances des modèles n-grammes de base. Néanmoins, beaucoup d'entre eux ont été souvent utilisés avec des vocabulaires limités. De plus, l'historique traité est souvent limité aux 5 derniers mots (maximum), ce qui reste insuffisant dans certains cas.

3.4.6 Les modèles n-grammes distants

Dans les modèles n-grammes cités ci-dessus, la prédiction d'un mot est calculée seulement en fonction des occurrences des n derniers mots dans le corpus. Ceci constitue l'inconvénient principal de ce modèle, ainsi que le modèle n-classes. En effet, en français, les historiques "*Les chercheurs*" et "*Les chercheurs d'aujourd'hui*" doivent normalement être similaires dans la prédiction du mot "*inventent*". Or ceci n'est pas vrai dans le cas des modèles n-grammes. Normalement, on devrait se servir de la puissance prédictive de "*Les chercheurs*" dans "*inventent*", même lorsque ces deux expressions sont séparées par un ou plusieurs mots.

Pour pallier ce problème, une autre approche, se fondant sur l'utilisation des n-grammes à longue distance, a été proposée [86, 122]. Ces modèles cherchent à capturer directement la dépendance qui existe entre le mot courant et la suite de $n-1$ mots qui se trouvent à une certaine distance dans l'historique. Par exemple, dans la suite de mots $W_1^{i-1} = w_1,w_2,\dots w_{i-1}$, un modèle trigrammes distant d'ordre 2 prédit w_i en se basant sur (w_{i-3},w_{i-2}). Par conséquent, un modèle n-grammes distant d'ordre 1, est tout simplement le modèle n-grammes classique.

3.4.7 Les modèles cache et trigger

Ce type de modèle a pour objet d'étendre l'historique tout en évitant le problème du manque de données dans le corpus. Ceci est dû au fait qu'il existe une quantité d'information significative dans l'historique lointain. En effet, Huang *et al.* [86] ont montré qu'un modèle combinant les bigrammes distants d'ordre $d = 1,2,3,4$ et 5 (cf. §3.4.6) est plus performant, en terme de perplexité, qu'un modèle bigrammes classique (cf. §3.4.1). De plus, le jeu de Shannon (cf. §2.3.2) développé par IBM ([165] page 17) montre que l'être humain est beaucoup plus performant qu'un modèle trigrammes classique dans la prédiction du prochain mot, sachant un historique entier. En analysant les résultats de ce jeu, il s'est avéré que dans 40% des cas, le mot à prédire ou un autre mot qui lui est associé, apparaît dans son historique. Par conséquent, deux modèles sont apparus : le modèle cache et le modèle trigger.

Le modèle cache se fonde sur le fait que les mots existant dans l'historique ont plus de chance de réapparaître. Le principe, derrière ce modèle, consiste à prendre en compte les fréquences de mots dans l'historique afin de prédire plus précisément ces derniers [117, 120, 118]. Si on se limite à un historique de M mots, la probabilité du mot w_i sachant un historique $h_M = w_{i-M},\dots w_{i-1}$, fourni par le modèle cache, est estimée par :

$$P_{Cach}(w_i/h_M) = \frac{1}{M}\sum_{m=1}^{M}\delta(w_n/w_{n-m}) \tag{3.10}$$

où $\delta(w/v) = 1$ si et seulement si $w = v$.

Dans le cas du modèle trigger, on se fonde sur les couples de mots (triggers) qui ont une forte corrélation entre eux. Par conséquent, l'apparition du premier mot w_i d'un de ces couples (w_i,w_j) augmente la prédiction du deuxième mot w_j [191, 143, 167]. Ces couples sont, bien sûr, calculés à partir d'un corpus d'apprentissage en se fondant sur une mesure telle que l'information mutuelle.

Ces modèles sont généralement combinés avec d'autres, par exemple n-grammes, dans le but d'extraire en même temps les contraintes à court et à long terme dans l'historique. En effet, un modèle n-grammes estimé sur un corpus regroupant des textes économiques, se comporte de la même manière, que l'on parle d'exploitations agricoles ou de sport. Pourtant nul doute que le mot "blé" a plus de chance d'apparaître dans les mots prédits si le premier sujet est abordé. En combinant les modèles, la probabilité d'apparition du mot "blé" va donc varier en fonction du contexte.

Plusieurs méthodes de combinaison ont été présentées dans la littérature, les plus connues sont la méthode fondée sur l'interpolation linéaire [191] et la méthode à maximum d'entropie [165]. Pour la première méthode (interpolation linéaire), en combinant les modèles n-grammes, cache et trigger, la probabilité d'apparition du mot w_i sachant un historique $h_M = w_{i-M},\dots w_{i-1}$

est définie comme suit :

$$P(w_i/h_M) = \\ (1-\lambda_1-\lambda_2)P_{Gram}(w_i/w_{i-n+1},\dots w_{i-1}) + \lambda_1 P_{Cach}(w_i/h_M) + \lambda_2 P_{Trig}(w_i/h_M) \quad (3.11)$$

où $M > n$, les λ_i représentent les paramètres d'interpolation et somment à 1, $P_{Trig}(.)$ est la probabilité du modèle trigger, $P_{Cach}(.)$ est la probabilité du modèle cache et $P_{Gram}(.)$ est la probabilité du modèle n-grammes.

D'autres variantes dans la combinaison de ces modèles ont été proposées dans [132, 143, 191]. Les modèles cache et trigger actuels se limitent à l'utilisation des mots seuls, ce qui est insuffisant. En effet, dans une langue, c'est généralement toute une suite de mots qui influe sur la prédiction de ce qui suit. Nous proposons dans le chapitre 6 une extension de ces modèles, leur permettant de se fonder sur les couples de mots ou de séquences de mots qui sont fortement corrélés.

3.4.8 Les modèles utilisant le principe du maximum d'entropie

L'approche fondée sur le principe de maximum d'entropie est bien adaptée au problème de la combinaison de plusieurs sources de connaissances [165, 20]. En effet, l'objectif de cette approche est de concevoir un modèle combinant plusieurs contraintes de la langue, appelées caractéristiques, comme les n-grammes, le cache et le trigger. L'originalité de cette approche est sa capacité à respecter toutes les contraintes imposées par chaque source de connaissance, ce qui n'est pas le cas des méthodes d'interpolation. Ainsi, pour un historique h donné et pour un mot w à prédire, chaque fonction caractéristique i, spécifique d'une contrainte donnée, est définie par :

$$f_i(h,w) \in \{0,1\} \quad (3.12)$$

où la fréquence de chacune de ces fonctions f_i est supposée connue. Par conséquent, dans le cas d'un modèle bigrammes, la fonction caractéristique spécifique au couple de mots (A,B) est définie par la formule suivante :

$$f_{A,B}(h,w) = \begin{cases} 1 & si\ A\ est\ le\ dernier\ mot\ de\ h,\ w = B \\ 0 & sinon \end{cases} \quad (3.13)$$

dans le cas des modèles à longue distance (trigger), si on considère que le mot A influence l'apparition du mot B, la formule devient :

$$f_{A\rightarrow B}(h,w) = \begin{cases} 1 & si\ A \in h,\ w = B \\ 0 & sinon \end{cases} \quad (3.14)$$

Une fois les fonctions caractéristiques déterminées, le modèle à maximum d'entropie calcule la distribution générale qui satisfait toutes ces contraintes en fonction de leur fréquence d'apparition dans l'historique h [166]. La probabilité de prédire un mot w sachant un historique h est, par conséquent, estimée comme suit :

$$p(w/h) = \frac{exp[\sum_i \lambda_i f_i(h,w)]}{\sum_{w'} exp[\sum_i \lambda_i f_i(h,w')]} \quad (3.15)$$

où il existe un paramètre λ_i pour chaque fonction caractéristique f_i. En ce qui concerne le dénominateur, la somme est effectuée sur tous les mots w' du vocabulaire. Le modèle à maximum d'entropie est donc défini, en plus des fonctions caractéristiques, par l'ensemble des paramètres $\Lambda = \{\lambda_i\}$ estimé sur un corpus d'apprentissage.

Le modèle à maximum d'entropie estime la fonction de vraisemblance $G(\Lambda) = p(W_1^N)$, calculée sur un corpus d'apprentissage $W_1^N = w_1, \ldots w_n, \ldots w_N$, comme suit :

$$G(\Lambda) = p(W_1^N) = \sum_{n=1}^{N} log\ p(w_n/h_n) = \sum_{hw} N(h,w) log\ p(w/h) \tag{3.16}$$

où $N(.)$ est le nombre d'occurrences de la suite de mots en argument. Ainsi, pour définir l'ensemble Λ correspondant à une vraisemblance maximale, ce qui correspond à une perplexité minimale, il suffit d'extraire les paramètres λ_i qui rendent la dérivée de la fonction de vraisemblance $G(\Lambda)$ par rapport à λ_i nulle :

$$\frac{\partial G}{\partial \lambda_i} = \sum_{hw} N(h,w) \frac{\partial}{\partial \lambda_i} logp(w/h) = 0 \tag{3.17}$$

Après quelques manipulations, on obtient :

$$\frac{\partial G}{\partial \lambda_i} = -Q_i(\Lambda) + N_i = 0 \tag{3.18}$$

avec une fonction $Q_i(\Lambda)$ dépendante de Λ :

$$Q_i(\Lambda) = \sum_{hw} N(h) p(w/h) f_i(h,w) \tag{3.19}$$

et une fonction N_i indépendante de Λ :

$$N_i = \sum_{hw} N(h,w) f_i(h,w) \tag{3.20}$$

L'algorithme GIS (*generalized iterative scaling*) est bien connu en statistique pour définir une solution numérique aux formules de maximum de vraisemblance [44, 180]. Cet algorithme suppose que le nombre de fonctions caractéristiques est constant pour chaque mot à prédire w sachant un historique h. Il suppose également que le nombre de mots dans l'historique h est non nul. L'avantage d'utiliser l'algorithme GIS, dans l'estimation du modèle à maximum d'entropie, est qu'il garantit sa convergence vers une solution unique regroupant toutes les contraintes f_i définies auparavant.

L'avantage du modèle à maximum d'entropie est qu'il est extrêmement général. En effet, il possède la capacité de combiner toutes les contraintes, proposées par les autres modèles de langage, même si elles ne sont pas vérifiées dans le corpus. Néanmoins, l'inconvénient de ce modèle est qu'il nécessite un temps de calcul énorme pour son évaluation [165]. Même les améliorations proposées dans [157], pour l'optimisation du temps de calcul, ne sont pas encore satisfaisantes. De plus, l'ajout d'un ensemble de contraintes, non vérifié par le corpus d'apprentissage, peut rendre les caractéristiques théoriques du modèle (qui sont unicité, convergence et existence) non valides.

Plusieurs modèles à maximum d'entropie, combinant différentes caractéristiques à court et à long terme comme les bigrammes, trigrammes, bigrammes distant, cache et trigger ont été présentés dans la littérature [123, 165, 180].

3.4.9 Les modèles à base de grammaires

Les modèles n-grammes se sont révélés puissants pour le décodage linguistique, mais ils sont forcément limités par la taille de l'historique qu'ils utilisent, réduit à deux ou trois mots. Pour extraire les contraintes globales au niveau de la phrase, il est naturel d'utiliser des règles grammaticales, d'où les modèles à grammaires probabilistes. Ces modèles sont issus du mariage des modèles probabilistes et des modèles à syntaxe fixe (structurels) [24].

Les modèles à syntaxe fixe, très utilisés dans les systèmes de RAP du début des années soixante-dix, traduisent la langue sous forme de règles grammaticales à contexte libre ("*Context Free Grammar*") décrivant ainsi les séquences de mots valides. Ces règles sont souvent traduites sous forme de graphes. Dans le cas des langues naturelles, avec un grand vocabulaire, ces modèles ne sont pas valables. En effet, en plus de l'espace mémoire important nécessaire au stockage du graphe correspondant, la construction exhaustive des règles grammaticales est impossible [24]. Néanmoins, ces modèles restent valables pour des cas d'applications limitées (les linguistes parlent alors de sous-langages [107, 106]).

Les modèles probabilistes sont souvent contraints d'abdiquer devant certains problèmes courants tels les accords à longue portée entre sujets et verbes. En revanche, de tels accords ne posent pas de problème (ou beaucoup moins) aux modèles à syntaxe fixe. D'où l'idée de combiner ces deux modèles en décrivant des règles grammaticales probabilistes. J. Backer [13] a proposé une procédure automatique d'ajustement des probabilités pour une grammaire contextuelle. Cet algorithme est similaire à l'algorithme Forward-Backward (processus itératif d'estimation) [32]. F. Jelinek et H. Fusijaki ont utilisé les distributions obtenues par le modèle qui détermine l'arbre syntaxique le plus probable, dans le cas d'une phrase ambiguë, pour analyser la grammaire. Les résultats ont montré qu'un grand nombre de phrases pouvait être analysées de cette manière ([32] page 623).

Plusieurs modèles ont été présentés ces dernières années combinant les grammaires probabilistes avec d'autres modèles de langage, tels les modèles n-grammes ou les modèles n-classes [163, 83, 70]. Néanmoins, les modèles à base de grammaires probabilistes ne sont encore utilisés que dans de très petites applications ou alors comme post-traitement applicable : par exemple, sur une liste d'hypothèses produites par le système de RAP [19].

3.4.10 Les modèles à analyse sémantique latente

L'introduction d'une connaissance sémantique dans la prédiction est un phénomène qui préoccupe les chercheurs depuis longtemps. D'autant que la sémantique est l'une des principales connaissances utilisées par l'être humain dans la prédiction d'un mot sachant un contexte donné.

Les modèles fondés sur la méthode d'analyse sémantique latente[9] (LSA) sont considérés comme très prometteurs dans l'intégration d'une composante sémantique dans la prédiction. En effet, ils permettent de prédire les mots qui sont cohérents avec leurs contextes [16]. Ces modèles ont besoin, en plus d'un vocabulaire V de $|V|$ mots, d'un corpus d'apprentissage $\Re$ contenant N articles, de préférence du même domaine (par exemple, un article sur l'économie mondiale, un autre sur l'agronomie, etc). Il est donc nécessaire d'identifier, manuellement ou automatiquement, les thèmes du corpus [21]. Une fois le corpus subdivisé en articles, une matrice de correspondance W est définie où chaque ligne est associée à un mot w_i du vocabulaire et chaque colonne est associée à un article d_j. Chaque entrée de la matrice $w_{i,j}$ donne ainsi une idée sur le nombre d'occurrences du mot w_i dans le document d_j [17]. Une différence marquante avec les modèles n-grammes est que l'ordre d'apparition des mots dans l'historique n'est pas pris en compte : pour

9. Latent Semantic Analysis

déterminer la matrice W, on ne s'occupe que de l'occurrence des mots dans les articles et non pas de leur ordre d'apparition.

L'objectif principal de la matrice de correspondance W est de définir une matrice carrée S (de dimension $R \times R$), représentant l'*espace sémantique* de l'approche LSA. Le rôle de la matrice S est d'établir des liens entre les mots et les articles. Ainsi, pour calculer cette matrice, il suffit de subdiviser W en trois termes, suivant la méthode de décomposition en valeurs singulières :

$$W = USV^T \tag{3.21}$$

où ^{T} dénote la matrice transposée, U est une matrice (de dimension $|V| \times R$) qui représente les mots du vocabulaire dans l'espace sémantique S, et V (de dimension $R \times |V|$) est une matrice qui représente les articles dans ce même espace sémantique S. Ainsi, si la distance métrique entre deux vecteurs $u_i,u_j \in U$ est faible cela signifie que les deux mots correspondants $w_i,w_j \in V$ tendent à apparaître dans le même genre d'articles. Un exemple de métrique permettant de calculer la distance entre u_i et u_j est le cosinus de l'angle entre u_iS et u_jS, défini comme suit :

$$D(u_i,u_j) = cos(u_iS,u_jS) = \frac{u_iS^2u_j^T}{\parallel u_iS \parallel \quad \parallel u_jS \parallel}. \tag{3.22}$$

Les modèles fondés sur la méthode LSA définissent la probabilité de prédiction d'un mot w_t dans un contexte h spécifique (l'ensemble des mots de l'article en cours jusqu'au mot w_{t-1}, noté par $\tilde{d}_{t-1}$), comme suit :

$$p(w_t/h,S) = p(w_t/\tilde{d}_{t-1}) \tag{3.23}$$

où la condition sur S est de montrer que la probabilité dépend de la décomposition en valeurs singulières définie dans la formule 3.21. La probabilité $p(w_t/\tilde{d}_{t-1})$ est mesurée par une normalisation de la distance entre le vecteur u_i ($u_i \in U$) du mot à prédire w_t et l'ensemble des vecteurs représentatifs de $\tilde{d}_{t-1}$ dans l'espace S. Cette probabilité $p(w_t/\tilde{d}_{t-1})$ reflète ainsi la "pertinence" du mot w_t dans l'historique h, considéré comme un pseudo document. Pour les mots qui ont un sens qui s'accorde bien avec le domaine de $\tilde{d}_{t-1}$ (par exemple : économie mondiale, économie nationale, etc), la valeur de la probabilité $p(w_t/\tilde{d}_{t-1})$ est assez forte. En revanche, cette valeur est assez faible pour les mots qui n'apportent aucune information sur ce domaine.

L'inconvénient de ce modèle est qu'il est un mauvais prédicteur pour les mots communs (par exemple, les mots outils comme "de", "le", ...) et également pour les mots rares même s'ils sont représentatifs d'un ou de plusieurs articles. Il est ainsi intéressant de combiner cette approche avec d'autres modèles comme le modèle n-grammes par exemple [17, 39]. Une tentative d'utilisation de cette approche dans la classification des mots a été présentée dans [18]. Malheureusement, malgré sa réussite dans l'extraction des classes de mots qui ont un sens, cette tentative n'a pas pu améliorer les performances des modèles n-classes. Pour plus d'information sur cette approche le lecteur pourra se référer à [16].

3.4.11 Les modèles à base de séquences

Dans la langue, il arrive que plusieurs mots adjacents soient fortement corrélés (ex: "pomme de terre"). Par conséquent, il est plus naturel de considérer ces suites de mots (séquences) comme des unités de la langue. Si on demande à une personne de prédire la suite de la phrase tronquée "*le nom du président actuel est*", il est plus naturel pour elle de dire "*Jacques Chirac*" plutôt que "*Jacques*".

Plusieurs modèles de langage, se fondant sur cette idée, ont été proposés [93, 69, 46]. Leur objectif est principalement d'extraire les séquences qui méritent vraiment d'être considérées comme des unités de la langue. Ainsi, l'unité de base dans la prédiction n'est plus le mot mais plutôt la séquence (suite de mots). En effet, étant donné un historique de séquences, ces modèles essayent de prédire la séquence qui peut suivre.

Néanmoins, l'inconvénient des modèles à base de séquences existants est qu'ils se contentent de vocabulaires de taille limitée (quelques centaines de mots). En plus, la majorité de ces modèles ne s'occupent que des contraintes locales (unigrammes, bigrammes), en ignorant les contraintes qu'on pourrait rencontrer dans un historique plus grand. Pour plus d'informations sur les modèles à base de séquences existants dans la littérature, nous invitons le lecteur à se référer au chapitre 6 de ce document.

Motivés par l'originalité de cette approche, nous allons l'étendre, pour prendre en compte l'ensemble des contraintes existantes dans un historique lointain et pour pouvoir utiliser de très grands vocabulaires. Nous nous sommes également inspirés de cette approche pour introduire des connaissances linguistiques supplémentaires lors de la prédiction et de la reconnaissance. Nous définissons, dans les prochains chapitres, la théorie sous-jacente à nos modèles ainsi que leurs évaluations avec la perplexité et le jeu de Shannon. Ensuite, nous présentons leurs intégrations dans le système MAUD de RAP utilisant un grand vocabulaire.

3.5 Estimation des données manquantes

En pratique, toutes les suites de mots ne sont pas rencontrées dans un corpus d'apprentissage, alors qu'elles peuvent exister dans le langage considéré. De ce fait, plusieurs modèles de langage (par exemple, n-grammes de base utilisant seulement les occurrences des suites de mots) attribuent une probabilité d'apparition nulle pour certaines suites de mots qui sont pourtant valides. Le problème est le même pour les suites de mots peu rencontrées dans la langue. Une solution générale consiste à estimer la probabilité (distribution) conditionnelle $\hat{p}(w/h)$ en combinant deux composantes qui sont : le modèle de lissage et le modèle de redistribution.

Le modèle de lissage est lié au problème d'estimation des suites de mots de fréquence nulle [197]. En utilisant ce modèle, la probabilité de tous les mots, non observés dans un contexte h, est estimée par la réduction des fréquences des suites de mots observées. La fréquence relative d'un mot w sachant un contexte h, estimée suivant le principe du maximum de vraisemblance, est définie comme suit :

$$fr(w/h) = \frac{c(hw)}{c(h)} \tag{3.24}$$

où, $c(.)$ représente l'occurrence de la suite de mots en argument et, par définition, $c(h) = 0$ implique $fr(w/h) = 0$. Le modèle de lissage produit ainsi une *fréquence conditionnelle réduite* $fr^*(w/h)$, telle que :

$$0 \leq fr^*(w/h) \leq fr(w/h) \quad \forall hw \in V^n \tag{3.25}$$

où V est le vocabulaire et n est le nombre de mots dans la suite de mots hw. Plusieurs méthodes calculant le terme $fr^*(w/h)$ ont été proposées dans la littérature [48]. Nous citons à titre d'exemple la méthode de lissage par palier ("*floor discounting*"), la méthode de Good-Turing, la méthode de lissage absolu et la méthode de lissage linéaire. Pour notre travail, nous avons utilisé la méthode de Good-Turing. La justification de l'utilisation de cette méthode est sa bonne estimation des probabilités de suites de mots de fréquence nulle. De plus, des tests effectués avec les méthodes de lissage absolu et linéaire sur des modèles n-grammes et n-classes n'ont pas donné de meilleurs résultats.

Dans le cas de la méthode de Good-Turing le terme $fr^*(w/h)$ est défini comme suit :

$$fr^*(w/h) = \begin{cases} \frac{c(hw)^*}{c(h)} & \text{si } c(hw) > 0 \\ 0 & sinon \end{cases} \tag{3.26}$$

où

$$c(hw)^* = (c(hw) + 1)\frac{t_{c(hw)+1}}{t_{c(hw)}}. \tag{3.27}$$

Le terme t_r indique le nombre de suites de mots qui apparaissent exactement r fois dans le corpus d'apprentissage.

La probabilité $\lambda(h)$ des suites de mots de fréquence nulle est définie comme suit :

$$\lambda(h) = 1 - \sum_{w \in V} fr^*(w/h). \tag{3.28}$$

Ainsi, dans le cas de la méthode de Good-Turing, ce terme est défini par l'équation suivante :

$$\lambda(h) = 1 - \sum_{w:c(hw)>0} \frac{c(hw)^*}{c(h)}. \tag{3.29}$$

La probabilité $\lambda(h)$ est redistribuée sur tous les mots non rencontrés dont le contexte est h. Cette redistribution est faite proportionnellement à une distribution moins spécifique $p(w/h')$, où h' indique un contexte moins précis. Typiquement, dans le cas d'estimation d'un modèle trigrammes (3-grammes) c'est la distribution bigrammes (2-grammes) qui est utilisée, et c'est la distribution unigrammes (1-grammes) qui est choisie pour le calcul d'un modèle bigrammes. Des améliorations portées à cette méthode de base ont été réalisées dans [112]. Les deux principales approches utilisées dans la combinaison des modèles de lissage et de redistribution sont l'approche d'interpolation et l'approche de Katz nommée repli (*backing-off*).

L'approche de Katz [99]

Cette approche estime la distribution $\hat{p}(w/h)$ comme suit :

$$\hat{p}(w/h) = \begin{cases} fr^*(w/h) & si\ fr^*(w/h) > 0 \\ \alpha_h \lambda(h) p(w/h') & sinon \end{cases} \tag{3.30}$$

où α_h est un facteur de normalisation :

$$\alpha_h = \left(\sum_{w:fr^*(w/h)=0} p(w/h') \right)^{-1} \tag{3.31}$$

assurant ainsi la condition suivante : $\sum_w \hat{p}(w/h) = 1$.

Cette approche est souvent utilisée dans l'estimation des modèles n-grammes. Dans ce cas, c'est généralement une distribution de type $(n-1)$-grammes qui est utilisée quand la *fréquence conditionnelle réduite* est nulle ($fr^*(w/h) = 0$). En revanche, d'autres méthodes plus originales, utilisant des distributions $p(w/h')$ autres que les $(n-1)$-grammes, ont été proposées dans la littérature. Ces méthodes se fondent soit sur la même distribution (n-grammes) mais avec un historique semblable et aussi fréquent que celui utilisé dans la distribution initiale [43], soit sur une distribution de nature différente telle que la distribution n-classes [55, 134].

L'approche d'interpolation

Dans cette approche, proposée initialement par F. Jelinek et R. Mercer [96], les distributions sont directement combinées :

$$\hat{p}(w/h) = fr^*(w/h) + \lambda(h)p(w/h'). \tag{3.32}$$

Dans le cas d'une interpolation linéaire entre un modèle trigrammes et un modèle bigrammes, cette formule devient :

$$\hat{p}(w/h) = (1 - \lambda(h))p_{trig}(w/h) + \lambda(h)p_{big}(w/h') \tag{3.33}$$

où $p_{trig}(.)$ et $p_{big}(.)$ sont, respectivement, la probabilité trigrammes et la probabilité bigrammes de la suite de mots en argument [93]. Plusieurs extensions de cette approche d'interpolation ont été présentées dans la littérature [145, 197, 143, 108].

Il est clair que les deux approches, d'interpolation et de Katz, utilisent d'une façon récursive les distributions d'ordre inférieur dans l'estimation du modèle global. Ainsi, le modèle n-grammes peut être interpolé avec un modèle $(n-1)$-grammes qui lui aussi peut être interpolé avec un modèle $(n-2)$-grammes, et ainsi de suite.

3.6 La théorie d'estimation d'un modèle optimal

Dans ce paragraphe, nous présentons les méthodes les plus utilisées dans l'estimation d'un modèle optimal gouvernant l'activité d'une source supposée ergodique[10].

Soit le corpus d'apprentissage $W_1^T = w_1, \ldots, w_T$ contenant T symboles[11] appartenant à un vocabulaire fini V. Un modèle donné M se fonde généralement sur un contexte bien défini pour la prédiction d'un symbole[12]. Par conséquent et sans perte d'information, on peut associer à chaque symbole du corpus son contexte (ou plutôt le contexte utilisé, par le modèle M, pour sa prédiction), ce qui permet de noter le corpus W_1^T de la manière suivante :

$$C = (h_1)w_1, \ldots, (h_i)w_i, \ldots, (h_T)w_T \tag{3.34}$$

où h_i représente le contexte utilisé pour la prédiction du symbole w_i.

Ainsi, pour un contexte donné h, en prenant toutes les séquences commençant par h dans le corpus C, on obtient un sous corpus C_h, comme suit :

$$C_h = (h)w_{h_1}, \ldots, (h)w_{h_N} \tag{3.35}$$

où l'ensemble $\{w_{h_1}, \ldots w_{h_N}\} \subseteq \{w_1, \ldots w_T\}$, peut être considéré comme une suite de N symboles indépendants écrits suivant la loi de probabilité $p(w/h)$. En d'autres termes, le corpus C_h est constitué de N variables aléatoires indépendantes, ayant une même distribution (par exemple : distribution discrète, distribution symétrique).

Dans ce qui suit, nous allons nous limiter à un contexte donné (fixe) h, lors de la présentation de l'ensemble des méthodes utilisées pour l'estimation des probabilités conditionnelles,

10. Dans le cas de la langue, on suppose qu'une suite de mots suffisamment longue est représentative de son comportement.

11. Pour des raisons de généralité, on considère le corpus comme une suite de symboles, où le symbole est une unité qui peut être le phonème, la syllabe, le mot ou une suite de mots (séquence).

12. le modèle n-grammes se fonde sur les $n-1$ derniers symboles.

$\{p(v/h)\}$ $(v \in V)$, d'un modèle M. Il suffit donc de procéder de la même manière, pour l'estimation des probabilités conditionnelles, $\{p(v/h')\}$, utilisant d'autres contextes, $h' \neq h$. Ainsi, pour des raisons de simplicité, le contexte h sera supprimé de la notation (le corpus C_h défini dans l'équation 3.35 sera ainsi représenté par l'équation 3.36). Cependant, notre objectif est d'estimer la distribution de probabilité discrète p(v), $v \in V$, donnée par le modèle M à partir du corpus d'apprentissage :

$$C = w_1, \ldots, w_N. \tag{3.36}$$

Ce corpus C est constitué de N variables aléatoires indépendantes, ayant une même distribution. Il est à noter que v est utilisé pour noter les symboles du vocabulaire, en revanche w_i est utilisé pour désigner le $i^{ème}$ symbole du corpus C.

La distribution $p(v)$ appartient généralement à une *famille paramétrique* connue :

$$\{p(v;\theta), \theta \in \Theta\},$$

où θ est un vecteur de paramètres inconnus spécifiant la distribution, et Θ est l'espace de paramètres. Nous nous limiterons dans ce qui suit à deux d'entre elles : distribution discrète et distribution symétrique (discrète) [48].

Distribution discrète : L'espace de paramètres de la distribution discrète est défini comme suit :

$$\Theta = \{\theta = [\theta_v]_{v \in V} : \theta_v \geq 0 \text{ et } \forall v \in V, \sum_{v \in V} \theta_v = 1\}. \tag{3.37}$$

Ceci permet ainsi d'affecter un paramètre θ_v à chaque symbole v du vocabulaire V :

$$p(v;\theta) = \theta_v. \tag{3.38}$$

Ainsi, la vraisemblance du corpus $p(C;\theta)$ est définie par l'équation suivante :

$$p(C;\theta) = \frac{N!}{\prod_{v \in V} c(v)!} \prod_{v \in V} \theta_v^{c(v)} \quad \text{où} \quad c(v) = \sum_{i=1}^{N} \delta(w_i = v). \tag{3.39}$$

Le terme $\delta(w_i = v)$ vaut 1 ($\delta(w_i = v) = 1$) si le $i^{ème}$ symbole du corpus est v et 0 sinon. Le terme N représente le nombre de symboles dans le corpus C et le vecteur de paramètres $[c(v)]_{v \in V}$ représente les *statistiques suffisantes* du corpus C [137].

Distribution symétrique : Une distribution légèrement différente peut être définie en supposant que les symboles ayant la même fréquence dans le corpus C doivent avoir la même probabilité. D'où la formule suivante :

$$p(v;\theta) = \frac{\theta_r}{n_r} \text{ avec } r = c(v) \tag{3.40}$$

où θ_r représente la probabilité totale de tous les symboles $v \in V$ qui apparaissent r fois dans C, tandis que n_r représente le nombre de symboles qui apparaissent r fois dans ce même corpus C. La vraisemblance du corpus C avec la distribution symétrique est par conséquent définie comme suit :

$$p(C;\theta) = \frac{N!}{\prod_{v \in V} c(v)!} \prod_{v \in V} \left(\frac{\theta_{c(v)}}{n_{c(v)}}\right)^{c(v)} = \frac{N!}{\prod_{v \in V} c(v)!} \prod_{r \geq 0} \left(\frac{\theta_r}{n_r}\right)^{rn_r}. \tag{3.41}$$

Ainsi, pour estimer la valeur de probabilité $p(v;\theta)$, il suffit de définir la meilleure valeur pour son vecteur de paramètres θ. Nous trouvons dans la littérature plusieurs méthodes permettant l'estimation de θ. Nous présentons dans ce qui suit, trois d'entre elles : maximum de vraisemblance, bayésienne et validation croisée.

3.6.1 Estimation par maximum de vraisemblance

Cette approche considère le paramètre θ comme une quantité inconnue à déterminer. La meilleure estimation, suivant cette approche, est celle qui maximise la probabilité (ou la vraisemblance) du corpus d'apprentissage C. Ainsi, étant donné que le corpus C contient N variables aléatoires, l'estimation suivant le principe de maximum de vraisemblance (ML) de θ est définie comme suit :

$$\begin{aligned} \theta^{ML} &= arg \max_{\theta \in \Theta} p(C;\theta) \\ &= arg \max_{\theta \in \Theta} \prod_{i=1}^{N} p(w_i;\theta). \end{aligned} \tag{3.42}$$

L'estimation ML d'une distribution discrète peut être facilement calculée en maximisant le logarithme de la vraisemblance au lieu de la vraisemblance elle-même. Ainsi, en appliquant le logarithme sur la vraisemblance du corpus $p(C;\theta)$ (équation 3.39), en tenant compte de l'ensemble des contraintes sur Θ et en éliminant une constante, on obtient la fonction lagrangienne suivante :

$$L(\theta;\lambda) = \sum_{v \in V} c(v) log\theta_v + \lambda \left(1 - \sum_{v \in V} \theta_v\right). \tag{3.43}$$

En dérivant par rapport à θ_v ($v \in V$) et en ramenant la fonction à zéro, on obtient :

$$\frac{\partial L}{\partial \theta_v} = \frac{c(v)}{\theta_v} - \lambda = 0 \qquad \forall v \in V. \tag{3.44}$$

Après avoir réarrangé la formule 3.44, on obtient :

$$\sum_{v \in V} c(v) = \left(\sum_{v \in V} \theta_v\right) \lambda = \lambda. \tag{3.45}$$

En substituant λ, on obtient l'estimation suivante :

$$\theta_v^{ML} = \frac{c(v)}{\sum_{v \in V} c(v)} = \frac{c(v)}{N}. \tag{3.46}$$

Dans le cas d'une distribution symétrique, si on maximise le logarithme de la vraisemblance par rapport à θ, on obtient la formule suivante :

$$\theta_r^{ML} = \frac{n_r r}{N} \tag{3.47}$$

ce qui donne une probabilité similaire à celle de la distribution discrète.

3.6.2 Estimation bayésienne

La distribution bayésienne est conceptuellement différente de celle du maximum de vraisemblance. En effet, le vecteur de paramètres θ est considéré comme une variable aléatoire pour laquelle une distribution *a priori* est supposée être connue [137, 53]. Dans ce cas, le problème est d'estimer θ, étant donné le corpus d'apprentissage C et la distribution *a priori* $p(\theta)$ du paramètre θ. En appliquant la règle de Bayes, la distribution *a posteriori* est définie comme suit :

$$p(\theta/C) = \frac{p(C/\theta)p(\theta)}{p(C)}. \tag{3.48}$$

Il est à noter que les deux expressions $p(C/\theta)$ et $p(C;\theta)$ indiquent la même distribution. Cette différence au niveau de la notation sert juste à mettre en évidence que θ est un paramètre dans $p(C;\theta)$, en revanche θ est considéré comme une variable aléatoire dans $p(C/\theta)$.

L'estimation de θ peut être extraite de la distribution *a posteriori* de plusieurs manières. Nous citons à titre d'exemple l'approche du maximum *a posteriori* et l'approche bayésienne.

Approche du maximum *a posteriori* (MAP) :

cette approche cherche la valeur de θ qui maximise la probabilité *a posteriori*. Ainsi, en supprimant la constante $p(C)$ dans la formule 3.48, l'approche du maximum *a posteriori* peut être considérée comme une généralisation de l'approche ML (cf. §3.6.1). En effet :

$$\begin{aligned} \theta^{MAP} &= arg \max_{\theta \in \Theta} p(\theta/C) \\ &= arg \max_{\theta \in \Theta} p(C/\theta)p(\theta). \end{aligned} \tag{3.49}$$

Approche bayésienne (B) :

cette approche considère le vecteur de paramètres θ^B comme étant l'espérance de θ par rapport à la distribution *a posteriori*. En effet :

$$\theta^B = E[\theta/C]. \tag{3.50}$$

T. Kawabata et M. Tamoto [100] ont défini cette approche dans le cas d'un contexte n-grammes. Pour plus d'informations sur cette approche le lecteur pourra se référer à [187, 53].

3.6.3 Estimation de validation croisée

La validation croisée est une technique qui peut être utilisée pour estimer et évaluer un modèle paramétrique générique [189, 53]. Pour le problème concernant l'estimation d'une distribution paramétrique, la validation croisée peut être combinée avec l'approche de maximum de vraisemblance. En particulier, l'estimation de validation croisée (CV) peut être appliquée quand le modèle paramétrique $p(v;\theta)$ utilise des contraintes (statistiques) calculées sur le corpus C. En effet, la validation croisée permet de réduire le biais provoqué par l'utilisation du corpus dans l'estimation des paramètres inconnus et dans le calcul des statistiques du modèle. Afin de résoudre ce problème, on peut utiliser deux corpus différents, un pour calculer les statistiques et l'autre pour estimer les paramètres. Malheureusement cette technique, nommée "*held-out*", nécessite des données supplémentaires [53]. Une solution consiste à remplacer les produits, dans la formule de l'approche de maximum de vraisemblance (équation : 3.42), par d'autres, où la probabilité de chaque symbole du corpus utilise des statistiques calculées sur le corpus entier excepté ce symbole. Par conséquent, l'estimation CV de θ est définie comme suit :

$$\theta^{CV} = arg \max_{\theta \in \Theta} \prod_{i=1}^{N} p(w_i; \theta / C^{(i)}) \tag{3.51}$$

où $C^{(i)}$ est le corpus C excepté le symbole w_i. Cette technique, nommée estimation effacée (*deleted estimation*), simule d'une certaine façon l'apparition de nouveaux événements dans le corpus [96].

En appliquant l'estimation CV à une distribution discrète, on obtient la formule suivante :

$$p(C;\theta) = \prod_{i=1}^{N} \left(\frac{\theta_{c(w_i)-1}}{n_{c(w_i)-1}} \right) = \prod_{r \geq 1} \left(\frac{\theta_{r-1}}{n_{r-1}} \right)^{rn_r} . \tag{3.52}$$

où n_r est le nombre de symboles qui apparaissent r fois dans C. Pour chaque symbole w_i du corpus, la statistique $n_{c(w_i)}$ est calculée sur le corpus C après suppression du symbole w_i.

En prenant le logarithme de la formule 3.52, la fonction lagrangienne est définie comme suit :

$$L(\theta,\lambda) = \sum_{r \geq 0} (r+1) n_{r+1} log(\frac{\theta_r}{n_r}) + \lambda \left(1 - \sum_{r \geq 0} \theta_r \right) . \tag{3.53}$$

En ramenant à 0 la dérivée par rapport à θ, la formule 3.53 devient :

$$\frac{\partial L}{\partial \theta_r} = \frac{(r+1) n_{r+1}}{\theta_r} - \lambda \quad r = 0,1,2,\ldots \tag{3.54}$$

Après quelques réarrangements, on obtient :

$$\sum_{r \geq 0} (r+1) n_{r+1} = \lambda \left(\sum_{r \geq 0} \theta_r \right) = \lambda \tag{3.55}$$

dont la solution est :

$$\theta_r^{CV} = \frac{(r+1) n_{r+1}}{N}. \tag{3.56}$$

Si on suppose que le symbole $v \in V$ apparaît r fois dans le corpus, la probabilité $p(v;\theta)$ est définie comme suit :

$$p(v;\theta) = \frac{\theta_r^{CV}}{n_r} = \frac{r^*}{N} \quad \text{où} \quad r^* = (r+1) \frac{n_{r+1}}{n_r} \tag{3.57}$$

Le terme r^* est la formule de Good-Turing [71]. Son utilisation avec l'estimation CV est définie en détail dans [139].

3.7 Conclusion

Nous avons présenté, dans ce chapitre, les approches de base utilisées dans la modélisation du langage. Sans chercher à être exhaustif, au vu du nombre élevé de variantes qui existent dans la littérature, nous nous sommes surtout intéressés aux modèles les plus utilisés en RAP.

Nous avons également défini quelques moyens pour pouvoir combiner ces modèles ensemble dans le but de les rendre plus performants et de pouvoir estimer la vraisemblance des évènements non rencontrés dans le corpus d'apprentissage. Enfin, nous avons évoqué les techniques les plus utilisées dans l'estimation de ces modèles, en se fondant sur des approches connues en théorie de l'information.

Dans les prochains chapitres, nous décrivons d'autres modèles de langage originaux et nous évaluons leurs performances en terme de perplexité. Ensuite, nous testons leurs capacités prédictives avec le jeu de Shannon et nous abordons leur intégration dans notre système de RAP MAUD.

Chapitre 4

Une approche hybride pour la RAP

4.1 Introduction

Les modèles de langage classiques du genre n-grammes sont très limités dans la prise en compte des phénomènes complexes d'une langue naturelle telle que la restriction d'accord. En effet, dans la langue, il y a souvent des mots qui font référence à d'autres qui ne se trouvent pas forcément n mots avant le mot courant. Ces références peuvent être de nature sémantique ou purement syntaxique. Dans la phrase : "*les pommes que j'ai mangées*", le sixième mot "*mangées*" s'accorde avec le complément "*les pommes*" qui se trouve quatre mots avant. Il n'est donc pas possible à un simple modèle de langage trigrammes ou triclasses de prendre en compte des phénomènes d'accord de ce type.

Pour mieux tenir compte des contraintes de la langue, telle que la restriction d'accord, nous proposons une approche hybride. Cette approche est une combinaison d'un modèle probabiliste avec des connaissances linguistiques explicites mise en oeuvre dans un modèle formel. Ce modèle se fonde sur des règles grammaticales se servant des classes syntaxiques issues de la langue.

Le fonctionnement général de notre approche est présenté dans ce qui suit. Nous présentons ensuite l'ensemble des classes syntaxiques utilisées ainsi que l'algorithme d'étiquetage du corpus et des hypothèses. Dans le paragraphe 4.5, nous introduisons le modèle formel fondé sur les grammaires d'unification ; nous montrons également l'algorithme utilisé pour son implantation. Nous décrivons dans le paragraphe 4.6 les corpus d'apprentissage, de développement et de test. Enfin, nous évaluons notre approche.

4.2 Fonctionnement général

L'utilisation d'une approche hybride, combinant les modèles probabilistes et les modèles structurels, a passionné depuis longtemps de nombreux chercheurs [34, 175]. A la différence des modèles probabilistes, les modèles structurels utilisent des connaissances linguistiques explicites. L'ajout de telles connaissances aux modèles probabilistes permet une meilleure prise en compte de l'ensemble des contraintes linguistiques, ce qui rend les modèles de langage plus performants.

L'approche que nous proposons dans ce chapitre est hybride : elle combine un modèle de langage probabiliste avec un modèle formel. Le modèle probabiliste est une combinaison des modèles n-grammes, *cache* et *trigger*. En revanche, le modèle formel se fonde sur des classes syntaxiques, issues de la langue, et sur des règles grammaticales représentées par une grammaire d'unification (cf. §4.5). Ces règles grammaticales ont pour objectif d'examiner la restriction d'accord des phrases reconnues.

Dans le cas des langues naturelles, en utilisant un grand vocabulaire, la construction d'un ensemble exhaustif de règles grammaticales est quasiment impossible. Aussi, dans notre modèle formel, nous nous sommes limités à inclure les règles grammaticales les plus fréquentes de la langue. Étant donnée la fréquence de ces règles, nous espérons analyser un maximum d'hypothèses et donc améliorer les performances des systèmes de RAP.

Introduire le modèle formel lors de la phase de prédiction dans un système de RAP accroîtra considérablement le temps de calcul et l'espace mémoire requis. En effet, pour un vocabulaire important, le nombre de combinaisons possibles formant des phrases est très grand. Ainsi, l'utilisation directe de modèles de langage manipulant la totalité de l'historique, ce qui est le cas de notre modèle formel, demande un effort de calcul trop important pour l'amélioration du taux de reconnaissance qui en découle. En revanche, la liste des N meilleures suites de mots peut être facilement estimée de nouveau par des modèles de langage utilisant de grands historiques avec un coût de calcul relativement faible. De plus, pour examiner la restriction d'accord dans une phrase, il faut que sa structure linguistique soit valide ; or, cette décision de validité ne peut être prise qu'après avoir reconnu tous les mots de la phrase.

La meilleure phrase reconnue ne coïncide pas toujours effectivement avec la phrase prononcée. De récents travaux ont montré l'utilité de générer une liste des N meilleures suites de mots reconnues afin de trouver, avec l'aide d'informations supplémentaires, la phrase effectivement énoncée (par exemple, la phrase satisfaisant à des restrictions supplémentaires ou encore la meilleure phrase après une réévaluation des scores respectifs de chaque élément de la liste). Les N meilleures suites de mots obtenues doivent être différentes et non pas des alignements distincts d'une même hypothèse [50].

Nous avons ainsi opté pour l'introduction du modèle formel lors de l'étape de sélection. Les N meilleures suites de mots produites initialement par le système de RAP, avec un modèle de langage probabiliste, sont filtrées pour fournir la meilleure hypothèse, résultat final du système. Le modèle formel se sert des classes syntaxiques lors de l'analyse. Ainsi, son utilisation nécessite une étape préliminaire d'étiquetage, qui consiste à attribuer à chaque mot appartenant à la liste des hypothèses sa classe syntaxique en fonction du contexte. Une fois l'étiquetage effectué, ce modèle formel supprime toutes les hypothèses qui ne respectent pas les accords en genre, en nombre et en personne.

4.3 Les classes syntaxiques

L'objectif de la classification, que nous présentons dans ce paragraphe, est de répartir les mots en plusieurs sous-ensembles ayant un comportement syntaxique très proche. Chacun de ces sous-ensembles est désigné par une classe syntaxique et regroupe un certain nombre de mots sur la base de propriétés syntaxiques communes. Ainsi, nous procédons à une classification manuelle qui nous permet de bien prendre en compte ces genres de contraintes. Un mot w appartient à une classe C_i si et seulement si il peut être substitué à tous les mots w_i de C_i ($\forall w_i \in C_i$) dans la phrase, tout en préservant la validité syntaxique de celle-ci. Dans notre classification, nous nous sommes tenus au maximum à cette règle théorique, même si dans certains cas ce n'était pas vraiment possible.

Le problème de la classification syntaxique est bien connu chez les linguistes qui pensaient que les dénombrements de formes étaient irréalisables, étant donné le nombre considérable de formes différentes. M. Gross s'est intéressé aux propriétés des verbes du français. Pour 3000 verbes, son système de classification a fourni un ensemble de 2000 classes [74], ce qui fait en moyenne 1,5 verbe par classe. Gross conclut qu'il n'y a pas deux verbes qui ont les mêmes propriétés syntaxiques.

Cette affirmation se justifie par le fait qu'il suffit de prendre une nouvelle propriété syntaxique et d'examiner les éléments d'une même classe. On s'apercevra très vite que la classe se divise en plusieurs sous-classes. Nous sommes tout à fait d'accord avec cette conclusion à laquelle nous sommes arrivés, ainsi que K. Smaïli dans [182], après l'analyse de nos corpus et nos vocabulaires. Notre objectif n'est pas de pousser la classification aussi loin que celle effectuée par M. Gross : en effet, avoir un nombre très important de classes réduira leurs fréquences d'apparition dans le corpus.

Pour extraire les classes syntaxiques, nous avons utilisé l'étude approfondie faite par K. Smaïli [182], fondée sur de nombreux ouvrages de grammaire et d'orthographe [72, 98]. Dans cette classification, les mots ont été répartis dans les huit classes grammaticales de la langue française : adverbe, adjectif, article, conjonction, nom, préposition, pronom et verbe. Ainsi, les mots ayant plusieurs comportements syntaxiques (par exemple, le mot "*porte*" qui peut être considéré comme un nom ou comme un verbe) seront dupliqués et répartis dans les classes auxquelles ils appartiennent. Ensuite, les mots de chaque classe grammaticale ont été répartis en deux sous-ensembles formant ainsi des classes ouvertes et des classes fermées. Les classes ouvertes reçoivent les mots qui peuvent être formés à partir d'autres mots comme, par exemple, les adverbes qui se terminent en "**ment**". Elles peuvent recevoir également des mots comme les noms propres. Ainsi, dix classes ouvertes ont été créées contenant : des adverbes (AD1), des adjectifs (ADJ), des cardinaux (CAR), des noms communs (NOM), des noms propres (NOP), des ordinaux (ORD), des participes passés s'employant avec les deux auxiliaires (PPx), des participes présents (SEN), des verbes conjugués (VEC) et des verbes à l'infinitif (VEI). Pour plus de précision sur ces classes, le lecteur pourra se référer à l'annexe A. Le deuxième sous-ensemble, formant les classes fermées, contient les mots véhiculant des informations syntaxiques et même sémantiques très importantes. Les classes correspondant à ce sous-ensemble sont affinées par dichotomie : nous divisons en deux cette partition et nous regardons si les mots appartenant à chaque nouvelle classe vérifient bien la contrainte d'appartenance. Si ce n'est pas le cas, un nouveau découpage est effectué, jusqu'à ce que la contrainte soit vérifiée ou jusqu'à la formation de singletons. Les mots d'une même classe sont testés, autour de l'axe paradigmatique de la phrase, dans les différents contextes possibles. Le mot est définitivement ajouté s'il peut être substitué à tous les mots de la classe dans tous les contextes. Ainsi, pour la classe des adverbes contenant "*moins*" et "*plus*" notée MPA, ce procédé a été appliqué comme dans l'exemple suivant :

$$\text{Des hypothèses pour le} \left| \begin{array}{c} \text{moins} \\ \text{plus} \end{array} \right| \text{audacieuses.}$$

Nous rappelons que nous n'avons pas été très stricts quant à l'application de la contrainte d'appartenance d'un mot à une classe ; notamment pour minimiser le nombre de classes syntaxiques. Nous donnons, dans ce qui suit, quelques informations concernant la répartition des mots dans les classes grammaticales ainsi que les classes créées qui leurs appartiennent. Le détail de ces classes, qui sont au nombre de 233, se trouve dans l'annexe A.

Les adverbes

Un adverbe est un mot invariable que l'on joint à un verbe, à un adjectif ou à un autre adverbe, pour en modifier le sens. Cette classe contient deux partitions :

- la première contenant tous les adverbes qui se terminent par la syllabe "**ment**",
- la deuxième est composée de 84 sous-classes d'adverbes ; la plupart de ces sous-classes contiennent des singletons.

Le processus dichotomique utilisé a montré qu'il est difficile de trouver des critères de regroupement de ces adverbes. En effet, quel critère de regroupement peut on trouver entre les adverbes "*aujourd'hui*" et "*oui*" ?

Les adjectifs

L'adjectif est un mot que l'on joint au nom pour le qualifier ou pour le déterminer. Cette classe est également divisée en deux :

- une partition ouverte qui contient les adjectifs qualificatifs,
- une partition des autres adjectifs qui est divisée en 12 sous-classes représentant, entre autres, les adjectifs possessifs, démonstratifs, interrogatifs, exclamatifs, indéfinis, ...

Les articles

L'article est un mot que l'on place devant le nom pour indiquer si ce nom est pris dans un sens défini ou indéfini ; il sert aussi à préciser le genre et le nombre du nom. Cette classe ne contient pas de partition ouverte et elle est composée de trois sous-classes syntaxiques : les articles définis, les articles indéfinis et les articles contractés "*au*" et "*aux*".

Les conjonctions

La conjonction est un mot invariable qui sert à joindre et à mettre en rapport, soit deux propositions, soit deux mots de même fonction dans une proposition. Cette classe grammaticale est composée de 12 sous-classes syntaxiques.

Les noms

Le nom sert à désigner des êtres, des choses ou des idées. Nous avons décomposé cette classe grammaticale en 4 sous-classes syntaxiques : les noms communs, les noms propres, les jours de la semaine et les mois de l'année. Les deux dernières classes ont été séparées de la classe des noms communs à cause des informations sémantiques qu'elles véhiculent.

Les prépositions

La préposition est un mot invariable ; elle sert de charnière entre un mot et son complément. La classe des prépositions est composée de 25 sous-classes syntaxiques.

Les pronoms

Le pronom est un mot qui, en général, représente un nom, un adjectif, une proposition. La classe des pronoms contient 37 sous-classes syntaxiques. Dans ces sous-classes, nous distinguons surtout les pronoms démonstratifs, les pronoms indéfinis, les pronoms interrogatifs, les pronoms personnels, et les pronoms attributs renforcés (*moi-même, toi-même, etc.*).

Les verbes

Le verbe est un mot qui exprime, soit l'action faite ou subie par le sujet, soit l'existence ou l'état du sujet. Cette classe grammaticale est composée de 8 sous-classes syntaxiques réparties comme suit : deux classes de verbes auxiliaires, une classe de participes présents utilisés en forme verbale, trois classes de participes passés utilisés en forme verbale, une classe des verbes à l'infinitif et une classe contenant les verbes conjugués.

Les autres

Cette classe contient 14 sous-classes syntaxiques ; elle diffère des précédentes parce-qu'elle comporte des sous-classes qui ne partagent pas forcément les propriétés de celle qui les englobe. Certaines de ces sous-classes ont été mises dans cette classe à cause de la difficulté qu'elles peuvent poser au moment de l'étiquetage. En effet, le mot "*même*" par exemple peut avoir trois fonctions syntaxiques : adjectif (*les mêmes machines*), pronom (*il est resté le même*) et adverbe (*il lui arrive même de jeûner*). Nous avons donc préféré placer le mot "*même*" dans une classe à part. Nous trouvons également dans cette classe deux sous-classes contenant chacune une lettre euphonique. Rappelons qu'une lettre euphonique est placée entre deux voyelles pour éviter la réalisation d'une prononciation désagréable. Ainsi, nous avons créé la sous-classe PH1 contenant le mot "l' " ; ce mot n'a pas de rôle grammatical mais donne des informations concernant les mots qui peuvent le précéder ou le suivre. Nous avons créé également une deuxième classe PH2 contenant le mot "t' " ; ce mot est lui même une lettre euphonique. Cette classe (PH2) ne peut précéder que les mots suivants : "*il*", "*elle*" ou "*on*".

Nous notons que certaines de ces classes syntaxiques peuvent véhiculer des concepts sémantiques. En effet, l'examen de certaines de ces classes montre que les éléments qui les composent partagent un même concept. Citons, à titre d'exemple, la classe ANL qui est composée des adjectifs indéfinis "*aucun*" et "*nul*" et qui suggère une idée de quantité zéro ; la classe HEX, contenant "*excepté*" et "*hormis*", reflète l'idée de l'exception ; la classe VVO, contenant "*voici*" et "*voilà*", présente une idée de présentation.

4.4 Étiquetage du corpus

L'utilisation du modèle formel nécessite l'attribution de la classe syntaxique à chaque mot issu des hypothèses produites par le système de RAP. De plus, l'apprentissage des modèles de langage utilisant les classes syntaxiques citées ci-dessus, nécessite à son tour un texte syntaxiquement résolu. En effet, pour estimer les vraisemblances, nous avons besoin d'affecter à chaque mot du corpus sa classe syntaxique, en fonction du contexte. Par conséquent, étant donné qu'un mot peut appartenir à plusieurs classes syntaxiques, il est nécessaire de trouver une manière d'affecter la bonne classe à un mot donné du corpus.

La taille du corpus doit être assez grande pour avoir des vraisemblances représentatives de la langue considérée. Ainsi, l'étiquetage d'un tel corpus représente un travail considérable qu'il n'est pas envisageable de faire à la main. Nous utiliserons donc le processus d'apprentissage supervisé pour effectuer l'étiquetage de nos corpus [182]. L'idée est la suivante : au départ, nous étiquetons manuellement une partie du corpus et nous calculons les statistiques sur cette partie étiquetée. Ensuite, nous utilisons ces statistiques pour étiqueter une autre partie du corpus. Puis, nous corrigeons à la main les erreurs de l'étiquetage automatique et nous recalculons les statistiques sur la totalité du corpus étiqueté. Ces nouvelles statistiques seront ainsi utilisées dans l'étiquetage d'une autre partie du corpus, qui sera elle également corrigée manuellement. Ce processus sera répété jusqu'à épuisement de tout le corpus d'apprentissage. Dans notre cas, nous avons commencé par étiqueter manuellement un texte de quelques milliers de mots. Ensuite, nous avons utilisé les statistiques, estimées sur ce texte, pour étiqueter automatiquement un corpus de 0.5 million de mots extraits du journal "L'Est Républicain". Ainsi, une fois l'étiquetage automatique terminé, nous avons corrigé manuellement les erreurs d'étiquetage et nous avons estimé à nouveau les statistiques sur la totalité du corpus étiqueté. Nous notons que les premières statistiques, estimées sur un texte de quelques milliers de mots, nous ont donné un taux d'étiquetage correct d'environ 97%. Enfin, nous utilisons ces nouvelles statistiques pour étiqueter les 50 millions de mots extraits du journal "Le Monde".

Pour étiqueter automatiquement une phrase, nous procédons comme suit : nous affectons à chaque mot du corpus toutes ses classes syntaxiques hors contexte. Pour ceci, nous utilisons une composante lexicale qui cherche le mot en question dans un lexique comportant 230000 formes étiquetées. Si le mot ne se trouve pas dans ce lexique, nous lui affectons toutes les classes syntaxiques ouvertes, c'est-à-dire toutes les classes où nous risquons de ne pas disposer de tous les mots (cf. §4.3). Ensuite, pour chaque phrase et en fonction des classes syntaxiques auxquelles peut appartenir chacun des mots qui la composent, nous construisons un graphe représentant les différents chemins syntaxiques possibles. Prenons un exemple pour illustrer cette idée ; soit la phrase : "*Elle porte des chaussures de luxe*". La première phase d'affectation des classes donne :

Elle	porte	des	chaussures	de	luxe	.
PPE	NOM	DES	NOM	DDE	NOM	POT
PAT	VEC					

La deuxième phase de construction de graphe est présentée sur la figure 4.1.

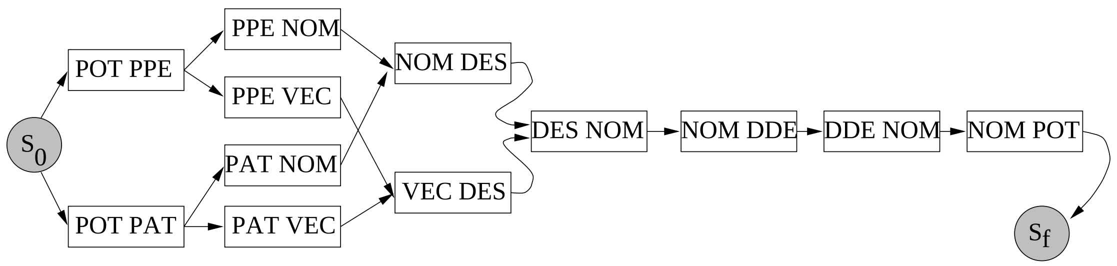


FIG. 4.1 – *Exemple de construction de graphe de la phrase : "Elle porte des chaussures de luxe"*

Chaque état de ce graphe est représenté par une suite de deux classes indiquant la classe précédente et celle en cours. La transition d'un état à un autre forme une suite de trois classes produisant ainsi une probabilité triclasses (3-classes) de transition. Cette définition ne s'applique pas aux états fictifs S_0 et S_f. Pour effectuer l'étiquetage automatique, il suffit donc de trouver le meilleur chemin reliant S_0 à S_f. Soit $G = (X,\Gamma)$ le graphe à étudier, où X représente l'ensemble des états de ce graphe et $\Gamma(j)$ l'ensemble des successeurs de l'état j ($j \in X$). Nous notons par $P(j,k)$ la probabilité de transition entre les états j et k, où $j \in X$ et $k \in \Gamma(j)$. Ainsi, pour calculer le meilleur chemin entre S_0 et S_f, nous procédons avec le principe de la programmation dynamique, comme suit :

$$\begin{aligned} F_1(j) &= P(S_0,j) \\ F_i(j) &= \max_{k \,\in\, \Gamma^{-1}(j)} \{F_{i-1}(k).P(k,j)\} \end{aligned} \tag{4.1}$$

où $\Gamma^{-1}(j)$ désigne l'ensemble des antécédents de l'état j dans G et $F_1(j)$ est initialisé à 1.

Pour estimer la probabilité de transition $P(j,k)$ nous avons utilisé le principe de maximum de vraisemblance défini dans le chapitre 3 pour les modèles n-grammes. Dans le but de résoudre le phénomène des suites de classes non rencontrées dans le corpus d'apprentissage, nous interpolons linéairement la distribution triclasses (3-classes) avec la distribution biclasses (2-classes), la distribution uniclasses (1-classes) et la distribution uniforme (0-classes) (cf. §3.5). Ainsi dans le cas où $k \in \Gamma(j)$, si j représente la suite des deux classes j_1,j_2 et k représente la suite des deux classes j_2,k_1, la probabilité de transition est estimée par la formule suivante :

$$P(j,k) = \alpha \frac{N(j_1,j_2,k_1)}{N(j_2,k_1)} + \beta \frac{N(j_2,k_1)}{N(k_1)} + \gamma \frac{N(k_1)}{T} + \delta \tag{4.2}$$

où $N(.)$ est le nombre d'occurrences de la suite de classes en argument dans le corpus d'apprentissage et T est la taille de ce corpus. Les variables α, β, γ et δ représentent les coefficients de pondération ($\alpha + \beta + \gamma + \delta = 1$). En gardant les mêmes notations que précédemment, ces variables sont estimées sur un corpus d'évaluation (différent de celui de l'apprentissage) de la façon suivante : pour chaque triclasse (suite de 3 classes c_1, c_2 et c_3) de ce corpus, nous regardons sa probabilité ($\frac{N(c_1,c_2,c_3)}{N(c_1,c_2)}$) estimée à partir du corpus d'apprentissage. Si elle n'est pas nulle, nous incrémentons la variable α de 1. Sinon, nous regardons la probabilité du biclasses $\frac{N(c_2,c_3)}{N(c_2)}$ (sur le corpus d'apprentissage, bien sûr). Si cette probabilité n'est pas nulle, nous incrémentons β de 1, sinon nous regardons la probabilité de c_1 ($\frac{N(c_1)}{T}$). Si elle est nulle nous incrémentons δ, sinon nous incrémentons γ. Une fois le corpus de développement parcouru, nous ajustons les variables α, β, γ et δ pour que leur somme soit égale à 1.

4.5 Le modèle de langage formel

En analysant les N meilleures hypothèses d'un système de RAP, nous remarquons que la meilleure, qui respecte les phénomènes d'accord et qui coïncide effectivement avec la phrase prononcée, ne se trouve pas souvent en première position. Nous avons ainsi construit un modèle formel ; il se fonde sur un ensemble de règles grammaticales lui permettant de supprimer toutes les hypothèses qui ne vérifient pas la restriction d'accord.

Pour examiner la structure syntaxique d'une hypothèse (phrase), le modèle est obligé de tenir compte de :

- la grammaire, qui est la spécification formelle des structures permises dans la langue ;
- la technique d'analyse, qui est la méthode d'analyse utilisée pour déterminer la structure d'une phrase selon une grammaire.

Nous abordons dans le sous-paragraphe qui suit, les grammaires formelles telle qu'elles sont considérées par Chomsky. Ensuite, dans le paragraphe 4.5.2, nous présentons les grammaires à traits, extension des grammaires formelles. Ces grammaires ont l'avantage de mieux gérer certains phénomènes du langage naturel, tels que les phénomènes d'accord. Nous exposons les traits que nous avons choisis pour notre modèle dans le paragraphe 4.5.3. Nous définissons dans le paragraphe 4.5.4 le formalisme des grammaires d'unification, qui font partie des grammaires à traits et qui ont été à la base de notre modèle. Nous montrons ensuite dans le paragraphe 4.5.5 les réseaux de transition augmentés, utilisés pour représenter notre grammaire d'unification. Nous abordons par la suite dans le paragraphe 4.5.6 la technique d'analyse utilisée pour la vérification de la restriction d'accord dans une phrase. Enfin, dans le paragraphe 4.5.7, nous exposons le fonctionnement général du modèle formel. Les classes syntaxiques utilisées sont celles présentées dans le paragraphe 4.3.

4.5.1 Définition des grammaires formelles

Nous présentons ici quelques notions essentielles de la théorie des grammaires formelles, telle qu'elle est considérée par Chomsky [37]. Le lecteur qui s'intéresse plus particulièrement aux propriétés intrinsèques de ces formalismes pourra se référer à [75] ; en ce qui nous concerne, dans cette thèse, nous insistons surtout sur certaines caractéristiques dues à l'utilisation de cette théorie dans le cadre du traitement automatique des langues.

Le premier concept important, pour la définition d'une grammaire, est le vocabulaire : noté V. Ce vocabulaire V est composé de deux sous-ensembles finis et disjoints : le vocabulaire terminal V_T (l'ensemble des symboles pouvant apparaître dans les phrases d'une langue) et le vocabulaire non terminal V_N (l'ensemble des symboles nécessaires à la description de cette langue, nommés également "variables" ou "catégories syntaxiques"). Ainsi, la phrase sera beaucoup plus structurée grâce à la considération de catégories syntaxiques comme groupe nominal, groupe prépositionnel, groupe verbal, etc. En outre, cela permet de présenter de manière beaucoup plus nette les régularités de la langue : l'existence de la notion de *groupe nominal* permet de mettre en évidence les analogies de comportement, au niveau de la phrase, des différentes formes sous lesquelles il peut se réécrire (le nom, les groupements nom+adjectif, nom+relative, etc. jouent des rôles interchangeables). Un langage sur V_T est alors défini comme un ensemble (potentiellement infini) de chaînes de longueurs finies formées avec des éléments de V_T. V^*, le monoïde libre construit sur V_T, est donc l'ensemble des chaînes finies formées en combinant les symboles terminaux de toutes les façons possibles. Un langage est donc défini comme un sous-ensemble de V^*.

L'élément essentiel qui va ensuite permettre de tenir compte de la structure interne de la phrase est la règle de réécriture (appelée également règle de production). Un tel ensemble de règles spécifie les relations permises entre des chaînes formées de symboles de V (terminaux ou non). Ainsi, par exemple, le fait qu'une phrase (P) puisse être composée d'un groupe nominal (GN) suivi d'un groupe verbal (GV) sera représenté par une règle de la forme :

$$P \rightarrow GN + GV$$

où le + est le symbole de la concaténation, et la flèche se lit comme une instruction ordonnant la réécriture du symbole de gauche en utilisant les symboles de droite.

Enfin, parmi les symboles non terminaux de V_N on distingue un symbole particulier : le symbole origine (il est en général noté P [pour "Phrase"] ou S [pour "Sentence"]).

Une grammaire formelle est alors définie par le quadruplet :

$$G = (V_N, V_T, R, P)$$

où R représente l'ensemble des règles de réécriture et P représente le symbole d'origine. Ainsi, on appelle langage engendré par la grammaire G, l'ensemble de toutes les chaînes terminales que l'on peut dériver du symbole origine en appliquant toutes les séquences possibles des règles de réécriture.

N. Chomsky a défini quatre types de grammaires : les grammaires régulières, les grammaires à contexte libre, les grammaires sensibles au contexte et les grammaires générales.

Les grammaires régulières

On distingue les grammaires régulières à gauche et les grammaires régulières à droite. Notons par X et Y des éléments non terminaux et "b" un élément terminal ($X,Y \in V_N$ et $b \in V_T$), les règles sont de la forme :

$$\begin{array}{llll} X \rightarrow b + Y & \text{ou} & X \rightarrow b & \text{pour les premières} \\ X \rightarrow Y + b & \text{ou} & X \rightarrow b & \text{pour les secondes.} \end{array}$$

Ces grammaires sont également appelées grammaires d'états finis. En effet, on peut montrer que le mécanisme de génération correspond à un automate d'état fini ; en d'autres termes, cela signifie qu'au cours de la génération, la seule information nécessaire pour terminer la phrase correctement est la connaissance de l'état dans lequel on se trouve : la connaissance de tout ce qui précède est inutile pour trouver une continuation correcte de la phrase. Cette caractéristique montre clairement que ces types de grammaires sont insuffisants pour traiter les divers aspects de la langue.

Les grammaires à contexte libre

Les règles de ces grammaires sont telles que le membre gauche de la règle ne peut être qu'un (et un seul) symbole non terminal, appartenant à V_N. Ces grammaires n'expriment aucune contrainte sur le membre droit de ces règles. Elles sont dites indépendantes du contexte car les règles signifient que le membre gauche peut être réécrit sous la forme du membre droit, indépendamment du contexte (des symboles qui l'entourent). La grammaire composée des deux règles qui suivent en est un exemple :

$$P \rightarrow a + P + b \qquad P \rightarrow a + b$$

où les symboles a et b appartiennent à V_T et le symbole P appartient à V_N.

Nous donnons dans la table 4.1 un exemple très simplifié d'un fragment de la langue française.

1.	$P \rightarrow GN + GV$	5.	$PPE \rightarrow$ *il* / *elle* /...
2.	$GN \rightarrow PPE$	6.	$VEC \rightarrow$ *mange* / *lit* /...
3.	$GN \rightarrow ARI + NOM$	7.	$ARI \rightarrow$ *une* / *un* /...
4.	$GV \rightarrow VEC + GN$	8.	$NOM \rightarrow$ *pomme* / *livre* /...

TAB. 4.1 – *Exemple d'une grammaire à contexte libre.*

Cette grammaire permet de décrire et de produire la phrase "*il mange une pomme*" par application successive des règles numéros 1, 2, 4, 5, 6, 3, 7 et 8 (ce n'est bien sûr pas le seul ordre possible). Elle peut engendrer également : "*elle lit un livre*" et permet de rejeter des phrases comme : "*livre lit pomme elle*" ou "*pomme livre un mange*". L'ensemble des règles utilisées pour décrire ou produire une phrase peut être représenté par un arbre de dérivation. Nous présentons sur la figure 4.2, l'arbre de dérivation de la phrase "*il mange une pomme*" en utilisant la grammaire de la table 4.1. Ce formalisme correspond à la notion d'automate à pile.

D'après les quelques règles ci-dessus, on voit que les phrases qui n'ont pas de sens et même des phrases syntaxiquement incorrectes peuvent être engendrées. Nous citons à titre d'exemple les deux phrases suivantes : "*il lit un pomme*" et "*elle mange une livre*". Ces exemples n'ont bien sûr qu'une valeur d'illustration, et ne sont pas une démonstration puisque la grammaire présentée est excessivement simplifiée. Néanmoins, vu leur facilité d'implantation sur ordinateur, beaucoup de travaux ont été réalisés sur ces grammaires non contextuelles qui sont malgré tout suffisantes dans des applications restreintes.

Les grammaires sensibles au contexte

Ces grammaires sont considérées comme une extension des grammaires précédentes pour qu'elles puissent tenir compte du contexte. En effet, considérons les phrases formées d'un nombre

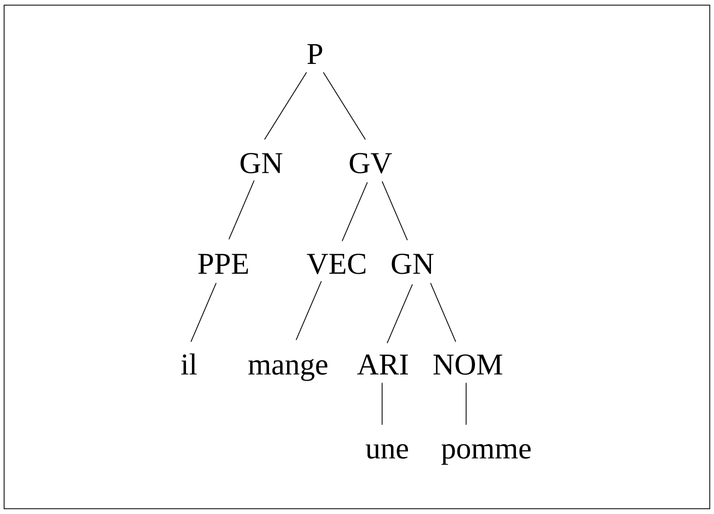


FIG. 4.2 – *Arbre de dérivation de la phrase : “il mange une pomme”.*

quelconque de “a” suivi du même nombre de “b”, suivi du même nombre de “c”. Ce langage ne peut être engendré par une grammaire non contextuelle : ses propriétés sont en fait des contraintes récursives de croisement (le premier “a”, le premier “b” et le premier “c” se correspondent, les deuxièmes “a,b,c” se correspondent, de même les troisièmes, etc.). Les règles de la table 4.2 permettent d'engendrer ce langage.

P → a + B + C	P → a + P + B + C
C + B → B + C	a + B → a + b
b + B → b + b	b + C → b + c
c + C → c + c	

TAB. 4.2 – *Exemple d'une grammaire sensibles au contexte.*

Nous présentons ici une définition équivalente, expliquant pourquoi ces grammaires sont appelées sensibles au contexte. En effet, les règles sont de la forme :

$$u + x + v \rightarrow u + y + v$$

où $u,v,y \in V^*$ et $x \in V_N$. Ceci peut être interprété comme : x peut se réécrire en y dans le contexte $u - -v$ (c'est-à-dire s'ils sont l'un et l'autre entourés de u et v).

La première définition montre que les langages engendrés par ces grammaires sont décidables. En effet, elle implique que le nombre de symboles de la chaîne engendrée ne peut que croître lors de l'application des règles de réécriture. Étant donnée une phrase de longueur n, si on ne l'a pas rencontrée après avoir engendré toutes les chaînes de longueur au moins égale à n, on est sûr qu'on ne la rencontrera pas par la suite. Elle n'appartient donc pas au langage. Ces grammaires engendrent ainsi des langages qui peuvent être reconnus par un automate linéaire borné (un processus déterministe utilisant une mémoire proportionnelle à la longueur de la phrase en entrée).

Le formalisme pur des grammaires contextuelles fut assez peu utilisé tel quel. En toute généralité, ce type de règles reste difficile à mettre en oeuvre dans des programmes. Les applications informatiques ont surtout tenté d'augmenter la puissance générative des grammaires non contextuelles afin de les rendre équivalentes aux grammaires contextuelles [170].

Les grammaires générales

On n'introduit aucune contrainte sur la forme des règles de réécriture. Les langages correspondants sont reconnus par une machine de Turing générale. Ce type de système non limité est trop peu structuré pour pouvoir servir de grammaire. Certes, en imposant des contraintes supplémentaires, on perd de la puissance, mais on obtient des systèmes qui présentent plus d'intérêt linguistique. En outre, G. Sabah annonce qu'il a pu remarquer que toutes les grammaires qui ont été écrites pour des langues naturelles pouvaient être réécrites en utilisant le formalisme des règles contextuelles, même si l'utilisation des règles générales de réécriture permet une écriture plus souple [170].

4.5.2 Les grammaires à traits

Les grammaires formelles et tout particulièrement les grammaires à contexte libre sont souvent à la base de nombreux travaux en traitement automatique des langages. Néanmoins, ces grammaires ne permettent pas d'analyser tous les aspects du langage naturel tels que les phénomènes d'accord. Ainsi, pour pouvoir gérer certains phénomènes du langage naturel, une extension a été apportée à ces grammaires. En effet, une des premières idées fut l'adjonction, aux constituants de la structure de la phrase, de traits ne correspondant pas forcément à des mots [113].

La notion de traits

Notre objectif ici est d'illustrer la notion de traits utilisée par ces grammaires. Soit le groupe nominal "*le garçons*". Ce groupe nominal n'est pas valide dans la langue française, car l'article "*le*" est au singulier tandis que le nom "*garçons*" est au pluriel. Pour qu'une grammaire formelle classique tienne compte de cette contrainte (accord en nombre), il est nécessaire de dupliquer les règles de réécriture (une fois pour le singulier et une autre fois pour le pluriel). Ainsi, la taille de la grammaire sera multipliée par deux :

$$\begin{array}{lll} GN(Singulier) & \rightarrow & ARD(Singulier)\ \ NOM(Singulier) \\ GN(Pluriel) & \rightarrow & ARD(Pluriel)\ \ NOM(Pluriel) \end{array} \tag{4.3}$$

Si, de plus, nous voulons tenir compte d'autres contraintes langagières, tel que l'accord en genre, on sera encore obligé de doubler la taille de la grammaire.

Pour alléger cette taille, une solution consiste à ajouter un ensemble de traits aux constituants de la grammaire de base. On pourra donc définir un trait "NOMBRE" qui prend comme valeur soit "s" (singulier) soit "p" (pluriel). Ainsi, la règle de réécriture 4.3 devient :

$$\begin{array}{ll} GN \rightarrow ARD\ NOM & (\text{seulement si } NOMBRE \text{ de } ARD \\ & \quad \text{s'accorde avec } NOMBRE \text{ de } NOM). \end{array} \tag{4.4}$$

Par conséquent, chaque constituant de la grammaire est défini par une *structure de traits*, contenant un certain nombre de ces valeurs appropriées.

Discussions et choix

L'objectif du modèle formel que nous développons est surtout de tenir compte au maximum des restrictions d'accord imposées par la langue. Les grammaires formelles et particulièrement les grammaires à contexte libre sont assez restreintes, de sorte que des programmes d'analyse syntaxique efficaces puissent être établis facilement pour analyser les phrases. Nous trouvons en revanche, parmi les grammaires à traits, les grammaires fonctionnelles qui donnent un rôle primordial au lexique (un dictionnaire de mots) [101, 168]. Celles-ci développent une analyse fondée principalement sur les caractéristiques syntaxiques et sémantiques des mots de la phrase. Alors que le phénomène d'insertion lexicale fut toujours un point délicat de la grammaire générative, les théories de ces grammaires visent à expliciter les articulations entre le niveau lexical, le niveau syntaxique et le niveau sémantique ; elles aboutissent ainsi, en général, à des modèles plus riches et d'une souplesse plus grande que celui de Chomsky. De plus, elles répondent parfaitement à nos besoins de vérification des phénomènes d'accord.

Le lexique (dictionnaire) joue également un rôle très important dans les systèmes de RAP actuels. En effet, le choix du lexique influe fortement sur le taux de reconnaissance du système. Ainsi, étant donné que les grammaires fonctionnelles accordent également un rôle primordial au lexique, nous les avons retenues pour la construction de notre modèle. Ces grammaires considèrent les connaissances lexicales, les connaissances sur les structures et les règles de grammaire de façon uniforme : ce sont des expressions de contraintes. Nous trouvons dans la littérature plusieurs types de grammaires fonctionnelles à traits : les grammaires d'unification, les grammaires lexicales fonctionnelles, les travaux de M. Gross et le modèle Sens-Texte [170]. Pour choisir le type de grammaire qui nous convient le mieux, nous nous sommes surtout fondés sur celles qui s'intègrent et qui s'implantent le plus facilement.

Le principal intérêt des grammaires d'unification réside dans l'utilisation d'un système unique de représentation, ce qui permet une écriture aisée et une utilisation conjointe des divers types de connaissances (lexicales, syntaxiques, sémantiques et éventuellement pragmatiques). De plus, ces grammaires sont très adaptées au traitement informatique. En effet, contrairement à de nombreuses théories linguistiques, elles ont été développées selon de bons critères de calculabilité informatique [5]. Pour la définition de notre modèle formel, nous avons donc retenu les grammaires d'unification. Nous présentons en détail, dans le paragraphe 4.5.4, le fonctionnement de ces grammaires. Préalablement, nous définissons dans ce qui suit l'ensemble des traits choisis.

4.5.3 Les traits utilisés

Étant donné que notre objectif est surtout de vérifier les phénomènes d'accord, nous nous sommes limités à un nombre restreints de traits : la forme du verbe, le genre, le nombre et la personne. Cela permet déjà d'éliminer une quantité importante d'hypothèses que les modèles statistiques laissent passer.

Le trait de la forme du verbe

Le trait **VFORM** a pour but de spécifier la forme d'un verbe conjugué (VEC) : passé simple, passé composé, présent, futur, ... Les valeurs de ce trait appartiennent à l'ensemble des temps possibles du français.

Le trait du genre

Le trait **GENRE** ne peut prendre que les valeurs "m" et "f" pour masculin et féminin.

Le trait du nombre

Le trait **NOMBRE** se limite à deux valeurs : "s" pour le singulier et "p" pour le pluriel.

Le trait de la personne

Ce trait nous permet de vérifier la restriction d'accord en nombre qui existe à différents niveaux de l'analyse, surtout au niveau de l'accord entre le sujet et le verbe. Cependant, il est nécessaire, en plus de la restriction d'accord en nombre, que les verbes et les sujets s'accordent à une autre dimension qui est la personne. Nous construisons ainsi le trait **NBP**, regroupant le nombre et la personne, dont les valeurs sont comme suit :

- 1s : pour spécifier la première personne du singulier,
- 2s : pour spécifier la deuxième personne du singulier,
- 3s : pour spécifier la troisième personne du singulier,
- 1p : pour spécifier la première personne du pluriel,
- 2p : pour spécifier la deuxième personne du pluriel,
- 3p : pour spécifier la troisième personne du pluriel.

4.5.4 Les grammaires d'unification

Le principe de base des grammaires d'unification est d'utiliser le même formalisme pour représenter les mots du dictionnaire, les règles de grammaire et les structures internes des phrases [102, 179]. Un constituant sera donc décrit par un ensemble de couples (traits) :

$$\{(attribut = valeur)\}$$

où chaque couple est considéré comme une description partielle de ce constituant, indépendante des autres. L'ordre de ces couples n'est pas pertinent.

Ainsi, chaque constituant est défini par une structure de traits, contenant un certain nombre de couples (attribut, valeur) appropriés. La *structure de traits* du constituant ARD1, par exemple, représentant le mot "le", peut être écrite comme suit :

$$\begin{array}{ll} ARD1: & (\textbf{CAT}\ \ ARD \\ & \textbf{RACINE}\ \ \text{le} \\ & \textbf{GENRE}\ \ m \\ & \textbf{NOMBRE}\ \ s) \end{array}$$

où les mots en gras représentent les attributs. Généralement, l'attribut de la catégorie (CAT) n'est pas représenté. Ainsi la formule ci-dessus s'écrit :

$$ARD1: \ (ARD,\ \textbf{RACINE}\ \text{le},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ s).$$

Les *structures de traits* peuvent également être utilisées pour représenter des constituants plus grands, composés de sous-constituants. Pour ce faire, il suffit de considérer la *structure de traits* comme une simple valeur et de représenter ainsi les sous-constituants par des attributs spéciaux fondés sur des numéros (1,2, ...). Par conséquent, le constituant $GN1$ ($GN \rightarrow ARD\ \ NOM$) de la suite de mots "*le garçon*" est représenté comme suit :

$$\begin{array}{lll} GN1: & (GN, & \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ s, \\ & & \textbf{1}\ (ARD,\ \textbf{RACINE}\ \text{le},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ s), \\ & & \textbf{2}\ (NOM,\ \textbf{RACINE}\ \text{garçon},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ s)). \end{array}$$

Ces structures de traits peuvent également représenter les ambiguïtés qui existent pour certains constituants. Dans le mot "temps" par exemple, il existe une ambiguïté du nombre : singulier ou pluriel. Ainsi, en se fondant sur ce qui est mentionné ci-dessus, nous devons avoir deux occurrences du mot “temps” qui diffèrent seulement par le contenu de l'attribut NOMBRE :

$$(NOM,\ \textbf{RACINE}\ \text{temps},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ s)$$
$$(NOM,\ \textbf{RACINE}\ \text{temps},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ p).$$

Or, pour des raisons de simplicité, il suffit de garder une seule occurrence et d'affecter aux attributs entachés d'ambiguïté, toutes les valeurs possibles. Ainsi, le constituant $NOM1$, représentant le mot “temps”, est défini comme suit :

$$NOM1:\ \ (NOM,\ \textbf{RACINE}\ \text{temps},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ \{s{,}p\}).$$

La notion de superposition

Le principal concept sur lequel se fondent les grammaires d'unification est la notion de superposition. Cette notion de base va permettre un fonctionnement efficace lors de la construction des représentations internes. Elle autorise également l'utilisation de descriptions partielles, là où elles sont nécessaires.

La superposition, tout en étant différente de l'unification, présente des idées voisines : il s'agit de construire une structure de traits (nommée également description fonctionnelle) qui synthétise deux autres structures de traits, en singularisant éventuellement certains aspects. On vérifiera d'abord que les deux structures de traits à superposer ne présentent pas de contradictions (présence de deux attributs identiques avec des valeurs incompatibles, comme par exemple $[Nombre = s]$ et $[Nombre = p]$). On construit ensuite la structure de traits, somme de tous les attributs présents dans les structures de traits initiales, en donnant les valeurs les plus spécifiques aux attributs communs.

Soient les structures de traits suivantes :

$$\begin{array}{ll}(\textbf{CAT} & \text{VEC} \\ \ \textbf{RACINE} & \text{travaille})\end{array} \qquad\qquad \begin{array}{ll}(\textbf{CAT} & VEC \\ \ \textbf{VFORM} & \text{présent}).\end{array} \tag{4.5}$$

Pour superposer ces deux structures, nous constatons la présence d'un attribut commun, la catégorie, pour laquelle les deux valeurs sont identiques. Les autres attributs étant distincts, ils sont ajoutés tels quels au résultat, ce qui produira la structure de traits qui suit :

$$\begin{array}{ll}(\textbf{CAT} & VEC \\ \ \textbf{RACINE} & \text{travaille} \\ \ \textbf{VFORM} & \text{présent}).\end{array} \tag{4.6}$$

Nous construisons ainsi une description plus complète, à partir de descriptions partielles du concept “travaille” et de la catégorie lexicale verbe au présent.

Ce mécanisme permet également de résoudre des ambiguïtés lexicales. Soit le mot “temps” dont la structure de traits est définie comme suit :

$$NOM1:\ \ (NOM,\ \textbf{RACINE}\ \text{temps},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ \{s{,}p\}).$$

La superposition avec une structure contenant le seul attribut [**NOMBRE** p] donnera la seule solution :

$$NOM1:\ \ (NOM,\ \textbf{RACINE}\ \text{temps},\ \textbf{GENRE}\ m,\ \textbf{NOMBRE}\ p).$$

En revanche, les structures :

$$\begin{pmatrix} \textbf{CAT} & \text{VEC} \\ \textbf{NBP} & \text{3s} \end{pmatrix} \qquad \begin{pmatrix} \textbf{CAT} & VEC \\ \textbf{NBP} & 3p \end{pmatrix} \tag{4.7}$$

ne peuvent pas se superposer. En effet, il n'existe aucune structure de traits qui les synthétise, parce que les valeurs respectives de l'attribut **NBP** sont contradictoires ($3s$ et $3p$).

On décrit ainsi un langage qui permet :

– d'expliciter les règles de grammaire,
– de représenter les éléments du dictionnaire,
– de représenter les structures internes des phrases.

Ce langage est considéré équivalent à une grammaire formelle à contexte libre [170]. Nous allons donc présenter quelques ajouts qui permettent de le rendre plus puissant, pour une meilleure modélisation de la restriction d'accord dans le langage naturel.

Contraintes

Une structure de traits peut être enrichie par des contraintes qui, elles, précisent des exigences comme l'accord (le nom et l'adjectif s'accordent en genre et en nombre). Nous en citons quelques unes à titre d'exemple :

$$\begin{array}{l} EstNonVide\_\cap(vt_1,vt_2,\dots) \\ EGAL(vt_1,vt_2,\dots) \\ PRESENT(vt_1) \\ SUPERPOSABLES(vt_1,vt_2) \\ \dots \end{array}$$

qui testent respectivement si l'intersection entre deux (ou plusieurs) valeurs de traits est non nul, si les valeurs correspondant à plusieurs traits sont identiques, si un trait donné existe dans une structure de traits ou si les valeurs de deux traits peuvent se superposer. Ainsi, si l'on trouve dans une structure de traits un attribut tel que :

$$EstNonVide\_\cap(\ < DET, GENRE > , \quad < NOM, GENRE >)$$

on cherche les deux structures de traits qui décrivent le genre du déterminant et le genre du nom ; ensuite on vérifie si l'intersection entre les valeurs de l'attribut DET et NOM est non nulle.

Ainsi, une règle de réécriture ne peut être appliquée que si l'ensemble des contraintes qui lui est associé est vérifié. La structure de traits, résultat de l'application de cette règle, est alors la superposition de l'ensemble des structures qui la compose.

Soit la règle de réécriture suivante :

$$P \rightarrow GN \quad GV$$

où la personne et le nombre du constituant groupe nominal (GN) et du constituant groupe verbal (GV) ne sont pas contradictoires (intersection non nulle). En utilisant la théorie des grammaires d'unification, cette règle sera définie comme suit :

$$P \rightarrow GN\ GV \qquad EstNonVide\_\cap(< GN, NBP > , \ < GV, NBP >) \tag{4.8}$$

L'interprétation de cette règle se fait de la manière suivante : le constituant P ne peut être construit à partir de GN et GV que si l'intersection entre les valeurs des attributs NBP des constituants GN et GV est non vide. La structure de traits de la phrase (P) est alors la superposition des deux structures GN et GV. Par conséquent, la forme du verbe $VFORM$ de la phrase P sera celle du groupe verbal GV. En revanche, la valeur de l'attribut NBP (la personne et le nombre) de la phrase (P) sera le résultat de l'intersection entre les valeurs des attributs NBP des constituants GN et GV. Ainsi, la règle de réécriture 4.8 peut être écrite comme suit :

$$\begin{array}{ll} P \rightarrow GN\ GV & EstNonVide\_\cap(<GN, NBP>,\ <GV, NBP>) \\ & NBP := \cap(<GN, NBP>,\ <GV, NBP>) \\ & VFORM := <GV, VFORM> \end{array} \tag{4.9}$$

Nous avons alors tout ce qu'il faut pour décrire une grammaire de taille raisonnable, qui tient mieux compte de la restriction d'accord. Nous avons élaboré une grammaire d'unification, contenant un ensemble de règles grammaticales, qui se fonde sur les classes syntaxiques du français présentées dans le paragraphe 4.3 et qui intègre les contraintes d'accord (voir l'annexe B). Nous expliquons, dans ce qui suit, comment nous avons procédé pour l'implantation de notre grammaire.

4.5.5 Les réseaux de transition augmentés

Le formalisme des réseaux de transition augmentés correspond à la notion d'automate à pile et est, de ce fait, assez simple à mettre en oeuvre sur le plan informatique [5]. Nous les avons donc choisis pour représenter notre grammaire d'unification.

Un réseau de transition se compose d'un ensemble de noeuds et d'arcs étiquetés reliant ces noeuds. Ces noeuds sont répartis en deux sous-ensembles : les noeuds terminaux et les noeuds non terminaux. Un de ces noeuds est spécifié comme l'état initial du réseau et chacun de ces arcs est étiqueté par un symbole s_i de V. Ces réseaux possèdent également une notion de récursion, qui leur permet d'avoir la puissance descriptive des grammaires à contexte libre. En effet, chaque arc du réseau peut se rapporter à des symboles non terminaux (autres réseaux de transition) aussi bien qu'à des symboles terminaux (des mots du dictionnaire). Ainsi, une grammaire à contexte libre peut être représentée par un ensemble de réseaux de transition. Chacun de ces réseaux correspond à un symbole non terminal de la grammaire, qui définit son nom. Les règles de réécriture, commençant par le symbole d'origine P, seront représentées par un réseau de transition particulier, nommé réseau de départ.

Soient les réseaux de transition de la figure 4.3, qui reconnaissent les mêmes expressions que la grammaire définie dans la table 4.1.

Le réseau de départ P correspond à la première règle de réécriture. Le réseau numéro 2 a GN comme état de départ et représente la deuxième et la troisième règle de réécriture. En revanche, les réseaux 3,4,... 7 se rapportent respectivement aux règles de réécriture 4,5,... 8. Ainsi, une phrase est considérée valide, par l'ensemble de ces réseaux de transition et par conséquent par la grammaire, si et seulement si il existe un chemin dans ce même réseau qui parcourt toute la phrase. Ce chemin commence au noeud initial et traverse les arcs pour arriver à un noeud terminal du réseau de départ. Dans le cas où l'arc est étiqueté par un symbole terminal (par exemple, l'arc étiqueté par le mot "il" entre les noeuds PPE et $PPE1$ dans le réseau numéro 4), il ne peut être traversé que si le mot en cours d'analyse lui correspond. En revanche, un arc étiqueté par un symbole non terminal (par exemple, l'arc étiqueté par la catégorie syntaxique GN entre les noeuds P et $P1$) ne peut être traversé que si le réseau correspondant est traversé avec succès jusqu'à un noeud terminal. Dans ce qui suit, pour des raisons simplificatrices, nous considérons les classes syntaxiques comme des symboles terminaux de la grammaire (les étiquettes des arcs représentant ces classes seront donc transformées en minuscules). Ainsi, un arc étiqueté par une

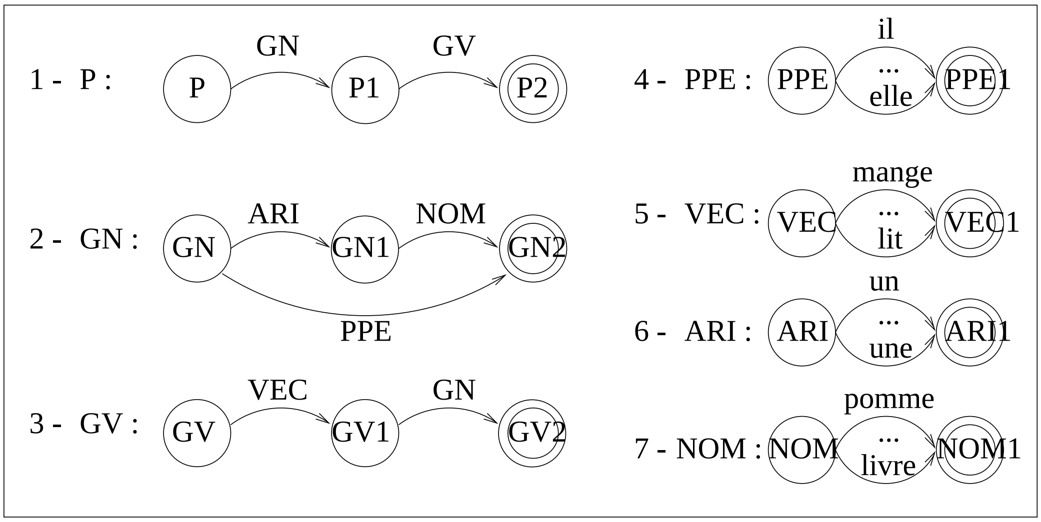


FIG. 4.3 – *Exemple de réseaux de transition.*

classe syntaxique ne peut être traversé que si le mot en cours d'analyse appartient à cette même classe syntaxique. Nous présentons alors sur la figure 4.4 l'équivalent des réseaux de transition présentés sur la figure 4.3.

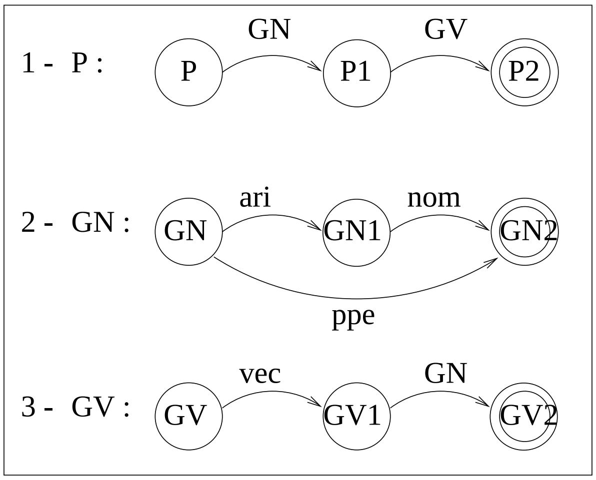
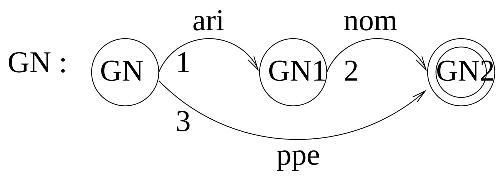


FIG. 4.4 – *Exemple de réseaux de transition simplifié.*

Les réseaux de transition augmentés ont la particularité d'intégrer des informations supplémentaires, sous forme de registres, leur permettant de représenter les traits des grammaires d'unification. Ainsi, à chaque fois qu'un réseau est à parcourir, un nouvel ensemble de registres est créé. Ces registres sont mis à jour par des actions, associées à chacun des arcs du réseau. Une fois le réseau traversé jusqu'à un noeud terminal, les registres se regroupent pour former une structure de traits dont la valeur de l'attribut catégorie (CAT) est le nom du réseau. Cette structure de traits, superposition de l'ensemble des structures de traits des constituants du réseau (cf. §4.5.4), est obtenue au fur et à mesure de l'application des actions affectées au réseau.

Nous présentons sur la figure 4.5 un exemple de réseau de transition augmenté. Chaque fois qu'un symbole terminal est traversé (par exemple, l'arc numéro 1 étiqueté par "ari" sur la figure 4.5), le constituant construit à partir du mot en cours d'analyse est inséré dans un registre spécial, nommé $*$ (première action du réseau présenté sur la figure 4.5). La deuxième action de cet arc, $NBP =< *, NBP >$, a pour but d'affecter au registre NBP du réseau la valeur du trait NBP du mot en cours d'analyse (le constituant qui est dans le registre $*$). La vérification d'accord est spécifiée dans la partie *contraintes*. Ainsi, dans le cas où la contrainte est non vérifiée, l'arc ne pourra pas être traversé. L'arc étiqueté par "nom" ne peut donc être traversé que si l'intersection, entre la valeur du registre NBP du réseau et la valeur du trait NBP du mot en cours d'analyse (le constituant "nom" dans *), est non nulle. Une fois que l'arc étiqueté par "nom" est traversé, la valeur du registre NBP du réseau est la valeur de l'intersection ($NBP := \cap : (< ari, NBP > \;,\; < *, NBP >)$). En arrivant sur un noeud terminal ($GN2$), après l'application des actions, la structure de traits du réseau est ainsi construite. La valeur de l'attribut $RACINE$ de cette structure de traits est le constituant élaboré à partir du réseau en cours (GN). Si un arc étiqueté par un symbole terminal est traversé, le mot en cours d'analyse devient le mot qui suit dans la phrase à traiter.

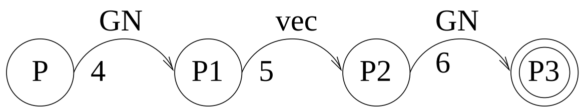


Arc	Contraintes	Actions
1	sans contraintes	$RACINE := *$ $NBP :=< *, NBP >$
2	$EstNonVide\_\cap (< ari, NBP > \,,\ < *, NBP >)$	$RACINE := *$ $NBP := \cap(< ari, NBP > \,,\ < *, NBP >)$
3	sans contraintes	$RACINE := *$ $NBP :=< *, NBP >$

FIG. 4.5 – *Exemple de réseau de transition augmenté.*

Dans le cas où l'arc est étiqueté par un symbole non terminal, les traits sont également traités de la même manière. En effet, la structure de traits, construite par la traversée d'un réseau correspondant à un symbole non terminal, est affecté à l'ensemble des registres associé à l'arc étiqueté par ce même symbole. Ainsi, le constituant construit par cette traversée est affecté au registre *. Dans la grammaire présentée sur la figure 4.6, l'action associée à l'arc reliant P et $P1$ ($RACINE := *$) a pour objectif d'affecter au registre $RACINE$, le constituant retourné par le réseau GN. La contrainte affectée à l'arc étiqueté par "vec" est vérifiée si et seulement si l'intersection, entre la valeur du registre NBP résultat de la traversée de l'arc étiqueté par "GN" et la valeur du trait NBP du mot en court d'analyse (le verbe conjugué), est non nulle. Ce test renforce l'accord entre le sujet et le verbe.

P :



Arc	Contraintes	Actions
4	sans contraintes	$RACINE := *$
5	$EstNonVide\_ \cap (< GN, NBP >,$ $< *, NBP >)$	$RACINE := *$ $NBP := \cap(< GN, NBP >,$ $< *, NBP >)$
6	sans contraintes	$RACINE := *$

FIG. 4.6 – *Exemple de réseau de transition augmenté.*

4.5.6 Technique d'analyse

La technique d'analyse que nous avons utilisée pour vérifier la structure et la restriction d'accord d'une phrase, correspond à la notion d'automate à pile. Elle est, de ce fait, assez simple à mettre en oeuvre sur le plan informatique.

Les analyseurs utilisant les automates à pile ont besoin de connaître leur état à chaque étape de l'analyse. Ainsi, à tout moment, l'analyseur doit avoir les informations suivantes, définissant son état :

- *la position courante* : c'est un registre contenant le mot en cours de traitement dans la phrase.
- *le noeud courant* : le noeud dans lequel se trouve l'analyseur dans le réseau.
- *les points de retour* : une pile de noeuds dans d'autre réseau où nous effectuerons le traitement une fois que nous sortirons du réseau en cours, après l'avoir parcouru jusqu'à un noeud terminal.
- *la liste de structures de traits*, nommé LST : c'est une liste qui contient l'ensemble des structures de traits des constituants déjà analysés.

Nous sommes donc maintenant capable de définir l'algorithme d'analyse de phrases, utilisé par notre modèle de langage. Pour des raisons de clarté, nous allons procéder en deux étapes : l'algorithme $A1$ ensuite l'algorithme $A2$.

L'algorithme $A1$

Supposons que nous sommes au milieu de l'analyse et que nous connaissons *la position courante*, *le noeud courant*, *les points de retour* et la liste LST. Pour traverser un arc, cet algorithme procède comme suit :

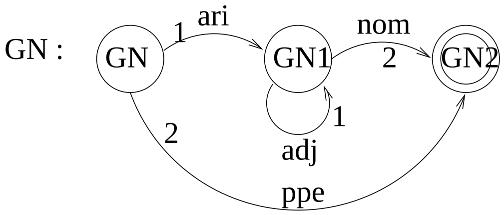
1. **Si** l'arc est étiqueté par un symbole terminal (une classe syntaxique), si le mot courant (*la position courante*) correspond à ce symbole (appartient à cette classe) et si les contraintes affectées à cet arc sont vérifiées,
   alors
   (a) appliquer l'ensemble des actions associé à l'arc, ce qui construit la structure de traits du constituant élaboré à partir du mot en cours d'analyse ;
   (b) insérer cette structure de traits dans la liste LST ;
   (c) mettre à jour *la position courante* avec le prochain mot ;

   (d) mettre à jour *le noeud courant* avec le noeud destination de l'arc.
2. **Sinon Si** l'arc est étiqueté par un autre réseau K,
   alors
   (a) ajouter le noeud destination de l'arc aux *points de retour*;
   (b) mettre à jour *le noeud courant* avec le noeud de départ du réseau K.
3. **Sinon Si** *le noeud courant* est un noeud terminal et si la liste des *points de retour* est non vide,
   alors
   (a) remplacer, dans LST, les structures de traits des constituants du réseau courant, par la structure de traits du constituant élaboré par sa traversée ;
   (b) supprimer le premier *point de retour* et l'affecter au *noeud courant*.
4. **Sinon Si** *le noeud courant* est un noeud terminal, la liste des *points de retour* est vide et il n'existe plus de mot à traiter,
   alors
   (a) l'analyse est terminé avec succès.
5. **Sinon** l'arc ne pourra pas être traversé : on se trouve sur un mauvais chemin.

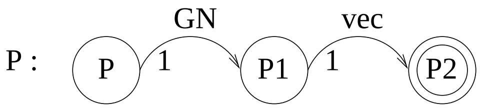
Exemple

Soient les réseaux de transition augmentés de la figure 4.7, présentant un exemple très simplifié d'un fragment de la langue française ; le terme $\otimes$, utilisé dans cette figure, est pour représenter le nombre de fois que l'arc $GN1/1$ est traversé (si $\otimes = 0$, l'ensemble $\{< adj, NBP >\}^{\otimes}$ devient vide). Pour des raisons de clarté, pour mieux expliquer le fonctionnement de l'algorithme, nous nous limitons à la vérification de la personne et du nombre (l'attribut NBP). Il suffit par conséquent de procéder de la même manière pour la vérification du genre et de la forme du verbe. Les numéros mentionnés sur les arcs ont pour but de définir l'ordre de leur traitement dans le cas où il existe une ambiguïté de choix (dans le cas où plusieurs arcs débutent en un même noeud, l'analyseur les traite suivant l'ordre croissant des numéros). Dans cet exemple, nous identifions chaque arc par son noeud de départ et par un numéro. Ainsi, l'arc $P/1$ est l'arc numéro 1 dont le noeud de départ est P.

Nous montrons dans la table 4.3 que les réseaux de la figure 4.7 valident la phrase : "*un grand garçon mange*". En effet, nous trouvons dans cette table la trace des états (noeud courant, position courante et points de retour) générés par l'algorithme $A1$ présenté ci-dessus. En commençant par le noeud P, le seul arc possible à traverser est l'arc GN. Par conséquent, comme il est mentionné dans la deuxième étape de l'algorithme $A1$, le nouvel état de l'analyseur est défini par la mise à jour du *noeud courant* avec GN et par l'insertion du noeud $P1$ dans la liste des *points de retour*. Ensuite, à partir du noeud GN, l'analyseur va essayer de traverser l'arc $GN/1$. Pour ce faire, il doit tout d'abord contrôler, comme il est spécifié dans la première étape de l'algorithme $A1$, si le mot courant (*la position courante*) appartient à la classe "*ari*" (article indéfini) et si les contraintes (dans le cas où elles existent) sont vérifiées. Une fois le contrôle effectué avec succès, l'arc est traversé, la *position courante* est mise à jour et la structure de traits du constituant élaboré à partir du mot "un" est insérée dans la liste LST (troisième étape de la table 4.3). L'analyseur continue de cette façon jusqu'à la cinquième étape (voir la table 4.3), où un noeud terminal est atteint, imposant au *noeud courant* d'être initialisé à $P1$ (l'arc GN est parcouru avec succès). Les structures de traits des constituants (les constituants construits à partir des mots "*un*", "*grand*" et "*garçon*") du réseau GN sont ainsi remplacées par la structure de traits du constituant "*un grand garçon*" élaborée par sa traversée. L'analyse se termine avec succès, une



Arc	**Contraintes**	**Actions**
$GN/1$	sans contraintes	$RACINE := *$ $NBP := < *, NBP >$
$GN1/1$	$EstNonVide\_ \cap (< ari, NBP >, \{< adj, NBP >\}^{\otimes}, < *, NBP >)$	$RACINE := *$ $NBP := \cap(< ari, NBP >, \{< adj, NBP >\}^{\otimes}, < *, NBP >)$
$GN1/2$	$EstNonVide\_ \cap (< ari, NBP >, \{< adj, NBP >\}^{\otimes}, < *, NBP >)$	$RACINE := *$ $NBP := \cap(< ari, NBP >, \{< adj, NBP >\}^{\otimes}, < *, NBP >)$
$GN/2$	sans contraintes	$RACINE := *$ $NBP := < *, NBP >$



Arc	**Contraintes**	**Actions**
$P/1$	sans contraintes	$RACINE := *$
$P1/1$	$EstNonVide\_ \cap (< GN, NBP >, < *, NBP >)$	$RACINE := *$ $NBP := \cap(< GN, NBP >, < *, NBP >)$

FIG. 4.7 – *Exemple de réseaux de transition augmentés.*

fois que le verbe conjugué "mange" est rencontré et lorsque les contraintes de l'arc $P1/1$ sont vérifiées (sixième étape), ramenant ainsi l'analyseur à un noeud terminal du réseau P (septième étape).

étape	état courant	arc à traverser	à insérer dans LST	commentaires
1.	(P,1,NIL)	$P/1$	NIL	position initiale.
2.	(GN,1,($P1$))	$GN/1$	NIL	analyse avec le réseau GN suivi d'un retour à $P1$.
3.	($GN1$,2,($P1$))	$GN1/1$	(ARD, **RACINE** un, **NBP** 3s)	traversé l'arc $GN/1$ ("*un*").
4.	($GN1$,3,($P1$))	$GN1/2$	(ADJ, **RACINE** grand, **NBP** 3s)	traversé l'arc $GN1/1$ ("*grand*").
5.	($GN2$,4,($P1$))	$GN2/2$	(NOM, **RACINE** garçon, **NBP** 3s)	traversé l'arc $GN1/2$ ("*garçon*"), $GN1/1$ est non vérifié.
6.	($P1$,4,NIL)	$P1/1$	(GN, **RACINE** un grand garçon, **NBP** 3s)	le réseau GN est analysé, nous retournons donc à $P1$.
7.	($P2$,5,NIL)	$P2/1$	(VEC, **RACINE** mange, **NBP** 3s)	traverser l'arc $P2/1$ ("mange").
8.			(P, **RACINE** un grand garçon mange, **NBP** 3s)	$P2$ est un noeud terminal. L'analyse se termine avec succès.

TAB. 4.3 – *Trace d'analyse de la phrase "un grand garçon mange"*

L'algorithme $A2$

L'analyse de la phrase "*un grand garçon mange*" est réussie parce que le premier arc traversé avec succès à chaque étape était finalement le bon. Cependant, avec une phrase comme "*un sportif travaille*", où le mot "*sportif*" peut être un adjectif ("adj") ou un nom ("nom"), l'algorithme d'analyse $A1$ échoue. En effet, $A1$ classera le mot "sportif" comme un "adj" (adjectif), et donc il ne trouvera pas de nom à la suite. Pour remédier à cette contrainte, nous devons sauvegarder tous les noeuds d'appuis durant l'analyse (les autres choix que pourrait prendre l'analyseur). Par conséquent, une fois que l'analyseur échoue, celui-ci effectue un retour en arrière pour essayer un autre choix. Nous présentons dans ce qui suit une extension apportée à l'algorithme $A1$, donnant l'algorithme $A2$, lui permettant de prendre en compte cette contrainte supplémentaire.

Soit une liste, dite *de possibilités*, initialisée avec le noeud initial du réseau de départ (P). Cette liste contient l'ensemble des noeuds d'appui sur lequel l'analyseur pourra revenir en cas d'échec. L'algorithme $A2$, qui est à la base de notre analyseur, utilise la *liste de possibilités* et procède comme suit :

1. sélectionner un noeud $N1$ de la *liste de possibilités*,
2. générer des nouveaux noeuds qui correspondent aux arcs qui peuvent être traversés à partir du noeud $N1$. Si l'arc à traverser est étiqueté par un symbole non terminal, le noeud généré est le noeud initial du réseau de ce symbole. Sinon, le noeud généré est le noeud d'arrivée de l'arc. Pour traverser un arc, c'est l'algorithme $A1$ qui est utilisé. Il peut y avoir aucun noeud généré, si on se trouve sur un mauvais chemin (arc) : étape numéro 5 de $A1$.
3. remplacer le noeud $N1$, dans la *liste de possibilités*, par les noeuds qui viennent d'être générés. Si aucun noeud n'a été généré, $N1$ sera supprimé de la *liste de possibilités* et les structures de traits des constituants élaborés depuis $N1$ seront supprimées de LST.
4. les étapes 1, 2 et 3 seront répétées tant que l'on a pas trouvé de chemin, parcourant tous les mots de la phrase et ramenant à un noeud terminal d'analyse. Si la *liste de possibilités* est vide et si aucun chemin n'est trouvé, la phrase est supposée non valide.

L'ordre selon lequel les noeuds de la liste sont sélectionnés permet de définir la méthode d'analyse : en profondeur ou en largeur. En effet, si la *liste de possibilités* est du genre dernier entré premier sorti, l'analyseur procède en profondeur d'abord (ce que nous utilisons). En revanche, si la *liste de possibilités* est du genre premier entré premier sorti, l'analyseur procède en largeur [5].

Exemple

Pour illustrer l'algorithme $A2$, nous allons l'appliquer sur la phrase écrite "*le sportif travaille*", en utilisant les réseaux de transition augmentés de la figure 4.7. Nous présentons dans la table 4.4 la trace des états générés par cet algorithme.

étape	état courant	arc à traverser	liste de possibilités	à insérer dans LST
1.	(P,1,NIL)	$P/1$	NIL	NIL
2.	(GN,1,($P1$))	$GN/1$	NIL	(ARI, **RACINE** un, **NBP**3s)
3.	($GN1$,2,($P1$))	$GN1/1$	($GN1$,2,($P1$))	(ADJ, **RACINE** sportif, **NBP**3s)
4.	($GN1$,3,($P1$))	NIL	(GN1, 2, (P1))	
5.	($GN1$,2,($P1$))	$GN1/2$	NILL	(NOM, **RACINE** sportif, **NBP**3s)
6.	($P1$,2,NIL)	$P1/1$	NILL	(GN, **RACINE** un sportif, **NBP**3s)
7.	($P2$,3,NIL)	$P2/1$	NILL	(VEC, **RACINE** travaille, **NBP**3s)
8.		(noeud terminal)		

TAB. 4.4 – *Trace d'analyse de la phrase "un sportif travaille" avec l'algorithme A2.*

Durant cette analyse, nous traitons les arcs suivant l'ordre croissant des numéros qui leurs sont affectés. Nous commençons donc par P où le seul arc possible à traverser est GN. Le nouvel état de l'analyseur est ainsi défini par la mise à jour du *noeud courant* avec GN et par l'insertion du noeud $P1$ dans la pile des *points de retour*. Ensuite, à partir du noeud GN, étant donné que le mot courant (*la position courante*) appartient à la classe "ari" (article indéfini), seul l'arc $GN/1$ peut être traversé. L'analyseur traverse alors cet arc, insère la structure de traits du constituant élaboré à partir du mot courant dans LST et met à jour la *position courante* avec le mot qui suit (deuxième étape). Le mot courant "*sportif*" peut être soit un "adj" (adjectif) soit un "nom" (nom). Ainsi, suivant la grammaire de la figure 4.7, l'analyseur pourra traverser deux arcs ($GN1/1$ et $GN1/2$). Or, étant donné que le premier arc qui devrait être analysé à partir de $GN1$ est $GN1/1$, l'arc "adj" sera traversé. La structure de traits du constituant élaboré à partir du mot "*sportif*" est insérée dans LST et l'état courant (*position courante, noeud courant et points de retour*) est inséré dans la *liste de possibilités* (troisième étape). Dans ce cas, après mise à jour de la *position courante*, l'analyseur s'attend à rencontrer un nom ou un autre adjectif. Ceci n'est pas le cas du mot "*travaille*" qui est un verbe conjugué. Ainsi, la structure de traits du constituant élaboré à partir du mot "*sportif*" est supprimée de LST et l'analyseur recule à un noeud de la *liste de possibilités*, sélectionné suivant une certaine stratégie (stratégie du dernier entré premier sorti pour nous). L'analyseur revient donc au noeud $GN1$ où il traverse cette fois-ci l'arc "nom", après la vérification des contraintes, pour arriver sur un noeud terminal du réseau GN. Dans ce cas il revient à l'état $P1$ après avoir traversé l'arc $P1/1$. Enfin, étant donné que les contraintes d'accord sont vérifiées entre le sujet (structure de traits du constituant élaboré à partir de la traversée du réseau GN) et le verbe conjugué "*travaille*", l'arc $P1/1$ est traversé pour arriver sur un noeud terminal d'analyse ($P2$). Ainsi, la phrase "*un garçon travaille*" est considérée comme valide.

4.5.7 Fonctionnement du modèle formel

Nous venons de définir les techniques et les formalismes qui sont à la base de notre modèle de langage formel. L'objectif de ce modèle est de filtrer les phrases qui ne vérifient pas la restriction d'accord. Nous avons ainsi construit une grammaire d'unification qui se fonde sur des règles grammaticales du français et sur des classes syntaxiques. Nous avons représenté cette grammaire par des réseaux de transition augmentés et nous avons utilisé l'algorithme $A2$ pour l'analyse.

L'objectif de notre modèle est de vérifier les phénomènes d'accord des mots d'une phrase ayant des structures simples [52, 72, 98]. Nous espérons par conséquent, étant donnée la fréquence des règles grammaticales de ces structures, analyser un maximum de phrases. Dans le cas où la phrase ne correspond à aucune des règles grammaticales de notre grammaire, les phénomènes d'accord ne seront pas vérifiés [203]. Nous garderons ainsi le résultat fourni par le modèle probabiliste.

4.6 Le corpus

Un même jeu de corpus a été utilisé pour tester tous les modèles de langage présentés dans ce document. Le corpus que nous employons contient environ 50 millions de mots, extraits du journal "*L'Est Républicain*" et du journal "*Le Monde*" des années $1987-1988$. Le corpus d'apprentissage regroupe 88% de la taille totale, le corpus de développement contient 3% et le corpus de test 9%. Le corpus de développement sert surtout pour l'estimation des paramètres d'interpolation.

4.7 Évaluation

Pour évaluer les performances du modèle formel, nous l'avons intégré dans notre système de dictée vocale, MAUD ; le vocabulaire utilisé contient environ 20000 mots. Nous avons utilisé le modèle probabiliste pour fournir les N meilleures hypothèses, correspondant à la phrase prononcée. Chacune de ces hypothèses possède un score de vraisemblance. Ensuite, nous avons utilisé le modèle formel pour filtrer ces hypothèses. Nous supprimons ainsi toutes celles qui ne vérifient pas la restriction d'accord selon nos règles. Le résultat du système est donc l'hypothèse restante qui possède le meilleur score [184]. Si le modèle formel supprime toutes les hypothèses, le résultat du système sera la meilleure hypothèse fournie par le modèle probabiliste. Avant le filtrage, une étape préliminaire d'étiquetage est lancée. Elle consiste à attribuer, à chaque mot de la liste des N meilleures hypothèses, sa classe syntaxique en fonction de son contexte. Pour ce faire, nous utilisons la procédure d'étiquetage présentée dans le paragraphe 4.4.

Les tests ont été lancés sur les 300 phrases de la campagne AUPELF-UREF[13] pour l'évaluation des systèmes de RAP traitant du français [4]. Pour chacune de ces phrases, les niveaux inférieurs de MAUD fournissent les 80 meilleures hypothèses, en utilisant un modèle de langage probabiliste. Ces hypothèses seront ensuite filtrées par le modèle formel. Le modèle de langage probabiliste utilisé est une combinaison des modèles n-grammes, *cache* et *trigger*. En se servant que des mots comme unités de base dans le vocabulaire (cf. chapitre 6) , c'est cette combinaison qui donne la meilleure perplexité et le meilleur taux d'erreur avec MAUD (cf. chapitre 8). Pour une valeur de n égale à 3 (trigrammes) et en utilisant le corpus présenté dans le paragraphe 4.6, la perplexité est de 72.69.

Nous avons constaté que le modèle formel élimine 18% de l'ensemble des hypothèses. Les hypothèses éliminées sont statistiquement probables, mais ne respectent pas la restriction d'accord. Cette approche a permis d'améliorer notablement le taux de reconnaissance en terme de phrases linguistiquement correctes.

En analysant les résultats, nous avons remarqué que plusieurs hypothèses de premier rang ont été éliminées parce qu'elles contiennent des mots qui ne respectent pas les contraintes d'accord. De plus, d'autres hypothèses linguistiquement correctes, mais dont la forme du verbe, du genre ou du nombre ne correspondent pas à la phrase d'origine, ont été conservées. Nous avons également remarqué quelques erreurs d'étiquetage qui ont influencé la prise de décision du modèle formel.

4.8 Conclusion

Nous venons de présenter une nouvelle approche hybride combinant un modèle probabiliste avec un modèle formel. Le modèle probabiliste est utilisé pour fournir les N meilleures hypothèses, qui correspondent à la phrase prononcée. En revanche, l'objectif du modèle formel est de filtrer ces hypothèses pour ne garder que celles qui vérifient la restriction d'accord. Ce modèle, défini en détail dans ce chapitre, se sert des classes syntaxiques issues de la langue et d'un ensemble de règles grammaticales représenté par une grammaire d'unification.

Pour évaluer le modèle formel, un jeu de tests de 300 phrases prononcées a été utilisé. Pour chacune de ces phrases, 80 hypothèses ont été conservées pour le filtrage. En utilisant le modèle formel, les résultats de filtrage montrent que 18% de ces hypothèses ont été éliminées. Les hypothèses éliminées sont toutes linguistiquement invalides. L'utilisation d'une telle approche a amélioré notablement le taux de reconnaissance en terme de phrases linguistiquement correctes.

13. Agence francophone pour l'enseignement supérieur et la recherche.

En analysant les résultats, nous notons qu'un nombre considérable d'hypothèses linguistiquement correctes ont été gardées, bien qu'elles ne correspondent pas à la phrase d'origine ; les mots constituant ces phrases sont des homonymes des mots de la phrase d'origine. Ajoutons à ceci le fait que plusieurs hypothèses de premier rang ont été supprimées bien qu'elles ne contiennent qu'un ou deux mots, qui ne respectent pas la restriction d'accord. Sans oublier les erreurs d'étiquetage qui parfois ont influencé le comportement du système.

En regardant l'ensemble des hypothèses après filtrage, nous avons remarqué que la meilleure ne se trouve pas souvent en premier. Ainsi, pour mieux évaluer la vraisemblance de ces hypothèses, nous présentons dans le chapitre qui suit deux nouveaux modèles de langage : multiclasses et hiérarchique. Ces modèles utilisent les classes syntaxiques définies dans le paragraphe 4.3 et agissent d'une manière probabiliste sur les structures linguistiques d'une phrase.

Chapitre 5

Modèles de langage multiclasses et hiérarchique

5.1 Introduction

Dans le langage, une phrase est soumise à différentes contraintes : des contraintes lexicales liées à la limitation du vocabulaire et des contraintes syntaxiques et sémantiques régissant l'ordre des mots. Dès lors, toutes les combinaisons possibles de mots ne sont pas observées, loin s'en faut. Certaines suites de mots forment un groupe homogène, pour des raisons philologiques, syntaxiques ou sémantiques. Ces séquences, naturellement de longueur variable, véhiculent souvent soit une idée soit une structure langagière particulière.

Le modèle multigrammes, tel qu'il est défini par S. Deligne [46], tient compte de ces séquences. En effet, la particularité de ce modèle provient de sa capacité à extraire des séquences clés, supposées être capables de traduire la structure d'une phrase. Néanmoins, l'ensemble des expérimentations effectuées sur ce modèle ont montré que dans le cas d'une utilisation de grands vocabulaires, l'apprentissage des paramètres nécessite des corpus énormes et des machines très puissantes.

Nous exposons dans ce chapitre deux modèles probabilistes différents inspirés du modèle multigrammes : le modèle multiclasses et le modèle hiérarchique. L'originalité du modèle multiclasses est sa capacité à pouvoir utiliser de grands vocabulaires sans avoir besoin de machines puissantes ni d'énormes corpus. Le modèle hiérarchique ajoute une meilleure modélisation de la dépendance entre les séquences clés d'une phrase.

Nous présentons dans ce qui suit le principe de base du modèle multigrammes. Dans le paragraphe 5.3, nous abordons le modèle multiclasses : nous définissons son fonctionnement et la manière de l'estimer. Ensuite, nous exposons le modèle hiérarchique dans le paragraphe 5.4. Enfin, nous terminons par une évaluation.

5.2 Le modèle multigrammes

En utilisant le modèle multigrammes, la langue est considérée comme une source qui émet des séquences d'observation. Les observations d'une séquence ne sont pas indépendantes et leur dépendance est de longueur variable. Dans la langue parlée, l'observation est une entité qui peut représenter le phonème, la syllabe, le mot, etc. Dans cette thèse, étant donné que nous nous basons sur les mots pour la modélisation du langage, nous allons nous limiter à l'utilisation du terme mot pour désigner une observation. Ainsi, la source gouvernant l'activité d'une langue est supposée émettre des séquences de mots de longueur variable, appelées multigrammes, appartenant à un ensemble fini $M = \{m_i\}$. Toute suite de mots ($W_1^T = w_1, \dots w_T$ ($w_i \in V$)) émise par la langue, est supposée être la concaténation d'un ensemble de séquences appartenant à M. Localiser un ensemble de multigrammes dans une suite de mots W_1^T revient ainsi à définir une segmentation L de W_1^T, comme dans l'exemple suivant :

$$
\begin{array}{llllll}
\text{Suite de séquences } M: & m_{i_1} & m_{i_2} & m_{i_3} & \dots & m_{i_l} \\
 & \Uparrow & \Uparrow & \Uparrow & & \Uparrow \\
\text{Segmentation } L: & [w_1,w_2] & [w_3] & [w_4,w_5,w_6] & \dots & [w_j, \dots w_T] \\
 & & & \Uparrow & & \\
 & & & \Uparrow & & \\
\text{Suite de mots } W_1^T: & \multicolumn{5}{c}{w_1,w_2,w_3,w_4,w_5,w_6, \dots w_j, \dots w_T}
\end{array}
$$

Il s'agit de détecter dans W_1^T des formes récurrentes, se présentant comme des séquences de mots de longueurs variables appartenant à M.

L'utilisation du modèle multigrammes nécessite donc la détermination de l'ensemble optimal des séquences, $M^* = \{m_i^*\}$, représentant les multigrammes émis par la source. Pour ce faire, il suffit de trouver l'ensemble de séquences M^* qui maximise la vraisemblance d'un corpus représentatif du comportement de la langue. En effet, n'importe quel ensemble de séquences, répondant aux exigences de la dépendance entre les mots, est un candidat. Ainsi, pour un ensemble de multigrammes donné $M = \{m_i\}$, la vraisemblance du corpus doit prendre en compte la vraisemblance de l'ensemble $M = \{m_i\}$ lui-même. L'ensemble optimal de multigrammes M^* est donc défini par la maximisation conjointe de la vraisemblance du corpus et de la vraisemblance de l'ensemble $M = \{m_i\}$:

$$M^* = \{m_i^*\} = arg \max_{M=\{m_i\}} \mathcal{P}(W_1^T/M)\mathcal{P}(M) \tag{5.1}$$

où $\mathcal{P}(W_1^T/M)$ représente la vraisemblance du corpus W_1^T sachant un ensemble de multigrammes donné $M = \{m_i\}$, et $\mathcal{P}(M)$ la vraisemblance de l'ensemble de multigrammes $M = \{m_i\}$. En utilisant le modèle n-grammes, l'ensemble M est constitué d'un ensemble prédéfini d'unités linguistiques (par exemple un vocabulaire de mots). Ainsi, la vraisemblance $\mathcal{P}(M)$ est constante. La valeur de la vraisemblance $\mathcal{P}(W_1^T/M)$ évalue l'adéquation du modèle multigrammes aux données observées. Elle se calcule comme la vraisemblance cumulée de toutes les segmentations et combinaisons d'entités multigrammes (L,S) possibles :

$$\mathcal{P}(W_1^T/M) = \sum_{(L,S)} \mathcal{P}(W_1^T,L,S/M) \tag{5.2}$$

où $S = \{s_t\}$ représente l'ensemble des séquences de mots correspondant à la segmentation L de W_1^T. Chaque séquence $s_t \in S$ est étiquetée par un multigramme $m_{i_t} \in M$. Cette même valeur de vraisemblance $\mathcal{P}(W_1^T/M)$ peut être également considérée comme la vraisemblance de la meilleure segmentation [23] :

$$\mathcal{P}(W_1^T/M) = \mathcal{P}^*(W_1^T/M) = \max_{(L,S)} \mathcal{P}(W_1^T,L,S/M). \tag{5.3}$$

La valeur de l'expression $\mathcal{P}(W_1^T,L,S/M)$ repose sur la dépendance entre les séquences dans (L,S). Dans l'approche multigrammes, les séquences sont supposées indépendantes. Par conséquent, la vraisemblance d'une segmentation L du corpus W_1^T avec les séquences S (W_1^T,L,S) est définie comme suit :

$$\mathcal{P}(W_1^T,L,S/M) = \prod_t^{N(L,S)} p(s_t,m_{i_t}) = \prod_t^{N(L,S)} p(s_t/m_{i_t})\ p(m_{i_t}) \tag{5.4}$$

où s_t représente une séquence correspondant à un multigramme m_{i_t} (de rang i_t) dans M, et $N(L,S)$ le nombre de séquences dans L (ce qui correspond également au nombre de multigrammes dans S). Le modèle est donc complètement défini par l'ensemble des paramètres suivants :

- la distribution *a priori* des multigrammes, $\{p(m_i)\}$, où $\sum_{m_i \in M} p(m_i) = 1$
- l'ensemble des distributions $\{p(\ldots/m_i)\}$ caractérisant la dispersion des séquences observées pour chaque multigramme m_i (une séquence peut appartenir à plusieurs multigrammes), nommés paramètres *a posteriori*.

Pour un modèle multigrammes donné, c'est-à-dire pour un ensemble M donné, la segmentation et la combinaison de multigrammes qui sous-tendent la suite W_1^T sont définies comme suit :

$$(L^*,S^*) = arg \max_{L,S} \mathcal{P}(W_1^T,L,S/M). \tag{5.5}$$

Dans la suite, étant donné qu'une valeur de vraisemblance est toujours calculée pour un modèle multigrammes donné (un ensemble $M = \{m_i\}$ donné), le symbole M sera omis. Par conséquent, pour des raisons de simplicité, nous utilisons le terme $\mathcal{P}(W_1^T,L,S)$ pour noter la vraisemblance $\mathcal{P}(W_1^T,L,S/M)$.

5.2.1 Estimation du modèle

L'estimation d'un modèle multigrammes optimal est liée à l'optimisation de l'équation 5.1, recherchée par une procédure itérative de type EM (Expectation-Maximization) [49]. Au cours de cette procédure, la vraisemblance des données et l'entropie du modèle multigrammes sont optimisées en alternance. La procédure débute avec l'initialisation d'un ensemble de multigrammes M^0 et de paramètres correspondants (probabilités). Ensuite, chaque itération comprend deux étapes :

- Modifier l'ensemble des multigrammes M^k par M^{k+1}, pour réduire l'entropie du modèle. Une heuristique, qui consiste à supprimer de M^k tous les multigrammes ayant une très petite probabilité, est appliquée.
- Estimer de nouveau l'ensemble des paramètres restants pour maximiser la vraisemblance des données $\mathcal{P}(W_1^T/M^{k+1})$.

La procédure s'arrête lorsque la vraisemblance des données cesse de croître, ou lorsque la composition de l'ensemble de multigrammes se stabilise. Pour réestimer l'ensemble de tous ces paramètres du modèle multigrammes, une méthode consiste à utiliser le principe de maximum de vraisemblance, comme défini ci-dessous.

Optimisation suivant le principe de maximum de vraisemblance

L'algorithme EM repose sur l'utilisation d'une fonction auxiliaire Q mettant en jeu les valeurs de vraisemblance, calculées avec les paramètres des itérations (k) et $(k+1)$ de la procédure d'estimation du maximum de vraisemblance [49]. En gardant les notations précédentes, cette fonction Q a pour expression :

$$Q(k,k+1) = \sum_{(L,S)} \mathcal{P}^{(k)}(W_1^T,L,S)\ \ log\ \mathcal{P}^{(k+1)}(W_1^T,L,S). \tag{5.6}$$

D'après [49], il est démontré que si :

$$Q(k,k+1) \geq Q(k,k) \tag{5.7}$$

alors,

$$\mathcal{P}^{(k+1)}(W_1^T) \geq \mathcal{P}^{(k)}(W_1^T). \tag{5.8}$$

Ainsi, si à chaque nouvelle itération $(k+1)$, on détermine les paramètres qui maximisent la fonction Q, on augmente du même coup la vraisemblance du corpus W_1^T. Le modèle est ainsi amélioré à chaque itération, dans le sens où il conduit à une augmentation de la vraisemblance des données, jusqu'à ce que l'algorithme converge vers un point critique de la courbe de vraisemblance. Ce point de convergence peut correspondre à un optimum local et non global. Par conséquent, la formule de réestimation de l'ensemble des paramètres de l'itération $(k+1)$ est obtenu en maximisant la fonction auxiliaire $Q(k,k+1)$.

En utilisant l'équation 5.4, le *log* de la vraisemblance de la suite de mots W_1^T est définie par l'équation suivante :

$$log\ \mathcal{P}(W_1^T,L,S) = \sum_{t=1}^{N(L,S)} log\ p(m_{i_t}) + log\ p(s_t/m_{i_t}). \tag{5.9}$$

En regroupant les multigrammes m_{i_t} identiques, on obtient :

$$log\ \mathcal{P}(W_1^T,L,S) = \sum_{t=1}^{N(L,S)} \sum_{i=1}^{z} \delta_{t,i}(log\ p(m_i) + log\ p(s_t/m_i)) \tag{5.10}$$

où z est le nombre de multigrammes distincts et où la variable de Kronecker $\delta_{t,i}$ vaut 1 si le multigramme de rang t dans S est m_i, et 0 sinon.

En notant $N(m_i/L,S)$ le nombre d'occurrences de m_i dans la suite S associée à la segmentation L, on a :

$$log\ \mathcal{P}(W_1^T,L,S) = \sum_{i=1}^{z} N(m_i/L,S)\ log\ p(m_i) + \sum_{i=1}^{z} \sum_{t=1}^{N(L,S)} \delta_{t,i}\ log\ p(s_t/m_i). \tag{5.11}$$

Ainsi, en remplaçant le *log* de la vraisemblance de l'itération $(k+1)$ dans l'équation 5.6 par son expression définie dans 5.11, la fonction Q s'écrit comme la somme de deux fonctions Q_1 et Q_2. la fonction Q_1 dépend de la distribution des multigrammes : $\{p(m_i)\}$. En revanche, la fonction Q_2 dépend de la distribution $\{p(\ldots/m_i)\}$ caractérisant la dispersion des séquences observées pour chaque multigramme m_i. En effet :

$$Q(k,k+1) = Q_1(k,k+1) + Q_2(k,k+1) \tag{5.12}$$

avec

$$Q_1(k,k+1) = \sum_{i=1}^{z} \left(\sum_{L,S} N(m_i/L,S)\ \mathcal{P}^{(k)}(W_1^T,L,S) \right)\ log\ p^{(k+1)}(m_i) \tag{5.13}$$

et

$$Q_2(k,k+1) = \sum_{i=1}^{z} \sum_{L,S} \left(\mathcal{P}^{(k)}(W_1^T,L,S) \sum_{t=1}^{N(L,S)} \delta_{t,i}\ log\ p^{(k+1)}(s_t/m_i) \right). \tag{5.14}$$

Les formules de réestimation des paramètres $\{p(m_i)\}$ et $\{p(\ldots/m_i)\}$ peuvent ainsi être établies indépendamment, en maximisant respectivement la fonction Q_1 et la fonction Q_2.

Distribution a priori des multigrammes

La formule d'estimation des probabilités *a priori* des multigrammes m_i s'obtient en maximisant l'équation 5.13 par rapport à l'ensemble $\{p^{(k+1)}(m_i)\}$, avec la contrainte $\sum_i p^{(k+1)}(m_i) = 1$.

La fonction Q_1 est de la forme :

$$\sum_{i=1}^{z} y_i \;\; log\ x_i \quad \text{avec} \;\; \sum_{i=1}^{z} x_i = 1.$$

La maximisation de cette somme par rapport à l'ensemble des x_i est un problème classique d'optimisation sous contraintes. Ainsi, en utilisant la méthode des multiplicateurs de Lagrange par exemple, on atteint un maximum global au point unique :

$$x_i = \frac{y_i}{\sum_{j=1}^{z} y_j} \quad \text{pour } i = 1 \text{ à } z.$$

Il vient donc :

$$p^{(k+1)}(m_i) = \frac{\sum_{L,S} N(m_i/L,S)\ \mathcal{P}^{(k)}(W_1^T,L,S)}{\sum_{j=1}^{z} \sum_{L,S} N(m_j/L,S)\ \mathcal{P}^{(k)}(W_1^T,L,S)} \tag{5.15}$$

qui, compte-tenu du fait que $\sum_{j=1}^{z} N(m_j/L,S)$ vaut $N(L,S)$, le nombre de séquences dans L se simplifie en :

$$p^{(k+1)}(m_i) = \frac{\sum_{L,S} N(m_i/L,S)\ \mathcal{P}^{(k)}(W_1^T,L,S)}{\sum_{L,S} N(L,S)\ \mathcal{P}^{(k)}(W_1^T,L,S)}. \tag{5.16}$$

La probabilité de tout multigramme m_i est ainsi estimée par le rapport de deux moyennes. En effet, le numérateur représente le nombre moyen d'occurrences de m_i, sur l'ensemble de toutes les segmentations et combinaisons possibles (L,S). De la même manière, le dénominateur indique le nombre de multigrammes figurant en moyenne dans une décomposition du corpus. Ainsi, à chaque itération, en augmentant la vraisemblance du corpus, la performance du modèle s'améliore. Il converge éventuellement vers un point critique qui peut être un optimum local.

Distribution caractérisant la dispersion des observations

La réestimation des distributions qui caractérisent la dispersion des séquences, observées autour des multigrammes sous-jacentes, se déduit de la maximisation de l'équation 5.14 par rapport aux paramètres *a posteriori* de l'itération $(k+1)$. Pour cela, il faut au préalable définir précisément le modèle permettant d'expliciter $p(s_t/m_i)$ en fonction de l'ensemble des paramètres à estimer. Ensuite, le calcul est mené d'une manière similaire au précédent, en annulant la dérivée de Q_2 par rapport à ces paramètres. On obtient de la sorte une formule de réestimation qui prend en compte la contribution de toutes les segmentations et combinaisons de multigrammes L et S possibles.

Comme nous ne nous intéressons qu'au cas où le multigramme est une séquence distincte de mots, chaque séquence s_t ne peut être étiquetée que par un seul multigramme m_i. Nous nous trouvons ainsi dans un cas où la distribution *a posteriori* des séquences observées pour chaque multigramme m_i est une fonction de Dirac : $p(s_t/m_i) = 1$ si $s_t = m_i$, et 0 sinon.

Par conséquent, pour une segmentation L donnée, il n'existe qu'une seule combinaison de vraisemblance non nulle. Il s'agit de la combinaison S_L telle que pour toute séquence de L : $p(s_t/m_{i_t}) = 1$. Pour toute suite S différente de S_L :

$$\mathcal{P}(W_1^T,L,S) = 0$$

tandis que,

$$\mathcal{P}(W_1^T,L,S_L) = \mathcal{P}(W_1^T,L).$$

Dès lors, la formule de réestimation 5.16 se simplifie comme suit :

$$p^{(k+1)}(m_i) = \frac{\sum_L N(m_i/L,S_L)\ \mathcal{P}^{(k)}(W_1^T,L,S_L)}{\sum_L N(L,S_L)\ \mathcal{P}^{(k)}(W_1^T,L,S_L)}. \tag{5.17}$$

L'estimation de cette équation peut se faire avec un algorithme de type *forward-backward* ou de type Viterbi. Nous présentons dans le paragraphe 5.3 une solution permettant l'estimation de l'équation 5.17.

5.2.2 Conclusion sur l'approche multigrammes

Le modèle multigrammes part de l'hypothèse qu'une phrase est décomposée en entités sous-jacentes appartenant à un ensemble fini M^*. Ces entités sont des séquences de mots de longueur variable, indépendantes entre elles. Le modèle extrait donc l'ensemble fini M^* des entités et attribue ensuite une valeur de vraisemblance pour chacune d'entre elles. Ainsi, pour estimer la vraisemblance d'une suite de mots, il suffit au modèle multigrammes de calculer le produit des probabilités des entités qui la composent. Un intérêt de ce modèle réside dans le fait qu'il permet une structuration non supervisée de successions de mots (appartenant à un vocabulaire donné V) en unités plus longues et de longueur variable, à partir de simples considérations statistiques.

Une des premières applications de ce modèle a été la recherche de séquences typiques dans un texte écrit dont les espaces entre les mots ont été supprimés. Cette application, qui utilise un vocabulaire de 26 lettres, a conduit à la définition d'unités segmentales intermédiaires entre le caractère et le mot. Une autre application consistait à modéliser, d'une manière probabiliste, l'enchaînement entre les mots dans un corpus de texte. Cette application, fondée sur un vocabulaire limité d'environ 900 mots, a démontré l'intérêt du modèle multigrammes en modélisation du langage [23]. Plusieurs autres applications ont suivi, mais à chaque fois, le vocabulaire utilisé est de taille limitée [46]. En effet, pour définir l'ensemble de multigrammes ($M^* = \{m_i^*\}$) et évaluer la valeur de vraisemblance de chacune des entités m_i^* qui le composent, le modèle commence par choisir un ensemble de multigrammes M^0 qu'il améliore au fur et à mesure des itérations jusqu'à ce qu'il converge vers un optimum. Le modèle aura donc à modifier, lors de chaque itération i, l'ensemble des multigrammes M^i en cours et par conséquent à réestimer leurs valeurs de vraisemblance. Ainsi, dans le cas où on utilise de grands vocabulaires, le nombre de multigrammes à évaluer, lors de chaque itération, devient énorme. L'estimation des paramètres dans ce cas engendre le besoin d'un espace mémoire important et d'une machine très puissante, ce qui n'est pas toujours évident. C'est pour ces raisons que les domaines d'application du modèle multigrammes demeurent restreints.

Pour pouvoir utiliser de grands vocabulaires, nous présentons ci-dessous un nouveau modèle de langage : le modèle multiclasses. Ce modèle emploie également une composante linguistique supplémentaire, permettant de mieux prendre en compte la structure d'une phrase.

5.3 Le modèle multiclasses

Ce modèle, inspiré de l'approche multigrammes, se fonde sur des classes syntaxiques. Il suppose qu'une source, gouvernant l'activité d'une langue, émet des séquences de mots dont les classes syntaxiques correspondantes ne sont pas indépendantes et que leurs dépendances est de longueur variable. Chaque séquence de classes, nommée multiclasse, est supposée correspondre à une entité sous-jacente issue d'un ensemble fini $M_C^* = \{m_{c_i}^*\}$. Dans le cas idéal, M_C^* sera l'ensemble des syntagmes représentant la structure du langage naturel. Localiser un ensemble de multiclasses dans une suite de mots, $W_1^T = w_1,w_2,\ldots w_T$, revient à définir une segmentation L_C de la suite de classes syntaxiques correspondante $C_1^T = c_1,c_2,\ldots c_T$, comme dans l'exemple suivant :

$$
\begin{array}{llllll}
\text{Suite de séquences } M_C : & m_{ci_1} & m_{ci_2} & m_{ci_3} & \ldots & m_{ci_l} \\
 & \Uparrow & \Uparrow & \Uparrow & & \Uparrow \\
\text{Segmentation } L_C : & [c_1,c_2] & [c_3] & [c_4,c_5,c_6] & \ldots & [c_j,\ldots c_T] \\
 & & & \Uparrow & & \\
 & & & \Uparrow & & \\
\text{Suite de classes syntaxiques } C_1^T : & \multicolumn{5}{c}{c_1,c_2,c_3,c_4,c_5,c_6,\ldots c_j,\ldots c_T} \\
 & & & \Uparrow & & \\
\text{Suite de mots } W_1^T : & \multicolumn{5}{c}{w_1,w_2,w_3,w_4,w_5,w_6,\ldots w_j,\ldots w_T}
\end{array}
$$

Ainsi, étant donné que le nombre de classes utilisées est limité à 233 classes syntaxiques, le modèle multiclasses est capable de se servir de très grands vocabulaires.

Dans la langue, un mot peut appartenir à plusieurs classes syntaxiques ; le mot "*orange*", par exemple, est un nom (NOM) dans la phrase "*l'orange est délicieuse*" et un adjectif (ADJ) dans la phrase "*la porte orange est ouverte*". Ainsi, pour définir les classes syntaxiques C_1^T d'une suite de mots W_1^T, le modèle multiclasses utilise la méthode d'étiquetage présentée dans le paragraphe 4.4.

L'utilisation du modèle multiclasses nécessite la détermination des séquences de classes syntaxiques optimales $M_C^* = \{m_{c_i}^*\}$. Elle demande également de définir l'approche permettant l'étiquetage des mots avec leurs classes syntaxiques. La vraisemblance d'une suite de mots W_1^T, étiquetés par la suite de classes syntaxiques C_1^T, est définie comme suit :

$$\mathcal{P}(W_1^T) = (\prod_{i=1}^{T} p(w_i/c_i)) \times \mathcal{P}(C_1^T) \tag{5.18}$$

où $p(w_i/c_i)$ représente la probabilité conditionnelle que le mot w_i soit étiqueté par la classe syntaxique c_i, et où $\mathcal{P}(C_1^T)$ est la vraisemblance de la suite de classes C_1^T. La probabilité $p(w_i/c_i)$ est définie par l'équation suivante :

$$p(w_i/c_i) = \frac{N(w_i/c_i)}{N(c_i)} \tag{5.19}$$

où $N(w_i/c_i)$ représente le nombre de fois où le mot w_i est étiqueté par la classe c_i dans le corpus d'apprentissage, et où $N(.)$ représente le nombre d'occurrences de la classe en argument. Dans le cas où chaque classe c_k ne contient qu'un seul mot w_k, la probabilité conditionnelle $p(w_k/c_k)$ vaut 1. Ainsi, le premier terme de l'équation 5.18 peut être supprimé :

$$\mathcal{P}(W_1^T) = \mathcal{P}(C_1^T). \tag{5.20}$$

Ceci nous ramène au cas de l'approche multigrammes (cf. §5.2).

L'estimation de la vraisemblance de la suite de classes syntaxiques $\mathcal{P}(C_1^T)$ de l'équation 5.18, peut être réalisée de la même manière que dans l'approche multigrammes, en utilisant soit l'équation 5.2, soit l'équation 5.3. Cette méthode nécessite la détermination de l'ensemble des séquences optimales $M_C^* = \{m_{c_i}^*\}$ qui maximisent la vraisemblance d'un corpus de classes syntaxiques, dont le corpus de mots correspondant est représentatif du comportement de la langue. Ainsi, $\mathcal{P}(C_1^T)$ est soit définie comme la somme des vraisemblances de toutes les segmentations possibles de C_1^T :

$$\mathcal{P}(C_1^T) = \mathcal{P}(C_1^T/M_C^*) = \sum_{(L_C,S_C)} \mathcal{P}(C_1^T,L_C,S_C/M_C^*) \tag{5.21}$$

soit comme la vraisemblance de la meilleure segmentation de C_1^T :

$$\mathcal{P}(C_1^T) = \mathcal{P}^*(C_1^T/M_C^*) = \max_{(L_C,S_C)} \mathcal{P}(C_1^T,L_C,S_C/M_C^*) \tag{5.22}$$

où (L_C,S_C) représente une segmentation L_C possible de C_1^T avec les séquences S_C. Dans le modèle multiclasses, les séquences de classes sont supposées indépendantes. La vraisemblance d'une segmentation L_C du corpus C_1^T avec les séquences S_C (C_1^T,L_C,S_C) est donc définie comme suit :

$$\mathcal{P}(C_1^T,L_C,S_C/M_C^*) = \prod_{t}^{N(L_C,S_C)} p(s_{c_t}/m_{c_{i_t}}^*)\ p(m_{c_{i_t}}^*) \tag{5.23}$$

où s_{c_t} représente une séquence correspondant à une multiclasse $m_{c_{i_t}}^*$ (de rang i_t) de M_C^*, et $N(L_C,S_C)$ le nombre de séquences dans L_C (ce qui correspond également au nombre de multiclasses dans S_C).

Dans notre cas, chaque séquence de classes s_{c_t} ne peut être étiquetée que par une seule multiclasse $m_{c_{i_t}}^*$. Nous nous trouvons ainsi dans un cas où la distribution *a posteriori* des séquences observées pour chaque multiclasse $m_{c_{i_t}}^*$ est une fonction de Dirac : $p(s_{c_t}/m_{c_i}^*) = 1$ si $s_{c_t} = m_{c_i}^*$, et 0 sinon. Par conséquent, pour une segmentation L_C donnée, il n'existe qu'une seule combinaison de vraisemblance non nulle. Il s'agit de la combinaison S_{L_C} telle que pour toute séquence de L_C : $p(s_{c_t}/m_{c_{i_t}}^*) = 1$. L'équation 5.23 devient :

$$\begin{aligned} \mathcal{P}(C_1^T,L_C,S_C/M_C^*) &= \mathcal{P}(C_1^T,L_C,S_{L_C}/M_C^*) \\ &= \mathcal{P}(C_1^T,L_C/M_C^*) \\ &= \textstyle\prod_t^{N(L_C)} p(m_{c_{i_t}}^*) \end{aligned} \tag{5.24}$$

où $N(L_C)$ représente le nombre de séquences dans L_C qui est en fait le nombre de multiclasses dans S_{L_C}. Le terme $m_{c_{i_t}}^*$ est une multiclasse qui représente une séquence s_{c_t} dans L_C. Par conséquent, la vraisemblance d'une segmentation L_C d'un corpus C_1^T est le produit des probabilités des multiclasses qui la composent.

En limitant à 3 mots la taille maximale d'une séquence et en utilisant l'équation 5.22, la vraisemblance de la suite de mots "$w_1\ w_2\ w_3\ w_4$" étiquetée par la suite de classes "$c_1\ c_2\ c_3\ c_4$", est définie comme suit :

$$\mathcal{P}(w_1\ w_2\ w_3\ w_4) = (\prod_{i=1}^{4} p(w_i/c_i)) \times \max \begin{cases} p([c_1\ c_2\ c_3])\ \ p([c_4]), \\ p([c_1\ c_2])\ \ p([c_3\ c_4]), \\ p([c_1])\ \ p([c_2\ c_3\ c_4]), \\ p([c_1\ c_2])\ \ p([c_3])\ \ p([c_4]), \\ p([c_1])\ \ p([c_2\ c_3])\ \ p([c_4]), \\ p([c_1])\ \ p([c_2])\ \ p([c_3\ c_4]), \\ p([c_1])\ \ p([c_2])\ \ p([c_3])\ \ p([c_4]) \end{cases} \tag{5.25}$$

Dans le cas où c'est l'équation 5.21 qui est utilisée pour l'estimation de la vraisemblance de la suite de classes "$c_1\ c_2\ c_3\ c_4$", il suffit de remplacer dans l'équation 5.25 le maximum (max) par la somme ($\sum$).

5.3.1 Distribution a priori des multiclasses

Pour estimer l'ensemble optimal des multiclasses $M_C^* = \{m_{c_i}^*\}$ et l'ensemble des paramètres (probabilités) correspondant $\{p(m_{c_i}^*)\}$, nous procédons de la même manière que dans l'approche multigrammes. Nous débutons avec l'initialisation d'un ensemble de multiclasses M_C^0 et des paramètres correspondants. Ensuite, à chaque itération, nous procédons en deux étapes :

- Modifier l'ensemble des multiclasses M_C^k par M_C^{k+1}, pour réduire l'entropie du modèle. Une heuristique, qui consiste à supprimer de M_C^k toutes les multiclasses ayant une très petite probabilité, est appliquée.
- Estimer de nouveau l'ensemble des paramètres restants pour maximiser la vraisemblance des données $\mathcal{P}(C_1^T/M^{k+1})$.

La procédure s'arrête soit lorsque la vraisemblance des données cesse de croître, soit lorsque la composition de l'ensemble des multiclasses se stabilise. Les résultats de cette procédure sont l'ensemble des multiclasses optimales $M_C^* = \{m_{c_i}^*\}$ et l'ensemble des paramètres correspondants $\{p(m_{c_i}^*)\}$. Pour réestimer l'ensemble de ces paramètres, nous utilisons le principe de maximum de vraisemblance comme défini au paragraphe 5.2. Ainsi, la probabilité d'une multiclasse à l'itération $(k+1)$ est définie de la même façon que l'équation 5.17, comme suit :

$$p^{(k+1)}(m_{c_i}) = \frac{\sum_{L_C} N(m_{c_i}/L_C,S_{L_C})\ \mathcal{P}^{(k)}(C_1^T,L_C,S_{L_C})}{\sum_{L_C} N(L_C,S_{L_C})\ \mathcal{P}^{(k)}(C_1^T,L_C,S_{L_C})} \tag{5.26}$$

où $N(m_{c_i}/L_C,S_{L_C})$ représente le nombre d'occurrences de m_{c_i} dans la suite de séquences S_{L_C} associée à la segmentation L_C. L'estimation de cette équation peut se faire avec un algorithme de type *forward-backward* ou de type Viterbi, comme présenté ci-dessous.

5.3.2 Estimation au moyen de l'algorithme Forward-Backward

La formule de réestimation 5.26 peut être évaluée au moyen d'un algorithme *forward-backward* [14], de sorte que la complexité de l'algorithme croît linéairement avec le nombre de mots dans le corpus d'apprentissage. L'algorithme *forward-backward* repose sur la définition d'une variable *forward* α et d'une variable *backward* β. Dans un premier temps, on établit les relations de récurrence permettant de calculer α et β. On réécrit ensuite la formule de réestimation 5.26 en fonction de $\alpha^{(k)}$ et $\beta^{(k)}$, variables *forward-backward* de l'itération (k).

Soit C_{t1}^{t2} la suite de classes $c_{t_1},c_{t_1+1},c_{t_1+2},\dots c_{t2}$. La variable *forward* $\alpha(t)$ est définie comme la vraisemblance de la suite C_1^t contenant les t premières classes :

$$\alpha(t) = \mathcal{P}(C_1^t) \tag{5.27}$$

les séquences étant supposées indépendantes. Ainsi, la vraisemblance de la suite C_1^t associée à une segmentation L_{C_i}, dont la dernière séquence est de longueur l, est définie comme suit :

$$\alpha(t) = \alpha(t-l)p([c_{t-l+1},\dots c_t]). \tag{5.28}$$

Dans le cas du modèle multiclasses, toute segmentation L_{C_i} se termine par une séquence de longueur égale soit à 1, 2, ... ou n. Par conséquent, en notant T le nombre de classes dans le

pour $1 \leq t \leq T$:			
$\alpha(t)$	$=$	$\sum_{l=1}^{n} \alpha(t-l) \;\; p([c_{t-l+1}, \ldots c_t])$	
où $\alpha(0) = 1$		et	$\alpha(t) = 0$ pour $t < 0$

TAB. 5.1 – *Formule de récurrence de la variable forward α.*

corpus d'apprentissage, la variable $\alpha(t)$ peut se calculer à partir des variables $\alpha(t-1)$,$\alpha(t-2)$,$\ldots \alpha(t-n)$, selon la procédure de récurrence indiquée dans la table 5.1.

On définit, de la même façon, la variable $\beta(t)$ comme la vraisemblance des $(T-t)$ dernières classes :

$$\beta(t) = \mathcal{P}(C_{t+1}^{T}). \tag{5.29}$$

Ainsi, la vraisemblance de suite de classes C_{t+1}^{T} associée à une segmentation L_{C_i} débutant par une séquence de longueur l est définie par l'équation suivante :

$$\beta(t) = p([c_{t+1}, \ldots c_{t+l}])\beta(t+l). \tag{5.30}$$

Toute segmentation L_{C_i} de C_{t+1}^{T} débute par une séquence de longueur égale soit à 1, 2, ... ou n. Par conséquent, la variable $\beta(t)$ peut se calculer à partir des variables $\beta(t+1)$,$\beta(t+2)$,$\ldots \beta(t+n)$, selon la procédure de récurrence indiquée dans la table 5.2.

pour $1 \leq t \leq T$:			
$\beta(t)$	$=$	$\sum_{l=1}^{n} p([c_{t+1}, \ldots c_{t+l}]) \;\; \beta(t+l)$	
où $\beta(T) = 1$		et	$\beta(t) = 0$ pour $t > T$

TAB. 5.2 – *Formule de récurrence de la variable backward β.*

La démonstration de la formule de réestimation 5.26 en fonction des variables *forward-backward* est exposée en annexe C ; on se contente ici d'en donner le résultat :

$$p^{(k+1)}(m_{c_i}) = \frac{\sum_{t=1}^{T} \sum_{l=1}^{n} \delta(t,l,m_{c_i}) \;\; \alpha^{(k)}(t-l) \;\; p^{(k)}(m_{c_i}) \;\; \beta^{(k)}(t)}{\sum_{t=1}^{T} \alpha^{(k)}(t) \;\; \beta^{(k)}(t)} \tag{5.31}$$

où le terme $\delta(t,l,m_{c_i})$ est défini comme suit :

$$\delta(t,l,m_{c_i}) = \begin{cases} 1 \text{ si } [c_{t-l+1}, \ldots c_t] = m_{c_i} \\ 0 \text{ sinon} \end{cases} \tag{5.32}$$

5.3.3 Estimation au moyen de l'algorithme de Viterbi

En ne conservant dans l'équation 5.26 que la contribution de la segmentation $L_C^{*(k)}$ la plus vraisemblable à l'itération (k), la formule de réestimation devient :

$$p^{(k+1)}(m_{c_i}) = \frac{N(m_{c_i}/L_C^{*(k)})}{N(L_C^{*(k)})} \tag{5.33}$$

où

$$L^{*(k)} = arg\ max_{L_C} \mathcal{P}^{(k)}(C_1^T, L_C). \tag{5.34}$$

La probabilité de la multiclasse m_{c_i} est donc estimée comme sa fréquence relative le long de la meilleure segmentation du corpus à l'itération (k). La détermination de la meilleure segmentation à chaque itération peut être effectuée par l'algorithme de programmation dynamique de Viterbi [46].

Les performances des modèles multigrammes utilisant l'algorithme *forward-backward* étaient légèrement supérieures à celles employant l'algorithme de Viterbi [46]. Nous nous sommes ainsi tournés vers l'algorithme *forward-backward* pour l'estimation des paramètres du modèle multiclasses.

5.3.4 Conclusion sur les multiclasses

Le modèle multiclasses suppose que les classes syntaxiques d'une suite de mots soient décomposées en entités appartenant à un ensemble fini M_C^*. Le modèle extrait donc, tout d'abord, l'ensemble fini M_C^* des entités multiclasses et il attribue ensuite une valeur de probabilité pour chacune d'entre elles. Ainsi, pour estimer la vraisemblance d'une suite de mots, il suffit de calculer la vraisemblance de la suite de classes correspondante, en tenant compte de l'étiquetage (équation 5.18). La vraisemblance de la suite de classes, pour une segmentation donnée, est le produit de probabilité des multiclasses qui la composent. Pour extraire d'une manière automatique les classes syntaxiques correspondant à une suite de mots donnée, le modèle utilise l'approche d'étiquetage probabiliste définie dans le paragraphe 4.4. Nous présentons dans le paragraphe 5.5.2 de ce chapitre une évaluation en terme de perplexité de ce modèle.

Un des avantages de ce modèle est sa capacité, tout comme le modèle multigrammes, à permettre une structuration non supervisée d'une succession de mots en unités plus longues et de longueur variable, à partir de simples considérations statistiques. De plus, à la différence du modèle multigrammes, le modèle multiclasses apporte une connaissance linguistique supplémentaire et permet d'utiliser de grands vocabulaires. En effet, en regardant les multiclasses M_C^*, nous remarquons que beaucoup d'entre elles correspondent à des groupes syntaxiques de la langue. Nous présentons dans la table 5.3 quelques multiclasses types.

ARD NOM : l'état, la mesure, le mécanisme, ...
ARD NOP : le RPR, le FLN, la Nouvelle-Calédonie, ...
ARD NOM VEC : l'état entend, la situation évolue, le marketing devrait, ...
ADJ NOM ADJ : inépuisables réserves électorales, grande partie intégrante, grand public britannique, ...
VEC ARD NOM : réclament la création, oubliera les sommes, demande l'addition, ...

TAB. 5.3 – *Un échantillon de multiclasses avec quelques séquences de mots correspondantes.*

Nous notons également que beaucoup d'autres multiclasses ne correspondent à aucune structure linguistique connue, malgré les vraisemblances importantes qu'elles possèdent. Nous présentons dans la table 5.4 un échantillon de ces multiclasses.

“ NOM	NOM “ , COM
NOM “ E_O	(ARD :
...	

TAB. 5.4 – *Un échantillon de multiclasses non conventionnel.*

Comme le modèle multiclasses utilise un nombre limité de classes syntaxiques (233), il est donc capable de se servir d'un vocabulaire illimité de mots, sans avoir recours à une grande capacité de calcul.

En utilisant ce modèle, les multiclasses d'une phrase sont supposées indépendantes, ce qui est en contradiction avec la structure du langage naturel. Dans la langue, une phrase est caractérisée par la cohésion interne de ses mots. Les mots se regroupent et constituent des sous-ensembles (syntagmes), qui eux-mêmes s'assemblent pour former d'autres sous-ensembles, et ainsi de suite jusqu'à la construction de la structure de la phrase. Par exemple, les mots d'une phrase simple se regroupent pour former un groupe nominal et un groupe verbal, qui eux-mêmes s'assemblent pour construire sa structure syntaxique. Une phrase est donc une arborescence dont les feuilles sont les mots qui la composent. Pour remédier au problème d'indépendance entre les séquences (multiclasses) et pour se rapprocher le plus possible de la structure du langage naturel, nous proposons un nouveau modèle de langage : le modèle hiérarchique. Nous présentons en détail le fonctionnement de ce modèle dans le prochain paragraphe.

S. Deligne a présenté une nouvelle approche permettant de tenir compte de la dépendance entre les séquences dans le modèle multigrammes [47]. L'originalité de cette approche est sa capacité à combiner un modèle bigrammes et le modèle multigrammes, avec un nombre de paramètres restreint. Néanmoins, cette nouvelle approche, bien qu'elle nécessite un temps de calcul conséquent, n'a pas réussi à dépasser les performances d'un modèle trigrammes classique.

5.4 Le modèle hiérarchique

L'idée de se fonder sur une hiérarchie particulière, lors de l'estimation de la vraisemblance, a été déjà utilisée depuis longtemps. Par exemple, les modèles de langage à grammaires probabilistes tentent de représenter la langue sous forme de règles grammaticales probabilistes décrivant la vraisemblance des phrases (cf. §3.4.9). Ces règles sont souvent traduites sous forme de graphe ou d'arbre, construisant ainsi les arborescences les plus probables. Un autre modèle, utilisant une hiérarchie particulière, vient également d'être proposé [89]. Ce modèle utilise les n-grammes les plus fréquents pour transformer une phrase en suites de mots de longueur variable, construisant ainsi un ensemble fini d'entités. Ensuite, ces entités sont analysées par un modèle n-grammes pour former les entités du niveau supérieur de la hiérarchie. Ce processus est répété jusqu'à la construction d'une entité regroupant toute la phrase (la racine de l'arborescence).

Pour améliorer les performances du modèle multiclasses, lors de l'estimation de la vraisemblance d'une phrase, il nous a paru important de prendre en compte la hiérarchie représentant ses différents composants. Nous avons ainsi construit un modèle hiérarchique. L'utilisation de ce modèle permet de remédier à la contrainte d'indépendance qui existe entre les multiclasses dans l'approche multiclasses. En effet, le modèle hiérarchique suppose que les classes syntaxiques des mots dans une phrase soient dépendantes ; le regroupement des classes et leurs rattachements à des entités forme le premier niveau de la hiérarchie. A leur tour les entités de ce premier niveau vont être rattachées à d'autres et construisent ainsi un deuxième niveau, ainsi de suite jusqu'à un niveau donné. Chaque entité, d'un niveau de hiérarchie j donné, est supposée appartenir à un ensemble fini $M^*_{Cj} = \{m^*_{c^j_i}\}$. Pour des raisons de clarté, nous utilisons le même terme multiclasses pour définir les entités d'un niveau donné de la hiérarchie. Localiser l'ensemble des multiclasses à différents niveaux de la hiérarchie, dans une suite de mots ($W_1^T = w_1, w_2, \dots w_T$), revient donc à définir une segmentation L_C de la suite de classes syntaxiques correspondante $C_1^T = c_1, c_2, \dots c_T$. Cette segmentation se fait d'une manière hiérarchique en plusieurs niveaux de regroupement, comme présenté sur la figure 5.1 où nous nous limitons à deux niveaux de hiérarchie. Il s'agit donc de construire une sorte d'arbre syntaxique.

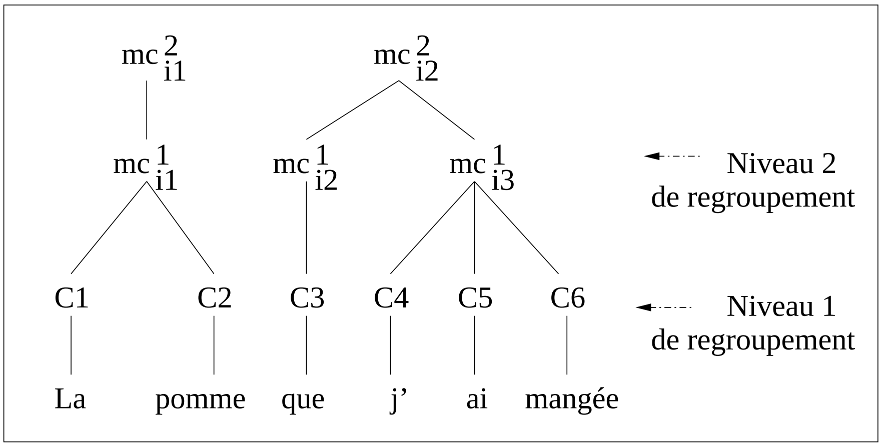


FIG. 5.1 – *Le modèle hiérarchique appliqué sur la phrase : "la pomme que j'ai mangée".*

5.4.1 Formulation du modèle

L'utilisation du modèle hiérarchique nécessite la détermination du niveau optimal de la hiérarchie ν et, pour chaque niveau j de la hiérarchie, les entités multiclasses optimales $M^*_{Cj} = \{m^*_{c^j_i}\}$. Elle oblige également à définir l'approche permettant l'étiquetage des mots avec leurs classes syntaxiques. Pour ce faire, nous utilisons la même approche que celle utilisée pour le modèle multiclasses (cf. §4.4). Pour déterminer la vraisemblance d'une suite de mots W_1^T, dont la suite de classes syntaxiques correspondante est C_1^T, le modèle hiérarchique évalue la vraisemblance de l'étiquetage qu'il multiplie par la vraisemblance de la suite de classes C_1^T :

$$\mathcal{P}(W_1^T) = (\prod_{i=1}^{T} p(w_i/c_i)) \times \mathcal{P}(C_1^T) \tag{5.35}$$

où $p(w_i/c_i)$ représente la probabilité conditionnelle pour que le mot w_i soit étiqueté par la classe syntaxique c_i. Pour estimer cette probabilité, nous utilisons le même principe que celui présenté pour le modèle multiclasses (équation 5.19). Ainsi, la seule différence avec le modèle multiclasses réside dans la manière d'estimer la vraisemblance $\mathcal{P}(C_1^T)$. En effet, selon le modèle hiérarchique à ν niveaux, $\mathcal{P}(C_1^T)$ équivaut à la vraisemblance de la meilleure segmentation de la suite de séquences du niveau ν, laquelle dépend de la meilleure segmentation de la suite de séquences du niveau $\nu - 1$, qui elle-même dépend de la meilleure segmentation du niveau inférieur jusqu'au niveau 1.

Soit Ω_j la suite de séquences (multiclasses ou classes syntaxiques) du niveau j ($1 \leq j \leq \nu$) correspondant à la meilleure segmentation Ω_{j-1} du niveau $j-1$. Le terme Ω_1 représente ainsi la suite de classes syntaxiques C_1^T correspondant à la suite de mots W_1^T à analyser. En utilisant cette notation, la vraisemblance $\mathcal{P}(C_1^T)$ est définie comme suit :

$$\mathcal{P}(C_1^T) = \mathcal{P}^*_{MC_n^\nu}(\Omega_\nu) \tag{5.36}$$

où $\mathcal{P}^*_{MC_n^\nu}(.)$ représente la vraisemblance au niveau ν de hiérarchie de la suite de séquences (multiclasses ou classes syntaxiques) en argument. Pour limiter la complexité, la taille maximale d'une séquence d'un niveau donné est limitée à n entités.

5.4.2 Estimation des paramètres du modèle hiérarchique

L'estimation de la vraisemblance $\mathcal{P}(C_1^T)$ nécessite la détermination du niveau maximal de la hiérarchie ν et de l'ensemble des séquences multiclasses optimales $M^*_{Cj} = \{m^*_{c_i^j}\}$ pour chaque niveau de la hiérarchie ($1 \leq j \leq \nu$). Pour estimer l'ensemble de ces séquences M^*_{Cj} nous procédons à partir du premier niveau de la hiérarchie comme suit : soit le corpus de classes $\Omega_1 = C_1^T$; l'ensemble des séquences optimales du premier niveau $M^*_{C^1}$ est celui qui maximise la vraisemblance du corpus de classes Ω_1. La meilleure segmentation de ce corpus Ω_2 servira de base à l'apprentissage au niveau supérieur (niveau 2). L'ensemble des séquences optimales de ce deuxième niveau $M^*_{C^2}$ est donc celui qui maximise la vraisemblance du corpus Ω_2 ; ce processus est réitéré et construit l'ensemble des séquences optimales M^*_{Cj} de chaque niveau j de la hiérarchie, en maximisant la vraisemblance du corpus du niveau inférieur Ω_j. Le processus s'arrête quand la vraisemblance du corpus cesse de croître :

$$\mathcal{P}^*_{MC_n^j}(\Omega_j) \leq \mathcal{P}^*_{MC_n^{j-1}}(\Omega_{j-1}).$$

Ce processus permet de définir la valeur du nombre optimal de niveaux, ν.

A chaque niveau de la hiérarchie j, la vraisemblance de la suite de séquences Ω_j, $\mathcal{P}^*_{MC_n^j}(\Omega_j)$, peut être estimée de deux manières différentes, en utilisant l'équation 5.21 ou l'équation 5.22. En effet, $\mathcal{P}^*_{MC_n^j}(\Omega_j)$ peut être considérée comme la somme des vraisemblances de toutes les segmentations possibles de Ω_j. Dans ce cas, elle sera estimée comme suit :

$$\mathcal{P}^*_{MC_n^j}(\Omega_j) = \mathcal{P}_{MC_n^j}(\Omega_j/M^*_{Cj}) = \sum_{(L_{C_j},S_{C_j})} \mathcal{P}_{MC_n^j}(\Omega_j,L_{C_j},S_{C_j}/M^*_{Cj}) \tag{5.37}$$

Sinon, $\mathcal{P}^*_{MC_n^j}(\Omega_j)$ est la vraisemblance de la meilleure segmentation de Ω_j :

$$\mathcal{P}^*_{MC_n^j}(\Omega_j) = \mathcal{P}^*_{MC_n^j}(\Omega_j/M^*_{Cj}) = \max_{(L_{C_j},S_{C_j})} \mathcal{P}_{MC_n^j}(\Omega_j,L_{C_j},S_{C_j}/M^*_{Cj}) \tag{5.38}$$

où (L_{C_j},S_{C_j}) représente une segmentation L_{C_j} possible (au niveau j de la hiérarchie) de Ω_j avec les séquences S_{C_j}.

La vraisemblance d'une segmentation L_{C_j}, du corpus Ω_j, avec les séquences S_{C_j} $(\Omega_j,L_{C_j},S_{C_j})$, est définie comme suit :

$$\mathcal{P}_{MC_n^j}(\Omega_j,L_{C_j},S_{C_j}/M^*_{Cj}) = \prod_{t}^{N(L_{C_j},S_{C_j})} p(s^j_{c_t}/m^*_{c^j_{i_t}})\ p(m^*_{c^j_{i_t}}) \tag{5.39}$$

où $s^j_{c_t}$ représente une séquence qui correspond à une multiclasse $m^*_{c^j_{i_t}}$ (de rang i_t au niveau j de la hiérarchie) dans M^*_{Cj}, et où $N(L_{C_j},S_{C_j})$ représente le nombre de séquences dans L_{C_j}.

En ce qui nous concerne, à chaque niveau j de la hiérarchie, chaque séquence $s^j_{c_t}$ ne peut être étiquetée que par une seule multiclasse $m^*_{c^j_{i_t}}$:

$$p(s^j_{c_t}/m^*_{c^j_i}) = 1 \qquad \text{si } s^j_{c_t} = m^*_{c^j_i}, \qquad \text{et 0 sinon.}$$

Par conséquent, pour une segmentation L_{C_j} donnée, au niveau j de la hiérarchie, il n'existe qu'une seule combinaison de vraisemblance non nulle. Il s'agit de la combinaison $S_{L_{C_j}}$ telle que pour toute séquence de L_{C_j} : $p(s^j_{c_t}/m^*_{c^j_i}) = 1$. Ainsi, l'équation 5.39 devient :

$$\begin{aligned}
\mathcal{P}_{MC^j_n}(\Omega_j, L_{C_j}, S_{C_j}/M^*_{C^j}) &= \mathcal{P}_{MC^j_n}(\Omega_j, L_{C_j}, S_{L_{C_j}}/M^*_{C^j}) \\
&= \mathcal{P}_{MC^j_n}(\Omega_j, L_{C_j}/M^*_{C^j}) \\
&= \textstyle\prod_t^{N(L_{C_j})} p(m^*_{c^j_{i_t}})
\end{aligned} \tag{5.40}$$

où $N(L_{C_j})$ représente le nombre de séquences dans L_{C_j}. Le terme $m^*_{c^j_{i_t}}$ est une multiclasse qui représente une séquence $s^j_{c_t}$ dans L_{C_j}. La vraisemblance d'une segmentation L_{C_j} d'un corpus Ω_j est donc le produit des probabilités des multiclasses qui la composent à ce niveau j de la hiérarchie.

5.4.3 Exemple

En utilisant l'équation 5.38, en limitant à 3 classes la taille maximale d'une multiclasse et en limitant à 2 les niveaux de la hiérarchie, la vraisemblance de la suite de mots "$w_1\ w_2\ w_3\ w_4$" étiquetée par la suite de classes "$c_1\ c_2\ c_3\ c_4$" est définie comme suit :

$$\mathcal{P}^*(w_1\ w_2\ w_3\ w_4) = (\prod_{i=1}^{4} p(w_i/c_i)) \times \mathcal{P}^*_{MC^2_3}(\Omega_2) \tag{5.41}$$

où $\mathcal{P}^*_{MC^2_3}(.)$ représente la vraisemblance du deuxième niveau de hiérarchie de la suite de classes en argument. Le terme Ω_2 désigne la suite des multiclasses du deuxième niveau, qui correspond à la meilleure segmentation du niveau inférieur $\Omega_1 = c_1\ c_2\ c_3\ c_4$. En effet, soit $\mathcal{P}^*_{MC^1_3}(\Omega_1)$ la vraisemblance obtenue au premier niveau de la hiérarchie :

$$\mathcal{P}^*_{MC^1_3}(\Omega_1) = \mathcal{P}^*_{MC^1_3}(c_1\ c_2\ c_3\ c_4) = \max \left\{ \begin{array}{l}
p([c_1\ c_2\ c_3])\ \ p([c_4]), \\
p([c_1\ c_2])\ \ p([c_3\ c_4]), \\
p([c_1])\ \ p([c_2\ c_3\ c_4]), \\
p([c_1\ c_2])\ \ p([c_3])\ \ p([c_4]), \\
p([c_1])\ \ p([c_2\ c_3])\ \ p([c_4]), \\
p([c_1])\ \ p([c_2])\ \ p([c_3\ c_4]), \\
p([c_1])\ \ p([c_2])\ \ p([c_3])\ \ p([c_4])
\end{array} \right. \tag{5.42}$$

Si nous supposons que le résultat de l'équation 5.42 est la valeur $p([c_1])p([c_2c_3])p([c_4])$, et si nous notons X la multiclasse $[c_2c_3]$ ($X \equiv [c_2c_3]$), la suite d'entités Ω_2 est tout simplement la séquence

c_1Xc_4 ($\Omega_2 = c_1Xc_4$). Ainsi, la valeur de vraisemblance $\mathcal{P}^*_{MC_3^2}(\Omega_2)$ est définie comme suit :

$$\mathcal{P}^*_{MC_3^2}(\Omega_2) = \mathcal{P}^*_{MC_3^2}(c_1\ X\ c_4) = \max \begin{cases} p([c_1])\ \ p([X\ c_4]), \\ p([c_1\ X])\ \ p([c_4]), \\ p([c_1\ X\ c_4]), \\ p([c_1])\ \ p([X])\ \ p([c_4]) \end{cases} \tag{5.43}$$

Dans le cas où le niveau de hiérarchie est limité à 1 ($\nu = 1$), le modèle hiérarchique se réduit au modèle multiclasses.

En utilisant l'équation 5.37, pour estimer la valeur de vraisemblance $\mathcal{P}^*_{MC_3^2}(\Omega_2)$, il suffit de remplacer le maximum de l'équation 5.43 par une somme.

5.4.4 Distribution a priori des multiclasses

Pour chaque niveau de la hiérarchie j, l'estimation de l'ensemble de multiclasses optimales $M^*_{Cj} = \{m^*_{c_i^j}\}$ et de l'ensemble des paramètres correspondants $\{p(m^*_{c_i^j})\}$ ($\sum_{m^*_{c_i^j} \in M^*_{Cj}} p(m^*_{c_i^j}) = 1$), se fait de la même manière que dans l'approche multiclasses. En effet, pour chaque niveau j, nous débutons avec l'initialisation d'un ensemble de multiclasses M^0_{Cj} et de paramètres correspondants (probabilités). Ensuite, à chaque itération, nous procédons en deux étapes :

- Modifier l'ensemble des multiclasses M^k_{Cj} par M^{k+1}_{Cj}, pour réduire l'entropie du modèle. Une heuristique, qui consiste à supprimer de M^k_{Cj} toutes les multiclasses, de niveau j de la hiérarchie, ayant une très petite probabilité, est appliquée.
- Estimer de nouveau l'ensemble des paramètres restants pour maximiser la vraisemblance des données au niveau j, $\mathcal{P}_{MC_n^j}(\Omega_j/M^{k+1})$.

La procédure s'arrête lorsque la vraisemblance des données au niveau j cesse de croître, ou lorsque la composition de l'ensemble de multiclasses de ce même niveau se stabilise. Le résultat de cette procédure est l'ensemble des multiclasses optimales $M^*_{Cj} = \{m^*_{c_i^j}\}$ et l'ensemble des paramètres correspondants $\{p(m^*_{c_i^j})\}$. Pour réestimer l'ensemble de ces paramètres, nous utilisons le principe de maximum de vraisemblance, comme défini dans le paragraphe 5.2. Ainsi, la probabilité d'une multiclasse du niveau j à l'itération $(k+1)$ est définie comme suit :

$$p(k+1)(m_{c_i^j}) = \frac{\sum_{L_{C_j}} N(m_{c_i^j}/L_{C_j}, S_{L_{C_j}})\ \mathcal{P}^{(k)}_{MC_n^j}(\Omega_j, L_{C_j}, S_{L_{C_j}})}{\sum_{L_{C_j}} N(L_{C_j}, S_{L_{C_j}})\ \mathcal{P}^{(k)}_{MC_n^j}(\Omega_j, L_{C_j}, S_{L_{C_j}})} \tag{5.44}$$

où $N(m_{c_i^j}/L_{C_j}, S_{L_{C_j}})$ représente le nombre d'occurrences de $m_{c_i^j}$ dans la suite $S_{L_{C_j}}$ associée à la segmentation L_{C_j}. L'estimation de cette équation peut à nouveau se faire avec un algorithme de type *forward-backward* ou de type Viterbi.

5.5 Évaluation

Dans ce paragraphe, nous allons tout d'abord évaluer les performances des modèles multiclasses et hiérarchique en terme de perplexité. Ensuite, nous allons les comparer entre eux et aux modèles biclasses et triclasses (cf. §5.5.1). Nous présentons au chapitre 8 leur évaluation à travers le système MAUD.

Pour l'évaluation des modèles de langage n-classes, multiclasses et hiérarchique, nous utilisons un corpus d'environ 50 millions de mots (cf. §4.6). Pour obtenir les classes syntaxiques de chacun des mots de ce corpus, nous utilisons la méthode d'étiquetage définie dans 4.4. Le vocabulaire de mots utilisé regroupe 20000 mots environ, tandis que le vocabulaire de classes contient 233 classes syntaxiques (cf. §4.3). Il est à noter qu'un mot peut appartenir à plusieurs classes syntaxiques.

5.5.1 Le modèle n-classes

Lors de l'estimation de la vraisemblance d'une suite de mots $W_1^T = (w_1,w_2,\ldots,w_T)$, ce modèle suppose connue la classe syntaxique de chacun des mots qui la composent, formant ainsi la suite de classes $C_1^T = (c_1,c_2,\ldots,c_T)$. En utilisant le modèle n-classes, la vraisemblance de la suite de mots W_1^T est définie comme suit :

$$P(w_1^T) = p(w_1,\ldots w_{n-1}) \times \prod_{i=n}^{T} p(w_i/w_{i-n+1}\ldots,w_{i-1}) \tag{5.45}$$

où $p(w_i/w_{i-n+1}\ldots,w_{i-1})$ est la probabilité conditionnelle n-classes du mot w_i sachant les $n-1$ derniers mots qui précèdent. La probabilité $p(w_1,\ldots w_{n-1})$ est estimée par les modèles m-classes, où $m < n$.

La probabilité conditionnelle $p(w_i/w_{i-n+1}\ldots,w_{i-1})$ est définie suivant le même principe que l'approche n-classes présentée dans le paragraphe 3.4.2 :

$$p(w_i/w_{i-n+1}\ldots,w_{i-1}) = p(w_i/C(w_i)) \times p(C(w_i)/C(w_{i-n+1})\ldots,C(w_{i-1})) \tag{5.46}$$

où $C(.)$ est la classe syntaxique du mot en argument. La probabilité d'appartenance d'un mot à sa classe syntaxique, $p(w_i/C(w_i))$, est calculée comme suit :

$$p(w_i/C(w_i)) = \frac{N(w_i)}{N(C(w_i))} \tag{5.47}$$

où $N(.)$ est le nombre d'occurrences dans le corpus de l'argument. Bien sûr, pour estimer cette probabilité, le corpus doit être étiqueté (chaque mot du corpus doit être accompagné de sa classe syntaxique). En ce qui concerne la probabilité de succession de classes, $p(C(w_i)/C(w_{i-n+1})\ldots,C(w_{i-1}))$, nous l'avons estimée suivant le principe du maximum de vraisemblance défini au chapitre 3 pour les modèles n-grammes. Dans le but de résoudre le phénomène des occurrences des suites de classes non rencontrées dans le corpus d'apprentissage, nous interpolons linéairement la distribution n-classes avec les distributions d'historiques plus restreints, (n-1)-classes, (n-2)-classes, … (cf. §3.5).

Ainsi, la probabilité triclasses (3-classes), d'une classe $C(w_i)$ sachant les deux classes qui précèdent $C(w_{i-2})$ et $C(w_{i-1})$, est définie comme suit :

$$\begin{aligned} p(C(w_i)/C(w_{i-2}),C(w_{i-1})) = \; & \alpha_1 \tfrac{N(C(w_{i-2}),C(w_{i-1}),C(w_i))}{N(C(w_{i-2}),C(w_{i-1}))} + \\ & \beta_1 \tfrac{N(C(w_{i-1}),C(w_i))}{N(C(w_{i-1}))} + \\ & \gamma_1 \tfrac{N(C(w_i))}{T} + \delta_1 \end{aligned} \tag{5.48}$$

où T représente la taille de ce corpus. Les variables α_1, β_1, γ_1 et δ_1 représentent les coefficients de pondération ($\alpha_1 + \beta_1 + \gamma_1 + \delta_1 = 1$).

En utilisant les mêmes notations que précédemment, la probabilité biclasses $p(C(w_i)/C(w_{i-1}))$ est calculée par une simple interpolation avec la distribution uniclasses et la distribution uniforme :

$$p(C(w_i)/C(w_{i-1})) = \beta_2 \frac{N(C(w_{i-1}),C(w_i))}{N(C(w_{i-1}))} + \gamma_2 \frac{N(C(w_i))}{T} + \delta_2 \tag{5.49}$$

où α_2, β_2 et γ_2 représentent les coefficients de pondération ($\alpha_2 + \beta_2 + \gamma_2 = 1$).

Nous présentons dans la tables 5.5 les paramètres d'interpolation des modèles biclasses et triclasses, en utilisant les données citées ci-dessus. Ainsi, en se fondant sur ces paramètres d'interpolation, le modèle biclasses a une valeur de perplexité égale à 138.97. En revanche, la perplexité du modèle triclasses vaut 87.73.

	α_i	β_i	γ_i	δ_i	PP
triclasses (i=1)	0.97	2.04×10^{-2}	9.6×10^{-3}	0	87.73
biclasses (i=2)		0.99	0.01	0	138.97

TAB. 5.5 – *Les paramètres d'interpolation et la perplexité (PP) des modèles triclasses et biclasses.*

5.5.2 Le modèle multiclasses

Les modèles n-classes et multiclasses s'appuient sur des hypothèses différentes, de sorte que la vraisemblance d'une phrase n'est pas calculée de la même manière dans les deux cas. En effet, la probabilité d'une triclasse $p(w_1w_2w_3)$ par exemple, ne dépend que du nombre d'occurrences de w_1 par rapport à sa classe $c(w_1)$ et du nombre d'occurrences conjointes des classes correspondantes $c(w_1)c(w_2)c(w_3)$. Éventuellement, si une technique d'interpolation est mise en oeuvre, $p(w_1w_2w_3)$ dépend également du nombre d'occurrences de la suite de classes $c(w_1)c(w_2)$, ainsi que du nombre d'occurrences de $c(w_1)$. Mais dans tous les cas l'inégalité $p(w_1w_2w_3) \leq p(w_1w_2) \leq p(w_1)$ est vérifiée, car le nombre d'occurrences de la classe qui correspond à w_1, $c(w_1)$, est au moins égal au nombre d'occurrences de la suite de classes $c(w_1)c(w_2)$, qui est lui-même au moins égal au nombre d'occurrences conjointes de $c(w_1)$, $c(w_2)$ et $c(w_3)$.

Dans le cadre de l'approche multiclasses, la probabilité d'une séquence de classes $[c(w_1)c(w_2)c(w_3)]$ dépend de la probabilité de toutes les autres séquences du corpus, et ce par le biais de la vraisemblance $\mathcal{P}^{(k)}(C,L)$ de chaque segmentation L (ou seulement de la meilleure segmentation) de la suite de classes C (équation de réestimation 5.26) :

$$p^{(k+1)}([c(w_1)c(w_2)c(w_3)]) = \frac{\sum_L N([c(w_1)c(w_2)c(w_3)]/L)\ \mathcal{P}^{(k)}(C_1^T,L)}{\sum_L N(L)\ \mathcal{P}^{(k)}(C_1^T,L)}$$

Par conséquent, il se peut que $p([c(w_1)]) \leq p([c(w_1)c(w_2)]) \leq p([c(w_1)c(w_2)c(w_3)])$. Pour cela, il suffit que les segmentations, où la séquence $[c(w_1)c(w_2)c(w_3)]$ apparaît, aient des valeurs de vraisemblance plus élevées que celles où apparaissent les séquences $[c(w_1)c(w_2)]$ ou $[c(w_1)]$. Les séquences qui sont issues de segmentations très peu vraisemblables ont, au cours de la procédure itérative d'estimation, leurs probabilités qui décroissent jusqu'à éventuellement atteindre la valeur nulle. De la sorte, le nombre de séquences, entre lesquelles la masse de probabilité est répartie, se trouve réduit au nombre des séquences qui confèrent une forte vraisemblance au corpus d'apprentissage. En somme, les modèles n-classes et multiclasses reposent sur des hypothèses différentes. L'approche multiclasses exploite le fait que les dépendances entre les mots peuvent être plus ou moins fortes et de longueurs variables. De cette façon, la masse de probabilité est principalement affectée à un nombre restreint de séquences effectivement caractéristiques des données.

Pour évaluer le modèle multiclasses, nous nous sommes fondés sur les données (corpus et vocabulaire) cités ci-dessus. En utilisant l'approche multiclasses, la vraisemblance d'une suite de mots W_1^T peut être évaluée suivant deux manières différentes : soit en utilisant la somme des vraisemblances de toutes les segmentations possibles (équation 5.21), soit en utilisant la vraisemblance de la meilleure segmentation (équation 5.22). Nous montrons, sur la figure 5.2, les valeurs de la perplexité obtenues par ces deux méthodes d'estimation avec différentes valeurs de n et de C_0, où n représente le nombre maximal de classes dans une multiclasse et C_0 la fréquence minimale d'une multiclasse dans le corpus d'apprentissage.

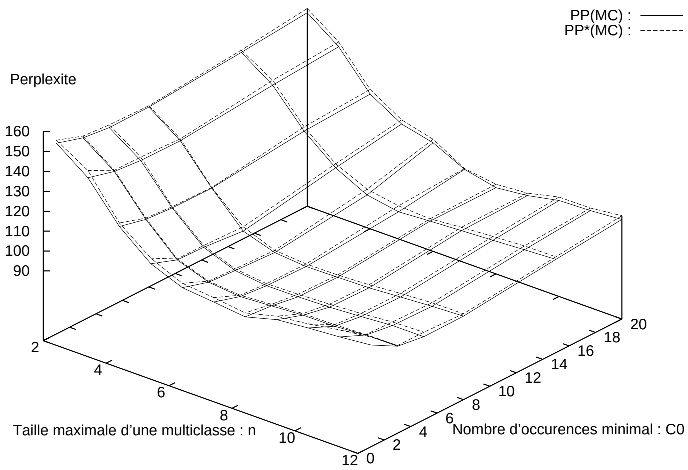


FIG. 5.2 – *Performance, en terme de perplexité du modèle multiclasses, utilisant les méthodes de réestimation de l'équation 5.21 ($PP(MC)$) et de l'équation 5.22 ($PP^*(MC)$), pour différentes valeurs de n et de C_0, où n est le nombre maximal de classes dans une multiclasse et C_0 le nombre d'occurrences minimal d'une multiclasse.*

Les résultats obtenus, avec l'ensemble de ces données, montrent que la méthode de réestimation définie par l'équation 5.21 (PP_{MC}) est légèrement plus performante que la méthode définie par l'équation 5.22 (PP^*_{MC}) ; ceci est dû au fait que, dans la méthode de réestimation définie dans l'équation 5.21, nous utilisons la somme des vraisemblances de toutes les segmentations possibles, contenant entre autres la meilleure segmentation. La figure 5.2 montre également que le modèle multiclasses atteint son optimum avec une perplexité de 93.52, ceci pour une valeur de n égale à 7 et de C_0 égale à 8. Nous remarquons que le nombre d'occurrences minimal d'une multiclasse est toujours aux alentours de 8. Nous présentons ainsi sur la figure 5.3, avec les deux méthodes de réestimation définies dans les équations 5.21 et 5.22, les performances en terme de perplexité du modèle multiclasses pour différentes valeurs de n, en fixant à 8 le nombre d'occurrences minimal d'une multiclasse ($C_0 = 8$). Par conséquent, en se fondant sur ces résultats, nous remarquons que le modèle multiclasses ($PP_{MC} = 93.52$), malgré sa capacité à améliorer d'une manière considérable les performances du modèle biclasses ($PP_{2-cl} = 138.97$), reste toujours limité par rapport à un modèle triclasses ($PP_{3-cl} = 87.73$).

En utilisant le modèle multiclasses optimal ($n = 7$ et $C_0 = 8$), nous présentons ci-dessous la

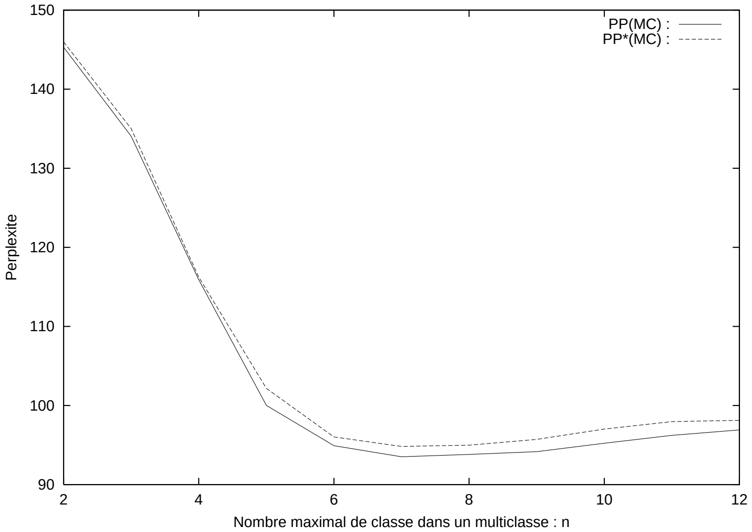


FIG. 5.3 – *Performance, en terme de perplexité du modèle multiclasses, utilisant les méthodes de réestimation de l'équation 5.21 ($PP(MC)$) et de l'équation 5.22 ($PP^*(MC)$), pour différentes valeurs de n, où n est le nombre maximal de classes dans une multiclasse.*

meilleure segmentation de la phrase : "*le RPR se dit le parti héritier du gaullisme*" :

[ARD NOP] [PCR VEC] [ARD NOM ADJ] [DDU NOM]

ARD NOP PCR VEC ARD NOM ADJ DDU NOM

le RPR se dit le parti héritier du gaullisme

5.5.3 Le modèle hiérarchique

Comme mentionné précédemment et à la différence du modèle multiclasses, le modèle hiérarchique tient compte de la dépendance qui existe entre les multiclasses lors de la modélisation. Ainsi, les modèles n-classes, multiclasses et hiérarchique s'appuient sur des hypothèses différentes, de sorte que la vraisemblance d'une phrase n'est pas calculée de la même façon dans les trois cas. En effet, à la différence des modèles n-classes et multiclasses, le modèle hiérarchique suppose que les classes syntaxiques des mots dans une phrase ne sont pas indépendantes et que leur dépendance est de longueur variable.

Pour une meilleure comparaison entre les modèles, les données (corpus et vocabulaire) utilisées pour l'évaluation du modèle hiérarchique sont les mêmes que celles utilisées pour les modèles multiclasses et n-classes. Nous présentons, sur la figure 5.4, les performances en terme de perplexité du modèle hiérarchique, en utilisant deux méthodes différentes de réestimation. Dans la première ($PP_{MC_n^\nu}$), nous considérons la vraisemblance d'une suite de classes C_1^T comme la somme des vraisemblances de toutes les segmentations possibles des multiclasses du dernier niveau ν de la hiérarchie (équation 5.37). Dans la deuxième méthode ($PP^*_{MC_n^\nu}$), nous considérons la vraisemblance d'une suite de classes C_1^T comme la vraisemblance de la meilleure segmentation des multiclasses du dernier niveau ν de la hiérarchie (équation 5.38). Les perplexités, données sur la figure 5.4, sont obtenues pour différentes valeurs de n et de ν, où n représente la taille maximale d'une multiclasse et ν le niveau maximal de la hiérarchie. Pour l'ensemble de ces expérimentations, nous avons fixé la fréquence minimale d'une multiclasse à 8. Les résultats obtenus, avec l'ensemble de ces données, montrent que la méthode de réestimation définie dans l'équation 5.37 ($PP_{MC_n^\nu}$) est légèrement meilleure que la méthode définie dans l'équation 5.38 ($PP^*_{MC_n^\nu}$) ; ceci est dû, comme cité auparavant pour l'approche multiclasses, au fait que dans l'équation 5.37 nous prenons la somme des vraisemblances de toutes les segmentations possibles, contenant entre autres la meilleure segmentation. Nous remarquons également, sur la figure 5.4, que le modèle hiérarchique atteint son optimum avec une perplexité de 74.98 et ceci pour une valeur de n égale à 5 et une valeur de ν égale à 4.

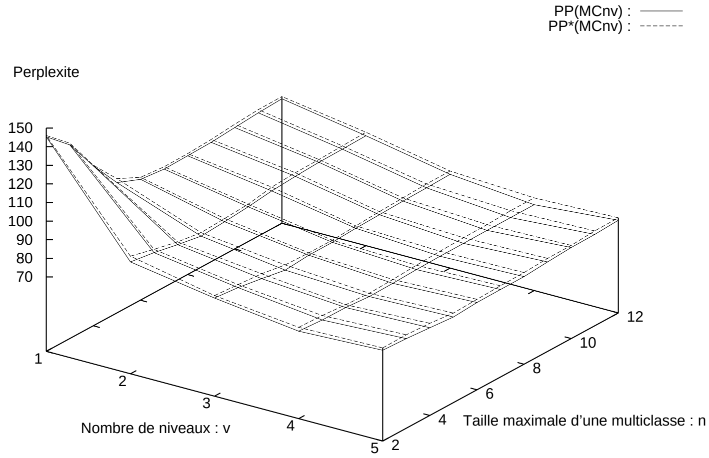


FIG. 5.4 – *Performance, en terme de perplexité du modèle hiérarchique, utilisant les méthodes de réestimation de l'équation 5.37 ($PP(MCnv)$) et de l'équation 5.38 ($PP^*(MCnv)$), pour différentes valeurs de n et de ν, où n est la taille maximale d'une multiclasse et ν le niveau maximal de la hiérarchie.*

Nous présentons, à la suite, les meilleures segmentations obtenues aux différents niveaux avec le modèle hiérarchique optimal ($n = 5$, $C_0 = 8$ et $\nu = 4$), pour la phrase : "*le RPR se dit le parti héritier du gaullisme*" :

[ARD NOP PCR VEC] [ARD NOM ADJ DDU NOM]
⇑ ⇑
[ARD NOP PCR VEC] [ARD NOM ADJ DDU NOM]
⇑ ⇑
[ARD NOP PCR VEC] [ARD NOM ADJ] [DDU NOM]
⇑ ⇑ ⇑
[ARD NOP] [PCR VEC] [ARD NOM ADJ] [DDU NOM]
⇑ ⇑ ⇑ ⇑
⇑ ⇑ ⇑ ⇑
ARD NOP PCR VEC ARD NOM ADJ DDU NOM
⇑ ⇑ ⇑ ⇑ ⇑ ⇑ ⇑ ⇑ ⇑
le RPR se dit le parti héritier du gaullisme

5.5.4 Bilan comparatif

Les modèles multiclasses et hiérarchique représentent une alternative compétitive aux modèles regroupant automatiquement les groupes syntaxiques d'une langue. En effet, en regardant les multiclasses, nous remarquons qu'une bonne partie d'entre elles correspond à des groupes syntaxiques de la langue française (par exemple : le groupe nominal, le groupe verbal, ...).

L'évaluation, en terme de perplexité des modèles multiclasses et hiérarchique, valide l'hypothèse de dépendance introduite par le modèle hiérarchique entre les multiclasses. En effet, le modèle multiclasses ($PP_{MC} = 93.52$), malgré sa capacité à améliorer d'une manière considérable les performances du modèle biclasses ($PP_{2-cl} = 138.97$), reste quand même limité par rapport à un modèle triclasses ($PP_{3-cl} = 87.73$). En revanche, en introduisant une dépendance entre les multiclasses d'une phrase (le modèle hiérarchique), les performances augmentent considérablement pour atteindre une perplexité de $PP_{MC_n^\nu} = 74.78$. Ceci lui permet ainsi d'améliorer les performances du modèle triclasses. Néanmoins, l'utilisation du modèle hiérarchique avec de grands vocabulaires, sans avoir recours aux classes, reste encore très coûteuse en espace mémoire et en temps de calcul. La comparaison en terme de perplexité n'est donc pas significative, entre les modèles multiclasses et hiérarchique se fondant sur les classes et un modèle n-grammes se basant uniquement sur les mots. Malgré cela, en étudiant les résultats, nous remarquons que les performances en terme de perplexité du modèle hiérarchique sont encore similaire à ceux d'un modèle n-grammes : $PP_{3-gr} = 74.65$.

En gardant les mêmes notations que précédemment, nous présentons dans la table 5.6 une comparaison en terme de perplexité entre les modèles multiclasses et hiérarchique avec un niveau de hiérarchie maximal de 4 ($\nu = 4$), ceci pour différentes valeur de n. Pour l'ensemble des valeurs de perplexité présentées dans cette table, la fréquence minimale d'une multiclasse est fixée à 8 et la méthode de réestimation utilisée se fonde sur la somme des vraisemblances de toutes les segmentations possibles (l'équation 5.21 pour le modèle multiclasses et l'équation 5.37 pour le modèle hiérarchique).

n	2	3	4	5	6	7	8	9	10
PP_{MC}	145.31	134.05	115.91	100.02	94.92	<u>93.52</u>	93.81	94.16	95.23
$PP_{MC_n^4}$	76.98	76.13	75.95	<u>74.78</u>	75.17	75.67	75.94	76.00	76.08

TAB. 5.6 – *Les performances en terme de perplexité des modèles multiclasses (PP_{MC}) et hiérarchique ($PP_{MC_n^4}$) pour différentes valeur de n, où n représente la taille maximale d'une multiclasse.*

Les résultats de cette expérimentation montre que le modèle multiclasses atteint son optimum pour une valeur de n égale à 7. En revanche, le modèle hiérarchique atteint la limite de ces performances avec une valeur de n égale à 5.

5.6 Conclusion

Les modèles multiclasses et hiérarchique, présentés dans ce chapitre, se fondent sur des hypothèses différentes de celles des modèles classiques de type n-grammes. Ces modèles supposent que les classes syntaxiques d'une phrase sont décomposées en entités (multiclasses), considérées capables de traduire sa structure. Ils permettent également une structuration non supervisée d'une phrase en séquences de mots de longueur variable, à partir de simples considérations statistiques. A la différence du modèle multigrammes, ces modèles sont capables d'utiliser de grands vocabulaires sans avoir besoin d'énormes corpus ni de machines puissantes. La différence entre le modèle multiclasses et le modèle hiérarchique réside dans leur façon de traiter les dépendances entre les multiclasses. En effet, le modèle multiclasses suppose indépendantes les multiclasses d'une phrase ; en revanche, le modèle hiérarchique les considère dépendantes en construisant une hiérarchie. En se limitant à un seul niveau de hiérarchie, le modèle hiérarchique se réduit au modèle multiclasses.

Les résultats d'évaluation confirment l'intérêt de prendre en compte la dépendance entre les multiclasses d'une phrase. En effet, le modèle hiérarchique a augmenté d'environ 21% les performances du modèle multiclasses ; ce dernier qui pourtant dépasse de 43% les performances d'un modèle biclasses, est loin derrière le modèle triclasses de 6%. En revanche, par rapport à ce modèle triclasses, le modèle hiérarchique est meilleur de 15%.

L'utilisation des modèles multiclasses et hiérarchique, lors des premières étapes de reconnaissance, n'est pas évidente. En effet, ces modèles ont besoin de la totalité de la phrase (ou presque) pour agir. L'amélioration des performances d'un système de RAP reste limitée si nous n'agissons qu'au niveau du filtrage des hypothèses, une nouvelle approche sera ainsi exposée dans le prochain chapitre. Cette approche extrait automatiquement, à partir d'un corpus, un ensemble fini de séquences de mots linguistiquement viables. Ces séquences serviront de base lors de l'estimation des paramètres des modèles de langage pouvant intervenir dès les premiers niveaux de la reconnaissance.

Chapitre 6

Modèles de langage à base de séquences

6.1 Introduction

Comme nous l'avons mentionné dans le chapitre précédent, nous trouvons dans la langue plusieurs séquences clés traduisant la structure d'une phrase. Ces séquences de mots sont naturellement de longueur variable et permettent d'avoir une élocution naturelle.

Pour tenir compte de ces séquences, lors des premières étapes de la prédiction et de la sélection, nous avons adapté les modèles de langage les plus utilisés dans les systèmes de RAP. Tout d'abord, nous avons considéré l'ensemble de ces séquences comme des unités de la langue. Ensuite, nous les avons ajoutées au vocabulaire de base, construisant ainsi un nouveau vocabulaire qui sert à l'apprentissage des modèles de langage. Par conséquent, lors de la prédiction et même de la sélection, les modèles de langage utilisant ce vocabulaire se fondent sur un historique d'unités où chacune d'entre elles peut être, soit un mot, soit une séquence. Ceci donne aux modèles la possibilité d'utiliser un historique plus important et de mieux prendre en compte le rôle prédictif de ces séquences. Les modèles de langage correspondants sont également capables de prédire la totalité d'une séquence, et de ne plus se limiter à la prédiction d'un seul mot.

Nous exposons, dans le paragraphe 6.2, l'intérêt de prendre en compte ces séquences lors de la reconnaissance. Nous présentons ensuite différentes méthodes, exposées dans la littérature, d'extraction de séquences de mots. Nous définissons, dans le paragraphe 6.4, une nouvelle approche d'extraction de séquences linguistiquement viables ; elle se sert des classes syntaxiques et se fonde sur des critères d'optimalité connus en théorie de l'information. En 6.5, nous abordons une autre approche ne se servant que des mots, que nous comparons à celle exposée au paragraphe 6.4. Nous proposons ensuite, dans les paragraphes suivants, différents modèles de langage se servant de ces séquences. Enfin, nous donnons une évaluation.

6.2 Intérêt des séquences en RAP

L'utilisation de certaines séquences de mots linguistiquement viables, comme unités dans un vocabulaire, permet aux modèles de langage de tenir compte d'autres contraintes langagières supplémentaires ; ceci engendrera un accroissement de leurs performances. En effet, l'utilisation de ces séquences, comporte de nombreux avantages :

– l'historique traité par les modèles de langage à contexte fixe (n-grammes, n-classes, ...)

devient variable, en fonction de la taille des séquences, ce qui est plus proche du comportement humain lors de la prédiction (cf. §3.4.5) ;
- la prononciation orale d'une expression contenant plusieurs mots est souvent différente de celle où on prononce un à un (d'une façon indépendante) les mots qui la compose. Ainsi, une idée consiste à introduire un modèle de prononciation spécifique à chaque séquence de mots ;
- les résultats fournis par un système de RAP permettent d'obtenir des phrases plus cohérentes au niveau linguistique que celles obtenues par des modèles à base de mots. Ceci est dû au fait que les séquences possèdent souvent des structures linguistiques viables qui aident à la reconnaissance.

6.3 Extraction des séquences de mots : l'état de l'art

L'idée d'extraire des séquences de mots, afin de les utiliser comme unités dans un vocabulaire, a passionné les chercheurs depuis longtemps. Néanmoins, l'introduction de ces séquences peut augmenter le nombre des unités (mots ou séquences de mots) peu fréquentes dans le corpus d'apprentissage, ce qui risque de réduire les performances des modèles de langage. Les séquences ne doivent donc pas être introduites arbitrairement dans le vocabulaire initial.

Au début des années quatre-vingt-dix R.L. Mercer a proposé un des premiers algorithmes d'extraction de séquences de mots ; cet algorithme est décrit dans [93]. Son objectif était d'utiliser ces séquences comme des unités du vocabulaire et ainsi de produire un modèle n-grammes qui les utilise. L'inconvénient de ce modèle est qu'il n'a pas un bon critère d'optimalité. En effet, pour extraire les séquences candidates, R.L. Mercer utilise une méthode itérative se fondant sur la valeur de l'information mutuelle $J(w_1,w_2)$ qui existe entre deux mots adjacents w_1 et w_2 :

$$J(w_1,w_2) = log\left[\frac{P(w_1,w_2)}{P(w_1).P(w_2)}\right] = log\left[\frac{N(w_1,w_2).T}{N(w_1).N(w_2)}\right] \tag{6.1}$$

où $P(.)$ est la probabilité d'apparition de la séquence en argument, $N(.)$ le nombre d'occurrences de la suite de mots en argument dans le corpus d'apprentissage et T la taille de ce corpus. Une valeur assez grande de $J(w_1,w_2)$ indique que les mots w_1 et w_2 ne se retrouvent pas juxtaposés d'une manière arbitraire, mais qu'ils constituent plutôt une séquence. R.L. Mercer définit alors deux seuils S_{min} et S_j ; le seuil S_{min} est utilisé pour s'assurer que la valeur de l'information mutuelle $J(w_1,w_2)$ est estimée sur une quantité suffisante de données, tandis que le seuil S_j est employé pour valider le choix des séquences. En effet, les couples de mots (w_i,w_j) ayant une valeur d'information mutuelle supérieure à S_j sont considérés comme des séquences. Ainsi, R.L. Mercer ajoute à son vocabulaire V_0 toutes les séquences (w_1,w_2) qui vérifient les conditions suivantes :

$$J(w_1,w_2) > S_j \quad \text{et} \quad N(w_1,w_2) > S_{min}. \tag{6.2}$$

Une fois le vocabulaire enrichi par les nouvelles séquences, construisant ainsi le vocabulaire $V1$, le corpus est mis à jour pour prendre en compte ces nouvelles séquences et les considérer ainsi comme des mots. Ce processus est répété tant qu'il existe des séquences, extraites du vocabulaire en cours V_i, qui satisfont la formule 6.2. Généralement, le vocabulaire final contient une quantité importante de séquences de taille illimitée, ce qui engendre une augmentation du nombre de mots peu rencontrés à prendre en compte. Les performances des modèles de langage, utilisant ce nouveau vocabulaire, ont ainsi de fortes chances de se dégrader.

Une autre méthode, qui permet d'extraire ces séquences de mots, a été proposée par E. Giachin et *al.* [69, 68]. L'idée de base de cette méthode est la suivante : au départ, on répartit les mots dans des classes séparées. Ensuite, pour chaque couple d'unités adjacentes u_i et u_j (mots ou séquences de mots), un modèle biclasses considérant ce couple comme une unité dans le corpus est évalué. Le couple qui réduit le plus la perplexité sera considéré comme une unité de la langue et il sera ajouté au vocabulaire. Après mise à jour du corpus avec cette nouvelle séquence, le processus sera répété et ceci tant que la perplexité diminue. K. Ries utilise également la perplexité comme critère d'optimalité [164]. La seule différence avec E. Giachin, est que K. Ries extrait, à chaque itération, un ensemble de séquences candidates et qu'il n'intègre dans son vocabulaire que celles qui minimisent la perplexité. Il est clair que cette procédure est très coûteuse en temps de calcul, même avec l'utilisation des classes qui permettront la réduction du nombre de paramètres à évaluer.

B. Suhm et *al.* [190] ainsi que P.E. Kenne et *al.* [103] proposent une autre méthode qui repose sur le même critère d'optimalité que celui de E. Giachin : la perplexité. La particularité de cette méthode par rapport à celle de E. Giachin réside dans le choix des séquences à traiter lors de chaque itération. En effet, au lieu d'essayer tous les couples d'unités et de choisir le meilleur, c'est plutôt celui qui fournit l'information mutuelle la plus élevée qui sera pris en compte. Cette méthode, moins coûteuse en temps de calcul que celle de Giachin, reste tout de même assez lente.

Une autre approche vient d'être publiée récemment par C. Beaujard et M. Jardino [15]. Elle repose sur des mesures distinctes lors de l'extraction des séquences : l'occurrence des bigrammes (couple de mots ou de séquences), l'information mutuelle, la probabilité de l'unité courante sachant celle qui précède et la probabilité de l'unité courante sachant celle qui suit. Cette approche commence par trier les couples d'unités adjacentes par ordre décroissant suivant une des précédentes mesures. Ensuite, elle parcourt les couples un à un et considère comme une unité ceux qui améliorent la vraisemblance du corpus. Ce processus se répète tant que la vraisemblance du corpus s'améliore. L'amélioration des performances, apportée par cette approche, n'était pas très significative par rapport à un modèle bigrammes.

L'inconvénient de ces algorithmes est leur complexité. De ce fait, ils n'ont été utilisés que sur des vocabulaires de quelques centaines de mots et seulement avec des modèles de type n-grammes ($n < 3$).

L'approche multiclasses, définie au chapitre 4, est différente de celles présentées ci-dessus. En effet, elle considère la phrase comme une suite de séquences de mots de longueur variable. La vraisemblance d'une phrase est donc évaluée, soit par la vraisemblance de la meilleure segmentation, soit par la somme des vraisemblances de toutes les segmentations possibles. Si nous voulons utiliser cette approche dans l'extraction des séquences, une idée consiste à lancer ce modèle sur un corpus de test et d'extraire ainsi toutes les séquences ayant une forte probabilité. Ces séquences seront ensuite utilisées comme des unités dans un vocabulaire. La probabilité de ces séquences ne dépend que de leur fréquence d'apparition dans le corpus d'apprentissage.

Pour pouvoir utiliser de grands vocabulaires et bénéficier d'autres connaissances (en plus de la fréquence d'apparition) lors de l'extraction des séquences, nous proposons une nouvelle approche ; celle-ci, présentée dans le paragraphe suivant, se fonde sur deux critères d'optimalité : l'information mutuelle et la perplexité.

6.4 Une nouvelle méthode d'extraction de séquences à base de classes syntaxiques

L'information mutuelle est une bonne mesure du lien qui existe entre deux unités c_1 et c_2. Cette mesure détermine la quantité d'information existante dans c_1 pour la prédiction de l'unité c_2 [93]. Par conséquent, une valeur d'information mutuelle assez grande, entre deux classes syntaxiques, indique que ces deux classes ne sont pas apparues l'une à coté de l'autre, par un pur hasard, mais qu'elles constituent plutôt tout ou partie d'un groupe syntaxique. En revanche, nous ne pouvons pas nous limiter seulement aux classes syntaxiques et à cette mesure pour l'extraction des séquences. En effet, en ne se fondant que sur l'information mutuelle, nous risquons d'augmenter le nombre d'événements non rencontrés dans le corpus, ce qui peut réduire les performances des modèles de langage. Il est donc nécessaire d'utiliser un bon critère d'optimalité permettant d'avoir un modèle plus robuste. Étant donné que la perplexité est un des moyens les plus utilisés dans l'évaluation des performances des modèles de langage, nous l'avons utilisée, dans notre approche, comme critère d'optimalité.

L'approche proposée par B. Suhm et P.E. Kenne pour l'extraction des séquences de mots est séduisante [190, 103]. Pourtant, elle possède l'inconvénient d'être très coûteuse en temps de calcul surtout si nous utilisons de grands vocabulaires. Ainsi, nous proposons une extension de cette approche lui permettant d'être utilisée sur de grands vocabulaires et de mieux tenir compte des contraintes syntaxiques de la langue.

6.4.1 Proposition d'un algorithme

L'algorithme que nous proposons est entièrement non supervisé. Il repose sur plusieurs paramètres :

- un corpus de mots, $W = w_1, \ldots w_M$, ainsi que le corpus de classes, $C = c_1, \ldots c_M$, correspondant. Il est à noter que chacun de ces deux corpus est réparti en deux sous-corpus, constituant ainsi le corpus d'apprentissage (W_A,C_A) et celui de test (W_T,C_T) ;
- un vocabulaire de mots V ; chaque mot de V doit disposer des classes auxquelles il peut appartenir ;
- un nombre maximal de mots q dans une séquence ; cette valeur de q nous permet de contrôler la convergence de l'algorithme ;
- un nombre d'occurrences minimal, T_{min}, que doit avoir la suite de mots dans le corpus d'apprentissage W_A, pour pouvoir être considérée comme une séquence ;
- un coefficient p ($0 \leq p \leq 1$) qui permet de définir le seuil T_J, utilisé lors de l'extraction des séquences candidates. Les séquences, obtenues par concaténation des couples de classes ayant une valeur d'information mutuelle $J(c_i,c_j)$ supérieure à T_J, sont considérées comme candidates. Il est à noter que nous ne tenons compte que des classes adjacentes dans le corpus C_A, dont les mots correspondants dans W_A appartiennent au vocabulaire V. Le seuil T_J est défini par l'équation suivante :

$$T_J = (1 - p) \max_{c_i \in C_A, c_j \in C_A} J(c_i,c_j) \tag{6.3}$$

  où $J(c_i,c_j)$ est la valeur de l'information mutuelle du couple (c_i, c_j) dans le corpus d'apprentissage C_A (voir l'équation 6.1).

L'algorithme d'extraction des séquences procède comme suit :

1. Initialiser l'ensemble des nouvelles classes candidates S_c ainsi que l'ensemble des séquences candidates S_w à nul.

2. Mettre à jour S_c avec les couples de classes (c_i,c_j) ayant une valeur d'information mutuelle supérieure au seuil T_J. Ces couples sont adjacents dans le corpus C_A et les mots correspondants dans W_A appartiennent au vocabulaire V. Le nombre de classes de mots dans une séquence, obtenu par la concaténation de c_1 et de c_2, doit être inférieur à q. Une classe c_i peut être, soit une classe de mot, soit une séquence de classes de mots.
3. Extraire, à partir du corpus d'apprentissage (W_A, C_A), les séquences de mots $S_w = \{s_i\}$ qui correspondent à l'ensemble S_c. En effet, nous parcourons le corpus (W_A, C_A) et à chaque fois que nous rencontrons, dans le corpus C_A, un couple de classes appartenant à S_c, nous ajoutons à l'ensemble S_w, la séquence de mots qui lui correspond dans W_A. Nous ne gardons que les séquences les plus fréquentes dont les mots appartiennent tous au vocabulaire V.
4. Mettre à jour les corpus W et C (les corpus d'apprentissage et de test) avec les nouvelles séquences de l'ensemble S_w et S_c respectivement. En effet, nous parcourons le corpus (W, C) et à chaque fois qu'une séquence s_i de l'ensemble S_w est rencontrée, dans W, elle sera remplacée par une étiquette. Dans le corpus C, le couple de classes (c_i,c_j), qui correspond à la séquence s_i, sera également remplacé par une nouvelle classe $(c_i - c_j)$. Cette nouvelle classe $(c_i - c_j)$ est une séquence appartenant à l'ensemble S_c. En cas de conflit de chevauchement entre deux séquences, lors de la mise à jour, nous donnons la priorité à la séquence qui a la plus grande valeur d'information mutuelle. Nous obtenons ainsi un nouveau corpus W' étiqueté par le corpus C'. Nous gardons la répartition de ces corpus en deux sous-ensembles : le corpus d'apprentissage (W'_A,C'_A) et le corpus de test (W'_T,C'_T).
5. Considérer l'ensemble des séquences S_w comme des unités et les additionner au vocabulaire V. Nous construisons ainsi un nouveau vocabulaire V'. Chaque unité (mot ou séquence de mots) de ce vocabulaire connaît les classes auxquelles elle peut appartenir.
6. Utiliser le vocabulaire V' pour estimer le modèle de langage sur le corpus d'apprentissage (W'_A,C'_A), et ainsi calculer la perplexité correspondante sur le corpus de test (W'_T,C'_T).
7. Si la perplexité diminue :
   (a) affecter V' à V,
   (b) affecter W' à W,
   (c) affecter C' à C,
   (d) retourner à 1
8. Sinon, extraire à partir des séquences candidates S_w, celles qui peuvent encore réduire la perplexité. Nous construisons ainsi un sous-ensemble d'unités (séquences de mots), S'_w, qui sera additionné au vocabulaire V.
9. Résultat : le vocabulaire V où chacune de ses unités (mot ou séquences de mots) connaît les classes (classes de mots ou séquences de classes de mots) auxquelles elle peut appartenir.

L'ensemble S_w de la huitième étape pourra contenir des séquences qui réduisent encore la perplexité. Pour extraire ces séquences, une solution consiste à les tester une à une et à prendre celles qui font converger l'algorithme ; l'opération est coûteuse en temps de calcul. Ainsi, nous proposons une autre méthode permettant l'extraction d'une grande partie de ces séquences en un temps raisonnable. Celle-ci procède comme suit : tout d'abord, nous trions les séquences de S_w de la plus grande à la plus petite en fonction de la valeur de l'information mutuelle des classes correspondantes dans S_c. En cas d'égalité, nous trions du plus grand au plus petit suivant le nombre d'occurrences dans W_A. Ensuite, nous procédons par récurrence sur chacune des deux moitiés de l'ensemble S_w. En effet, nous prenons les séquences de la première moitié de S_w et nous vérifions si elles réduisent la perplexité ; si c'est le cas, nous additionnons ce sous-ensemble de séquences au vocabulaire V, et nous traitons l'autre moitié. Sinon, nous divisons de nouveau la moitié en cours en deux sous-ensembles et nous les traitons à nouveau. Nous appliquons ce processus sur tout l'ensemble S_w.

Le seuil T_J joue un rôle important dans la vitesse d'exécution de notre algorithme : une valeur assez faible de T_J permet d'extraire beaucoup plus de séquences à chaque itération, accélérant ainsi la convergence de l'algorithme vers un optimum local. En revanche, une très grande valeur de T_J peut provoquer l'effet inverse rendant ainsi l'algorithme très coûteux en temps de calcul. Une très faible valeur de T_J augmentera énormément l'ensemble des séquences S_w qui seront traitées lors de la huitième étape de l'algorithme ; ceci peut rendre l'algorithme très coûteux en temps de calcul et en espace mémoire. Pour estimer la valeur de T_J, il suffit de bien approximer le seuil p (voir l'équation 6.3). Nous avons appliqué l'algorithme ci-dessus sur un petit corpus d'environ 4 millions de mots (2 mois du corpus "*Le Monde*"), avec différentes valeurs de p. Cette étude, présentée sur la figure 6.1, a montré que l'algorithme atteint son optimum en temps de calcul pour une valeur de p égale à 23%, avec des résultats similaires. Les valeurs de la perplexité sont presque les mêmes à moins de 1% de différence. Nous avons donc choisi cette valeur pour l'extraction des séquences de mots avec un corpus plus important (cf. §6.4.5).

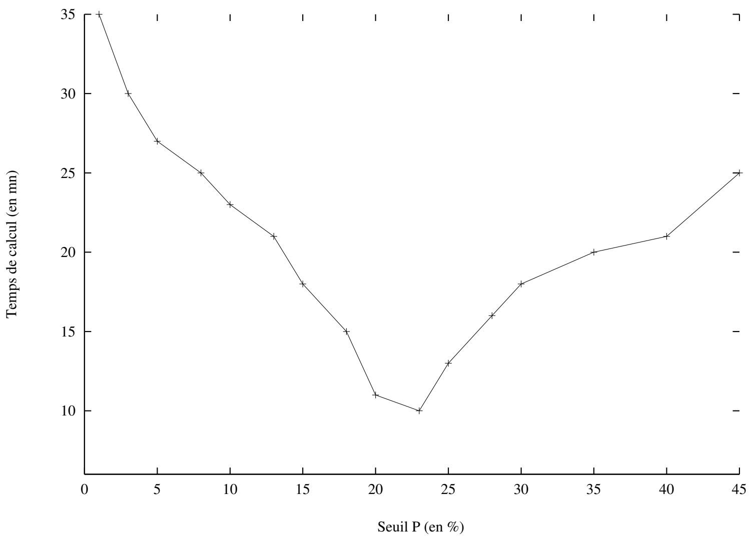


FIG. 6.1 – *Temps de calcul de l'algorithme d'extraction des séquences de mots pour différentes valeurs de p.*

6.4.2 Estimation du modèle de langage et calcul de la perplexité

Une des principales étapes de notre algorithme est le calcul de la perplexité. En effet, à chaque itération, nous devons calculer la valeur de la perplexité sur le corpus de test (C_T,W_T), avant et après la mise à jour des corpus et du vocabulaire avec les séquences de l'ensemble S_w. Pour tenir compte des liens syntaxiques entre les mots, nous avons utilisé les classes dans l'estimation de la perplexité. Ainsi, la perplexité du corpus, $W_T = w_1, \ldots, w_N$, étiqueté en, $C_T = c_1, \ldots c_N$, est définie par l'équation suivante :

$$PP = 2^{\frac{1}{N} log_2 P(w_1, \ldots, w_N)} \tag{6.4}$$

où N est le nombre de mots dans le corpus, et P(.) la vraisemblance de la suite de mots ou de séquences en argument. Dans le cas d'un modèle biclasses, utilisé dans notre algorithme, cette vraisemblance est définie comme suit :

$$P(w_1,\ldots,w_N) = p(w_1/c_1)p(c_1)\prod_{i=2}^{N} p(w_i/c_i)p(c_i/c_{i-1}) \qquad (6.5)$$

où c_i représente la classe qui correspond à l'unité (mot ou séquence) w_i.

Pour estimer les paramètres du modèle biclasses, nous utilisons le principe du maximum de vraisemblance sur les corpus d'apprentissage (W_A,C_A). Nous employons également le principe d'interpolation linéaire, pour mieux estimer les événements pas ou peu rencontrés dans le corpus d'apprentissage. Ainsi, nous interpolons linéairement le modèle biclasses avec le modèle uniclasses et zéroclasses (distribution uniforme). Les paramètres d'interpolation sont estimés à partir d'un corpus de développement (W_E,C_E), extrait du corpus d'apprentissage. Nous construisons ainsi un grand corpus pour l'apprentissage proprement dit (W_{A_1},C_{A_1}) et un autre corpus beaucoup plus petit pour l'évaluation des paramètres d'interpolation (W_E,C_E). La probabilité $p(c_j/c_i)$, utilisée dans l'équation 6.5, est par conséquent estimée comme suit :

$$p(c_j/c_i) = \alpha\frac{N(c_i,c_j)}{N(c_j)} + \beta\frac{N(c_i)}{T} + \gamma \qquad (6.6)$$

où $N(.)$ est le nombre d'occurrences de ce qui est en argument et T le nombre de classes syntaxiques de ce corpus, (W_{A_1},C_{A_1}). α, β et γ représentent les coefficients de pondération et leur somme vaut 1. Ces coefficients sont estimés sur le corpus de développement (W_E,C_E).

La probabilité d'appartenance d'une unité (mot ou séquence) w_i à une classe c_i, $p(w_i/c_i)$, utilisée dans l'équation 6.5, est définie par :

$$p(w_i/c_i) = \frac{N(w_i/c_i)}{N(c_i)} \qquad (6.7)$$

$N(w_i/c_i)$ est le nombre de fois où l'unité w_i est étiquetée par la classe c_i, dans le corpus d'apprentissage.

6.4.3 Accélération du calcul de la perplexité

L'évaluation de la perplexité nécessite de la puissance de calcul, car cette valeur doit être évaluée à chaque itération. Nous proposons donc une méthode permettant d'accélérer son traitement d'une manière très significative. En effet, nous ne sommes pas obligés de prendre en compte, à chaque itération, tous les mots du corpus pour calculer la perplexité. Il nous suffit d'estimer la vraisemblance des nouvelles séquences, ajoutées au vocabulaire, et des séquences voisines. Pour illustrer cette méthode, nous allons l'appliquer sur une phrase du corpus. Soit la phrase suivante W, étiquetée par C :

Elle	porte	des	chaussures	.
PPE	VEC	DES	NOM	POT

dont la perplexité est définie par l'équation 6.4 :

$$PP = 2^{\frac{1}{N} log_2 P(W)}$$

où N représente le nombre de mots dans la phrase (5 dans notre cas). La vraisemblance de la phrase W, $P(W)$, est définie comme suit :

$$\begin{aligned} P(W) &= p(elle/PPE)p(PPE) \times p(porte/VEC)p(VEC/PPE) \\ &\times p(des/DES)p(DES/VEC) \times p(chaussures/NOM)p(NOM/DES) \\ &\times p(./POT)p(POT/NOM). \end{aligned} \qquad (6.8)$$

Si nous supposons que seule la séquence "*des-chaussures*" étiquetée "*DES-NOM*" est à mettre à jour dans cette phrase, il nous suffit de remplacer la vraisemblance

$$p(des/DES)p(DES/VEC)p(chaussures/NOM)p(NOM/DES)p(POT/NOM)$$

par

$$p(des - chaussures/DES - NOM)p(DES - NOM/VEC)p(POT/DES - NOM)$$

dans le calcul de $P(W)$ (voir équation 6.8). Par conséquent, nous n'avons pas besoin d'estimer à nouveau la vraisemblance de toute la phrase. Comme certaines expressions ont été remplacées par des étiquettes, le nombre d'unités totales dans le corpus diminue. En revanche, comme la perplexité est estimée par rapport aux mots, nous gardons toujours constante la valeur de N utilisée dans le calcul de la perplexité (voir l'équation 6.4) [2]. Cette valeur est égale au nombre de mots dans le corpus.

6.4.4 Affinement des séquences

Notre algorithme utilise les séquences courtes pour générer des séquences de mots de taille plus élevée. Par conséquent, les séquences de mots de taille n (par exemple, "*quelle heure est il* ") sont construites à partir des séquences de taille plus petite m ($m < n$) qui sont incluses dans celles-ci (par exemple : "*quelle heure*"). Plusieurs de ces courtes séquences ne sont donc plus utiles une fois qu'elles ont participé à la génération des séquences plus longues. Nous supprimons ainsi du vocabulaire toutes les séquences qui réduisent la perplexité une fois remplacées par les mots qui les composent. Nous ne retenons que les séquences utilisées dans la génération d'une séquence plus longue. L'approche que nous utilisons pour affiner les séquences procède comme suit :

- Soit V le vocabulaire résultat de l'algorithme de génération de séquences (cf. §6.4.1).
- Extraire toutes les séquences de classes V_C qui ont généré les séquences de mots du vocabulaire V.
- Pour chaque séquence s_c de V_C, incluse dans une autre séquence plus longue de ce même V_C :
  1. définir le vocabulaire V', en prenant toutes les séquences de mots $\{s_m\}$, étiquetées par s_c et appartenant au vocabulaire V.
  2. construire le corpus (W', C'), en remplaçant dans le corpus W (respectivement, C), chaque séquence $s_m \in \{s_m\}$, (respectivement, la séquence de classes s_c) par les mots (respectivement, les classes de mots) qui la composent.
  3. si la perplexité diminue en utilisant le vocabulaire V' et le corpus (W',C') :
     - affecter le vocabulaire V' à V,
     - affecter le corpus (W',C') à (W,C),
  4. sinon, pour chaque séquence s_m de $\{s_m\}$:
     - définir le vocabulaire V', en enlevant la séquence s_m du vocabulaire V,
     - construire le corpus (W', C'), en remplaçant dans (W,C) la séquence de mots s_m (respectivement, la séquence de classes s_c correspondante) par les mots (respectivement, les classes de mots) qui la composent.
     - si la perplexité diminue en utilisant le vocabulaire V' et le corpus (W',C'), mettre à jour le vocabulaire V (en enlevant la séquence s_m ($V \longleftarrow V'$)) et affecter (W',C') au corpus (W, C),

– Résultat : le vocabulaire V.

Ainsi, à partir d'un vocabulaire de mots, après l'application des algorithmes cités ci-dessus, nous construisons un nouveau vocabulaire V. Ce nouveau vocabulaire, en plus des mots qui lui appartenaient initialement, contient toutes les séquences qui vont être utilisées comme unités au niveau de l'estimation des modèles de langage. Ces séquences participent également à la prédiction et à la sélection dans un système de RAP (voir chapitre 8).

6.4.5 Évaluation et résultats

Pour extraire les séquences, nous avons utilisé les 233 classes syntaxiques, définies au paragraphe 4.3, et un vocabulaire d'environ 20000 mots, V. Pour chaque mot de ce vocabulaire, nous possédons les classes syntaxiques auxquelles il peut appartenir. Les corpus que nous utilisons pour l'apprentissage, l'évaluation des paramètres d'interpolation et le test, sont ceux définis dans le paragraphe 4.6

Nous appliquons ensuite l'algorithme cité ci-dessus sur l'ensemble de ces données : nous obtenons les séquences que nous désirons insérer en tant qu'unités supplémentaires dans notre vocabulaire. Nous présentons, sur la courbe A de la figure 6.2, pour différentes valeurs de q (nombre maximal de mots dans une séquence), la convergence de l'algorithme d'extraction des séquences présentées ci-dessus. Cette courbe montre la perplexité donnée par l'ajout des séquences candidates de l'itération en cours au vocabulaire (sixième étape de l'algorithme présenté dans le sous-paragraphe 6.4.1). Pour une meilleure comparaison entre cet algorithme d'extraction de séquences et celui cité dans le paragraphe suivant (cf. §6.5), nous avons présenté la perplexité bigrammes. En effet, la perplexité de la courbe A et la courbe B de la figure 6.2 sont estimées sur le même corpus de test, par un modèle bigrammes utilisant les équations 6.10, 6.11 et 6.12.

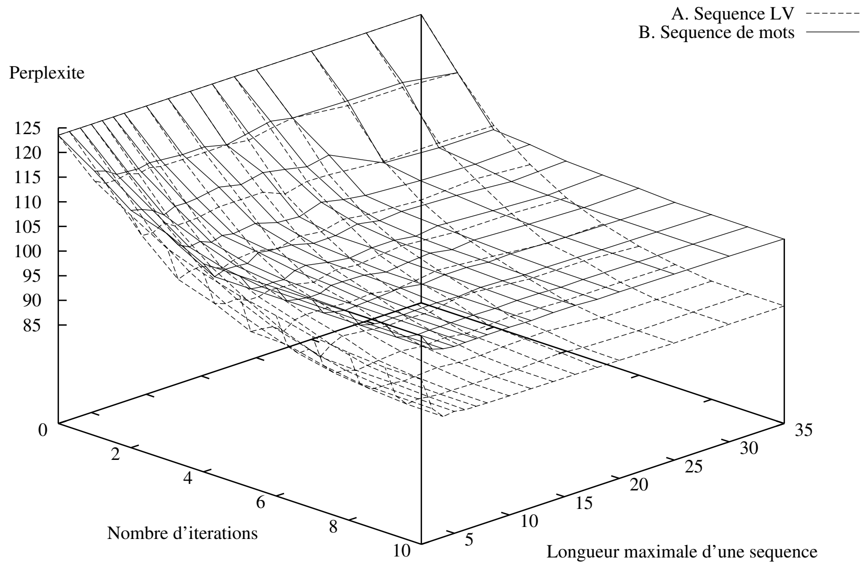


FIG. 6.2 – *Convergence des algorithmes d'extraction de séquences pour différentes valeurs de q (nombre maximal de mots dans une séquence).*

L'étude de la courbe A de la figure 6.2 montre qu'à partir d'une valeur de q égale à 4, la mesure de la perplexité converge. La valeur de la perplexité commence à 124.14 (perplexité bigrammes avec le vocabulaire initial) et converge vers une valeur qui est comprise entre 86.57 pour $q = 4$ et 85.63 pour $q = 35$. Nous nous limitons ainsi à une valeur de q égale à 4.

En analysant les séquences résultant de cet algorithme, nous avons remarqué que la majorité d'entre elles ont une structure syntaxique correcte. De plus, plusieurs de ces séquences peuvent être considérées comme des séquences sémantiques. Dans la suite, pour des raisons de clarté, nous nommons les séquences extraites par cet algorithme, séquences linguistiquement viables ou séquences LV. Nous présentons dans la table 6.1 un échantillon des classes extraites, ainsi qu'un échantillon de séquences correspondant à chacune d'entre elles.

D'autres séquences non conventionnelles de mots sont également retenues par cet algorithme, par exemple : "*a-de*", "*a-le*", "*cherche-les*", "*chez-les*", etc. Ces séquences sont extraites du fait de leur fréquence d'apparition. Certaines de ces séquences non conventionnelles, en revanche, reflètent le contenu du corpus. En effet, comme le corpus en notre possession provient des années 87-88, des séquences relatives à cette époque ont été prélevées (exemple : "*Balladur-avait*", "*l'-URSS*", "*la-RDA*", "*la-RFA*", etc.). Pour résoudre ce problème, nous pensons qu'il est utile d'effectuer une mise à jour du corpus avec des données traitant de sujets récents.

6.5 Une méthode d'extraction de séquences sans classes

Cette nouvelle approche n'utilise pas les classes syntaxiques et se fonde sur l'information mutuelle et la perplexité. Par conséquent, nous n'avons plus besoin du corpus de classes C. La différence avec l'algorithme proposé dans le paragraphe précédent se situe aux niveaux du calcul de la perplexité et du choix des séquences candidates à chaque itération. En effet, nous prenons à chaque itération les couples d'unités ayant une information mutuelle supérieure à un seuil T_J. Ce seuil, défini auparavant sur le corpus des classes (équation 6.3), est estimé sur le corpus de mots comme suit :

$$T_J = p \max_{w_1 \in V, w_2 \in V} J(w_1,w_2) \tag{6.9}$$

où $J(w_1,w_2)$ est la valeur de l'information mutuelle du couple w_1,w_2 dans le corpus de mots W (voir l'équation 6.1) et p le coefficient utile au calcul de T_J (cf. §6.4.1). Nous additionnons ensuite ces séquences candidates au vocabulaire de base, nous mettons à jour le corpus avec ces nouvelles séquences et nous calculons la valeur de la perplexité. Si cette valeur diminue, nous recommençons le même traitement. Sinon, nous procédons comme dans la huitième étape de l'algorithme précédent [202]. Les mêmes techniques que ci-dessus sont appliquées pour accélérer le calcul de la perplexité et affiner les séquences résultats (cf. §6.4).

La valeur de la perplexité est estimée sur un corpus de mots seulement, avec un modèle bigrammes. Ainsi, en gardant les mêmes notations qu'au paragraphe précédent, la vraisemblance de l'équation 6.5 devient :

$$P(w_1,\ldots,w_N) = p(w_1) \prod_{i=2}^{N} p(w_i/w_{i-1}). \tag{6.10}$$

où la probabilité $p(w_j/w_i)$ est estimée comme suit :

$$p(w_j/w_i) = \alpha_1 \frac{N(w_i,w_j)}{N(w_i)} + \beta_1 \frac{N(w_j)}{T} + \gamma_1. \tag{6.11}$$

ARD NOM :	l'administration, l'aéroport, la circulation, la civilisation, le chômage, le congrès, les capitaux, les fonctionnaires, ...
NOM DDE NOM :	chef de gouvernement, conditions de travail, conférence de presse, fin de semaine, hommes d' affaires, juge d' instruction, milliards de francs, pouvoir d' achat, ...
NEG ETR PAS :	n' était pas, n' est pas, ne sera pas, ne serait pas, ne soient pas, ne soit pas, ne sont pas.
CAR NOM :	cent ans, cinquante francs, deux jours, deux mille, un an, un exemple, un milliard, un mois, ...
ARI NOM DDE :	un an de, un état de, un gouvernement de, une réunion de, un langage d', un million de, une mission de, une politique de, ...
PPE PCR VEC :	il s' agit, il s' agissait, il se fait, il se trouve, je me sens, je me souviens, on se demande, on se trouve, ...
PCR ETR PPx :	s' est déclaré, s' est déroulé, s' est dit, s' est engagé, se sont déclarés, se sont déroulées, se sont mis, ...
ARD NOP :	l' Afrique, l' Amérique, l' Occident, la CEE, la France, le Canada, le Monde, ...
NOP NOP :	Arabie Saoudite, Edouard Balladur, George Bush, Jacques Chirac, Lionel Jospin, Los Angeles, Maison Blanche, Raymond Barre, ...

TAB. 6.1 – *Un échantillon des classes extraites et de séquences correspondantes.*

α_1, β_1 et γ_1 représentent les coefficients de pondération et somment à 1. Le même principe est utilisé pour l'estimation de la probabilité unigrammes :

$$p(w_i) = \alpha_2 \frac{N(w_i)}{T} + \beta_2 \tag{6.12}$$

où α_2 et β_2 sont des coefficients de pondération ($\alpha_2 + \beta_2 = 1$).

Nous présentons sur la courbe B de la figure 6.2 la convergence de l'algorithme d'extraction de séquences cité dans ce paragraphe. La courbe 6.2.B montre, pour différentes valeurs de q, la valeur de la perplexité donnée par l'ajout au vocabulaire des séquences de l'itération en cours.

La comparaison des deux courbes de la figure 6.2 montre que l'approche, fondée sur les classes syntaxiques, est plus performante. En effet, cette approche converge vers une valeur de perplexité beaucoup plus petite que celle vers laquelle converge l'algorithme de ce paragraphe. En utilisant cette nouvelle approche, il est à remarquer que nous obtenons beaucoup moins de séquences avec un temps de réponse très élevé. Ainsi, au vu des résultats, nous abandonnons cette approche en faveur de la méthode d'extraction des séquences se fondant sur les classes syntaxiques (cf. §6.4). Nous présentons dans la table 6.2 un échantillon de séquences obtenues par l'approche citée dans ce paragraphe.

adjoint des services, ah si, n' agit pas, d'action, de nos, est le, faire face, fin de, Jacques Chirac, mot d'ordre, Maison Blanche, un gouvernement, ville de, ...

TAB. 6.2 – *Un échantillon de séquences de mots.*

6.6 Les modèles n-SeqGrammes

Les modèles n-grammes s'avèrent performants pour tenir compte des contraintes locales de la langue. Cependant, ils montrent leurs limites si les contraintes apparaissent dans un historique lointain. Pour modéliser les contraintes qui existent dans un historique lointain avec les modèles n-grammes, il est nécessaire d'augmenter la valeur de n. Or, pour des valeurs de n peu élevées ($n > 3$), le nombre de suites de mots non rencontrées ainsi que le nombre de paramètres à estimer, deviennent très élevés.

Dans le but de mieux modéliser les contraintes qui existent dans un historique plus grand que celui utilisé par les modèles n-grammes ($n \leq 3$), nous avons construit les modèles n-SeqGrammes. La particularité de ces modèles est qu'ils combinent les avantages d'utilisation des modèles n-grammes avec ceux des séquences (cf. §6.4). En effet, en plus des mots, nous utilisons les séquences lors de l'estimation des modèles n-grammes ; nous bénéficions ainsi de la structure syntaxique et sémantique de ces séquences au niveau de la prédiction. De plus, étant donnée la longueur variable des séquences, la taille de l'historique que nous utilisons n'est plus constante. Ceci nous rapproche du comportement humain, qui se fonde généralement sur un historique variable pour définir le contexte et ainsi prédire ce qui suit.

En utilisant un modèle n-SeqGrammes, la vraisemblance de toutes suites de séquences (mots ou séquences de mots) $s_1^T = s_1, \ldots, s_T$, formant la suite de mots $w_1^N = w_1, \ldots, w_N$ où $N \geq T$, est définie comme suit :

$$P(s_1, \ldots, s_T) = p(s_1, \ldots, s_{n-1}) \prod_{i=n}^{T} p(s_i / s_{i-n+1}, \ldots, s_{i-1}) \tag{6.13}$$

où $p(s_i/s_{i-n+1},\ldots,s_{i-1})$ est la probabilité conditionnelle de la séquence s_i sachant la suite des séquences $s_{i-n+1},\ldots,s_{i-1}$. L'ensemble de ces probabilités conditionnelles définit les paramètres du modèle n-SeqGrammes. La probabilité $p(s_1,\ldots,s_{n-1})$ des $n-1$ premières séquences est définie par les paramètres des modèles m-SeqGrammes, où $m < n$:

$$p(s_1,\ldots s_{n-1}) = p(s_1) \times p(s_2/s_1) \times \cdots \times p(s_{n-1}/s_1,\ldots s_{n-2}). \tag{6.14}$$

6.6.1 Estimation des paramètres

Les paramètres du modèle n-SeqGrammes sont estimés suivant la théorie de validation croisée (cf. §3.6), en utilisant la formule de Good-Turing. L'avantage de cette méthode est son efficacité à réduire le biais provoqué par les suites de séquences non rencontrées dans le corpus [99]. Nous utilisons la méthode de Katz [14] pour estimer la probabilité des suites de séquences non rencontrées.

La probabilité de la suite de séquences $s_1^n = s_1,\ldots,s_n$, estimée à l'aide de la fréquence empirique, est définie comme suit :

$$P_{ML}(s_1^n) = \frac{N(s_1^n)}{T} \tag{6.15}$$

où T est la taille du corpus et $N(.)$ le nombre d'occurrences de la suite de séquences en argument, comme mentionné précédemment. Dans le cas d'une estimation avec la méthode de Good-Turing (validation croisée), la probabilité de cette même séquence s_1^n est définie par la formule suivante :

$$P_T(s_1^n) = \frac{N^*(s_1^n)}{T} \tag{6.16}$$

où $N^*(x)$ est définie par l'équation suivante :

$$N^*(x) = (N(x)+1)\frac{t_{N(x)+1}}{t_{N(x)}} \tag{6.17}$$

t_r indique le nombre de n-SeqGrammes (suite de n séquences) qui apparaissent exactement r fois dans le corpus. Par conséquent, la taille du corpus T peut être définie par la formule suivante :

$$T = \sum_r r t_r. \tag{6.18}$$

En utilisant la méthode de Good-Turing, la somme des probabilités de toutes les n-SeqGrammes, visibles dans le corpus et se terminant par une séquence s_n donnée, est définie par :

$$\sum_{s_1^n : N(s_1^n)>0} P_T(s_1^n) = 1 - \frac{t_1}{T}. \tag{6.19}$$

Dans le cas de l'utilisation de la fréquence empirique, cette somme vaut 1. L'équation 6.19, avec la méthode de Good-Turing, permet ainsi d'évaluer la probabilité des n-SeqGrammes qui se terminent par la séquence s_n et qui n'apparaissent pas dans le corpus :

$$\sum_{s_1^n : N(s_1^n)=0} P_T(s_1^n) = \frac{t_1}{T}. \tag{6.20}$$

14. Backoff method

D'autre part, la différence entre la somme des probabilités de toutes ces n-SeqGrammes (se terminant par s_n et appartenant au corpus), utilisant la fréquence empirique et la théorie de Good-Turing, donne :

$$\sum_{s_1^n:N(s_1^n)>0}(P_{ML}(s_1^n)-P_T(s_1^n))=\sum_{s_1^n:N(s_1^n)>0}\delta_{N(s_1^n)}=\sum_{r>0}t_r(1-d_r)\frac{r}{T}=\frac{t_1}{N} \tag{6.21}$$

où le terme δ_i est définie par l'équation suivante :

$$\delta_i=\frac{i}{T}-\frac{i^*}{T}=(1-d_i)\frac{i}{T} \tag{6.22}$$

et où d_i représente le coefficient de *discounting* de Good-Turing. Ainsi, nous déduisons l'équation suivante :

$$\sum_{s_1^n:N(s_1^n)>0}\delta_{N(s_1^n)}=\sum_{s_1^n:N(s_1^n)=0}P_T(s_1^n) \tag{6.23}$$

où $\delta_{N(s_1^n)}$ est interprété comme la "contribution" de la suite de séquences s_1^n à la valeur de probabilité des n-SeqGrammes qui se terminent par s_n et qui ne sont pas rencontrées dans le corpus [99].

Pour estimer la probabilité conditionnelle $p(s_n/s_1^{n-1})$, nous utilisons une nouvelle entité δ_i^{cond}, analogue à δ_i :

$$\delta_{N(s_1^n)}^{cond}=(1-d_{N(s_1^n)})\frac{N(s_1^n)}{N(s_1^{n-1})}. \tag{6.24}$$

Par conséquent, dans le cas où la suite de séquences à estimer apparaît dans le corpus ($N(s_1^{n-1})>0$), la probabilité conditionnelle $p(s_n/s_1^{n-1})$ du modèle n-SeqGrammes est approximée comme suit :

$$p(s_n/s_1^{n-1})=\tilde{p}(s_n/s_1^{n-1})=d_{N(s_1^n)}\frac{N(s_1^n)}{N(s_1^{n-1})} \tag{6.25}$$

où $\tilde{p}(.)$ représente la probabilité estimée par la méthode de Good-Turing. Nous approximons alors la somme des probabilités conditionnelles, de toutes les séquences s_1^{n-1} qui n'ont jamais précédé s_n ($N(s_1^n)=0$), par la formule suivante :

$$\begin{aligned}\tilde{\beta}(s_1^{n-1}) &= \sum_{s_n:N(s_1^n)>0}\delta_{N(s_1^n)}^{cond}\\ &= 1-\sum_{s_n:N(s_1^n)>0}\tilde{p}(s_n/s_1^{n-1}).\end{aligned} \tag{6.26}$$

La valeur de $\tilde{\beta}(s_1^{n-1})$ est répartie sur toutes les suites de n séquences (n-SeqGrammes) se terminant par s_n, dont l'occurrence est nul ($N(s_1^n)=0$). Cette répartition nous permet ainsi de calculer la probabilité des n-SeqGrammes qui se terminent par s_n et qui ne sont pas rencontrées dans le corpus. Pour estimer la probabilité de ces n-SeqGrammes nous utilisons un modèle d'ordre inférieur : le modèle $(n-1)$-SeqGrammes. Ce modèle, défini de la même manière que le modèle n-SeqGrammes, agit sur un historique de séquences plus restreint de taille $n-1$. Par conséquent, nous utilisons la probabilité $p(s_n/s_2^{n-1})$ pour calculer la probabilité conditionnelle $p(s_n/s_1^{n-1})$ de la suite de séquences s_1^n non rencontrée dans le corpus. La probabilité conditionnelle $p(s_n/s_1^{n-1})$, des n-SeqGrammes non vue ($N(s_1^n)=0$), est approximée comme suit :

$$p(s_n/s_1^{n-1}) = \alpha p(s_n/s_2^{n-1}) \tag{6.27}$$

où α est une constante de normalisation définie par :

$$\begin{aligned} \alpha &= \alpha(s_1^{n-1}) = \frac{\tilde{\beta}(s_1^{n-1})}{\sum_{s_n:N(s_1^n)=0} p(s_n/s_2^{n-1})} \\ &= \frac{1-\sum_{s_n:N(s_1^n)>0}\tilde{p}(s_n/s_1^{n-1})}{1-\sum_{s_n:N(s_1^n)>0}\tilde{p}(s_n/s_2^{n-1})}. \end{aligned} \tag{6.28}$$

Dans le cas où le nombre d'occurrences du $(n-1)$-SeqGramme s_1^{n-1} est nul ($N(s_1^{n-1})=0$), la probabilité du n-SeqGramme s_1^n devient identique à celle du $(n-1)$-SeqGramme s_2^n :

$$p(s_n/s_1^{n-1}) = p(s_n/s_2^{n-1}). \tag{6.29}$$

Ainsi, les valeurs de $\tilde{p}$ et de $\tilde{\beta}$, dans le cas où le nombre d'occurrences du $(n-1)$-SeqGramme s_1^{n-1} est nul ($N(s_1^{n-1})=0$), sont définies comme suit :

$$\tilde{p}(s_n/s_1^{n-1}) = 0$$

et

$$\tilde{\beta}(s_1^{n-1}) = 1.$$

En combinant les résultats des équations 6.25, 6.27 et 6.29, nous obtenons l'expression récursive de la probabilité conditionnelle du modèle n-SeqGrammes :

$$p(s_n/s_1^{n-1}) = \tilde{p}(s_n/s_1^{n-1}) + \theta(\tilde{p}(s_n/s_1^{n-1})).\alpha(s_1^{n-1})p(s_n/s_2^{n-1}) \tag{6.30}$$

où le terme $\theta(x)$ est définie comme suit :

$$\theta(x) = \begin{cases} 1, & \text{si } x=0 \\ 0, & \text{sinon.} \end{cases} \tag{6.31}$$

La version que nous utilisons pour l'estimation du modèle n-SeqGrammes est légèrement différente. En effet, nous gardons la valeur de $\frac{t_1}{T}$ comme la vraisemblance des n-SeqGrammes invisibles et nous n'appliquons la discontinuité de Good-Turing que sur les n-SeqGrammes d'occurrences inférieures à une certaine valeur k. Nous supposons ainsi fiable les n-SeqGrammes dont le nombre d'occurrences est supérieur à k. Les coefficients de discontinuité de Good-Turing d_r sont donc définis comme suit :

$$d_r = 1, \qquad \text{si } r > k. \tag{6.32}$$

Il nous reste ainsi à évaluer la valeur de ces coefficients pour des valeur de r inférieur, ou égale, à k ($r \leq k$). Rappelons que E_{s_n} représente l'ensemble de toutes les n-SeqGrammes se terminant par la séquence s_n. Nous remarquons que, même avec cette contrainte imposée par k, la somme des probabilités des n-SeqGrammes de E_{s_n} n'appartenant pas au corpus est toujours égale à la somme des "contributions" de ceux de E_{s_n} visibles dans ce même corpus :

$$\sum_{s_n:N(s_1^n)>0} \delta_{N(s_1^n)} = \sum_{1\leq r\leq k} t_r(1-d_r)\frac{r}{T} = \frac{t_1}{T}. \tag{6.33}$$

Ceci rend analogue les formules 6.33 et 6.21. En regardant la solution de l'équation 6.33, nous obtenons ainsi un ajustement de d_r de la forme :

$$(1 - d_r) = \mu \left(1 - \frac{r^*}{r}\right) \tag{6.34}$$

où r^* est estimé par la formule 6.17 et μ est une constante. Une solution unique existe à cette équation et elle est définie comme suit :

$$d_r = \frac{\frac{r^*}{r} - \frac{(k+1)t_{k+1}}{t_1}}{1 - \frac{(k+1)t_{k+1}}{t_1}}, \quad \text{pour } 1 \leq r \leq k. \tag{6.35}$$

La valeur de k, dans la formule 6.35, est déterminée expérimentalement sur un corpus. S.M. Katz montre qu'une valeur de k aux alentours de 6 est un très bon choix [99]. Ainsi, pour le modèle n-SeqGrammes, nous avons choisi la valeur 7. De plus, étant donné le bruit (par exemple, erreurs de saisie) qui peut apparaître dans un corpus, nous supposons invisibles les n-SeqGrammes rencontrés une seule fois ; en effet, il n'y a pas une grande différence entre ces n-SeqGrammes et ceux non visibles dans le corpus. Cette opération possède également l'avantage de réduire considérablement le nombre de paramètres à estimer.

6.6.2 Les modèles construits

Nous avons construit les modèles n-SeqGrammes pour $n \leq 3$, en se fondant sur les équations 6.28, 6.30 et 6.35. Ces modèles ont été évalués sur un corpus d'environ 50 millions de mots (cf. §4.6). Pour avoir ce corpus étiqueté avec les séquences LV, nous utilisons le corpus résultat de l'algorithme d'extraction des séquences, présenté dans le paragraphe 6.4. Cet algorithme fournit, en plus des séquences, le corpus de mots étiqueté avec les séquences résultats ; l'évaluation de ce modèle, en terme de perplexité, est présentée au paragraphe 6.10.

6.7 Les modèles n-SeqClasses

Les modèles n-SeqClasses, que nous présentons ici, se fondent sur la notion de classes et de séquences. En effet, pour estimer la vraisemblance d'une suite de mots, ces modèles se fondent sur les séquences LV, leurs classes et les classes syntaxiques (cf. §4.3). Les classes syntaxiques sont construites manuellement et regroupent les mots ayant un même comportement syntaxique. Les classes des séquences LV, en revanche, sont construites automatiquement à partir des classes syntaxiques et de l'algorithme d'extraction des séquences (cf. §6.4). En étudiant l'allure de ces classes, nous remarquons que la majorité d'entre elles ont une structure syntaxique connue de la langue, comme le groupe nominal, le groupe verbal, etc. L'utilisation de ces classes peut engendrer ainsi l'ajout de contraintes syntaxiques et même syntagmatiques supplémentaires, permettant d'améliorer les performances des modèles de langage. De plus, le regroupement des séquences et des mots dans des classes, peut réduire sensiblement l'espace de paramètres par rapport aux modèles n-SeqGrammes, ce qui pourrait améliorer leur fiabilité d'évaluation. Ce regroupement a également le notable avantage de réduire le problème de manque de données. En effet, étant donné que le nombre de classes est généralement inférieur au nombre de séquences dans le vocabulaire, il est alors plus réaliste de rencontrer beaucoup plus souvent les suites de classes que les suites de séquences de mots correspondantes.

En utilisant un modèle n-SeqClasses, la vraisemblance de la suite de séquences $s_1^T = s_1, \ldots, s_T$ est définie de la même façon qu'un modèle n-SeqGrammes :

$$P(s_1, \ldots, s_T) = p(s_1, \ldots, s_{n-1}) \prod_{i=n}^{T} p(s_i / s_{i-n+1}, \ldots, s_{i-1}). \tag{6.36}$$

La seule différence réside dans la manière de définir la probabilité conditionnelle $p(s_i/s_{i-n+1}, \ldots, s_{i-1})$. En effet, dans les modèles n-SeqClasses, nous nous appuyons sur une information supplémentaire : les classes. Ainsi, étant donné qu'une séquence peut appartenir à plusieurs classes, nous procédons comme suit pour estimer la probabilité conditionnelle de la séquence s_n sachant la suite de séquences $s_1, \ldots, s_{n-1}$:

$$\begin{aligned}
p(s_n/s_1, \ldots, s_{n-1}) &= \sum_{c(s_n) \in C_{s_n}} p(s_n/c(s_n))\ p(c(s_n)/s_1, \ldots, s_{n-1}) \\
&= \sum_{c(s_n) \in C_{s_n}} p(s_n/c(s_n)) \\
&\times \left[\sum_{c(s_{n-1}) \in C_{s_{n-1}}} p(c(s_{n-1})/s_{n-1})\ p(c(s_n)/s_1, \ldots, s_{n-2}, c(s_{n-1})) \right] \\
&= \sum_{c(s_n) \in C_{s_n}} p(s_n/c(s_n)) \\
&\times \Bigg(\sum_{c(s_{n-1}) \in C_{s_{n-1}}} p(c(s_{n-1})/s_{n-1}) \cdots \sum_{c(s_1) \in C_{s_1}} p(c(s_1)/s_1) \\
&\times [p(c(s_n)/c(s_1) \ldots c(s_{n-1}))] \Bigg)
\end{aligned} \tag{6.37}$$

où C_{s_i} indique l'ensemble de classes auxquelles la séquence s_i peut appartenir, et $c(s_i)$ une classe possible de la séquence s_i appartenant à l'ensemble C_{s_i}. Ainsi, dans le cas d'un modèle 2-SeqClasses, l'équation 6.37 devient :

$$\begin{aligned}
p(s_n/s_{n-1}) &= \sum_{c(s_n) \in C_{s_n}} p(s_n/c(s_n))\ p(c(s_n)/s_{n-1}) \\
&= \sum_{c(s_n) \in C_{s_n}} p(s_n/c(s_n)) \sum_{c(s_{n-1}) \in C_{s_{n-1}}} p(c(s_{n-1})/s_{n-1})\ p(c(s_n)/c(s_{n-1})).
\end{aligned} \tag{6.38}$$

Les paramètres d'un modèle n-SeqClasses se limitent donc aux probabilités conditionnelles $p(s_i/c(s_i))$, $p(c(s_i)/s_i)$ et $p(c(s_n)/c(s_1) \ldots c(s_{n-1}))$. Si nous notons par $N(s_i/c(s_i))$ le nombre de fois où la séquence s_i est étiquetée par la classe $c(s_i)$, la probabilité d'appartenance d'une séquence s_i à une classe $c(s_i)$ est estimée par l'équation suivante :

$$p(s_i/c(s_i)) = \frac{N(s_i/c(s_i))}{N(c(s_i))} \tag{6.39}$$

où $N(.)$ indique le nombre d'occurrences de ce qui est en argument. La probabilité pour que la classe $c(s_i)$ corresponde au mot s_i est approximée comme suit :

$$p(c(s_i)/s_i) = \frac{N(s_i/c(s_i))}{N(s_i)}. \tag{6.40}$$

Toutes les occurrences, $N(.)$, citées ci-dessus dans les équations 6.39 et 6.40, sont calculées sur le même corpus d'apprentissage. Pour estimer la probabilité conditionnelle $p(c(s_n)/c(s_1)\dots c(s_{n-1}))$, nous utilisons la méthode définie dans le paragraphe 6.6. Nous nous fondons sur la théorie de la validation croisée et la formule de Good-Turing, pour approximer les probabilités conditionnelles $p(c(s_n)/c(s_1)\dots c(s_{n-1}))$. Nous utilisons également la méthode de Katz pour estimer les probabilités conditionnelles des événements non rencontrés dans le corpus d'apprentissage.

Le corpus utilisé pour l'apprentissage et l'évaluation des modèles n-SeqClasses ($n \leq 3$) contient environ 50 million de mots (cf. §4.6). Pour estimer les paramètres de ces modèles, nous avons besoin de définir les classes de chaque séquence (mot ou séquence LV) du vocabulaire. Nous disposons d'avance des classes syntaxiques de chaque mot. En revanche, les classes des séquences LV sont extraites automatiquement à partir des corpus (W,C), résultats de l'algorithme d'extraction des séquences (cf. §6.4). En effet, cet algorithme fournit, en plus des séquences LV, le corpus de mots ainsi que le corpus de classes correspondant. Ces corpus sont étiquetés avec les séquences résultats. L'évaluation de ces modèles, en terme de perplexité, est présentée dans le paragraphe 6.10.

6.8 Le modèle SeqCache

L'utilisation du modèle *cache* (cf. §3.4.7) a surtout pour objectif d'adapter les modèles de langage au sujet en cours [120, 118]. En effet, quand on traite d'un sujet bien particulier, on remarque que plusieurs expressions ou mots clés identifiant ce sujet se répètent très souvent. Ainsi, la fréquence d'apparition de ces expressions ou de ces mots dans l'historique influe sur la prédiction. Or, le modèle *cache* ne se fonde que sur l'occurrence des mots dans l'historique. Par conséquent, ce modèle ne peut tenir compte des expressions clés, formées de suites de mots, qui caractérisent le sujet en cours. En effet, dans le cas où un utilisateur place souvent l'expression "*n'est-ce pas*" dans ses phrases, le modèle *cache* qui ne se fonde que sur les mots ne pourra pas tenir compte en même temps de ces quatre mots, pendant la prédiction.

Ce modèle, que nous nommons *SeqCache*, se fonde sur la fréquence d'apparition des séquences dans le *cache*. Les séquences utilisées par le modèle *SeqCache* peuvent être des mots ou des séquences LV. Par conséquent, si nous fixons le *cache* aux M derniers mots de l'historique, la probabilité de la séquence s_i sachant un historique $s_{i-M}^{i-1} = s_{i-M},\dots,s_{i-1}$ est estimée par la formule suivante :

$$P_{SeqCache}(s_i/s_{i-M}^{i-1}) = \frac{1}{M}\sum_{m=1}^{M}\delta(s_n/s_{n-m}) \qquad (6.41)$$

où $\delta(w/v) = 1$ si et seulement si $w = v$.

Pour mieux modéliser les contraintes qui existent dans le *cache* s_{i-M}^{i-1}, nous avons combiné le modèle *SeqCache* avec le modèle n-SeqGrammes, pour $n \leq 3$. Ainsi, la probabilité de la séquence s_i sachant la suite de séquences s_{i-M}^{i-1} est approximée comme suit :

$$P(s_i/s_{i-M}^{i-1}) = (1-\lambda)P_{SeqGrammes}(s_i/s_{i-n+1},\dots s_{i-1}) + \lambda P_{SeqCache}(s_i/s_{i-M}^{i-1}) \qquad (6.42)$$

où λ représente le paramètre d'interpolation et où $M > n$. Pour définir la taille du *cache*, nous avons effectué plusieurs expérimentations avec un historique variant entre 30 et 80 mots. Les performances, calculées en terme de perplexité, du modèle combinant *SeqCache* et n-SeqGrammes ont convergé avec un cache regroupant les 50 derniers mots. Nous avons donc choisi un historique de 50 mots pour l'évaluation et la comparaison de ce modèle avec les autres (cf. §6.10).

6.9 Le modèle SeqTrigger

L'objectif de base du modèle *SeqTrigger* est de tenir compte des informations significatives, introduites par des séquences de mots, existant dans un historique lointain. Le modèle utilisant les *triggers* présenté au paragraphe 3.4.7 se fonde sur des couples de mots ayant une forte corrélation entre eux [191, 143, 167] ; l'apparition d'un mot w_i, faisant partie d'un *trigger* $(w_i \rightarrow w_j)$, augmente le taux de prédiction du mot w_j. Ces modèles, combinés aux modèles n-grammes, ont amélioré les performances des modèles de langage. Néanmoins, dans la langue, nous remarquons que c'est une suite de mots, ayant une structure bien particulière et appartenant à un historique lointain, qui intervient lors de la prédiction. Nous construisons ainsi un nouveau modèle de langage que nous nommons le modèle SeqTrigger. Ce modèle se fonde sur des séquences de mots ainsi que sur le principe utilisé par le modèle *trigger*. En effet, le modèle *SeqTrigger* commence par la définition des couples de séquences ayant une forte corrélation entre eux. Ces séquences peuvent être soit des mots soit des séquences LV. Ainsi, l'apparition d'une séquence s_i, appartenant à un couple de séquences fortement corrélées $(s_i \rightarrow s_j)$, augmente le taux de prédiction de la séquence s_j. Ces couples de séquences sont calculés à partir d'un corpus d'apprentissage, en se fondant sur l'information mutuelle comme critère d'optimalité. Nous prenons donc tous les couples de séquences ayant une forte information mutuelle. En s'appuyant sur un historique h_M de M séquences, l'information mutuelle du couple de séquences (s_i,s_j) est définie par :

$$J(s_i,s_j) = log\left[\frac{N_{h_M}(s_i,s_j).T}{N(s_i).N(s_j)}\right] \tag{6.43}$$

où $N(s_i)$ est le nombre d'occurrences de la séquence s_i dans le corpus, et T la taille du corpus ; le terme $N_{h_M}(s_i,s_j)$ définit le nombre de fois où l'en rencontre s_i, sachant que s_j apparaît dans les M dernières séquences précédant s_i.

La vraisemblance de la séquence s_i sachant l'historique $s_{i-M}^{i-1} = s_{i-M},\ldots,s_{i-1}$ est estimée comme suit :

$$P_{SeqTrigger}(s_i/s_{i-M},\ldots,s_{i-1}) = \frac{1}{M}\sum_{m=1}^{M}\alpha(s_i/s_{i-m}) \tag{6.44}$$

où $\alpha(b/a)$ est définie par :

$$\alpha(b/a) = \frac{q(b/a)}{\sum_{b'} q(b'/a)}. \tag{6.45}$$

Pour une séquence s_i à prédire et pour un historique s_{i-M}^{i-1}, le paramètre $q(b/a)$ est estimé par la méthode de maximum de vraisemblance :

$$q(b/a) = \frac{F(a,b)}{F(a,b) + F(a,\bar{b})} \tag{6.46}$$

avec :

$$\begin{aligned}
F(a,b) &= \sum_{a \in s_{i-M}^{i-1}, b = s_i} 1 \\
F(a,\bar{b}) &= \sum_{a \in s_{i-M}^{i-1}, b \neq s_i} 1 \\
F(\bar{a},b) &= \sum_{a \notin s_{i-M}^{i-1}, b = s_i} 1 \\
F(\bar{a},\bar{b}) &= \sum_{a \notin s_{i-M}^{i-1}, b \neq s_i} 1 \qquad (6.47)
\end{aligned}$$

Pour extraire le maximum d'informations de l'historique s_{i-M}^{i-1}, nous avons combiné le modèle *SeqTrigger*, le modèle *SeqCache* (cf. §6.8) et un modèle n-SeqGrammes (cf. §6.6).

Un modèle n-SeqGrammes, approximé sur un corpus traitant de l'économie, se comporte généralement de la même manière, que l'on parle d'exploitation agricole ou de sport. Pourtant, nul doute que la séquence "pomme de terre" a plus de chance d'être prédite si le premier sujet est abordé. En combinant les modèles, la probabilité d'apparition de la séquence "pomme de terre" va donc varier en fonction du contexte. Pour combiner le modèle *SeqTrigger*, le modèle *SeqCache* et un modèle n-SeqGrammes, nous avons choisi la méthode d'interpolation linéaire. Ainsi, la probabilité de la séquence s_i sachant l'historique s_{i-M}^{i-1}, est estimée comme suit :

$$\begin{aligned}
P(s_i/s_{i-M}^{i-1}) &= \lambda_1 P_{SeqGrammes}(s_i/s_{i-n+1}, \ldots s_{i-1}) + \\
&\quad \lambda_2 P_{SeqCache}(s_i/s_{i-M}^{i-1}) + \lambda_3 P_{SeqTrigger}(s_i/s_{i-M}^{i-1}) \qquad (6.48)
\end{aligned}$$

où $M > n$ et où λ_i représente un paramètre d'interpolation ($\lambda_1 + \lambda_2 + \lambda_3 = 1$). En pratique, nous nous contentons d'une valeur de M égale à 50.

Pour extraire les couples de séquences ($s_i \rightarrow s_j$), nous employons le même corpus utilisé pour l'apprentissage des paramètres des modèles n-SeqGrammes. De plus, avec l'objectif de ne prendre en compte que les expressions et mots clés qui peuvent vraiment intervenir lors de la prédiction, nous ne prenons pas en considération les mots outils de la langue (le, la, de, ...) lors de l'extraction des couples ($s_i \rightarrow s_j$). Pour un vocabulaire de V séquences, nous avons V^2 couples possibles ; or ces couples de séquences ne sont pas tous utiles pour la prédiction. Ainsi, plusieurs expérimentations ayant pour objectif de déterminer le nombre de couples de séquences suffisant K ont été effectuées. Nous présentons sur la figure 6.3, pour différentes valeurs de K, la mesure de la perplexité donnée par une interpolation linéaire entre le modèle 3-SeqGrammes et le modèle *SeqTrigger* (la courbe A). Nous donnons également sur cette figure, pour les mêmes valeurs de K, la valeur de la perplexité donnée par une interpolation linéaire entre le modèle 3-SeqGrammes, le modèle *SeqCache* et le modèle *SeqTrigger* (la courbe B).

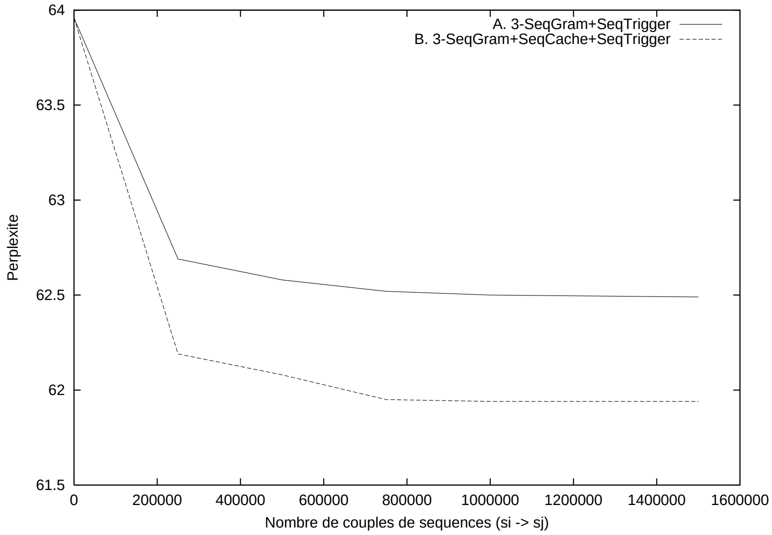


FIG. 6.3 – *Performances en terme de perplexité pour différents nombres de triggers retenus.*

Les résultats obtenus montrent que les 750000 meilleurs couples $(s_i \rightarrow s_j)$ sont largement suffisants pour atteindre les meilleures performances au niveau perplexité. Nous nous sommes ainsi limités, lors de l'estimation des vraisemblances, aux 750000 meilleures couples de séquences ayant une fréquence d'apparition supérieure à 7 dans le corpus. En regardant ces couples $(s_i \rightarrow s_j)$, pour différentes séquences s_j du vocabulaire, nous remarquons que ceux de la forme $(s_i \rightarrow s_i)$ apparaissent souvent dans les premiers rangs. Nous présentons dans la table 6.3 un échantillon des couples de séquences trouvés $(s_i \rightarrow s_j)$.

président directeur → société,	Fahd → Arabie Saoudite,
sous préfet → sous préfet,	couvre feu → a été,
millions de barils → cours,	le gouvernement → Jacques Chirac,
cohabitation → Chirac,	Montparnasse → Paris,
Françaises → Français,	observation militaire → programme,
la Syrie → Damas,	RPR → gaullisme,
Matignon → le premier ministre,	intersyndicale → CFDT,
symphonie → orchestre,	les cheminots → grève, ...

TAB. 6.3 – *Un échantillon de couple de séquences fortement corrélées* $(s_i \rightarrow s_j)$.

6.10 Évaluation

Dans ce paragraphe, nous allons évaluer, en terme de perplexité, tous les modèles utilisant les séquences, présentées dans ce chapitre. Nous allons également comparer ces modèles entre eux et avec leur équivalent n'utilisant que les mots. Pour ce faire, nous avons utilisé les mêmes corpus d'apprentissage, de développement et de test (cf. §4.6). Le vocabulaire de base employé contient environ 20000 mots, auxquels nous avons ajouté l'ensemble des séquences LV. Ces séquences, au nombre de 4400 environ, ont été extraites en utilisant l'algorithme défini au paragraphe 6.4.

Nous présentons, dans la table 6.4, les valeurs de la perplexité des différents modèles cités dans ce chapitre : des modèles n-grammes, des modèles n-classes, des modèles n-grammes avec *cache*, des modèles n-grammes avec *trigger* et des modèles n-grammes avec *cache* et *trigger* (cf. §3.4.7), pour $n \leq 3$. Les modèles n-classes, présentés dans cette table, sont estimés de la même façon que les modèles n-SeqClasses (cf. §6.7), avec la différence de n'utiliser que les mots sans séquences.

	3-grammes	3-SeqGrammes
Perplexité	74.65	63.96
	3-classes	3-SeqClasses
Perplexité	84.18	73.80
	3-grammes+cache	3-SeqGrammes+SeqCache
Perplexité	73.37	61.31
	3-grammes+Trigger	3-SeqGrammes+SeqTrigger
Perplexité	73.53	61.52
	3-grammes+cache+Trigger	3-SeqGrammes+SeqCache+SeqTrigger
Perplexité	72.69	60.95

	2-grammes	2-SeqGrammes
Perplexité	121.53	83.63
	2-classes	2-SeqClasses
Perplexité	135.11	89.12
	2-grammes+cache	2-SeqGrammes+SeqCache
Perplexité	118.84	80.98
	2-grammes+Trigger	2-SeqGrammes+SeqTrigger
Perplexité	119.02	81.47
	2-grammes+cache+Trigger	2-SeqGrammes+SeqCache+SeqTrigger
Perplexité	117.53	80.00

	1-grammes	1-SeqGrammes
Perplexité	409.07	205.26
	1-classes	1-SeqClasses
Perplexité	400.13	223.71
	1-grammes+cache	1-SeqGrammes+SeqCache
Perplexité	393.32	196.05
	1-grammes+Trigger	1-SeqGrammes+SeqTrigger
Perplexité	396.35	197.19
	1-grammes+cache+Trigger	1-SeqGrammes+SeqCache+SeqTrigger
Perplexité	392.16	195.73

TAB. 6.4 – *Evaluation en terme de perplexité de plusieurs variantes de modèles de langage.*

Les résultats obtenus montrent que l'utilisation des séquences permet d'améliorer considérablement les performances des modèles de langage. En effet, les modèles de langage utilisant les séquences (63.96 pour le 3-SeqGrammes) sont toujours meilleurs, en terme de perplexité, que leurs équivalents n'utilisant que les mots (74.65 pour le 3-grammes).

En revanche, nous remarquons que l'emploi du *cache* et du *trigger* n'apporte pas beaucoup aux modèles de base qui les utilisent (une perplexité de 61.95 pour un modèle utilisant une interpolation linéaire entre le 3-SeqGrammes, le *SeqCache* et le *SeqTrigger*). Nous pensons que ceci est dû surtout à la nature généraliste de notre corpus. Pour mieux évaluer les performances des modèles *SeqCache* et *SeqTrigger*, nous pensons qu'il est nécessaire de les appliquer sur des corpus traitant des thèmes spécifiques. En effet, dans la littérature les modèles utilisant les *triggers* se sont avérés utiles pour le traitement de thèmes spécifiques (cf. §6.8 et 6.9) [191, 118].

L'objectif d'un modèle n-SeqClasses est de mieux prendre en compte la structure syntaxique sous-jacente, introduite par les classes. Ce modèle pourrait également mieux prendre en compte le problème de manque de données. Néanmoins, les résultats obtenus ne sont pas encourageants, comparativement à un modèle n-SeqGrammes. Pour améliorer les performances des modèles n-SeqClasses et n-classes, nous pensons qu'il serait intéressant d'utiliser une autre méthode de classification plus performante.

Nous avons réalisé une autre expérience qui consiste à utiliser l'interpolation linéaire pour combiner les modèles de langage utilisant les mots avec ceux utilisant les classes. Nous avons ainsi construit deux nouveaux modèles :

– le premier est une combinaison linéaire entre un modèle n-grammes et un modèle n-classes :

$$a \text{ n-grammes} + (1-a) \text{ n-classes} \tag{6.49}$$

où $0 \leq a \leq 1$;

– le deuxième modèle représente des modèles n-SeqGrammes et n-SeqClasses interpolés :

$$a \text{ n-SeqGrammes} + (1-a) \text{ n-SeqClasses.} \tag{6.50}$$

Nous présentons, sur la figure 6.4, les performances en terme de perplexité de ces deux modèles pour différentes valeurs de a.

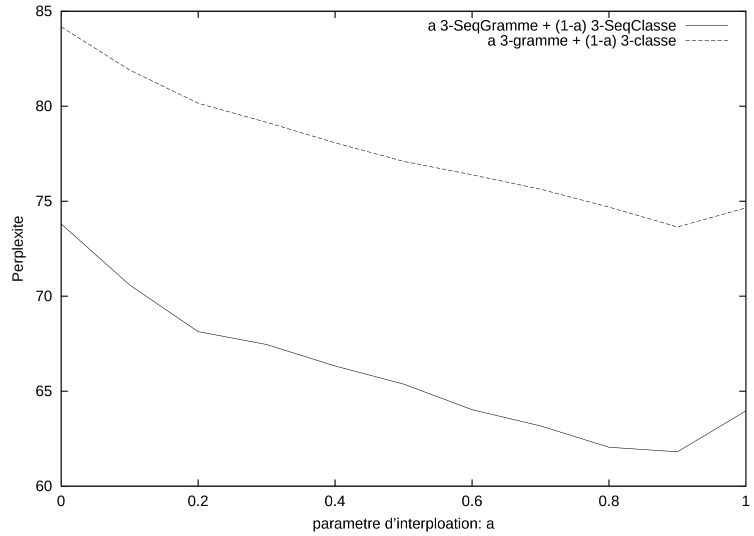


FIG. 6.4 – *Performances en terme de perplexité, des deux modèles de langage des équations 6.49 et 6.50, pour différentes valeurs de a.*

Cette figure montre que les deux modèles atteignent leur optimum pour une valeur de a égale à 0.9 (73.64 pour le premier modèle et 61.81 pour le deuxième). Les résultats obtenus avec cet optimum ($a = 0.9$) ne sont que légèrement inférieurs aux modèles de langage n'utilisant que les mots (n-grammes et n-SeqGrammes).

6.11 Conclusion

Nous venons de présenter dans ce chapitre une nouvelle approche d'extraction de séquences de mots linguistiquement viables. Les séquences extraites sont introduites comme des unités dans un vocabulaire de base. Nous utilisons ensuite ce nouveau vocabulaire pour l'apprentissage des modèles de langage et pour la prédiction. Ceci permet à ces modèles de mieux prendre en compte la capacité prédictive de ces séquences.

En analysant les séquences de mots extraites (séquences LV), nous avons remarqué que la majorité d'entre elles ont des structures syntaxiques viables linguistiquement. L'utilisation des séquences LV dans la modélisation de la langue a permis d'introduire des connaissances linguistiques supplémentaires, rendant ainsi les modèles de langage plus performants.

Les modèles de langage, présentés dans ce chapitre, sont tous à base de séquences. Évalués en terme de perplexité, ils donnent des performances supérieures aux modèles de langage classiques. La différence entre les modèles que nous avons proposés se situe surtout dans leur façon de traiter l'historique lors de la prédiction. En effet, à la différence d'un modèle n-SeqGrammes qui se fonde sur les n dernières séquences, un modèle n-SeqClasses utilise leurs classes syntaxiques lors de la prédiction ; les résultats obtenus avec ce dernier ne sont pas encourageants comparativement au modèle n-SeqGrammes. Pour améliorer les performances des modèles n-SeqClasses, nous pensons qu'il serait intéressant d'utiliser une autre méthode de classification plus performante.

L'objectif des modèles *SeqCache* et *SeqTrigger* est de mieux tenir compte des informations qui peuvent exister dans un historique lointain ; ceci en supposant, à la différence des modèles *cache* et *trigger*, que ces informations peuvent être introduites par des séquences de mots. Une interpolation linéaire de ces différents modèles (3-SeqGrammes, *SeqCache* et *SeqTrigger*) a donné la plus petite valeur de perplexité. Le modèle qui les combine devient ainsi meilleur. Pour mieux évaluer les capacités prédictives de ce modèle, nous comptons les utiliser avec des corpus thématiques (cf. §6.10).

Nous présentons, dans les deux chapitres qui suivent, les résultats d'évaluation de nos modèles de langage dans le cadre du jeu de Shannon et avec notre système de dictée vocale MAUD.

Chapitre 7

Évaluation avec le jeu de Shannon

7.1 Introduction

Les performances d'un modèle de langage sont souvent estimées en terme de perplexité sur un corpus de test différent de celui de l'apprentissage. Il existe également une autre méthode permettant d'évaluer les performances de ces modèles. Cette méthode, adaptation du jeu de Shannon [178], procède comme suit : soit un ensemble de phrases tronquées ; pour chacune de ces phrases, le modèle de langage doit distribuer un capital de 1 entre les mots du vocabulaire. La valeur de la distribution dépend de la vraisemblance de chaque mot à la suite de la phrase tronquée. Une fois le capital distribué, les mots du vocabulaire sont triés par ordre décroissant de la mise. Nous évaluons ainsi les performances des modèles de langage en fonction du nombre de fois où le bon mot est prédit en dessous d'un rang donné.

Nous présentons dans ce qui suit le principe de base du jeu de Shannon, ainsi que quelques extensions. Nous exposons au paragraphe 7.3 l'approche que nous avons utilisée pour l'évaluation de nos modèles de langage. Enfin, nous évoquons les résultats d'évaluation.

7.2 Le jeu de Shannon

C.E. Shannon propose une méthode permettant d'estimer l'entropie d'un langage [178] : on demande à un usager de prédire la première lettre d'un texte, en proposant successivement des candidats parmi les 27 possibilités (les 26 lettres de l'alphabet et l'espace), jusqu'à ce qu'il trouve le bon. Une fois la première lettre trouvée, il lui est demandé de prédire la seconde, et ainsi de suite. C.E. Shannon évalue alors les statistiques du nombre d'essais nécessaires pour définir une réponse correcte aux limites supérieure et inférieure de l'entropie du langage.

T.M. Cover et R.C. King ont montré que le jeu de Shannon peut être généralisé en une approche de jeu de hasard, où l'usager est invité à placer une certaine fraction d'un capital sur chaque lettre possible [41]. F. Jelinek a proposé également des variantes du jeu de Shannon pour comparer les capacités prédictives d'un modèle de langage et d'un être humain [93]. P. O'Boyle et *al.* ont utilisé ce jeu pour évaluer la qualité d'un modèle de langage à partir du nombre de fois où il prédit le bon mot en dessous d'un rang donné [146]. F. Bimbot et *al.* ont apporté une extension à ces approches [22] : ils proposent une alternative de calcul de la perplexité, en se fondant sur le formalisme du jeu de Shannon ; cette nouvelle approche permet la comparaison des différents modèles de langage dans le cadre d'une campagne d'évaluation.

7.3 Le jeu de hasard

L'approche que nous avons choisie pour l'évaluation des modèles de langage est similaire à celle du jeu de hasard. Elle peut être formalisée comme suit : soit les fragments successifs d'une phrase[15] W, obtenus par la découverte progressive de chacun de ces mots :

$$W =< w_1 w_2 \dots w_k \dots w_n >$$

1. $W_0^0 =<$
2. $W_0^1 =< w_1$
   ...
3. $W_0^{k-1} =< w_1 w_2 \dots w_{k-1}$
   ...
4. $W_0^{n-1} =< w_1 w_2 \dots w_{k-1} w_k \dots w_{n-1}$

où les mots w_i appartiennent à un vocabulaire $V = \{v_j\}_{1 \leq j \leq T}$. Pour chaque phrase tronquée W_0^{k-1}, le modèle de langage effectue une mise de $\beta_k(v_j)$ sur chaque mot du vocabulaire :

$$\beta_k(v_j) = p(v_j / W_0^{k-1}) \tag{7.1}$$

où $p(v_j / W_0^{k-1})$ est la probabilité conditionnelle du mot v_j sachant la phrase tronquée W_0^{k-1}. La somme des mises sur l'ensemble des mots du vocabulaire vaut 1 :

$$\sum_{j=1}^{T} \beta_k(v_j) = 1 \tag{7.2}$$

où T est la taille du vocabulaire. Dans le cas d'un modèle N-grammes, la probabilité conditionnelle de l'équation 7.1 est tout simplement la probabilité N-grammes :

$$\beta_k(v_j) = p(v_j / w_{k-N+1} \dots w_{k-1}). \tag{7.3}$$

Les mots du vocabulaire sont ensuite triés d'une manière décroissante suivant les mises. Les performances des modèles sont ainsi comparées au nombre de mots correctement prédits en dessous d'un rang donné.

Au lieu de prédire les mots successifs d'un texte, les données de test W sont considérées comme un ensemble de phrases distinctes, tronquées d'une façon aléatoire, à des endroits différents. Le modèle doit ainsi prédire le mot qui suit immédiatement chaque phrase tronquée. Une bonne représentation du comportement de la langue est alors obtenue avec un échantillon de phrases relativement petit.

7.3.1 Limitation de la liste des hypothèses

Pour contrôler le volume des données utilisées durant l'évaluation, le nombre de mots candidats pour chaque phrase tronquée peut être réduit à T_l, $T_l \leq T$. Dans le cas où $T_l < T$, la somme A_k des mises, affectées à la liste réduite des T_l mots, est strictement inférieure à 1 :

$$A_k = \sum_{r=1}^{T_l} \beta_k(u_r) < 1 \tag{7.4}$$

15. Une phrase est comprise entre < et >

où u_r représente le $r^{\text{ième}}$ mot candidat, une fois la liste des mots candidats a été ordonnée selon l'ordre décroissant des mises. Par conséquent, si le mot à prédire w_k n'appartient pas à la liste ordonnée des mots candidats $U = (u_r)_{1 \leq r \leq T_l}$, la valeur de sa mise sera estimée comme suit :

$$\beta_k^* = \frac{1 - A_k}{T - T_l}. \tag{7.5}$$

Cette approche suppose alors une répartition uniforme de la fraction non affectée des mises $(1 - A_k)$ sur l'ensemble des mots non candidats (les $T - T_l$ mots de rang supérieur à T_l). Comme les T_l mots candidats dans U correspondent aux T_l meilleures hypothèses proposées par le modèle, la valeur de β_k^* doit être plus petite que $\beta_k(u_{T_l})$. Ainsi, cette valeur de β_k^* est toujours inférieure à la plus petite mise dans U.

7.3.2 Une alternative au calcul de la perplexité

En utilisant l'approche de Bimbot et *al.* [22], la perplexité d'un modèle de langage $PP(W)$ est approximée par l'inverse de la moyenne géométrique des mises placées sur les mots corrects :

$$PP(W) = \left[\prod_{k=1}^{n} \beta_k(w_k)\right]^{-\frac{1}{n}} \tag{7.6}$$

Dans le cas d'un vocabulaire ouvert, si le mot à prédire n'appartient pas au vocabulaire, la valeur de la mise qui lui est associée sera nulle. Ceci engendre un terme de valeur égale à zéro dans la moyenne géométrique. Pour éviter ce problème, le vocabulaire doit inclure un mot particulier (UNK) représentant le mot hors vocabulaire. Le modèle de langage doit ainsi miser aussi bien sur ce mot que sur les autres mots du vocabulaire.

Si le modèle de langage à évaluer utilise des unités autres que les mots (les séquences de mots en ce qui nous concerne), nous avons besoin d'un traitement particulier pour estimer les mises. En effet, soit une séquence de r mots. Si les $i - 1$ ($i - 1 < r$) premiers mots de cette séquence correspondent aux $i - 1$ derniers mots de la phrase tronquée, il est clair qu'il faut prédire le $i^{\text{ème}}$ mot avec une une forte mise. Or, étant donné que le modèle à évaluer est à base de séquences, il faut trouver un protocole particulier permettant de miser sur ces mots. Une solution consiste à miser avec un modèle trigrammes ; en utilisant cette méthode, la perplexité (calculée à partir de l'équation 7.6) ne correspondra pas vraiment à celle du modèle qu'on désire évaluer.

Le calcul de la perplexité, à partir de l'équation 7.6, n'est qu'une alternative supplémentaire à la méthode classique (voir chapitre 2). De plus, notre objectif dans ce chapitre est loin d'être le calcul de la perplexité, puisque nous l'avons calculé auparavant. Nous avons donc abandonné cette mesure et nous nous sommes limités à l'évaluation des capacités prédictives des modèles.

7.4 Évaluation des modèles de langage

Pour l'évaluation des modèles de langage, nous avons utilisé un ensemble de 10000 phrases tronquées, extrait aléatoirement du journal “*Le Monde Diplomatique*” (1990 – 1996) [92]. Cet ensemble de phrases ne fait pas partie du corpus d'apprentissage. Le nombre de mots candidats pour chaque phrase est limité à 5000 (T_l = 5000). Ainsi, le modèle de langage fournit, pour chaque phrase tronquée, une liste U de 5000 mots candidats avec leurs mises respectives. Nous présentons dans les sous-paragraphes qui suivent les capacités prédictives de nos modèles de langage. Nous exposons pour chaque modèle le nombre de mots prédits correctement en dessous d'un rang donné. Nous nous limitons aux rangs numéros 1, 5, 25, 100 et 5000. Nous donnons en plus le rang moyen de prédiction pour chaque modèle ; ce rang correspond à la somme des rangs des mots prédits correctement, divisée par le nombre total de phrases (10 000 pour notre cas). Le nombre de mots à prédire n'appartenant pas au vocabulaire (UNK) est de 1271.

Pour l'apprentissage des modèles de langage, nous avons utilisé un corpus d'environ 50 millions de mots (cf. §4.6). Les classes employées par les modèles n-classes et 3-SeqClasses sont celles définies dans le paragraphe 4.3. Pour extraire le corpus de classes, nécessaire à l'apprentissage de ces modèles, nous avons appliqué l'algorithme d'étiquetage du paragraphe 4.4 sur le corpus de mots cité ci-dessus. Le vocabulaire de base utilisé contient 20000 mots environ. Dans le cas des modèles n-SeqGrammes et n-SeqClasses, nous avons ajouté à ce vocabulaire l'ensemble des séquences extraites à partir de l'algorithme présenté dans le paragraphe 6.4.

L'objectif de cette évaluation est de tester les capacités prédictives apportées par les séquences. Nous évaluons ainsi, par la suite, nos modèles de langage conventionnels ne se servant que des mots lors de la prédiction. Nous testons dans 7.4.2 les capacités prédictives des modèles équivalents se fondant, cette fois, sur les séquences. Nous exposons enfin dans 7.4.3 les capacités prédictives de l'approche hybride (voir chapitre 4) avec et sans séquences. Cependant, comme le modèle hiérarchique a besoin de la totalité de la phrase pour agir, nous ne l'avons pas évalué avec ce formalisme.

7.4.1 Les modèles conventionnels

Les modèles de langage conventionnels ne se servent que des mots lors de la prédiction. Nous nous sommes limités ici à quatre d'entre eux : le modèle trigrammes, le modèle triclasses, le modèle trigrammes avec *cache* et *trigger*, ainsi qu'un modèle obtenu à partir d'une interpolation linéaire entre les modèles trigrammes et triclasses.

Le modèle trigrammes

En utilisant le modèle trigrammes, la mise ne dépend que des deux derniers mots de la phrase tronquée ($w_{k-2}w_{k-1}$) :

$$\beta_k(v_j) = p(v_j/w_{k-2}w_{k-1}) \tag{7.7}$$

où v_j représente le $j^{\text{ème}}$ mot du vocabulaire. Dans le cas où la phrase tronquée ne contient qu'un seul mot w_1, nous misons avec le modèle bigrammes :

$$\beta_k(v_j) = p(v_j/w_1) \tag{7.8}$$

Nous exposons dans la table 7.1 les résultats de prédiction obtenus avec ce modèle trigrammes.

rang	**nombre de mots prédits correctement**
1	1829
$2-5$	2375
$6-25$	1920
$26-100$	1419
$101-5000$	2170
rang moyen : 207	
taux de prédiction : 97.13%	

TAB. 7.1 – *Résultats de prédiction du modèle trigrammes.*

Le modèle triclasses

Le modèle triclasses utilisé se fonde sur les classes syntaxiques du français (cf. §4.3). La mise effectuée par ce modèle est fonction des deux derniers mots de la phrase tronquée. Comme un mot peut appartenir à plusieurs classes, la mise affectée à un mot v_j du vocabulaire sachant une phrase tronquée se terminant par $w_{k-2}w_{k-1}$ est estimée comme suit :

$$\begin{aligned}\beta_k(v_j) &= p(v_j/w_{k-2}w_{k-1}) \\ &= \sum_{c(v_j)\in C_{v_j}} p(v_j/c(v_j)) \times \Bigg[\sum_{c(w_{k-1})\in C_{w_{k-1}}} p(c(w_{k-1})/w_{k-1}) \\ &\times \Bigg(\sum_{c(w_{k-2})\in C_{w_{k-2}}} p(c(w_{k-2})/w_{k-2}) \\ &\times p(c(v_j)/c(w_{k-2})c(w_{k-1}))\Bigg)\Bigg] \end{aligned} \quad (7.9)$$

où C_w indique l'ensemble de classes auxquelles le mot w peut appartenir, et $c(w)$ indique une classe possible du mot w appartenant à l'ensemble C_w. Si nous notons par $N(w/c(w))$ le nombre de fois où le mot w est étiqueté par la classe $c(w)$, la probabilité d'appartenance d'un mot w à une classe $c(w)$, $p(w/c(w))$, est approximée par :

$$p(w/c(w)) = \frac{N(w/c(w))}{N(c(w))} \quad (7.10)$$

où $N(.)$ indique le nombre d'occurrences de ce qui est en argument. En revanche, la probabilité de faire correspondre la classe $c(w)$ au mot w est estimée comme suit :

$$p(c(w)/w) = \frac{N(w/c(w))}{N(w)}. \quad (7.11)$$

Dans le cas où la phrase tronquée ne contient qu'un seul mot, nous utilisons le modèle biclasses :

$$\begin{aligned}\beta_k(v_j) &= p(v_j/w_{k-1}) \\ &= \sum_{c(v_j)\in C_{v_j}} p(v_j/c(v_j)) \times \Bigg[\sum_{c(w_{k-1})\in C_{w_{k-1}}} p(c(w_{k-1})/w_{k-1}) \\ &\times p(c(v_j)/c(w_{k-1}))\Bigg]. \end{aligned} \quad (7.12)$$

Nous présentons dans la table 7.2 les résultats de prédiction de ce modèle.

rang	nombre de mots prédits correctement
1	1270
2 – 5	1839
6 – 25	1681
26 – 100	1165
101 – 5000	3376
rang moyen : 510	
taux de prédiction : 93.31%	

TAB. 7.2 – *Résultats de prédiction du modèle triclasses.*

Le modèle trigrammes avec cache et trigger

Le modèle de langage que nous utilisons ici est le résultat d'une interpolation linéaire entre le modèle trigrammes, le modèle *cache* et le modèle *trigger*. Ce modèle se fonde sur la totalité de l'historique lors de la prédiction. En effet, pour une phrase tronquée $W_0^{k-1} =< w_1 w_2 \ldots w_{k-1}$, la mise sur un mot v_j du vocabulaire est estimée comme suit :

$$\begin{aligned} \beta_k(v_j) &= P(v_j/W_0^{k-1}) \\ &= (1-\lambda_1-\lambda_2)P_{Gram}(v_j/W_0^{k-1}) + \lambda_1 P_{Cach}(v_j/W_0^{k-1}) + \lambda_2 P_{Trig}(v_j/W_0^{k-1}) \end{aligned} \tag{7.13}$$

où les λ_i représentent les paramètres d'interpolation, $P_{Trig}(.)$ est la probabilité du modèle *trigger*, $P_{Cach}(.)$ celle du modèle *cache* et $P_{Gram}(.)$ celle du modèle trigrammes. Dans le cas où la phrase tronquée ne contient qu'un seul mot, nous nous limitons à un modèle bigrammes.

La table 7.3 présente les résultats de prédiction obtenus par ce modèle.

rang	nombre de mots prédits correctement
1	1830
2 – 5	2378
6 – 25	1922
26 – 100	1419
101 – 5000	2168
rang moyen : 206	
taux de prédiction : 97.17%	

TAB. 7.3 – *Résultats de prédiction du modèle trigrammes avec cache et trigger.*

Le modèle combinant trigrammes et triclasses

Ce modèle est le résultat d'une interpolation linéaire entre un modèle trigrammes et un modèle triclasses. La mise affectée à un mot v_j du vocabulaire, sachant une phrase tronquée, ne dépend ainsi que des deux derniers mots $(w_{k-2}w_{k-1})$ de la phrase :

$$\begin{aligned} \beta_k(v_j) &= P(v_j/w_{k-2}w_{k-1}) \\ &= (1-\lambda_1)P_{Gram}(v_j/w_{k-2}w_{k-1}) + \lambda_1 P_{Class}(v_j/w_{k-2}w_{k-1}) \end{aligned} \tag{7.14}$$

où λ_1 est un paramètre d'interpolation $(0 \leq \lambda_1 \leq 1)$, $P_{Gram}(.)$ la probabilité du modèle trigrammes, $P_{Class}(.)$ celle du modèle triclasses. Dans le cas où la phrase tronquée ne contient qu'un seul mot, nous utilisons les modèles bigrammes et biclasses.

Les résultats de prédiction de ce modèle sont exposés dans la table 7.4, pour $\lambda_1 = 0.1$ (valeur qui a donné la plus petite perplexité (chapitre 6)).

rang	**nombre de mots prédits correctement**
1	1829
$2-5$	2377
$6-25$	1921
$26-100$	1418
$101-5000$	2170
rang moyen : 207	
taux de prédiction : 97.15%	

TAB. 7.4 – *Résultats de prédiction d'une interpolation linéaire entre les modèles trigrammes et triclasses.*

Bilan comparatif

En analysant les résultats, nous remarquons que les capacités prédictives des modèles triclasses sont assez limitées. En effet, par rapport au modèle trigrammes, le taux de prédiction a baissé d'environ 4% et le rang moyen a plus que doublé. Pour améliorer les performances des modèles de langage utilisant ces classes, il serait intéressant de proposer une autre méthode de classification prenant mieux en compte le comportement de la langue [185].

Les capacités prédictives des autres modèles sont presque identiques. En effet, l'interpolation linéaire entre les modèles trigrammes et triclasses n'a pu accroître que de deux, le nombre de mots prédits correctement en dessous du rang 5000. Le rang moyen, par contre, est resté constant par rapport au modèle trigrammes.

L'utilisation du cache et des triggers n'a pas apporté non plus d'amélioration significative aux capacités prédictives. Le nombre de mots prédits correctement en dessous du rang 5000 ne s'est augmenté que de 4 mots et le rang moyen ne s'est amélioré que de 1, par rapport à un modèle trigrammes. Nous pensons que ceci est du à la nature généraliste de nos triggers et des phrases tronquées, extraits à partir de textes de journaux. En effet, les modèles utilisant les triggers se sont avérés intéressants surtout dans le traitement des thèmes spécifiques, ce qui n'est pas notre cas.

7.4.2 Les modèles à base de séquences

Nous présentons ici des modèles équivalents à ceux étudiés ci-dessus, mais en se servant cette fois des séquences. Les modèles exposés sont les suivants : 3-SeqGrammes, 3-SeqClasses et 3-SeqGrammes avec *SeqCache* et *SeqTrigger* (voir chapitre 6). Nous exposons également les résultats de prédiction d'un modèle issu d'une interpolation linéaire entre les modèles 3-SeqClasses et 3-SeqGrammes.

L'unité de prédiction de ces modèles est la séquence de mots. Il est donc nécessaire d'établir un traitement particulier, qui permet de bénéficier des avantages apportés par l'utilisation des séquences lors de la prédiction des mots candidats. Nous avons ainsi procédé en deux étapes lors de la prédiction : prédiction à partir des séquences seules et prédiction à partir des modèles.

La première étape est commune à tous les modèles utilisant les séquences. Elle ne se sert que des séquences et procède comme suit : soit une séquence s_r de r mots ; si les $i-1$ $(i-1<r)$ premiers mots de cette séquence correspondent aux $i-1$ derniers mots de la phrase tronquée, nous additionnons le $i^{\text{ème}}$ mot de la séquence à la liste des mots candidats ; nous répétons ce traitement pour toutes les séquences. Les mots prédits seront triés par le modèle trigrammes et apparaîtront dans les premiers rangs. Nous ne gardons que les mots distincts. Soit t $(t<T_l)$ le nombre de mots extraits.

Les modèles de langage sont utilisés lors de la deuxième étape pour compléter la liste des mots candidats à T_l mots. Pour ce faire, nous étiquetons la phrase tronquée avec l'ensemble des séquences LV (cf. §6.4). Nous affectons à chaque unité (mot ou séquence) du vocabulaire une mise, en fonction du modèle de langage utilisé. Ensuite, nous remplaçons chaque séquence du vocabulaire par son premier mot, construisant ainsi une liste de mots candidats. Enfin, nous trions les mots suivant l'ordre décroissant des mises et nous ne gardons que les $T_l - t$ meilleurs. Les T_l mots candidats doivent être distincts.

Nous présentons ci-dessous les résultats d'évaluation de nos modèles de langage se servant des séquences LV.

Le modèle 3-SeqGrammes

Pour une phrase tronquée se terminant par $s_{i-2}s_{i-1}$, la mise associée à une unité z_j du vocabulaire est estimée par l'équation suivante :

$$\beta_k(z_j) = p(z_j/s_{i-2}s_{i-1}) \tag{7.15}$$

où la valeur de vraisemblance $p(z_j/s_{i-2}s_{i-1})$ est approximée par le modèle 3-SeqGrammes. Dans le cas où la phrase tronquée ne contient qu'un seul mot, nous utilisons un modèle 2-SeqGrammes.

La table 7.5 donne les résultats de prédiction de ce modèle.

rang	**nombre de mots prédits correctement**
1	1842
$2-5$	2396
$6-25$	2143
$26-100$	1865
$101-5000$	1477
rang moyen : 192	
taux de prédiction : 97.23%	

TAB. 7.5 – *Résultats de prédiction du modèle 3-SeqGrammes.*

Le modèle 3-SeqClasses

Soit $s_{i-2}s_{i-1}$ les deux dernières unités (séquences ou mots) de la phrase tronquée en cours de traitement. La mise associée à une unité z_j du vocabulaire est approximée comme suit :

$$\begin{aligned}\beta_k(z_j) &= p(z_j/s_{i-2}s_{i-1}) \\ &= \sum_{c(z_j)\in C_{z_j}} p(z_j/c(z_j)) \times \Bigg[\sum_{c(s_{i-1})\in C_{s_{i-1}}} p(c(s_{i-1})/s_{i-1}) \\ &\times \Bigg(\sum_{c(s_{i-2})\in C_{s_{i-2}}} p(c(s_{i-2})/s_{i-2}) \\ &\times p(c(z_j)/c(s_{i-2})c(s_{i-1}))\Bigg)\Bigg] \end{aligned} \tag{7.16}$$

où C_s indique l'ensemble de classes auxquelles la séquence s peut appartenir, et $c(s)$ une classe possible de la séquence s appartenant à l'ensemble C_s. Les termes $p(s/c(s))$ et $p(c(s)/s)$ sont estimés de la même façon que dans le modèle triclasses ci-dessus. Si la phrase tronquée ne contient qu'un seul mot, nous utilisons un modèle 2-SeqClasses.

Les résultats de prédiction obtenus avec ce modèle sont exposés dans la table 7.6.

rang	**nombre de mots prédits correctement**
1	1700
$2-5$	2202
$6-25$	2088
$26-100$	1687
$101-5000$	1760
rang moyen : 313	
taux de prédiction : 94.37%	

TAB. 7.6 – *Résultats de prédiction du modèle 3-SeqClasses.*

Le modèle 3-SeqGrammes avec SeqCache et SeqTrigger

Dans ce modèle, la mise est calculée par une interpolation linéaire entre les modèles 3-SeqGrammes, *SeqCache* et *SeqTrigger*. Par conséquent, pour une phrase tronquée, étiquetée par la suite de séquences $S_0^{i-1} =< s_1 s_2 \ldots s_{i-1}$, la valeur de la mise associée à chaque unité z_j du vocabulaire est approximée comme suit :

$$\begin{aligned}\beta_k(s_j) &= P(s_j/S_0^{i-1}) \\ &= (1-\lambda_1-\lambda_2)P_{SeqGram}(s_j/S_0^{i-1}) + \lambda_1 P_{SeqCach}(s_j/S_0^{i-1}) + \lambda_2 P_{SeqTrig}(s_j/S_0^{i-1})\end{aligned} \tag{7.17}$$

où les λ_i représentent les paramètres d'interpolation, $P_{SeqTrig}(.)$ est la probabilité du modèle *SeqTrigger*, $P_{SeqCach}(.)$ celle du modèle *SeqCache* et $P_{SeqGram}(.)$ celle du modèle 3-SeqGrammes. Si la phrase tronquée ne contient qu'un seul mot, nous nous limitons au modèle 2-SeqGrammes.

Nous donnons dans la table 7.7, les résultats de prédiction de ce modèle.

rang	nombre de mots prédits correctement
1	1842
$2-5$	2398
$6-25$	2144
$26-100$	1869
$101-5000$	1472
rang moyen : 191	
taux de prédiction : 97.25%	

TAB. 7.7 – *Résultats de prédiction d'une interpolation linéaire entre les modèles 3-SeqGrammes, SeqCache et SeqTrigger.*

Le modèle combinant 3-SeqGrammes et 3-SeqClasses

La mise associée à une unité z_j du vocabulaire, sachant une phrase tronquée se terminant par les deux unités (séquences ou mots) $s_{i-2}s_{i-1}$, est approximée comme suit :

$$\begin{aligned} \beta_k(z_j) &= P(z_j/s_{i-2}s_{i-1}) \\ &= (1-\lambda_1)P_{SeqGram}(z_j/s_{i-2}s_{i-1}) + \lambda_1 P_{SeqClass}(z_j/s_{i-2}s_{i-1}) \end{aligned} \tag{7.18}$$

où λ_1 est un paramètre d'interpolation ($0 \leq \lambda_1 \leq 1$), $P_{SeqGram}(.)$ la probabilité du modèle 3-SeqGrammes, $P_{SeqClass}(.)$ celle du modèle 3-SeqClasses. Si la phrase tronquée ne contient qu'un seul mot, nous utilisons les modèles 2-SeqGrammes et 2-SeqClasses.

La table 7.8 présente les résultats de prédiction de ce modèle.

rang	nombre de mots prédits correctement
1	1842
$2-5$	2397
$6-25$	2143
$26-100$	1866
$101-5000$	1476
rang moyen : 192	
taux de prédiction : 97.24%	

TAB. 7.8 – *Résultats de prédiction d'une interpolation linéaire entre les modèles 3-SeqGrammes et 3-SeqClasses.*

Bilan comparatif

L'utilisation des séquences dans le modèle triclasses (3-SeqClasses) a permis d'augmenter considérablement les capacités prédictives du modèle. En effet, le nombre de mots prédits correctement, en dessous du rang 5000, a augmenté de 100 et le rang moyen est passé de 510 à 313. En revanche, les autres méthodes se servant des séquences, n'ont que peu amélioré les capacités prédictives de leurs équivalents n'utilisant que les mots. En se fondant sur les séquences, le nombre de mots prédits correctement en premier rang est majoré d'une quinzaine de mots et celui en dessous du rang 5000 d'environ une dizaine. Le rang moyen également n'a progressé que de 15 unités. Nous pensons que cette légère amélioration est due à la méthode d'évaluation, laquelle est plus adaptée aux modèles ne se fondant que sur les mots lors de la prédiction. En effet, en utilisant les modèles se servant des séquences, nous devons apporter une adaptation permettant d'extraire les mots candidats, ce qui n'est pas le cas des modèles n'employant que les mots.

Les remarques effectuées précédemment, lors de la comparaison des modèles conventionnels entre eux, restent valables pour les modèles de ce paragraphe.

7.4.3 L'approche hybride

Notre objectif ici est d'évaluer les capacités prédictives de l'approche hybride (voir chapitre 4) avec et sans séquences. Nous utilisons ainsi le modèle formel pour supprimer les mots du vocabulaire qui ne vérifient pas la contrainte d'accord avec la phrase tronquée en cours de traitement. Nous appliquons ce modèle seulement s'il contient une règle grammaticale dont les constituants de départ correspondent à la structure de la phrase tronquée en cours. Ensuite, nous utilisons un modèle probabiliste pour miser sur les mots restants. Deux expérimentations ont été menées avec deux modèles probabilistes différents : le premier modèle est le meilleur en ne se servant que des mots (modèle obtenu par une interpolation linéaire entre les modèles trigrammes, cache et trigger). Le deuxième modèle est le plus performant en se fondant sur les séquences (modèle obtenu par une interpolation linéaire entre les modèles 3-SeqGrammes, SeqCache et SeqTrigger).

La table 7.9 montre les résultats d'application de l'approche hybride sur les mots.

rang	**nombre de mots prédits correctement**
1	1830
$2-5$	2380
$6-25$	1921
$26-100$	1428
$101-5000$	2160
rang moyen : 205	
taux de prédiction : 97.19%	

TAB. 7.9 – *Résultats de prédiction de l'approche hybride, en utilisant une interpolation linéaire entre les modèles trigrammes, cache et trigger.*

La table 7.10 présente les résultats de l'approche se servant des séquences.

rang	**nombre de mots prédits correctement**
1	1843
$2-5$	2400
$6-25$	2143
$26-100$	1872
$101-5000$	1468
rang moyen : 191	
taux de prédiction : 97.26%	

TAB. 7.10 – *Résultats de prédiction de l'approche hybride, en utilisant une interpolation linéaire entre les modèles 3-SeqGrammes, SeqCache et SeqTrigger.*

En analysant les résultats, nous avons constaté que le modèle hybride n'a pu intervenir que sur 7% des phrases tronquées. Comme nous ne possédons qu'une portion de la phrase, le modèle formel n'est pas souvent en mesure de prendre une décision.

7.5 Conclusion

Nous venons d'évaluer les capacités prédictives des différents modèles de langage développés dans cette thèse, avec une variante du jeu de Shannon. Nous avons ainsi présenté des modèles utilisant les mots comme unité de base, ainsi que leurs équivalents se fondant sur les séquences. Les résultats obtenus prouvent que les modèles à base de séquences sont toujours meilleurs, en terme de prédiction, que leurs équivalents n'utilisant que les mots.

Le modèle 3-SeqClasses a augmenté les capacités prédictives du modèle triclasses de plus de 100 mots et il a diminué le rang moyen de moitié. Les performances des autres modèles se servant des séquences sont également supérieures à celles de leurs équivalents n'employant que les mots : environ 10 mots ont été prédits en plus. Nous pensons que cette légère amélioration est due à la nature de ce jeu qui est plus adapté aux modèles n'utilisant que les mots. En effet, pour utiliser les modèles à base de séquence, nous étions obligés d'effectuer un traitement particulier lors de la prédiction des mots (cf. §7.4.2).

L'approche hybride n'a que peu amélioré les capacités de prédiction : Le rang moyen a baissé d'une unité et le nombre de mots prédits correctement a augmenté de deux. Ceci s'explique par le fait que le modèle formel n'a pu intervenir que sur une partie des phrases tronquées (7%). Comparativement aux autres approches, celle-ci a tout de même amélioré le nombre de mots prédits correctement, en dessous du rang 100.

Nous présentons, dans le chapitre 8, le système de RAP MAUD, développé durant cette thèse, ainsi que les résultats apportés par l'intégration de nos modèles de langage.

Chapitre 8

Le système MAUD

8.1 Introduction

Dans ce chapitre, nous allons évaluer les différents modèles de langage cités auparavant, en utilisant le système de RAP MAUD que nous avons contribué à développer. La version de base du système MAUD a participé à la première campagne d'évaluation B1 d'Aupelf-UREF[16] [51, 35], avec de nombreux autres systèmes issus de différents laboratoires. Nous exposons brièvement dans le paragraphe 8.2 ces différents systèmes. Nous définissons dans le paragraphe 8.3 le fonctionnement de la version de base de MAUD. Nous présentons ensuite dans le paragraphe 8.4 plusieurs versions de ce système qui ne diffèrent que par le type de modèle de langage utilisé. Enfin, nous donnons les résultats apportés par l'intégration de nos modèles de langage et ceux obtenus par l'utilisation des modèles probabilistes conventionnels.

8.2 Les systèmes ayant participé à la campagne B1 de l'Aupelf-UREF

La campagne B1 est l'une des actions de recherches lancées par l'Aupelf-UREF ; elle vise la mise au point et l'utilisation d'une méthodologie commune, pour l'évaluation comparative et contrastive de systèmes de RAP sur une ou plusieurs tâches de référence. Le but est d'accompagner et de faire progresser la recherche fondamentale et appliquée dans le domaine.

La campagne B1 est destinée aux systèmes fonctionnant en français. Les ressources mises à la disposition des participants de la catégorie $P0$, dont nous étions, sont les suivantes [4] :

- données de parole *BREF-80* [121] et les textes correspondants (B80),
- données de parole *BREF-TOTAL* [121] et les textes correspondants (BT),
- textes du journal *Le Monde* couvrant les années 1987 et 1988 pour l'apprentissage des modèles de langage (corpus *Monde/87-88*).
- liste *20k/87-88* des 20000 mots les plus fréquemment rencontrés dans le corpus *Monde/87-88*, constituant le vocabulaire des systèmes de RAP testés.

8.2.1 Le système de l'INRS-Télécommunication de Montréal

Le système de l'INRS-Télécommunication est fondé sur une recherche en deux passes. La première passe de recherche utilise des modèles de HMMs (HMM à 3 états) de phonèmes dépen-

16. Association des universités partiellement ou entièrement de langue française - université des réseaux d'expression française (*http://www.refer.qc.ca/*)

dants du contexte droit (diphones) et un modèle de langage bigrammes pour produire un graphe de mots. L'algorithme de recherche utilisé est de type A^*. Une deuxième passe de recherche est effectuée sur le graphe de mots, en utilisant des modèles de HMMs de phonèmes dépendants du contexte droit et gauche (triphones) et un modèle de langage trigrammes. Cette dernière passe fournit le résultat final. Dans ce système, la parole est échantillonnée à $16KHz$. Ensuite, tous les $10ms$, il calcule un ensemble de 14 MFCC, les dérivées premières de ces 14 coefficients et la dérivée première de l'énergie, soit un total de 29 composantes, ceci sur une fenêtre de Hamming de $30ms$. Ce système a donné un taux d'erreur de 38.49% [172].

8.2.2 Le système du LIMSI-Orsay

Pour reconnaître une phrase prononcée, ce système procède en trois passes. Lors de la première passe, il réduit l'espace de recherche à explorer dans les prochaines passes. Pour ce faire, il utilise un modèle de langage bigrammes et un modèle acoustique de triphones dépendants de leur position dans le mot. Les triphones utilisés sont à base de HMMs. A l'issue de cette étape, le système dispose d'un graphe de mots hypothèses. Lors de la deuxième passe, la phrase est décodée avec un modèle acoustique de triphones indépendants et un modèle de langage trigrammes, en limitant l'espace de recherche aux graphes de mots générés lors de la première passe. Ensuite, les hypothèses produites sont utilisées pour adapter les modèles acoustiques au locuteur. La dernière passe de reconnaissance utilise ces modèles acoustiques adaptés pour fournir la meilleure hypothèse. Un modèle de langage biclasses est utilisé d'une manière optionnelle lors de cette dernière passe [3]. Les classes de ce modèle sont obtenues automatiquement par recuit simulé et minimisation de la perplexité. Dans la classification utilisée, un mot n'admet qu'une seule classe. Ce système a donné un taux d'erreur d'environ 11%. Il est à noter qu'il utilise un corpus de 270 millions de mots pour l'apprentissage des modèles de langage. Ceci est largement supérieur à la taille de corpus, utilisée par les autres participants (40 millions de mots). En plus, lors de la reconnaissance, le système du LIMSI s'appuie sur un vocabulaire largement supérieur aux 20000 mots donnés, ce qui limite le nombre de mots hors vocabulaire.

8.2.3 Le système du CRIM pour le français

Le système du CRIM (Centre de Recherche en Informatique de Montréal) utilise un modèle acoustique à base de HMMs de 4 états et un modèle de langage bigrammes. Le modèle acoustique utilisé est à trois distributions continues organisées en arbres de classification de distributions [125]. Le modèle de langage bigrammes est appris sur le corpus *Monde/87-88*. Le système procède en une seule passe, en utilisant un algorithme de type Viterbi synchrone. Le taux d'erreur obtenu par ce système est de l'ordre de 38,78% [124].

8.3 Le système MAUD

La version de base du système MAUD (Machine AUtomatique à Dicter) utilise un vocabulaire d'environ 20000 mots et procède en trois passes : construction d'un treillis de mots, construction des N meilleures phrases et filtrage de phrases [61, 62]. Cette version, avec des HMMs de diphones, a participé à la campagne B1 d'Aupelf-UREF et a donné un taux d'erreur de 31.85%. Nous présentons sur la figure 8.1 l'architecture générale de la version employée dans ce document.

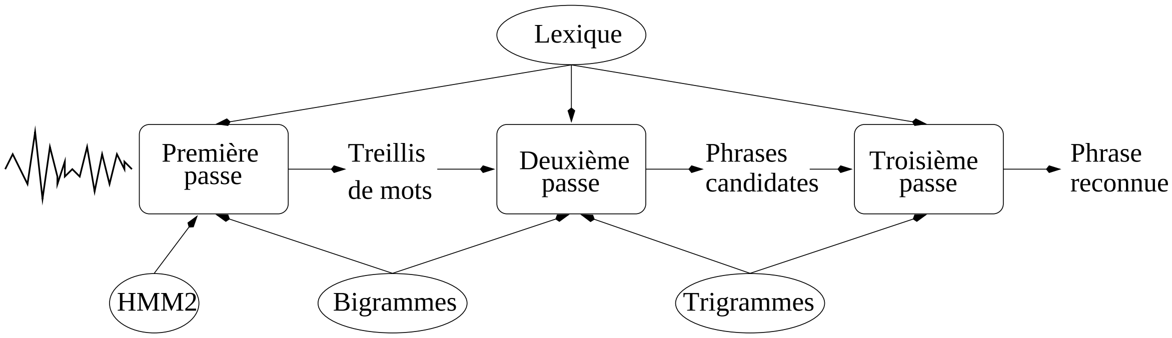


Fig. 8.1 – *Architecture générale du système MAUD.*

8.3.1 Construction d'un treillis de mots

La première passe a pour but d'obtenir, à partir du signal de parole, un treillis de mots en utilisant des modèles acoustiques, le lexique et un modèle de langage à contraintes locales (bigrammes). Pour paramétrer le signal, nous utilisons l'énergie du signal plus les 12 coefficients MFCC ainsi que leurs dérivées premières et secondes [36, 45]. La fenêtre d'analyse utilisée est de 32 ms. En revanche, l'espace entre les trames est de 8 ms. Chaque phonème est défini par un modèle de Markov caché continu du second ordre à trois états (HMM2) [131]. Ces HMM2 sont appris à l'aide de *BREF80* [121]. Nous avons ainsi construit 36 modèles de phonèmes non contextuels. Chaque mot du lexique est représenté par un HMM2 obtenu par la concaténation de HMM2 de phonèmes. Pour obtenir le treillis de mots, nous utilisons un algorithme de type "Viterbi-Bloc" (cf. §8.3.6) avec un modèle bigrammes. A la fin de cette étape, nous obtenons une liste de mots candidats, avec pour chacun d'entre eux, son score acoustique d'alignement et ses instants de début et de fin [60].

8.3.2 Construction des N meilleures phrases

Dans cette deuxième étape, nous cherchons à construire les N meilleures phrases à partir du treillis de mots obtenus à l'étape précédente. La méthode utilisée est une recherche en faisceau avec un modèle de langage trigrammes. Dans cette étape, nous n'utilisons pas les modèles acoustiques (HMM2), puisque nous disposons du score d'alignement de la passe précédente. Le résultat de cette étape est une liste de phrases ordonnées suivant un score combinant la distance de recalage acoustique et la valeur de vraisemblance donnée par le modèle trigrammes.

8.3.3 Filtrage de phrases

A la différence des deux premières étapes où nous n'utilisons que des contraintes locales, nous employons lors de cette passe des contraintes à plus long terme. La première phrase respectant ces nouvelles contraintes est la phrase reconnue. Dans la version de base du système, nous nous sommes limités aux deux premières étapes. Ainsi, la meilleure hypothèse produite par la deuxième passe est considérée comme le résultat du système. Nous présenterons dans le paragraphe 8.4 d'autres versions de MAUD, utilisant le modèle hybride et le modèle hiérarchique pour filtrer les N meilleures hypothèses produites par les niveaux inférieurs.

8.3.4 Liaison et phonologie

Un des problèmes inhérents à la reconnaissance de la parole continue en français, est la possibilité de faire des liaisons entre les mots. Pour prendre en compte ce phénomène, nous traitons cette information au niveau du modèle markovien. Comme nous l'avons déjà mentionné, nous représentons chaque mot par un HMM. Quand un mot peut se prononcer en liaison avec le mot qui le suit, nous ajoutons à la chaîne de phonèmes qui le compose, le phonème correspondant à la consonne latente ; par exemple, le mot “irons” sera représenté par la suite de phonèmes $/ir\tilde{o}z/$ et nous alignerons $/ir\tilde{o}/$ si le mot suivant commence par une consonne. Si le mot suivant commence par une voyelle, nous alignons $/ir\tilde{o}/$ et $/ir\tilde{o}z/$ et nous ne gardons que le meilleur alignement. Lors du traitement informatique, pour éviter d'avoir à traiter plusieurs cas particuliers de phonologie, nous considérons que toute liaison est facultative. Pour savoir si un mot est susceptible de provoquer une liaison, nous utilisons les connaissances phonologiques contenues dans le lexique BDLEX [156].

Dans le cas où un mot peut se prononcer de plusieurs manières (par exemple, le mot “médecin”), nous avons choisi de créer un modèle de Markov par prononciation. Quand un mot se termine par un “e muet”, nous alignons systématiquement les deux prononciations (avec et sans le “e muet”) et nous ne gardons que le meilleur alignement. En revanche, nous ne traitons pas les assimilations de sonorité provoquées par la disparition du “e muet” (noté @), comme dans “banque de France”. Nous essayons d'aligner “$/b\tilde{a}k@d@fR\tilde{a}s/$” ou “$/b\tilde{a}kd@fR\tilde{a}s/$” mais pas “$/b\tilde{a}gd@fR\tilde{a}s/$”.

8.3.5 Algorithme de Viterbi-Bloc

L'algorithme de Viterbi-Bloc utilise des critères basés sur une notion d'optimalité locale plutôt que globale. De telles contraintes ont été appliquées avec succès à la reconnaissance de mots isolés [78], de traits acoustiques [115, 116] ou encore de phonèmes [26] dans le discours continu.

Comme mentionné au chapitre 1, l'algorithme de Viterbi construit et préserve à chaque instant tous les chemins partiels s'achevant sur chaque état du HMM2 ; pourtant, il n'utilise qu'un seul chemin, issu de l'état final associé à la dernière trame, pour retrouver la suite des états empruntés. En procédant localement, l'algorithme met en oeuvre des structures de données moins volumineuses, puisqu'il ne conserve qu'une partie des données nécessaires aux retours-arrière. Cet algorithme effectue simultanément une segmentation et la reconnaissance ; il donne une réponse en temps réel au fur et à mesure de la prononciation de la phrase sans attendre la fin de celle-ci. Le principe général consiste à utiliser une fenêtre glissante (voir figure 8.2) de N trames de parole dans laquelle une décision est prise localement. Nous décrivons ci-dessous le processus de reconnaissance de Viterbi-Bloc.

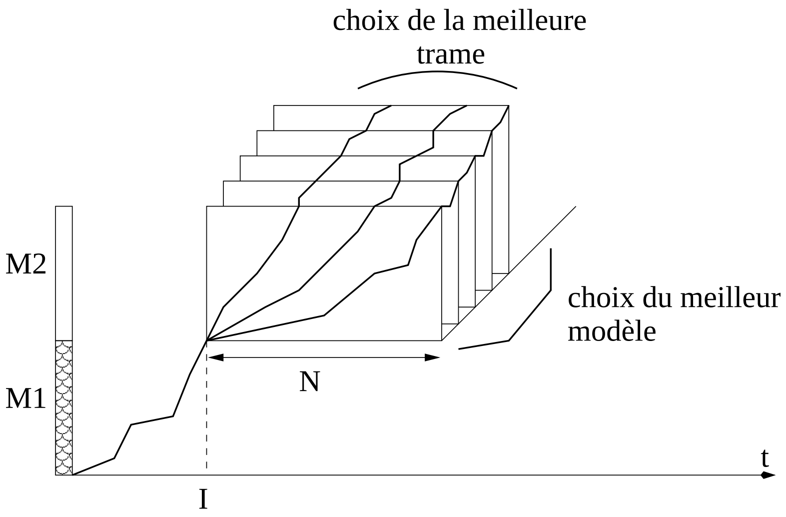


FIG. 8.2 – *Schéma de principe de l'algorithme de Viterbi-Bloc.*

Algorithme de Viterbi-Bloc (O,T,M,K) **:**

entrée : $O_1,O_2,\ldots,O_T$ la suite de T trames représentant l'élocution ;

sortie : $m_1,m_2,\ldots,m_K$ la suite de K modèles qui maximise le critère choisi ;

$I = 1$ (I est le début de la fenêtre glissante de longueur N)

tant que $I + N \leq T$ faire :

① exécution de l'algorithme de Viterbi (cf. §1.8.2) entre I et $I+N$ pour chaque $HMM2$ de référence ;

② mémorisation des probabilités d'alignement dans la fenêtre de comparaison pour chaque modèle ;

③ comparaison de ces probabilités ;

④ choix du segment correspondant au modèle favorable (soit t_1 son extrémité) au sens d'un critère heuristique de choix prédéfini ;

⑤ éliminer tous les autres alignements ;

⑥ $I = I + t_1 + 1$.

fin tant que

Description

- étape numéro 1 : on applique l'algorithme de Viterbi entre les trames $[I,I+1],[I,I+2],\ldots[I,I+N]$;
- étape numéro 2 : on sauvegarde les N probabilités d'alignement de chaque fenêtre ;
- étape numéro 3 : on étudie la dynamique de ces probabilités pour faire apparaître le meilleur modèle ainsi que la meilleure trame dans ce modèle. Soit t_1 cette trame. Les comparaisons sont faites à l'aide de critères heuristiques pour effectuer le choix du chemin localement optimal ;
- étapes numéro 4,5 et 6 : on retient un modèle et on élimine les autres. Le processus continue à partir de la trame $I + t_1 + 1$.

De par la nature même des choix réalisés dans l'étape numéro 3, l'algorithme de Viterbi-Bloc est sous-optimal. En effet, il ne garantit pas de retenir le chemin globalement optimal pour une phrase complète, c'est-à-dire maximisant la probabilité *a posteriori* de la suite de modèles connaissant l'évidence acoustique. Durant cette étape de comparaison, nous pouvons avantageusement utiliser des marques de segmentation qu'un niveau segmental pourrait avoir préalablement placées [6].

8.3.6 Application de Viterbi-Bloc à la recherche lexicale

Nous avons adapté l'algorithme de Viterbi-Bloc à la recherche lexicale dans le contexte de la machine à dictée vocale. Dans une première passe, l'algorithme construit un treillis de mots à l'aide de deux fenêtres : l'une, dite fenêtre de construction, de longueur L analogue à la fenêtre glissante de Viterbi-Bloc et l'autre de longueur $L + B$, dite fenêtre de prévision, dans laquelle un ou deux mots sont recherchés (figure 8.3). Il n'y a plus de comparaison de probabilités le long de l'arête supérieure de la fenêtre.

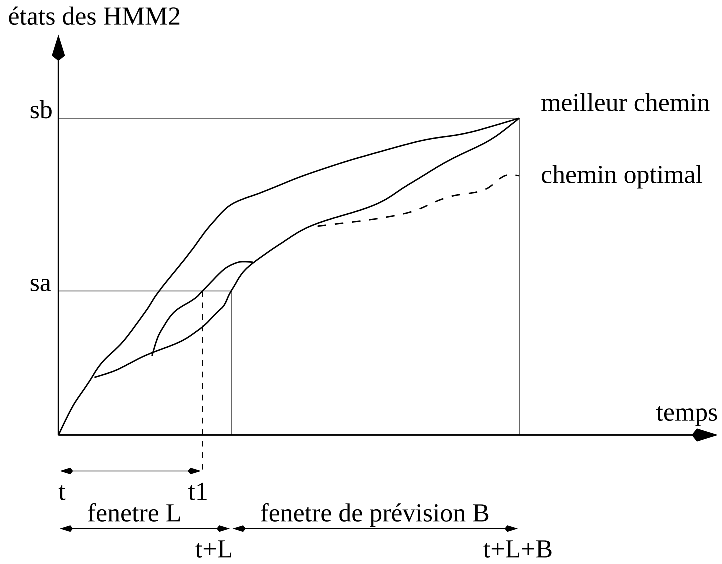


FIG. 8.3 – *Construction du treillis de mots.*

Supposons la reconnaissance effectuée jusqu'à la trame t. On appelle s_a l'état final du mot suivant, pour lequel on cherche les frontières temporelles. Cet état est en fait l'état final du dernier phonème du mot suivant. On applique l'algorithme de Viterbi entre les trames $[t,t+1],[t,t+2],\ldots[t,t+L]$. Aux frontières de mots, le processus de construction utilise un modèle de langage bigrammes. Nous poursuivons le processus de construction à l'aide des mots suivants jusqu'à la trame $t + L + B$. Soit s_b l'état possédant le meilleur score sur la trame $t + L + B$. Dans la fenêtre de prévision, nous venons de construire les chemins correspondant à deux mots supplémentaires.

A cet endroit, rien n'assure que l'on soit sur le chemin optimal donné par un algorithme de Viterbi classique. Mais, si la fenêtre de prévision est suffisamment grande (au plus, elle est égale à la portion restante de la phrase), nous retrouvons le chemin optimal en reculant dans cette fenêtre. En effet, l'hypothèse expérimentale qui a motivé la spécification de l'algorithme de Viterbi-Bloc est que le meilleur chemin s'achevant au point s_b $(t + L + B)$ est optimal entre

t et $t + L$. Finalement, nous nous trouvons à l'instant t_1 dans l'état s_a. Le terme t_1 représente l'extrémité recherchée du premier mot qui s'achève sur l'état s_a. Si ce mot n'a pas déjà été sauvegardé dans le treillis avec ses frontières temporelles, il y sera ajouté. Le début de la fenêtre de construction sera ainsi déplacé sur la trame t_1.

La place mémoire nécessaire à l'exécution ne dépend que de la taille des fenêtres de construction et de prévision, ce qui rend ces algorithmes intéressants pour la reconnaissance de longues phrases. Nous avons effectué différentes expériences sur le corpus de développement afin de déterminer expérimentalement les valeur de L et de B. A l'heure actuelle, les fenêtres de construction et de prévision ont des longueurs variables et correspondent aux durées des mots issus de t et s'achevant respectivement en s_a et s_b ; ceci revient à enregistrer l'avant dernier mot de chaque chemin partiel qui s'achève en s_b à l'instant $t + L + B$. Nous avons constaté que le nombre de mots qui s'achèvent en s_b à l'instant $t + L + B$ est 40 fois plus important que le nombre de mots qui s'achèvent en s_a à l'instant t_1.

Afin d'accélérer le processus, nous utilisons une recherche en faisceau qui détermine si nous devons conserver un alignement s'achevant sur un mot. Nous calculons, à chaque trame, la valeur du seuil qui détermine l'abandon d'un chemin, afin de ne conserver qu'un pourcentage fixe de l'ensemble des chemins.

Deux valeurs influent ainsi sur le bon fonctionnement de cet algorithme : la largeur du faisceau et la longueur de la fenêtre de prévision B.

8.4 Intégration des modèles de langage

Pour évaluer l'ensemble des modèles de langage exposés précédemment, plusieurs versions du système MAUD ont été développées. Pour des raisons de clarté, nous allons les répartir en 4 sous-ensembles : les versions conventionnelles, celles à base de séquences, celles utilisant les grammaires d'unification et celles se servant du modèle hiérarchique. Ces versions ne diffèrent que par le genre du modèle de langage utilisé. Pour les évaluer, nous utilisons les notations suivantes :

- **Corr** : le pourcentage de mots reconnus,
- **Sub** : le pourcentage de substitutions,
- **Supp** : le pourcentage de suppressions,
- **Ins** : le pourcentage d'insertions.

Les résultats sont calculés par les logiciels standards développés par le NIST (*National Institute of Standards and Technology*) [150], notamment les "*Scoring Package*" nommés SCLITE et SCORE [149].

Pour le développement des modèles de langage, nous utilisons un corpus d'environ 50 millions de mots (cf. §4.6). Le vocabulaire de base contient environ 20000 mots.

8.4.1 Les versions conventionnelles

L'objectif principal de ces versions est de tester les capacités de MAUD avec les modèles de langage conventionnels. Quatre versions ont été développées :

- la version de base citée ci-dessus : elle se limite à un modèle bigrammes lors de la première passe et à un modèle trigrammes lors de la deuxième passe de la reconnaissance. La phrase résultat est la meilleure hypothèse fournie par la deuxième étape. Pour estimer les modèles bigrammes et trigrammes, nous procédons avec le même principe développé dans le paragraphe 6.6 pour les modèles n-SeqGrammes. Nous notons par **VM1** cette version.

– une version utilisant un modèle n-classes, nommée **VM2** : dans cette version, nous remplaçons dans $VM1$ le modèle bigrammes de la première passe par un modèle biclasses et le modèle trigrammes de la deuxième passe par un modèle triclasses. Les modèles biclasses et triclasses utilisés sont estimés de la même façon que les modèles n-SeqClasses (cf. §6.7).
– une version se servant du *cache* et du *trigger*, nommée **VM3** : la version VM3 de MAUD se fonde sur un modèle bigrammes lors de la première passe. Dans la deuxième passe, nous utilisons un modèle de langage, résultat d'une interpolation linéaire entre les modèles trigrammes, *cache* et *trigger*. Le modèle trigrammes est celui de la version VM1. Les *triggers* sont estimés de la même manière que celle présentée dans le paragraphe 6.9, en se limitant aux mots.
– une version nommée **VM4** ; elle emploie un modèle bigrammes interpolé avec un modèle biclasses lors de la première passe. Lors de la deuxième passe, elle utilise une interpolation linéaire entre un modèle trigrammes et un modèle triclasses. Le paramètre d'interpolation a est fixé à 0.9 (voir la formule 6.49).

Nous présentons dans la table 8.1 les résultats d'évaluation de ces différentes versions.

	Corr	**Sub**	**Supp**	**Ins**
VM1	60.2	31.0	8.8	5.9
VM2	54.7	35.5	9.8	6.0
VM3	61.7	29.7	8.6	6.5
VM4	60.1	31.1	8.7	6.0

TAB. 8.1 – *Résultats d'évaluation des versions conventionnelles de MAUD.*

L'utilisation des modèles *cache* et *trigger* a peu amélioré les performances de MAUD, par rapport aux modèles n-grammes. Nous pensons que ceci est dû à la nature généraliste de nos *triggers* extraits à partir de textes de journaux. En effet, comme cité auparavant, l'utilisation des *triggers* est intéressante surtout lors de traitement d'un thème spécifique, ce qui n'est pas le cas ici.

La version $VM4$ n'a pas augmenté les performances du système : les résultats de la version utilisant un tel modèle et de celle ne se fondant que sur les n-grammes ($VM1$) sont presque identiques ; pourtant, la version $VM4$ a nécessité un effort de calcul largement supérieur à $VM1$.

En revanche, les performances apportées par l'utilisation des modèles n-classes s'avèrent limitées. A notre avis, ceci provient de la classification manuelle que nous utilisons. Il pourrait être intéressant d'essayer une autre méthode de classification, notamment automatique ; généralement, une telle classification prend mieux en compte la particularité d'un corpus ; ce qui engendre un modèle n-classes avec une meilleure valeur de perplexité par rapport à une classification manuelle [185]. Ceci pourrait ainsi améliorer les performances des versions utilisant les modèles n-classes ($VM2$ et $VM4$).

8.4.2 Les versions à base de séquences

L'objectif des quatre versions présentées dans ce paragraphe est d'évaluer la contribution des séquences *LV* à la modélisation du langage. Ces séquences, au nombre de 4400 environ, sont extraites par l'algorithme présenté dans le paragraphe 6.4. Nous définissons ainsi pour chaque version conventionnelle de MAUD, une version correspondante utilisant les séquences :

– une version employant le modèle n-SeqGrammes, nommée **VS1** : cette version utilise un modèle 2-SeqGrammes lors de la première passe et un modèle 3-SeqGrammes lors de la deuxième passe. L'estimation des modèles 2-SeqGrammes et 3-SeqGrammes se fait d'une manière identique à celle présentée dans le paragraphe 6.6.

- une version se servant du modèle n-SeqClasses, nommée **VS2** : dans cette version de MAUD, nous remplaçons dans $VS1$ le modèle 2-SeqGrammes de la première passe par un modèle 2-SeqClasses et le modèle 3-SeqGrammes de la deuxième passe par un modèle 3-SeqClasses (cf. §6.7).
- une version utilisant le *SeqCache* et le *SeqTrigger* développé dans le chapitre 6 : cette version nommée **VS3** emploie un modèle 2-SeqGrammes lors de la première passe. Elle se sert d'un modèle de langage, résultat d'une interpolation linéaire entre les modèles 3-SeqGrammes, *SeqCache* et *SeqTrigger*, lors de la deuxième passe.
- une version nommé **VS4** ; elle emploie une interpolation linéaire entre les modèles 2-SeqGrammes et 2-SeqClasses lors de la première passe. elle utilise un modèle 3-SeqGrammes interpolé avec un modèle 3-SeqClasses lors de la deuxième passe.

Nous présentons dans la table 8.2 les résultats d'évaluation de ces quatre versions.

	Corr	**Sub**	**Supp**	**Ins**
VS1	70.1	23.5	6.4	6.1
VS2	66.5	26.4	7.1	5.8
VS3	71.9	22.1	5.9	6.8
VS4	70.2	23.2	6.6	6.0

TAB. 8.2 – *Résultats d'évaluation des versions se servant des séquences.*

Les résultats d'évaluation ne font que confirmer l'intérêt de l'utilisation des séquences dans la modélisation du langage. En effet, les performances des versions employant les séquences sont largement supérieures à celles se limitant aux mots. Nous remarquons une amélioration d'environ 19% du taux de reconnaissance, d'environ 25% du taux de substitution et d'environ 29% du taux de suppression. Le taux d'insertion est resté sensiblement le même.

Le commentaires des résultats des versions $VM1$, $VM2$, $VM3$ et $VM4$ reste valable pour ces versions : $VS1$, $VS2$, $VS3$ et $VS4$ respectivement.

Construction du HMM2 d'une séquence :

Comme mentionné auparavant, chaque mot du lexique est représenté par un HMM2 obtenu par la concaténation de HMM2 de phonèmes. Pour construire le HMM2 d'une séquence s_i, nous procédons comme suit : entre tout couple de mots successifs m_1 et m_2 de s_i nous insérons un HMM2 optionnel du phonème silence, construisant ainsi la suite des phonèmes de la séquence s_i. Le HMM2 de s_i est la concaténation de l'ensemble de HMM2 de ces phonèmes. Dans l'état actuel du système MAUD, la probabilité de transition, entre le dernier phonème de m_1 et le premier phonème de m_2, est égale à la probabilité de transition entre ce dernier phonème de m_1 et le phonème silence. Pour améliorer les performances de ces modèles, nous pensons, dans un futur proche, estimer ces probabilités de transition sur le corpus de parole BREF80.

8.4.3 Les versions utilisant l'approche hybride

Nous voulons dans ce paragraphe tester les capacités de l'approche hybride développée dans le chapitre 4. Rappelons que cette approche utilise un modèle de langage probabiliste pour construire les N meilleures hypothèses d'une phrase prononcée. Ces hypothèses seront ensuite étiquetées par l'ensemble des classes syntaxiques et filtrées par le modèle formel (cf. §4.5). Pour la reconnaissance d'une phrase, nous nous limitons dans les versions citées ci-dessous aux 80 meilleures hypothèses.

Deux versions du système MAUD, utilisant des modèles probabilistes différents, ont été développées. La première version, nommée **VMG3**, est une extension de la meilleure version conventionnelle que nous possédons (VM3) : les deux premières passes de VMG3 sont identiques à VM3, et nous ajoutons dans VMG3 une troisième passe de filtrage, utilisant le modèle formel défini dans le chapitre 4. La deuxième version est une extension de la meilleure version se fondant sur les séquences ; elle est donc identique à la version VS3 lors des deux premières passes et intègre le modèle formel lors de la troisième passe. Nous notons par **VSG3** cette version.

Nous présentons dans la table 8.3 les résultats d'évaluation de VMG3 et de VSG3.

	Corr	**Sub**	**Supp**	**Ins**
VMG3	62.2	27.6	10.1	4.9
VSG3	72.0	20.4	7.3	5.3

TAB. 8.3 – *Résultats d'évaluation des versions utilisant l'approche hybride.*

Nous constatons que le modèle formel a éliminé 18% et 22% respectivement de l'ensemble total des hypothèses. Les hypothèses éliminées sont statistiquement probables, mais linguistiquement irréalisables. Nous remarquons aussi que, malgré la légère amélioration du taux de reconnaissance fournie par le modèle formel, son utilisation a apporté une diminution considérable des taux de substitution et d'insertion des mots. En effet, elle a permis de baisser le taux de substitution d'environ 8% et le taux d'insertion d'environ 25%. En revanche, nous avons remarqué une augmentation d'environ 19% du taux de suppression des mots.

Pour augmenter les performances de ces deux modèles, il pourrait être intéressant d'améliorer la méthode d'étiquetage. En effet, en analysant les résultats d'étiquetage des hypothèses fournies par la deuxième passe, nous avons constaté certaines erreurs dues à la nature probabiliste de notre étiqueteur qui se limite aux deux dernières classes (cf. §4.4). Il serait donc important, pour un meilleur étiquetage, d'utiliser un analyseur syntaxique [170]. Ceci permettrait au modèle formel de mieux agir, augmentant ainsi les performances de MAUD.

8.4.4 Les versions se servant des modèles multiclasses et hiérarchique

Pour évaluer les modèles multiclasses et hiérarchique exposés dans le chapitre 5, d'autres versions de MAUD ont été développées ; elles emploient le modèle multiclasses ou le modèle hiérarchique.

Les versions employant le modèle multiclasses :

L'utilisation du modèle multiclasses, avec la classification actuelle, lors des deux premières étapes du système MAUD, n'est pas simple. En effet, elle nécessite la connaissance de la bonne classe de chaque mot durant l'alignement. Or, par manque d'information sur le contexte des premiers mots alignés, ceci est loin d'être évident.

Pour remédier à ce problème, une solution consiste à utiliser une autre méthode de classification ou à transformer celle que nous avons, de façon à ce que chaque mot du lexique ne puisse appartenir qu'à une seule classe. Nous comptons dans un futur proche développer une telle approche.

Pour le moment, nous nous sommes limités à utiliser le modèle multiclasses lors de la troisième étape de filtrage. Nous avons ainsi développé deux versions : la première, nommée **VMM3**, est une extension de la meilleure version utilisant les mots (VM3) ; la deuxième version, nommée **VSM3**, est une extension de la meilleure version se fondant sur les séquences (VS3).

Les deux premières passes de VMM3 et VSM3 sont identiques à celles de VM3 et VS3 respectivement. En revanche, lors de la troisième passe, chacune de ces deux versions étiquette l'ensemble des N hypothèses produites par le niveau inférieur (cf. §4.4), pour les réévaluer par la suite avec le modèle multiclasses. Le résultat est l'hypothèse possédant le meilleur score. Dans le modèle multiclasses utilisé, le nombre maximal de classes, dans une multiclasse, est limité à 8 (optimum en terme de perplexité (cf. §5.5)). La table 8.4 expose les résultats d'évaluation de ces deux versions.

Les versions se servant du modèle hiérarchique :

Comme mentionné auparavant, le modèle hiérarchique a besoin de la totalité de la phrase pour agir. Nous ne l'avons ainsi appliqué que lors de la troisième étape. Deux versions ont été développées : **VMH3** et **VSH3**. Elles sont identiques aux VMM3 et VSM3, avec la différence que le modèle multiclasses a été remplacé par le modèle hiérarchique. Dans ces deux versions nous nous sommes limités à 4 niveaux de hiérarchie et à un nombre maximal de 5 entités dans une multiclasse (optimum en terme de perplexité (cf. §5.5)).

Résultats :

Nous présentons dans la table 8.4 les résultats d'évaluation de ces 4 versions, en se limitant aux 80 meilleures hypothèses produites par les niveaux inférieurs.

	Corr	**Sub**	**Supp**	**Ins**
VMM3	61.2	27.4	11.4	4.1
VSM3	70.8	20.7	8.6	4.1
VMH3	62.7	27.5	9.8	4.7
VSH3	72.5	20.4	7.1	4.8

TAB. 8.4 – *Résultats d'évaluation des versions employant les modèles multiclasses ou hiérarchique.*

L'utilisation du modèle multiclasses a légèrement dégradé le taux de reconnaissance (1.4%) et le taux de suppression (35% environ). En revanche, elle a amélioré le taux de substitution d'environ 8% et le taux d'insertion d'environ 38% par rapport aux $VM3$ et $VS3$. Nous pensons que cette dégradation du taux de reconnaissance est surtout due aux erreurs d'étiquetage (cf. §8.4.3) et aussi à la nature de la classification utilisées. En effet, nous comparons les performances de nos modèles utilisant les classes (triclasses et 3-SeqClasses) avec leurs équivalents ne se fondant que sur les mots (trigrammes et 3-SeqGrammes respectivement) ; nous remarquons alors une différence de plus de 10% en faveur de ceux qui ne se servent que des mots, que se soit en terme de perplexité ou même en terme de taux de reconnaissance.

L'emploi d'une autre méthode de classification, réduisant cet écart entre les deux genres de modèles (avec et sans les classes), pourrait donc augmenter considérablement les performances du modèle multiclasses pour dépasser celles des modèles n-grammes ($n \geq 3$). Comme pour les modèles n-classes, une des méthodes les plus prometteuses est la classification automatique. En effet, les performances en terme de perplexité des modèles de langage utilisant cette classification sont généralement meilleures que celles de leurs équivalents se servant de la classification manuelle [185]. Ceci est dû à la capacité des approches de classifications automatiques de mieux prendre en compte les particularités d'un corpus. L'intégration des modèles multiclasses dans MAUD avec une telle classification pourrait ainsi améliorer ces performances. Malgré, les carences de la classification actuelle, les taux de substitution et d'insertion ont baissé. Nous pensons ainsi que l'utilisation d'un jeu de classes déterministes permettrait d'améliorer les autres taux.

L'intégration du modèle hiérarchique a amélioré les performances de MAUD. En effet, nous remarquons une amélioration d'environ 8% du taux de substitution et d'environ 29% du taux d'insertion. En revanche, le taux de reconnaissance ne progresse que d'environ 1.2% et le taux de suppression se dégrade d'environ 17%. Comme pour le modèle multiclasses, nous pensons que ces deux derniers résultats sont surtout dues à la méthode de classification utilisée et également aux erreurs d'étiquetage. Pour améliorer les performances du modèle hiérarchique, nous pensons comme pour les modèles multiclasses, qu'il serait intéressant de choisir une autre méthode de classification plus performante ainsi qu'un autre étiqueteur plus puissant (analyseur syntaxique par exemple).

Nous avons déjà commencé à travailler dans ce sens en proposant une autre méthode de classification automatique [185]. Nous comptons, dans un futur proche, évaluer les performances des modèles multiclasses et hiérarchique avec cette nouvelle classification.

8.5 La nouvelle version du système MAUD

La version que nous proposons dans ce paragraphe est le résultat d'une combinaison des approches présentées dans ce document. Cette version, nommée **VSHG3**, procède en trois passes : la première passe se fonde sur le modèle 2-SeqGrammes pour la construction d'un treillis de mots ; la deuxième se sert d'un modèle de langage, résultat d'une interpolation linéaire entre les modèles 3-SeqGrammes, *SeqCache* et *SeqTrigger*. Cette passe fournit les N meilleures hypothèses. Enfin, une troisième passe réalise le filtrage et la réestimation de vraisemblances des hypothèses. Les hypothèses produites par la deuxième étape sont étiquetées puis filtrées par le modèle formel. Le score des hypothèses restantes est réévalué par le modèle hiérarchique. L'hypothèse ayant le meilleur score est ainsi considérée comme le résultat de MAUD.

Pour évaluer l'apport des séquences LV, nous avons construit une autre version nommée **VMHG3**. Cette version est identique à VSHG3 à la différence qu'elle utilise un modèle bigrammes lors de la première passe et un modèle trigrammes interpolé avec un modèle *cache* et un modèle *trigger* lors de la deuxième passe. Nous présentons dans la table 8.5 les résultats d'évaluation de ces deux versions, en se limitant aux 80 meilleures hypothèses pour chaque phrase prononcée.

	Corr	**Sub**	**Supp**	**Ins**
VMHG3	63.1	27.3	9.6	4.6
VSHG3	72.8	20.3	7.0	4.7

TAB. 8.5 – *Résultats d'évaluation des nouvelles versions de MAUD.*

Le modèle hiérarchique utilisé se limite à 4 niveaux de hiérarchie et à un nombre maximal de 5 entités dans une multiclasse (optimum en terme de perplexité (cf. §5.5)). Les résultats obtenus prouvent encore une fois l'intérêt de se servir des séquences LV lors de la reconnaissance.

8.6 Conclusion

Nous venons de décrire le système de dictée vocale MAUD auquel nous avons contribué. Ce système a servi à évaluer l'ensemble des modèles de langage cités dans ce document. Cette évaluation nous a permis de mener une étude comparative entre les différentes versions de MAUD et de proposer quelques extensions futures.

L'analyse des résultats des différentes versions de MAUD montre que chacune des approches, présentées dans ce document a contribué, à sa façon, à l'amélioration des performances. L'utilisation des séquences LV lors de la modélisation du langage a augmenté le taux de reconnaissance d'environ 19%. Nous avons aussi constaté que l'approche hybride prend mieux en compte les contraintes de la langue : l'emploi du modèle formel, même s'il a peu fait progresser le taux de reconnaissance (0.5% environ), a permis d'obtenir des solutions plus correctes linguistiquement. De plus, l'utilisation de ce modèle a réduit le taux de substitution d'environ 8% et le taux d'insertion d'environ 25%. Le modèle hiérarchique a contribué à son tour à l'amélioration des performances de MAUD ; même s'il n'est intervenu que lors de la dernière étape, son intégration a permis de perfectionner le taux de reconnaissance d'environ 1.2%, le taux de substitution d'environ 8% et le taux d'insertion d'environ 29%. Il a, par contre, dégradé le taux de suppression d'environ 17%. En revanche, le modèle multiclasses a dégradé les performances de MAUD (1.4% environ). Néanmoins, il a amélioré le taux de substitution de 8% et le taux d'insertion de 38%. Pour faire progresser les performances des modèles formel, hiérarchique et multiclasses, nous pensons qu'il serait intéressant de choisir une autre méthode de classification plus performante ainsi qu'un étiqueteur plus puissant (cf. §8.4.4).

La nouvelle version de MAUD, VSHG3 (cf. §8.5), permet de profiter pleinement des capacités de ces modèles. Par rapport à la version de base du système, elle a amélioré le taux de reconnaissance de 20%, le taux de substitution de 35%, le taux de suppression de 21% et le taux d'insertion de 21% également.

Conclusion

La reconnaissance automatique de la parole est une idée prometteuse dans le domaine de la technologie de l'information, secteur clé de l'industrie du $XXI^{\text{ème}}$ siècle. Néanmoins, beaucoup de problèmes liés à la reconnaissance vocale restent sans solution définitive, notamment ceux liés au respect des contraintes de la langue (chapitre 3). Nous avons présenté dans cette thèse de nouvelles approches en modélisation du langage ; elles définissent notre contribution dans le domaine. Ces approches ont été évaluées par la mesure de la perplexité, testées dans le cadre du jeu de Shannon et enfin intégrées dans notre machine de dictée vocale MAUD, traitant de la parole continue avec de grands vocabulaires.

L'utilisation de la composante formelle de l'approche hybride a permis de mieux prendre en compte la restriction d'accord lors de la reconnaissance. Elle a amélioré le taux de substitution, le taux d'insertion et légèrement le taux de reconnaissance de MAUD. Elle a sensiblement augmenté les capacités prédictives des modèles probabilistes dans le Jeu de Shannon. Dans MAUD, le modèle formel a été employé pour supprimer toutes les hypothèses, statistiquement probables, qui ne vérifient pas la restriction d'accord. Celles-ci sont construites par les niveaux inférieurs avec des modèles de langage probabilistes. Le modèle formel, mis en oeuvre à partir de connaissances linguistiques explicites, n'agit que s'il comporte une règle correspondant à la structure grammaticale de l'hypothèse en cours de traitement. Les hypothèses éliminées étaient toutes linguistiquement invalides. Dans le cas du jeu de Shannon, le modèle formel a été utilisé pour supprimer les mots du vocabulaire qui ne s'accordent pas avec la phrase tronquée. Le modèle probabiliste, en revanche, a été employé pour estimer la mise des mots restants. Étant donné que nous ne possédions qu'une portion de la phrase, le modèle formel n'a pu intervenir que sur un ensemble restreint d'hypothèses, ce qui explique la très légère amélioration du taux de prédiction.

La détection automatique de dépendances et de régularités, dans une phrase, peut être formulée comme un problème d'ajustement des classes syntaxiques de celle-ci en entités supposées capables de traduire sa structure. Dans les approches multiclasses et hiérarchique, ces entités (nommées multiclasses) sont de longueur variable et appartiennent à un ensemble fini. Elles sont construites automatiquement à partir d'un corpus. Les performances du modèle multiclasses, supposant indépendantes les multiclasses d'une phrase, ont été largement supérieures à celles d'un modèle bigrammes ou biclasses. Néanmoins, les capacités de ce modèle se sont avérées limitées par rapport à celles d'un modèle trigrammes ou triclasses. Comparativement à un modèle trigrammes, les performances de MAUD ont subi une légère dégradation du fait de l'utilisation du modèle multiclasses. En introduisant une certaine hiérarchie, prenant en compte la dépendance entre les multiclasses d'une phrase, les capacités prédictives ont augmenté considérablement pour dépasser celles des modèles triclasses et trigrammes. Bien qu'il ne soit intervenu que lors de l'étape du filtrage, le modèle hiérarchique a permis d'améliorer les performances de MAUD. Son pouvoir prédictif le désigne ainsi comme un complément possible aux modèles n-grammes et n-classes. Le mode de distribution des probabilités, entre les multiclasses, dispense d'élaborer des techniques d'interpolation visant à lisser les valeurs estimées des probabilités.

Les aptitudes à utiliser des séquences clés de mots comme unités de base, lors de la prédiction, ont été évaluées dans le contexte de la modélisation probabiliste du langage. Les capacités prédictives des modèles de langage utilisant ces séquences les désignent comme une alternative possible aux modèles conventionnels. En effet, l'utilisation des modèles de langage à base de séquences a considérablement augmenté les performances de MAUD. Elle a amélioré également les valeurs de la perplexité pour dépasser celles des modèles n'employant que les mots. Dans le cadre du jeu de Shannon, les capacités prédictives des modèles se servant de ces séquences ne se sont que légèrement améliorées. Ceci tient à la nature de ce jeu, plus adapté aux modèles n'utilisant que les mots. Ces séquences-clés sont de longueur variable et représentent des structures langagières qui s'apparentent, dans certains cas, à des syntagmes linguistiques. Elles sont détectées, de manière automatique, à partir d'un corpus de mots et de classes syntaxiques, en utilisant des mesures issues de la théorie de l'information. Une fois localisées, ces séquences sont introduites dans le vocabulaire de base et deviennent ainsi de simples entités lexicales.

Le vocabulaire de base joue un rôle essentiel lors de l'extraction des séquences. Pour tester son importance, deux expérimentations ont été menées avec deux vocabulaires différents : V_1 et V_2 ; V_1 est construit à partir des mots les plus fréquents du corpus *Monde/87-88* ; V_2 est le résultat d'un traitement effectué sur le premier vocabulaire. L'objectif de ce traitement est d'obtenir des mots plus cohérents d'un point de vue linguistique ; par exemple, "aujourd'hui", qui est perçu comme deux entrées distinctes dans V_1 ("aujourd" et "hui"), est considéré comme un seul mot dans V_2. Le vocabulaire V_2, utilisé pour l'évaluation des approches présentées dans ce document, a donné de meilleurs résultats. En se servant de V_1, le nombre de séquences extraites est largement supérieur à celui obtenu à partir de V_2 ; plusieurs séquences extraites à partir de V_1 constituent des unités connues *a priori*, appartenant à V_2. L'amélioration des performances apportée par ces séquences, aux modèles utilisant V_1, est largement supérieure à celle apportée par les séquences LV aux modèles se servant de V_2. Les séquences LV sont extraites à partir du vocabulaire V_2.

La version de base de notre système de dictée vocale MAUD se fonde sur les modèles de Markov cachés non contextuels de second ordre et sur un modèle de langage n-grammes. Cette version, mais avec un modèle diphones, a été classée seconde dans la première campagne d'évaluation de l'Aupelf-UREF. Dans la nouvelle version de MAUD, un modèle se servant des séquences LV (n-SeqGrammes), est utilisé lors des deux premières étapes pour fournir les N meilleures hypothèses, correspondant à la phrase prononcée. Ces hypothèses sont ensuite filtrées par le modèle formel, pour être enfin réévaluées avec le modèle hiérarchique. Le résultat du système est la meilleure hypothèse restante. Dans cette version de MAUD, les performances se sont améliorées considérablement par rapport à celles de la version de base : de 20% pour le taux de reconnaissance, 35% pour le taux de substitution et 21% pour les taux de suppression et d'insertion.

Nous sommes convaincus que les systèmes de reconnaissance vocale utilisant de grands vocabulaires auront toujours un taux d'erreurs différent de zéro. La réduction au minimum de ce taux d'erreurs nécessiterait l'introduction de connaissances encyclopédiques, mais surtout, d'un mécanisme d'inférence et d'auto-apprentissage très complexe. Néanmoins, dans le cadre de la dictée automatique, la présence d'un nombre minime d'erreurs ne présente pas un risque important dans le cas où le document est relu par un humain.

Perspectives

L'étude menée dans cette thèse nous a permis, en plus du travail effectué, d'ouvrir de nouvelles voies de recherches que nous envisageons d'explorer dans un futur proche.

Pour améliorer les performances des modèles se servant des classes syntaxiques, nous pensons qu'il est important d'utiliser une autre méthode d'étiquetage plus puissante. Nous suggérons ainsi l'utilisation d'une approche hybride combinant notre étiqueteur probabiliste avec un analyseur syntaxique se fondant sur des connaissances explicites [42].

Il pourrait être intéressant de donner au modèle formel un rôle correctif plutôt qu'un rôle de filtrage ; chaque fois qu'une hypothèse est jugée non valide, au lieu de la supprimer, nous demandons au modèle de la corriger, en mettant à jour l'ensemble des mots qui ne vérifient pas l'accord. Pour ce faire, une solution consiste à commencer tout d'abord par définir les valeurs de traits (le genre, le nombre, la personne, la forme du verbe) de chacune de ces hypothèses. Nous pouvons prendre, par exemple, la tendance majoritaire des valeurs de traits attribuées aux sous-constituants. Ensuite, il ne reste qu'à corriger les mots qui ne vérifient pas la restriction d'accord en fonction de ces valeurs. Dans le cas où une ambiguïté est rencontrée lors de la définition des valeurs de traits de l'hypothèse, une solution consiste à proposer plusieurs ensembles de valeurs produisant plusieurs hypothèses linguistiquement correctes. Enfin, une réévaluation de l'ensemble des hypothèses, après correction, est nécessaire pour déterminer l'hypothèse résultat. En utilisant cette méthode, nous risquons de proposer des solutions qui sont très loin de ce qui a été prononcé. Nous pensons ainsi qu'il est nécessaire de pénaliser les hypothèses subissant une correction et de ne corriger que celles qui sont très vraisemblables.

Nous comptons également tester les capacités de l'approche multiclasses dans l'extraction des séquences de mots : soit $M_C^* = \{m_{c_i}^*\}$ l'ensemble des entités multiclasses, appris sur un corpus C_A ; soit un corpus de mots différent C_W et le corpus de classes syntaxiques correspondant C_C. Nous commençons tout d'abord par définir la meilleure segmentation de C_C en utilisant les multiclasses M_C^*, construisant ainsi C_{C_1}. Nous traitons ensuite le corpus C_W pour extraire le corpus C_{W_1} qui correspond à C_{C_1} ; chaque entité de C_{W_1} est une séquence de mots (ou un mot) correspondant à une multiclasse de C_{C_1}. Enfin, nous trions toutes les séquences de C_{W_1} suivant leur fréquence d'apparition et nous ne prenons que celles qui minimisent la perplexité. Ces séquences seront introduites dans MAUD, de la même manière que les séquences LV, en utilisant les modèles n-SeqGrammes.

Une extension des modèles n-SeqGrammes et n-SeqClasses consisterait à augmenter la taille de l'historique traité lors de la prédiction (par exemple, $n \approx 4$ ou 5). Néanmoins, cette extension pourrait souffrir du manque de données et pourrait également augmenter considérablement le nombre de suites de séquences non rencontrées dans le corpus ; le nombre de paramètres à estimer pourrait devenir très élevé. Pour résoudre ce problème, une idée consisterait à utiliser un historique dont le nombre de séquences serait variable ; le nombre de séquences dans un historique serait limité à n. Nous définissons alors une fonction $\Im(.)$ qui prend en argument les n dernières séquences de l'historique, pour fournir les m ($m \leq n$) séquences utiles à la prédiction. Par conséquent, la probabilité conditionnelle $p(s_n/s_1^{n-1})$ peut être formulée comme suit :

$$p(s_n/s_1^{n-1}) = p(s_n/\Im(s_1^{n-1})) = p(s_n/s_{n-m+1}^{n-1})$$

où $m \leq n$. La probabilité $p(s_n/s_{n-m+1}^{n-1})$ peut être estimée de la même façon que pour les modèles n-SeqGrammes ou n-SeqClasses, en utilisant la méthode de Katz. Pour déterminer la fonction $\Im(.)$, il suffit de construire un arbre contenant l'ensemble des séquences les plus fréquentes, où chaque noeud de cet arbre représente un historique distinct. Cet arbre contient toutes les suites de séquences qui sont assez fréquentes dans le corpus. La fonction $\Im(.)$ extrait ainsi, à partir de cet arbre, le noeud qui correspond le mieux à l'historique en argument. Pour construire ce modèle, nous pouvons utilisé le même principe que celui utilisé par R. Kneser et T.R. Niesler dans [109, 144]. En effet, chacun de ces deux chercheurs propose une méthode permettant d'utiliser un historique de mots de taille variable lors de la prédiction. Nous proposons alors d'exploiter les avantages de cette approche au profit des modèles n-SeqGrammes et n-SeqClasses. Une autre extension consisterait à utiliser cette approche à base de séquences au niveau des modèles de langage distants (cf. §3.4.6).

Pour augmenter les performances des modèles de langage présentés dans ce document, nous proposons de les combiner de façon à prendre en compte les avantages de chacun. Nous proposons la construction d'une hiérarchie multidimensionnelle de partitions, où chacune d'entre elles correspond à un modèle de langage donné. Ensuite, pour estimer la vraisemblance, nous choisissons les partitions les plus prometteuses en fonction du contexte. Il est alors nécessaire d'établir un certain traitement, pour prendre en compte l'ensemble des contraintes de la théorie des probabilités et également pour définir automatiquement les partitions candidates. Pour ce faire, nous pouvons adapter l'approche proposée par P. Dupont et R. Rosenfeld dans [55]. Dans cet article, les auteurs proposent une idée originale qui consiste à choisir, parmi les modèles n-grammes et n-classes, celui qui estime le mieux la vraisemblance en fonction d'un historique donné. Cette idée peut être considérée comme une généralisation de la méthode d'interpolation de Katz. En effet, pour estimer la vraisemblance des événements non rencontrés dans le corpus d'apprentissage, au lieu d'utiliser le même modèle mais d'ordre inférieur (par exemple, (n-1)-SeqGrammes pour un modèle n-SeqGrammes), il suffit d'utiliser un modèle différent moins précis (par exemple, n-SeqClasses). Ce choix, qui peut être une combinaison de plusieurs modèles de langage différents, est donc variable en fonction du contexte.

Pour améliorer les performances de MAUD, nous pensons qu'il est nécessaire d'intervenir sur d'autres niveaux. Notre équipe travaille ainsi sur un modèle triphones pour une meilleure modélisation acoustique. Nous jugeons également important l'utilisation d'un modèle de langage plus puissant que le 2-SeqGrammes lors de la première passe ; nous songeons donc à l'utilisation du modèle 3-SeqGrammes. Pour chaque couple de mots (m_1,m_2) appartenant à une séquence donnée, la probabilité de transition entre le dernier phonème de m_1 et le premier phonème de m_2 est égale à la probabilité de transition entre ce dernier phonème de m_1 et le phonème silence. Nous comptons, dans un futur proche, estimer cette probabilité de transition sur le corpus de parole BREF80. Ceci nous permettrait de mieux prendre en compte les capacités prédictives de ces séquences (cf. §6.2).

Les approches, multiclasses, hiérarchiques et d'extraction des séquences LV, offrent plusieurs perspectives intéressantes dans le cadre de la modélisation statistique et dans celui de la compréhension du langage. En effet, il pourrait être intéressant d'utiliser ces approches pour regrouper automatiquement les suites de mots ayant sens en classes possédant un même comportement sémantique. Ces classes sémantiques, avec les classes syntaxiques que nous possédons déjà, permettraient d'améliorer les performances des modèles de langage, des systèmes de reconnaissance vocale et également des modèles de compréhension automatique de la langue.

Annexe A

Les classes syntaxiques

Les adverbes

AD1	→	les autres adverbes	
AQU	→	adv	peu, beaucoup, trop
AUS	→	adv	aussi
MOP	→	adv	moins, plus
PAS	→	adv	pas
X1S	→	adv	désormais
XAI	→	adv	ailleurs
XAL	→	adv	alentour
XAM	→	adv	jamais
XAN	→	adv	dedans
XAP	→	adv	après
XAR	→	adv	arrière
XAS	→	adv	assez
XAT	→	adv	attenant
XAU	→	adv	autant
XAV	→	adv	avant
XBI	→	adv	bien
XC5	→	adv	ça
XCE	→	adv	certes
XCI	→	adv	ci
XCO	→	adv	combien
XDA	→	adv	davantage
XDE	→	adv	debout
XDH	→	adv	dehors
XDO	→	adv	dorénavant
XE1	→	adv	déjà
XEM	→	adv	demain
XEN	→	adv	ensemble
XEP	→	adv	depuis
XES	→	adv	dessous
XEV	→	adv	devant
XEX	→	adv	exprès
XFO	→	adv	fort

XGU	→	adv	guère
XHI	→	adv	hier
XIC	→	adv	ici
XIE	→	adv	bientôt
XIN	→	adv	ainsi
XIR	→	adv	environ
XIT	→	adv	sitôt
XJA	→	adv	jadis
XLA	→	adv	là
XLO	→	adv	alors
Xlo	→	adv	loin
XMA	→	adv	mal
XMI	→	adv	mieux
XNC	→	adv	encore
XNF	→	adv	enfin
XNO	→	adv	non
XNS	→	adv	ensuite
XNT	→	adv	maintenant
XOM	→	adv	comment
XON	→	adv	longtemps
XOR	→	adv	lors
XOU	→	adv	souvent
Xou	→	adv	outre
XPA	→	adv	parfois
Xpa	→	adv	partout
XPI	→	adv	pis
XPL	→	adv	plutôt
XPR	→	adv	presque
Xpr	→	adv	près
XPU	→	adv	puis
XQU	→	adv	quasi
XRD	→	adv	tard
XRE	→	adv	très
XRO	→	adv	proche
XSI	→	adv	si
XSO	→	adv	soudain
XSS	→	adv	dessus
XTA	→	adv	tant
Xta	→	adv	tantôt
XTO	→	adv	tôt
Xto	→	adv	toujours
XTR	→	adv	contre
XU2	→	adv	où
XUE	→	adv	quelquefois
XUJ	→	adv	aujourd'hui
XUP	→	adv	auparavant
XUR	→	adv	autour
XUS	→	adv	aussitôt

XUT	→	adv	autrefois
XVI	→	adv	vite
XVO	→	adv	volontiers
XWI	→	adv	oui

Les adjectifs

ADE	→	adj démonstratifs	ce, cet, cette, ces
ADJ	→	les autres adjectifs	
ANL	→	adj indéfinis	aucun(e)(s), nul(le)(s)
APO	→	adj possessifs	ma, ta, sa, son, leur, ...
AUT	→	adj indéfinis	autre(s)
CER	→	adj indéfinis	certain(e)(s)
CHA	→	adj indéfinis	chaque, n'importe-quel(le)
DIF	→	adj indéfinis	différent(e)(s), divers(e)(s), maint(e)(s)
PLU	→	adj indéfini	plusieurs
QEL	→	adj indéfini	quel
QLQ	→	adj indéfinis	quelconque(s), quelque(s), tel(le)(s)
QUL	→	adj interrogatifs	quel(le)(s)
SEA	→	participe présent pouvant être adjectif	

Les articles

ARD	→	articles définis
ARI	→	articles indéfinis
AUX	→	articles contractés : au(x)

Les conjonctions

AFQ	→	locutions conjonctives se terminant par que (ex : de peur que)	
AIS	→	conjonction	mais
CAU	→	conjonctions	car, parceque
CON	→	les autres conjonctions qui n'apparaissent pas dans cette liste	
DON	→	conjonction	donc
E_O	→	conjonctions	et, ou, oubien
KEU	→	conjonction	que
MNI	→	conjonction	ni
OOR	→	conjonction	or
SIN	→	conjonction	sinon
SSI	→	conjonction	si
Xso	→	conjonction	soit

Les noms

JOU	→	jours de la semaine
MOI	→	mois de l'année
NOM	→	noms communs
NOP	→	noms propres

Les prépositions

AA2	→	préposition	a
APR	→	préposition	après
AVE	→	prépositions	avec, contre
CDD	→	prépositions	chez, devant, derrière, envers, vers
D2S	→	préposition	dès
DAN	→	prépositions	dans, via
DEP	→	prépositions	avant, depuis
EEN	→	préposition	en
HEX	→	prépositions	excépté, hormis
HOR	→	préposition	hors
JUS	→	préposition	jusque
LOP	→	locutions prépositives (ex : afin_de, ...)	
MAL	→	préposition	malgrè
MPL	→	prépositions	moins, plus
OUT	→	préposition	outre
PAR	→	prépositions	entre, par
PDT	→	préposition	pendant
PMI	→	préposition	parmi
PR2	→	prépositions	sous, sur
PRE	→	les autres prépositions	
PUR	→	préposition	pour
SAN	→	préposition	sans
SAU	→	préposition	sauf
SEL	→	préposition	selon
VVO	→	prépositions présentatives : voici, voilà	

Les pronoms

CCE	→	pronom	ce
CEC	→	pronoms	ça, ceci, cela
CEL	→	pronoms démonstratifs	celui, celle(s), ceux
CEU	→	pronoms	celui-ci, celui-là, ...
DDO	→	pronom relatif	dont
DKL	→	pronoms relatifs	duquel, de laquelle, desquelles, desquels
HAC	→	pronoms indéfinis	chacun(e)(s)
IN1	→	pronoms indéfinis	aucun(e)
IN2	→	pronoms indéfinis	nul(le)
IN3	→	pronom indéfini	personne
KWA	→	pronom relatif	quoi
LKL	→	pronoms relatifs	lequel(s), laquelle(s)
MME	→	pronoms attributs renforcés : moi même, toi même, ...	
OKL	→	pronoms relatifs	a laquelle, au(x)quel(s), auxquelles
PAT	→	pronoms attributs	moi, toi, elle(s), lui, nous, vous, eux
PCE	→	pronoms indéfinis	certain(e)(s), plusieurs
PCR	→	pronoms compléments D/I ou réfléchis : me, te, se, nous, vous	

PI1	→	pronom interrogatif	qui
PI2	→	pronom interrogatif	que
PI3	→	pronoms interrogatifs	lequel(s), laquelle(s)
PI4	→	pronom interrogatif	quoi
PID	→	pronoms indéfinis	beaucoup, peu, tant, trop, la plupart
PPD	→	pronoms personnels directs	la, le, les
PPE	→	pronoms personnels	je, tu, ...
PPI	→	pronoms personnels indirects	lui, leur
PPO	→	pronoms possessifs	le mien, le tien, ...
PQU	→	pronoms invariables	quiconque
PRO	→	les autres pronoms	
PTE	→	pronom indéfini	tel
QQN	→	pronoms indéfinis	quelqu'un, quelque chose, n'importe qui, n'importe quoi
QUE	→	pronom relatif	que
QUI	→	pronom relatif	qui
RIE	→	pronom indéfini	rien
SOI	→	pronom	soi
TRU	→	pronom indéfini	autrui
UNS	→	pronoms indéfinis	un(s)
UTR	→	pronoms indéfinis	autre(s)

Les verbes

AVO	→	auxiliaire avoir
ETR	→	auxiliaire être
PPa	→	participe passé s'employant uniquement avec l'auxiliaire avoir
PPe	→	participe passé s'employant avec l'auxiliaire être
PPx	→	participe passé s'employant avec les deux auxiliaires
SEN	→	participe présent
VEC	→	verbes conjugués
VEI	→	verbes à l'infinitif

Les autres

CAR	→	cardinaux	deux, trois, ...
COM	→		comme
DDE	→		de
DDU	→		du
DES	→		des
ENY	→		en & y
INT	→	les interjections	hélas, ...
MEM	→	adj/adv/pro	même(s)
NEG	→	négation	ne
ORD	→	ordinaux	deuxième, troisième, ...
OUU	→	adv/pro	où

PH1	$\rightarrow$	classe euphonique contenant le "l" apostrophe
PH2	$\rightarrow$	classe euphonique contenant le "t"
TOU	$\rightarrow$	adj/adv/nom/pro tout, tout(e)(s)

Annexe B

Les règles grammaticales

Le but de ces règles est de supprimer des phrases qui ne vérifient pas les règles d'accord. On utilisera dans ces règles les notations suivantes :

- * : pour désigner une classe syntaxique (voir annexe A) ;
- ** : pour désigner le mot ;
- GENRE : pour désigner le trait du genre ;
- NBP : pour désigner le trait du nombre et de la personne ;
- v_i : pour spécifier un élément (terminal ou non terminal) quelconque, de ceux du contexte droite de la règle de réécriture.

Une règle ne peut être appliquée que si l'ensemble des contraintes est vérifié. Le résultat de la contrainte "EstNonVide_∩" est un booléen indiquant si l'intersection entre les valeurs des traits en argument est non nulle. Le résultat de la contraint "Englobe" permet de définir la structure de trait, résultat de la superposition des structures de trait en argumant.

Les règles

P	→	GN1 / GN2 / GN2 **que PEGN2 / *VEC
P	→	GN1 GV1
		EstNonVide_∩ (<GN1, GENRE>, <GV1, GENRE>)
		EstNonVide_∩ (<GN1, NBP>, <GV1, NBP>)
		EstNonVide_∩ (<GN1, NOMBRE>, <GV1, NOMBRE>)
P	→	GN2 GV1
		EstNonVide_∩ (<GN2, GENRE>, <GV1, GENRE>)
		EstNonVide_∩ (<GN2, NBP>, <GV1, NBP>)
		EstNonVide_∩ (<GN2, NOMBRE>, <GV1, NOMBRE>)
P	→	GN2 **que PEGN2 *AVO
		EstNonVide_∩ (<PEGN2, GENRE>, <*AVO, GENRE>)
		EstNonVide_∩ (<PEGN2, NBP>, <*AVO, NBP>)
		EstNonVide_∩ (<PEGN2, NOMBRE>, <*AVO, NOMBRE>)
P	→	GN2 **que PEGN2 *AVO PGN
		EstNonVide_∩ (<PEGN2, GENRE>, <*AVO, GENRE>)
		EstNonVide_∩ (<PEGN2, NBP>, <*AVO, NBP>)
		EstNonVide_∩ (<PEGN2, NOMBRE>, <*AVO, NOMBRE>)

P → GN2 **que PEGN2 *AVO PPA-X
EstNonVide_∩ (<GN2, GENRE>, <PPA-X, GENRE>)
EstNonVide_∩ (<GN2, NBP>, <PPA-X, NBP>)
EstNonVide_∩ (<GN2, NOMBRE>, <PPA-X, NOMBRE>)
EstNonVide_∩ (<PEGN2, GENRE>, <*AVO, GENRE>)
EstNonVide_∩ (<PEGN2, NBP>, <*AVO, NBP>)
EstNonVide_∩ (<PEGN2, NOMBRE>, <*AVO, NOMBRE>)

P → GN2 **que PEGN2 *AVO PPA-X PGN
EstNonVide_∩ (<GN2, GENRE>, <PPA-X, GENRE>)
EstNonVide_∩ (<GN2, NBP>, <PPA-X, NBP>)
EstNonVide_∩ (<GN2, NOMBRE>, <PPA-X, NOMBRE>)
EstNonVide_∩ (<PEGN2, GENRE>, <*AVO, GENRE>)
EstNonVide_∩ (<PEGN2, NBP>, <*AVO, NBP>)
EstNonVide_∩ (<PEGN2, NOMBRE>, <*AVO, NOMBRE>)

GN1 → *PAT *PPE
EstNonVide_∩ (<*PAT, GENRE>, <*PPE, GENRE>)
EstNonVide_∩ (<*PAT, NBP>, <*PPE, NBP>)
EstNonVide_∩ (<*PAT, NOMBRE>, <*PPE, NOMBRE>)
GENRE := ∩ (<*PAT, GENRE>, <*PPE, GENRE>)
NBP := ∩ (<*PAT, NBP>, <*PPE, NBP>)
NOMBRE := ∩ (<*PAT, NOMBRE>, <*PPE, NOMBRE>)

GN1 → *PPE
GENRE := <*PPE, GENRE>
NBP := <*PPE, NBP>
NOMBRE := <*PPE, NOMBRE>

GN2 → DNA / *NOM / *NOP / TITRE
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

GN2 → DNA *E_O GN2
GENRE := Englobe(<DNA, GENRE>, <GN2, GENRE>)
NBP := Englobe(<DNA, NBP>, <GN2, NBP>)
NOMBRE := Englobe(<DNA, NOMBRE>, <GN2, NOMBRE>)

GN2 → DNA DEUU GN2
GENRE := <DNA, GENRE>
NBP := <DNA, NBP>
NOMBRE := <DNA, NOMBRE>

GN2 → TITRE *NOM
EstNonVide_∩ (<TITRE, GENRE>, <*NOM, GENRE>)
EstNonVide_∩ (<TITRE, NBP>, <*NOM, NBP>)
EstNonVide_∩ (<TITRE, NOMBRE>, <*NOM, NOMBRE>)
GENRE := ∩ (<TITRE, GENRE>, <*NOM, GENRE>)
NBP := ∩ (<TITRE, NBP>, <*NOM, NBP>)
NOMBRE := ∩ (<TITRE, NOMBRE>, <*NOM, NOMBRE>)

GN2 → TITRE *NOP
EstNonVide_∩ (<TITRE, GENRE>, <*NOP, GENRE>)
EstNonVide_∩ (<TITRE, NBP>, <*NOP, NBP>)
EstNonVide_∩ (<TITRE, NOMBRE>, <*NOP, NOMBRE>)
GENRE := ∩ (<TITRE, GENRE>, <*NOP, GENRE>)
NBP := ∩ (<TITRE, NBP>, <*NOP, NBP>)
NOMBRE := ∩ (<TITRE, NOMBRE>, <*NOP, NOMBRE>)

GN2 → TITRE DNA
EstNonVide_∩ (<TITRE, GENRE>, <DNA, GENRE>)
EstNonVide_∩ (<TITRE, NBP>, <DNA, NBP>)
EstNonVide_∩ (<TITRE, NOMBRE>, <DNA, NOMBRE>)
GENRE := ∩ (<TITRE, GENRE>, <DNA, GENRE>)
NBP := ∩ (<TITRE, NBP>, <DNA, NBP>)
NOMBRE := ∩ (<TITRE, NOMBRE>, <DNA, NOMBRE>)

GN2 → TITRE DNA *E_O GN2
EstNonVide_∩ (<TITRE, GENRE>, <DNA, GENRE>)
EstNonVide_∩ (<TITRE, NBP>, <DNA, NBP>)
EstNonVide_∩ (<TITRE, NOMBRE>, <DNA, NOMBRE>)
GENRE := Englobe(<TITRE, GENRE>, <DNA, GENRE>, <GN2, GENRE>)
NBP := Englobe(<TITRE, NBP>, <DNA, NBP>, <GN2, NBP>)
NOMBRE := Englobe(<TITRE, NOMBRE>, <DNA, NOMBRE>, <GN2, NOMBRE>)

GN2 → TITRE DNA DEUU GN2
EstNonVide_∩ (<TITRE, GENRE>, <DNA, GENRE>)
EstNonVide_∩ (<TITRE, NBP>, <DNA, NBP>)
EstNonVide_∩ (<TITRE, NOMBRE>, <DNA, NOMBRE>)
GENRE := ∩ (<TITRE, GENRE>, <DNA, GENRE>)
NBP := ∩ (<TITRE, NBP>, <DNA, NBP>)
NOMBRE := ∩ (<TITRE, NOMBRE>, <DNA, NOMBRE>)

GV1 → PPI-D *PAT GV
GENRE := <GV, GENRE>
NBP := <GV, NBP>
NOMBRE := <GV, NOMBRE>
VFORM := <GV, VFORM>

GV1 → PPI-D GV
GENRE := <GV, GENRE>
NBP := <GV, NBP>
NOMBRE := <GV, NOMBRE>
VFORM := <GV, VFORM>

GV1 → GV
GENRE := <GV, GENRE>
NBP := <GV, NBP>
NOMBRE := <GV, NOMBRE>
VFORM := <GV, VFORM>

GV1 → *PCR GV
GENRE := <GV, GENRE>
NBP := <GV, NBP>
NOMBRE := <GV, NOMBRE>
VFORM := <GV, VFORM>

GV2 → GN2 / PPI-D *VEI / PPI-D *VEI PGN / PREP *AVO *PPa
PREP *AVO *PPx / PREP *ETR *PPE /
PREP *ETR *PPx / PREP GN2 / PREP *VEI

GV → *VEC
GENRE := <VEC, GENRE>
NBP := <VEC, NBP>
NOMBRE := <VEC, NOMBRE>
VFORM := <VEC, VFORM>

GV → *VEC GV2
GENRE := <VEC, GENRE>
NBP := <VEC, NBP>
NOMBRE := <VEC, NOMBRE>
VFORM := <VEC, VFORM>

GV → *AVO PPI-D PGN
GENRE := <AVO, GENRE>
NBP := <AVO, NBP>
NOMBRE := <AVO, NOMBRE>
VFORM := <AVO, VFORM>

GV → *AVO *PPa
EstNonVide_∩: <AVO, GENRE><PPa, GENRE>
EstNonVide_∩: <AVO, NBP><PPa, NBP>
EstNonVide_∩: <AVO, NOMBRE><PPa, NOMBRE>
EstNonVide_∩: <AVO, VFORM><PPa, VFORM>
GENRE := ∩: (<AVO, GENRE>, <PPa, GENRE>)
NBP := ∩: (<AVO, NBP>, <PPa, NBP>)
NOMBRE := ∩: (<AVO, NOMBRE>, <PPa, NOMBRE>)
VFORM := ∩: (<AVO, VFORM>, <PPa, VFORM>)

GV → *AVO *PPx
EstNonVide_∩: <AVO, GENRE><PPx, GENRE>
EstNonVide_∩: <AVO, NBP><PPx, NBP>
EstNonVide_∩: <AVO, NOMBRE><PPx, NOMBRE>
EstNonVide_∩: <AVO, VFORM><PPx, VFORM>
GENRE := ∩: (<AVO, GENRE>, <PPx, GENRE>)
NBP := ∩: (<AVO, NBP>, <PPx, NBP>)
NOMBRE := ∩: (<AVO, NOMBRE>, <PPx, NOMBRE>)
VFORM := ∩: (<AVO, VFORM>, <PPx, VFORM>)

GV → *AVO *PPa PGN
EstNonVide_∩: <AVO, GENRE><PPa, GENRE>
EstNonVide_∩: <AVO, NBP><PPa, NBP>
EstNonVide_∩: <AVO, NOMBRE><PPa, NOMBRE>
EstNonVide_∩: <AVO, VFORM><PPa, VFORM>
GENRE := ∩: (<AVO, GENRE>, <PPa, GENRE>)
NBP := ∩: (<AVO, NBP>, <PPa, NBP>)
NOMBRE := ∩: (<AVO, NOMBRE>, <PPa, NOMBRE>)
VFORM := ∩: (<AVO, VFORM>, <PPa, VFORM>)

GV → *AVO *PPx PGN
EstNonVide_∩: <AVO, GENRE><PPx, GENRE>
EstNonVide_∩: <AVO, NBP><PPx, NBP>
EstNonVide_∩: <AVO, NOMBRE><PPx, NOMBRE>
EstNonVide_∩: <AVO, VFORM><PPx, VFORM>
GENRE := ∩: (<AVO, GENRE>, <PPx, GENRE>)
NBP := ∩: (<AVO, NBP>, <PPx, NBP>)
NOMBRE := ∩: (<AVO, NOMBRE>, <PPx, NOMBRE>)
VFORM := ∩: (<AVO, VFORM>, <PPx, VFORM>)

GV → *ETR PPI-D PGN
GENRE := <ETR, GENRE>
NBP := <ETR, NBP>
NOMBRE := <ETR, NOMBRE>
VFORM := <ETR, VFORM>

GV → *ETR *PPE
EstNonVide_∩: <ETR, GENRE><PPE, GENRE>
EstNonVide_∩: <ETR, NBP><PPE, NBP>
EstNonVide_∩: <ETR, NOMBRE><PPE, NOMBRE>
EstNonVide_∩: <ETR, VFORM><PPE, VFORM>
GENRE := ∩: (<ETR, GENRE>, <PPE, GENRE>)
NBP := ∩: (<ETR, NBP>, <PPE, NBP>)
NOMBRE := ∩: (<ETR, NOMBRE>, <PPE, NOMBRE>)
VFORM := ∩: (<ETR, VFORM>, <PPE, VFORM>)

GV → *ETR *PPx
EstNonVide_∩: <ETR, GENRE><PPx, GENRE>
EstNonVide_∩: <ETR, NBP><PPx, NBP>
EstNonVide_∩: <ETR, NOMBRE><PPx, NOMBRE>
EstNonVide_∩: <ETR, VFORM><PPx, VFORM>
GENRE := ∩: (<ETR, GENRE>, <PPx, GENRE>)
NBP := ∩: (<ETR, NBP>, <PPx, NBP>)
NOMBRE := ∩: (<ETR, NOMBRE>, <PPx, NOMBRE>)
VFORM := ∩: (<ETR, VFORM>, <PPx, VFORM>)

GV → *ETR *PPE PGN
EstNonVide_∩: <ETR, GENRE><PPE, GENRE>
EstNonVide_∩: <ETR, NBP><PPE, NBP>
EstNonVide_∩: <ETR, NOMBRE><PPE, NOMBRE>
EstNonVide_∩: <ETR, VFORM><PPE, VFORM>
GENRE := ∩: (<ETR, GENRE>, <PPE, GENRE>)
NBP := ∩: (<ETR, NBP>, <PPE, NBP>)
NOMBRE := ∩: (<ETR, NOMBRE>, <PPE, NOMBRE>)
VFORM := ∩: (<ETR, VFORM>, <PPE, VFORM>)

GV → *ETR *PPx PGN
EstNonVide_∩: <ETR, GENRE><PPx, GENRE>
EstNonVide_∩: <ETR, NBP><PPx, NBP>
EstNonVide_∩: <ETR, NOMBRE><PPx, NOMBRE>
EstNonVide_∩: <ETR, VFORM><PPx, VFORM>
GENRE := ∩: (<ETR, GENRE>, <PPx, GENRE>)
NBP := ∩: (<ETR, NBP>, <PPx, NBP>)
NOMBRE := ∩: (<ETR, NOMBRE>, <PPx, NOMBRE>)
VFORM := ∩: (<ETR, VFORM>, <PPx, VFORM>)

ADJ1 → *ADJ
GENRE := <*ADJ, GENRE>
NBP := <*ADJ, NBP>
NOMBRE := <*ADJ, NOMBRE>
ADJ1 → *ADJ *E_O *ADJ
GENRE := Englobe(<*ADJ, GENRE>, <*ADJ, GENRE>)
NBP := Englobe(<*ADJ, NBP>, <*ADJ, NBP>)
NOMBRE := Englobe(<*ADJ, NOMBRE>, <*ADJ, NOMBRE>)

DEUU → **au / **de / **du
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

DNA → TED NTN
EstNonVide_∩ (<TED, GENRE>, <NTN, GENRE>)
EstNonVide_∩ (<TED, NBP>, <NTN, NBP>)
EstNonVide_∩ (<TED, NOMBRE>, <NTN, NOMBRE>)
GENRE := ∩ (<TED, GENRE>, <NTN, GENRE>)
NBP := ∩ (<TED, NBP>, <NTN, NBP>)
NOMBRE := ∩ (<TED, NOMBRE>, <NTN, NOMBRE>)

DNA → TED NTN ADJ1
EstNonVide_∩ (<TED, GENRE>, <NTN, GENRE>,
<ADJ1, GENRE>)
EstNonVide_∩ (<TED, NBP>, <NTN, NBP>,
<ADJ1, NBP>)
EstNonVide_∩ (<TED, NOMBRE>, <NTN, NOMBRE>,
<ADJ1, NOMBRE>)
GENRE := ∩ (<TED, GENRE>, <NTN, GENRE>,
<ADJ1, GENRE>)
NBP := ∩ (<TED, NBP>, <NTN, NBP>, <ADJ1, NBP>)
NOMBRE := ∩ (<TED, NOMBRE>, <NTN, NOMBRE>,
<ADJ1, NOMBRE>)

DNA → TED NAD
EstNonVide_∩ (<TED, GENRE>, <NAD, GENRE>)
EstNonVide_∩ (<TED, NBP>, <NAD, NBP>)
EstNonVide_∩ (<TED, NOMBRE>, <NAD, NOMBRE>)
GENRE := ∩ (<TED, GENRE>, <NAD, GENRE>)
NBP := ∩ (<TED, NBP>, <NAD, NBP>)
NOMBRE := ∩ (<TED, NOMBRE>, <NAD, NOMBRE>)

NAD → ADJ1 NOM-P
EstNonVide_∩(<ADJ1, GENRE>, <NOM-P, GENRE>)
EstNonVide_∩(<ADJ1, NBP>, <NOM-P, NBP>)
EstNonVide_∩(<ADJ1, NOMBRE>, <NOM-P, NOMBRE>)
GENRE := ∩ (<ADJ1, GENRE>, <NOM-P, GENRE>)
NBP := ∩ (<ADJ1, NBP>, <NOM-P, NBP>)
NOMBRE := ∩ (<ADJ1, NOMBRE>, <NOM-P, NOMBRE>)

NAD → ADJ1 NOM-P ADJ1
EstNonVide_∩(<ADJ1, GENRE>, <NOM-P, GENRE>,
<ADJ1, GENRE>)
EstNonVide_∩(<ADJ1, NBP>, <NOM-P, NBP>,
<ADJ1, NBP>)
EstNonVide_∩(<ADJ1, NOMBRE>, <NOM-P, NOMBRE>,
<ADJ1, NOMBRE>)
GENRE := ∩ (<ADJ1, GENRE>, <NOM-P, GENRE>,
<ADJ1, GENRE>)
NBP := ∩ (<ADJ1, NBP>, <NOM-P, NBP>, <ADJ1, NBP>)
NOMBRE := ∩ (<ADJ1, NOMBRE>, <NOM-P, NOMBRE>,
<ADJ1, NOMBRE>)

NOM-P → *NOM / *NOP
GENRE := $<v_i$, GENRE>
NBP := $<v_i$, NBP>
NOMBRE := $<v_i$, NOMBRE>

NTN → *NOM / *NOP / TITRE
GENRE := $<v_i$, GENRE>
NBP := $<v_i$, NBP>
NOMBRE := $<v_i$, NOMBRE>

PEGN2 → *PPE / GN2
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

PEGN2 → *PPE PPI-D
GENRE := <*PPE, GENRE>
NBP := <*PPE, NBP>
NOMBRE := <*PPE, NOMBRE>

PEGN2 → GN2 PPI-D
GENRE := <GN2, GENRE>
NBP := <GN2, NBP>
NOMBRE := <GN2, NOMBRE>

PGN → GN2 / PREP GN2
GENRE := <GN2, GENRE>
NBP := <GN2, NBP>
NOMBRE := <GN2, NOMBRE>
VFORM := <GN2, VFORM>

PPA-X → *PPa / *PPx
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

PPI-D → *PPI / *PPD
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

PREP → *AA2 / *AVE / *DDE / *DDU / *DES / *PR2 / *PUR / *SAN
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

TED → *ADE / *ANL / *APO / *ARD / *ARI / *AUX / *CAR / *CCE / *CER / *CHA / *DDU / *DIF / *PLU
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

TITRE → **madame / **mademoiselle / **maître / **maîtresse / **mesdemoiselles / **messieurs / **monsieur
GENRE := <v_i, GENRE>
NBP := <v_i, NBP>
NOMBRE := <v_i, NOMBRE>

Annexe C

Estimation Forward-Backward des multiclasses

La formule de réestimation 5.26 peut être exprimée en fonction de $\alpha^{(k)}$ et $\beta^{(k)}$, les variables forward-backward de l'itération (k). Pour cela, on introduit le rang t des mots dans la formule 5.26 par le biais des quantités $N(L_C)$ et $N(m_{c_i}/L_C)$, qui représentent respectivement le nombre total de séquences, et le nombre d'occurrences de m_{c_i} dans la segmentation L_C.

Soit la quantité $\delta(t,l)$ qui vaut 1 lorsque la segmentation L_C comprend une séquence de longueur l se terminant en t, et qui vaut 0 sinon. Le nombre de séquences $N(L_C)$ de la segmentation L_C peut être définie comme suit :

$$N(L_C) = \sum_{t=1}^{T} \sum_{l=1}^{n} \delta(t,l) \tag{C.1}$$

où T représente le nombre total de classes dans le corpus d'apprentissage. Pour ne comptabiliser parmi les séquences de la segmentation L que les entités m_{c_i}, on définit la quantité $\delta(t,l,i)$, qui vaut 1 lorsque la séquence de longueur l se terminant en t est m_{c_i}, et 0 sinon.

$$N(m_{c_i}/L_C) = \sum_{t=1}^{T} \sum_{l=1}^{n} \delta(t,l,m_{c_i}). \tag{C.2}$$

Ainsi, en substituant $N(L_C)$ et $N(m_{c_i}/L_C)$ dans la formule de réestimation 5.26, et en modifiant l'ordre des sommations, il vient :

$$p^{(k+1)}(m_{c_i}) = \frac{\sum_{t=1}^{T} \sum_{l=1}^{n} \sum_{L_C} \delta(t,l,m_{c_i})\ \mathcal{L}^{(k)}(C_1^T,L_C)}{\sum_{t=1}^{T} \sum_{l=1}^{n} \sum_{L_C} \delta(t,l)\ \mathcal{L}^{(k)}(C_1^T,L_C)} \tag{C.3}$$

Le terme $\sum_{L_C} \delta(t,l)\mathcal{L}^{(k)}(C_1^T,L_C)$ qui apparaît au dénominateur représente la vraisemblance cumulée de toutes les segmentations qui comportent une séquence de longueur l se terminant en t. Elle peut se calculer au moyen des variables forward-backward de la façon suivante :

$$\sum_{L_C} \delta(t,l)\ \mathcal{L}^{(k)}(C_1^T,L_C) = \alpha^k(t-l)\ p^{(k)}([c_{t-l+1},\dots c_t])\ \beta^{(k)}(t) \tag{C.4}$$

De même, le terme $\sum_{L_C} \delta(t,l,m_{c_i})\mathcal{L}^{(k)}(C_1^T,L_C)$ qui apparaît au numérateur de l'équation C.3 représente la vraisemblance cumulée de toutes les segmentations où m_{c_i} figure entre les rangs

$(t-l+1)$ et t (si la longueur de m_{c_i} est différente de l, cette vraisemblance cumulée et nulle). Elle peut s'écrire de nouveau comme suit :

$$\begin{aligned}\textstyle\sum_{L_C} \delta(t,l,m_{c_i})\ \mathcal{L}^{(k)}(C_1^T,L_C) &= \delta(t,l,m_{c_i})\ \alpha^k(t-l)\ p^{(k)}([c_{t-l+1},\dots c_t])\ \beta^{(k)}(t) \\ &= \delta(t,l,m_{c_i})\ \alpha^k(t-l)\ p^{(k)}(m_{c_i})\ \beta^{(k)}(t)\end{aligned} \tag{C.5}$$

Ainsi, la formule de réestimation 5.26 devient comme suit :

$$p^{(k+1)}(m_{c_i}) = \frac{\sum_{t=1}^{T}\sum_{l=1}^{n} \delta(t,l,m_{c_i})\ \alpha^{(k)}(t-l)\ p^{(k)}(m_{c_i})\ \beta^{(k)}(t)}{\sum_{t=1}^{T}\sum_{l=1}^{n} \alpha^{(k)}(t-l)\ p^{(k)}([c_{t-l+1},\dots c_t])\ \beta^{(k)}(t)} \tag{C.6}$$

La somme sur l au dominateur se simplifie en utilisant la relation de récurrence de α donnée dans la table 5.1. La formule de réestimation des paramètres se réduit par conséquent à la formule suivante :

$$p^{(k+1)}(m_{c_i}) = \frac{\sum_{t=1}^{T}\sum_{l=1}^{n} \delta(t,l,m_{c_i})\ \alpha^{(k)}(t-l)\ p^{(k)}(m_{c_i})\ \beta^{(k)}(t)}{\sum_{t=1}^{T} \alpha^{(k)}(t)\ \beta^{(k)}(t)} \tag{C.7}$$

Publications personnelles

[1] D. Fohr, J.P. Haton, J.F. Mari, K. Smaïli et I. Zitouni. Traitements lexicaux autour de MAUD. Dans *Séminaire GDR-PRC lexique et communication parlée*, Toulouse, France, 1996.

[2] D. Fohr, J.P. Haton, J.F. Mari, K. Smaïli et I. Zitouni. Maud : Un prototype de machine à dicter vocale. Dans *Actes des premières JST Francil 1997*, pages 25–30, Avignon, France, April 1997.

[3] K. Smaïli, I. Zitouni, F. Charpillet et J.P. Haton. An hybrid language model for a continuous dictation prototype. Dans *Proceeding of the European Conference On Speech Communication and Technologie*, pages 2755–2758, Rhodes, Greece, 1997.

[4] I. Zitouni et K. Smaïli. Apport d'une grammaire d'unification dans un système de dicté automatique. Dans *Deuxièmes journées jeunes chercheurs en parole*, La Rochelle, France, Novembre 1997.

[5] D. Fohr, J.P. Haton, J.F. Mari, K. Smaïli et I. Zitouni. Towards an oral interface for data entry: The maud system. Dans *3rd ERCIM Workshop on "User Interface for All" (ERCIM: European Research Consortium for Informatics and Mathematics)*, Obernai, France, November 1997.

[6] M. Jardino, F. Bimbot, S. Igounet, K. Smaïli, I. Zitouni et M. El-Beze. A first evaluation campaign for language models. Dans *First International Conference on Language Resources and Evaluation*, pages 801–805, Granada, Spain, May 1998.

[7] I. Zitouni, K. Smaïli et J.P. Haton. Variable-length class sequences based on a hierarchical approach: Mc. Dans *International Workshop on Speech and Communication*, St Petersbourg Russia, Octobre 1998.

[8] I. Zitouni, K. Smaïli, J.P. Haton, S. Deligne et F. Bimbot. A comparative study between polyclass and multiclass models. Dans *International Conference on Language Resources and Evaluation*, Sydney, Australia, Novembre 1998.

[9] I. Zitouni. A language modeling based on a hierarchical approch: Mnv. Dans *The 7th Australian International Speech Science and Technology Conference*, Sydney, Australia, Novembre 1998.

[10] I. Zitouni, J.F. Mari, K. Smaïli et J.P. Haton. Variable-length sequence language model for large vocabulary continuous dictation machine: The n-seqgram approach. Dans *Proceeding of the European Conference On Speech Communication and Technologie*, Septembre 1999.

[11] K. Smaïli, A. Brun, I. Zitouni et J.P. Haton. Automatic and manual clustering for large vocabulary speech recognition: A comparative study. Dans *European Conference on Speech Communication and Technology*, Budapest, Hongrie, Septembre 1999.

[12] K. Smaïli, I. Zitouni et J.P. Haton. Towards a better collaboration between n-class and n-gram language models. Dans *International Workshop on Speech and Communication*, Moscou, 1999.

Bibliographie

[1] Abramson (N.). – *Information Theory and Coding.* – New-York, NcGraw-Hill, 1963.

[2] Adda (G.), Adda-Decker (M.), Gauvain (J.L.) et Lamel (L.). – Text normalisation and speech recognition in french. *In : Proceeding of the European Conference On Speech Communication and Technologie.* – Rhodes, Greece, 1997.

[3] Adda (G.), Adda-Decker (M.), Gauvain (J.L.) et Lamel (L.F.). – Le système de dictée du limsi pour l'évaluation aupelf'97. *In : Actes des premières JST Francil 1997*, pp. 35–40. – Avignon, France, April 1997.

[4] Adda (G.), DeCalmes (M.), Lamel (L.), Perennou (G.), Rajman (M.), Rosset (S.) et Zeiliger (J.). – Ressources pour l'apprentissage, le développement et l'évaluation des systèmes de dictée vocale en français : corpus de texte, de parole et lexical. *In : Actes des premières JST Francil 1997*, pp. 305–309. – Avignon, France, April 1997.

[5] Allen (J.). – *Natural Language Understanding.* – The Benjamin/Cummings Publishing Company Inc, 1995, deuxième édition.

[6] André-Olbrecht (R.). – A new statistical approach for the automatic segmentation of continuous speech signals. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 36, n1, 1988.

[7] Applebaum (T.) et Hanson (B.). – Regression features for recognition of speech in noise. *In : Proceeding of International Conference on Acoustics, Speech and Signal Processing.* – Toronto, 1991.

[8] Atal (B.). – Effectiveness of linear prediction characteristics of a speech wave for automatic speaker identification and verification. *In : Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 305–308. – 1991.

[9] Aubert (X.), Haeb-Umbach (R.) et Ney (H.). – Improvement in connected digit recognition using linear discriminant analysis and mixture densities. *In : Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 648–651. – 1993.

[10] Backer (J.K.). – The dragon system, an overview. *IEEE transactions on Acoustics Speech and Signal Processing*, vol. 23, n1, 1975, pp. 24–29.

[11] Bahl (L.R.), de Souza (P.V.), Gopalakrishnan (P.S.), Nahamoo (D.) et Picheny (M.A.). – Decision trees for phonological rules in continuous speech. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 185–188. – 1991.

[12] Bahl (L.R.), Jelinek (F.) et Mercer (R.L.). – A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Patt. Anal. and Machine Intell.*, vol. 5, 1983, pp. 179–190.

[13] Baker (J.K.). – Trainable grammars for speech recognition. *In : Proceeding of the 97th meeting of the Acoustical Society of America.* – 1979.

[14] Baum (L.E.). – An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities*, no3, 1972, pp. 1–8.

[15] Beaujard (C.) et Jardino (M.). – Language modeling based on automatic word concatenations. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1563–1566. – Budapest, Hungary, September 1999.

[16] Bellegarda (J.R.). – A multi-span language modeling framework for large vocabularyspeech recognition. *IEEE Transactions on Speech Audio Processing*, vol. 6, n5, September 1998.

[17] Bellegarda (J.R.). – Multi-span statistical language modeling for large vocabularyspeech recognition. *In : Proceeding of International Conference on Spoken Language Processing*. – 1998.

[18] Bellegarda (J.R.), Butzberger (J.W.), Chow (Y.L.), Coccaro (N.B.) et Naik (D.). – A novel word clustering algorithm based on latent semantic analysis. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 172–175. – 1996.

[19] Berger (A.) et Printz (H.). – Recognition performance of a large-scale dependency grammar language model. *In : Proceeding of International Conference on Spoken Language Processing*. – Sydney, Australia, December 1998.

[20] Berger (A.L.), DellaPietra (S.A.) et DellaPietra (V.J.). – A maximum entropy approach to natural language processing. *Computational Linguistics*, vol. 22, n1, 1996, pp. 39–71.

[21] Bigi (B.), DeMori (R.), El-Beze (M.) et Spriet (T.). – Detecting topic shifts using a cache memory. *In : International Conference on Spoken Language Processing*. – Sydney, Australia, December 1998.

[22] Bimbot (F.), El-Bèze (M.) et Jardino (M.). – An alternative scheme for perplexity estimation. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*. – Munich, Germany, 1997.

[23] Bimbot (F.), Pieraccini (R.), Levin (E.) et Atal (B.). – Modèles de séquences à horizon variable : Multigrams. *In : XXèmes Journées d'étude sur la parole*, pp. 467–472. – Juin 1994.

[24] Black (E.). – *Statistically-Driven Computer Grammars of English: The IBM/ Lancaster Approach*. – Amsterdam-Atlanta, Rodopi, 1993.

[25] Bonafonte (A.) et Marino (J.B.). – Language modeling using x-grams. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 394–397. – 1996.

[26] Boulianne (G.), Kenny (P.), Lennig (M.), O'Shaughessy (D.) et Mermelstein (P.). – Books on tape as training data for continuous speech recognition. *Speech Communication*, vol. 14, 1994, pp. 61–70.

[27] Bridle (J.S.), Brown (D.M.) et Chamberlain (R.M.). – An algorithm for connected word recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*. – May 1982.

[28] Brillouin (L.). – *La science et la théorie de l'information*. – Masson Paris, 1959.

[29] Brown (P.), Cocke (J.), DellaPietra (S.), DellaPietra (V.), Jelinek (F.), Lafferty (J.), Mercer (R.) et Roosin (P.). – A statistical approach to machine translation. *Computational Linguistics*, vol. 16, June 1990, pp. 79–85.

[30] Brown (P.F.), DellaPietra (V.J.), DeSouza (P.V.), Lai (J.C.) et Mercer (R.L.). – Class-based n-gram models of natural language. *Computational Linguistics*, vol. 18, n4, 1992, pp. 467–479.

[31] Brugnara (F.) et Federico (M.). – Techniques for approximating a trigram language model. *In : Proceeding of International Conference on Spoken Language Processing*. – Philadelphia, PA, 1996.

[32] Calliope. – *La parole et son traitement automatique.* – Masson, 1989.

[33] Cerf-Danon (H.) et El-Bèze (M.). – Three different probabilistic language models: Comparison and combination. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 297–300. – Toronto, Canada, 1991.

[34] Chelba (C.) et Jelinek (F.). – Recognition performance of a structured language model. *In : Proceeding of the European Conference On Speech Communication and Technologie.* – Budapest, Hongrie, Septembre 1999.

[35] Chibout (K.), Néel (F.), Mariani (J.) et Masson (N.). – Ressources et évaluation en ingénierie des langues. *In : Actualité Scientifique.* – Aupelf-UREF, 1999.

[36] Chollet (G.F.) et Gagnoulet (C.). – On the evaluation of speech recognizers and data bases using a reference system. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 2026–2029. – 1982.

[37] Chomsky (N.). – *Aspects de la théorie syntaxique.* – Paris, Le Seuil, 1971.

[38] Church (K.W.). – A stochastic parts program and noun phrase parser for unrestricted text. *In : Proceeding of the second Conference on Applied Natural Language Processing*, pp. 136–143. – Austin, Texas, 1988.

[39] Coccaro (N.) et Jurafsky (D.). – Toward better integration of semantic predictors in statistical language modeling. *In : Proceeding of International Conference on Spoken Language Processing*, pp. 2403–2406. – 1998.

[40] Costa-DeBeauregard (O.). – *Le second principe de la science et du temps.* – Denoël Paris, 1963.

[41] Cover (T.M.) et King (R.C.). – A convergent gambling estimate of the entropy of english. *IEEE Transactions on Information Theory*, vol. 24, n4, Juillet 1978, pp. 413–421.

[42] Cutting (D.), Pedersen (J.) et Sibun (P.). – A practical part-of-speech tagger. *In : Proceeding of the 3rd Conference on Applied Natural Language Processing*, pp. 133–140. – Torento, Italy, 1992.

[43] Dagan (I.), Marcus (S.) et Markovitch (S.). – Contextual word similarity and estimation from sparse data. *Computer Speech and Language*, vol. 9, n2, April 1995, pp. 123–152.

[44] Darroch (J.N.) et Ratcliff (D.). – Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics*, vol. 43, 1972, pp. 1470–1480.

[45] Davis (S.B.) et Mermelstein (P.). – Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 28, n4, 1980, pp. 357–366.

[46] Deligne (S.). – *Modèles de séquences de longeurs variables : application au traitement du langage écrit et de la parole.* – Thèse de PhD, Ecole Nationale Supérieure des Télécommunications, October 1996.

[47] Deligne (S.) et Sagisaka (Y.). – Modélisation statistique du langage avec un modèle bi-multigramme. *In : XXIIèmes Journées d'étude sur la parole*, pp. 355–358. – 1998.

[48] DeMori (R.) (édité par). – *Spoken Dialogues with Computers.* – ACADEMIC PRESS, 1998.

[49] Dempster (A.P.), Laird (N.M.) et Rubin (D.B.). – Maximum-likelihood from incomplete data via the em algorithm. *Journal of Royal Statistic Society, Ser. B (methodological)*, no39, 1977, pp. 1–38.

[50] Deroo (O.). – *Modèles dépendants du contexte et méthodes de fusion de données appliqués à la reconnaissance de la parole par modèles hybrides HMM/MLP.* – Thèse de PhD, Faculté Polytechnique de Mons, Décembre 1998.

[51] Dolmazon (J.M.), Bimbot (F.), Adda (G.), El-Bèze (M.), Caërou (J.C), Zeiliger (J.) et Adda-Decker (M.). – Organisation de la première campagne aupelf pour l'évaluation des systèmes de dictée vocale. *In : Actes des premières JST Francil 1997*, pp. 13–18. – Avignon, France, April 1997.

[52] Dubois (J.) et Lagane (R.). – *La grammaire du français.* – LAROUSSE, 1993.

[53] Duda (R.O.) et Hart (P.E.). – *Pattern Classification and Scene Analysis.* – New York, NY, John Wiley and Sons, 1973.

[54] Dugast (C.), Kneser (R.), Aubert (X.), Ortmanns (S.), Beulen (K.) et Ney (H.). – Continuous speech recognition tests and results for the nab'94 corpus. *In : 95 ARPA SLT Workshop*, pp. 156–161. – 1995.

[55] Dupont (P.) et Rosenfeld (R.). – *Lattice Based Language Models.* – Rapport technique, Pittsburg, PA 15213, Carnegie Mellon University Tech Report CMU-CS-97-173, September 1997.

[56] El-Bèze (M.) et Derouault (A.M.). – A morphological model for large vocabulary speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 577–580. – 1990.

[57] Ferretti (M.), Maltese (G.) et Scarci (S.). – Language model and acoustic model information in probabilistic speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 707–710. – 1989.

[58] Flanagan (J.). – Computers that talk and listen: man-machine communication by voice. *In : Proceeding IEEE, 64*, pp. 405–415. – 1976.

[59] Flanagan (J.L.). – *Speech analysis. Synthesis and perception.* – Berlin/ Heidelberg/ New York: Springer, 1965.

[60] Fohr (D.), Haton (J.P.), Mari (J.F.), Smaïli (K.) et Zitouni (I.). – traitements lexicaux autour de maud. *In : Séminaire GDR-PRC lexique et communication parlée.* – Toulouse, France, 1996.

[61] Fohr (D.), Haton (J.P.), Mari (J.F.), Smaïli (K.) et Zitouni (I.). – Maud : Un prototype de machine à dicter vocale. *In : Actes des premières JST Francil 1997*, pp. 25–30. – Avignon, France, April 1997.

[62] Fohr (D.), Haton (J.P.), Mari (J.F.), Smaïli (K.) et Zitouni (I.). – Towards an oral interface for data entry: The maud system. *In : 3rd ERCIM Workshop on "User Interface for All" (ERCIM: European Research Consortium for Informatics and Mathematics).* – Obernai, France, November 1997.

[63] Fontaine (V.), Ris (C.) et Boite (J.). – Non linear discriminant analysis for improved speech recognition. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 2071–2075. – 1997.

[64] Fourcin (A.) et al. – Progress overview of the sam project. *In : Proceeding of the European Conference On Speech Communication and Technologie.* – Paris, 1989.

[65] Furui (S.). – Cepstral analysis technique for automatic speaker verification. *IEEE Transactions on Acoustics Speech and Signal Processing*, 1981.

[66] Furui (S.). – Speaker-independent isolated word recognition using dynamic features of speech spectrum. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 34, n 1, 1986, pp. 52–59.

[67] Geutner (P.). – Introducing linguistic constraints into statistical language modeling. *In : Proceeding of International Conference on Spoken Language Processing*, pp. 402–405. – 1996.

[68] Giachin (E.). – Phrase bigrams for continuous speech recognition. *In : Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 225–228. – 1995.

[69] Giachin (E.), Baggia (P.) et Micca (G.). – Language models for spontaneous speech recognition: A bootstrap method for learning phrase bigrams. *In : Proceeding of International Conference on Spoken Language Processing*, pp. 843–846. – 1994.

[70] Gillett (J.) et Ward (W.). – A language model combining trigrams and stochastic context-free grammars. *In : Proceeding of International Conference on Spoken Language Processing.* – Sydney, Australia, 1998.

[71] Good (I.J.). – The population frequencies of species and the estimation of population parameters. *Biometrika*, vol. 40, 1953, pp. 237–264.

[72] Goosse (A.). – *Le bon usage de grammaire française.* – Duclout, 1986.

[73] Gopalakrisnan (P.S.), Bahl (L.R.) et Mercer (R.L.). – A tree search strategy for large vocabulary continuous speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 572–575. – Detroit, 1995.

[74] Gross (M.). – *Méthodes en syntaxes régime des constructions complétives.* – Herman, 1975.

[75] Gross (M.) et Lentin (A.). – *Notions sur les grammaires formelles.* – Paris, Gauthier-Villars, 1970, 2ème édition.

[76] Haeb-Umbach (R.), Geller (D.) et Ney (H.). – Linear discriminant analysis for improved large vocabulary continuous speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 13–16. – 1992.

[77] Haeb-Umbach (R.) et Ney (H.). – Improvement in time-synchronous beam search for 10000-word continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, vol. 2, 1994, pp. 353–356.

[78] Haton (J.P.). – Contribution à l'analyse, la paramétrisation et la reconnaissance de la parole. *Thèse de Doctorat d'état, Université de Nancy I*, 1974.

[79] Haton (J.P.), Pierrel (J.M.), Perennou (G.), Caelen (J.) et Gauvain (J.L.). – *Reconnaissance automatique de la parole.* – afcet DUNOD informatique, 1991.

[80] Hermansky (H.) et Morgan (N.). – Rasta processing of speech. *IEEE Transactions on Speech and Audio Processing*, vol. 2, 1994, pp. 578–589.

[81] Hochberg (M.), Cook (G.D.), Renals (S.), Robinson (A.J.) et Schechtman (R.S.). – The 1994 abbot hybrid connectionist-hmm large vocabulary recognition system. *In : Spoken Language Systems Technology Workshop*, pp. 170–176. – January 1995.

[82] Hormann (H.). – *Introduction à la psycholinguistique langue et langage.* – larousse, 1972.

[83] Hu (J.), Turin (W.) et Brown (M.K.). – Language modeling using stochastic automata with variable length contexts. *Computer Speech and Language*, vol. 11, n1, January 1997, pp. 1–16.

[84] Huang (E.F.), Soong (F.K.) et Wang (H.). – The use of tree-treillis search for large vocabulary mandarin polysyllabic word speech recognition. *Computer Speech and Language*, vol. 8, January 1994.

[85] Huang (X.), Belin (M.), Alleva (F.) et Huang (M.Y). – Unified stochastic engine (use) for speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing.* – 1993.

[86] Huang (X.D.), Alleva (F.), Hon (H.W.), Hwang (M.Y.), Lee (K.F.) et Rosenfeld (R.). – The sphinx-ii speech recognition system: An overview. *Computer Speech and Language*, 1993.

[87] Hyde (S.R.). – Automatic speech recognition: a critical survey of the litterature. *In : Human Communication, a Unified View*, éd. par David (E.E) et Denes (P.B.). Mc Graw Hill, pp. 399–438. – NY, USA, 1972.

[88] Iyer (R.), Ostendorf (M.) et Rohlicek (J.R.). – Language modeling with sentence-level mixtures. *In : Proceedings of the ARPA Human Language Technology Workshop*, pp. 82–86. – Plainsboro, NJ, 1994.

[89] Jang (P.J.) et Hauptmann (A.). – Hierarchical cluster language modeling with statistical rule extraction for rescoring n-best hypotheses during speech decoding. *In : International Conference on Spoken Language Processing*. – Sydney, Australia, December 1998.

[90] Jardino (M.) et Adda (G.). – Automatic word classification using simulated annealing. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 41–44. – Minneapolis, MN, 1993.

[91] Jardino (M.) et Adda (G.). – Language modeling for csr of large corpus using automatic classification of words. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1191–1194. – September 1993.

[92] Jardino (M.), Bimbot (F.), Igounet (S.), Smaïli (K.), Zitouni (I.) et El-Beze (M.). – A first evaluation campaign for language models. *In : First International Conference on Language Resources and Evaluation*, pp. 801–805. – Granada, Spain, May 1998.

[93] Jelinek (F.). – Self-organized language modeling for speech recognition. *Reading in Speech Recognition, Edited by Alex Waibel and Kai-Fu Lee*, - Morgan Kaufmann, San Mateo Calif., 1990, pp. 450–506.

[94] Jelinek (F.). – Continuous speech recognition by statistical methods. *Proceeding of the IEEE*, vol. 64, n4, 1976, pp. 532–556.

[95] Jelinek (F.). – Up from trigrams! the struggle for improved language models. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1037–1040. – 1991.

[96] Jelinek (F.) et Mercer (R.L.). – Interpolated estimation of markov source parameters from sparse data. *In : Pattern Recognition in Practice*, pp. 381–397. – Amsterdam, Holland, 1980.

[97] Jelinek (F.), Mercer (R.L.), Bahl (L.R.) et Baker (J.K.). – Perplexity : A mesure of difficulty of speech recognition tasks. *In : 94th Meeting of the Acoustic Society of America*. – Miami Beach, Florida, December 1977.

[98] Jouette (A.). – *Toute l'orthographe pratique : dictionnaire d'orthographe et de grammaire*. – Nathan, 1986.

[99] Katz (S.M.). – Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 35, n3, 1987, pp. 400–401.

[100] Kawabata (T.) et Tamoto (M.). – Back-off method for n-gram smoothing based on binomial posteriori distribution. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 192–195. – Atlanta, GA, 1996.

[101] Kay (M.). – Functional grammars. *In : Actes 5th. annual meeting of the Berkley linguistic society*, pp. 142–158. – 1979.

[102] Kay (M.). – Parsing in functional unification grammar. *In : Natural Language Parsing*, éd. par Karttunen (L.) et Zwicky (A.). pp. 251–278. – New York: Cambridge U. Press, 1982. réapparu dans RNLP.

[103] Kenne (P.E.), O'Kane (M.) et Pearcy (H.G.). – Language modeling of spontaneous speech in a court context. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1801–1804. – 1995.

[104] Kenny (P.), Hollan (R.), Gupta (V.), Lennig (M.), Mermelstein (P.) et O'Shaughnessy (D.). – A* admissible heuristics for rapid lexical access. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing.* – 1991.

[105] Khinchin (A.I.). – *Mathematical Foundations of Information Theory*, chap. The Entropy Concept in Probability Theory. – Dover, 1957.

[106] Kittredge (R.). – Sublanguage: Studies of language in restricted semantic domains. *De Gruyter*, 1982.

[107] Kittredge (R.). – Sublanguages. *American Journal of Computational Linguistics*, vol. 8, n 2, 1982.

[108] Klakow (D.). – Log-linear interpolation of language models. *In : Proceeding of International Conference on Spoken Language Processing*, pp. 1695–1698. – Sydney, Australia, 1998.

[109] Kneser (R.). – Statistical language modeling using a variable context. *In : Proceeding of International Conference on Spoken Language Processing.* – Philadelphia, PA, 1996.

[110] Kneser (R.) et Ney (H.). – Forming word classes by statistical clustering for statistical language modeling. *In : Proceeding of the Firt International Conference on Quantitative Linguistics*, éd. par Köhler (In R.) et Rieger (B.B.). pp. 221–226. – Trier, Germany, 1991.

[111] Kneser (R.) et Ney (H.). – Improved clustering technique for class-based statistical language modelling. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 973–976. – Berlin, Germany, 1993.

[112] Kneser (R.) et Ney (H.). – Improved backing-off for m-gram language modeling. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 181–184. – Detroit, MI, 1995.

[113] Knuth (D.E.). – Semantics for context-free languages. *Mathematical Systems Theory 2*, 1968, pp. 127–145.

[114] Kohonen (T.). – The 'neural' phonetic typewriter. *IEEE Computer, 21*, vol. 3, 1988.

[115] Krioule (A.), Mari (J.F.) et Haton (J.P.). – L'algorithme viterbi-bloc pour la reconnaissance de la parole continue. *In : XVIII journées d'étude sur la parole*, pp. 207–210. – 1990.

[116] Krioule (A.), Mari (J.F.) et Haton (J.P.). – Some improvements in speech recognition algorithms based on hmm. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 545–548. – April 1990.

[117] Kuhn (R.). – Speech recognition and the frequency of recently used words: a modified markov model for natural language. *In : Proceeding of COLING*, pp. 348–350. – Budapest, Hungary, 1988.

[118] Kuhn (R.) et DeMori (R.). – A cache-based natural language model for speech recognition. *IEEE Transactions Pattern Anal. Machine Intell.*, vol. 12, n6, 1990, pp. 570–582.

[119] Kukich (K.). – Techniques for automatically correcting words in text. *ACM Computing Surveys*, vol. 24, n4, 1992, pp. 377–439.

[120] Kupiec (J.). – Probabilistic models of short and long distance word dependencies in running text. *In : Proceeding of the DARPA Workshop on Speech and Natural Language*, éd. par Kaufmann (Morgan), pp. 290–295. – 1989.

[121] Lamel (L.), Gauvain (J.L.) et Eskenazi (M.). – Bref, a large vocabulary spoken corpus for french. *In : Proceeding of European Conference on Speech Communication and Technology.* – Gênes, 1991.

[122] Langlois (D.) et Smaïli (K.). – A new based distance language model for a dictation machine: Application to maud. *In: Proceeding of the European Conference On Speech Communication and Technologie*. Budapest, Hungary. – September 1999.

[123] Lau (R.), Rosenfeld (R.) et Roukos (S.). – Trigger-based language models: A maximum entropy approach. *In: Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. II 45–48. – 1993.

[124] Lazaridès (A.), Brousseau (J.), Lacouture (R.) et Dumouchel (P.). – Le système de dictée vocale en français du crim utilisé pour l'arc b1 de l'aupelf. – 14 April 1997. Exposé oral pour ARC-ILOR Thème B1, Reconnaissance de grands vocabualire : dictée vocale.

[125] Lazaridès (A.), Normandin (Y.) et Kuhn (R.). – Improving decision trees for acoustic modeling. *In: Proceeding on Spoken Language Processing*. – Philadelphie, Pennsylvanie, October 1996.

[126] Lea (W.A.). – Speech recognition: past, present and future. *In: Trends is speech recognition*, éd. par Lea (W.A.). Prentice Hall, Englewood Cliffs. – NJ, USA, 1980.

[127] Lee (K.F.), Hon (H.W.) et Reddy (R.). – An overview of the sphinx speech recognition system. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 38, n1, January 1990.

[128] Lonchamp (F.). – *Les sons du Français analyse acoustique descreptive: Cours de phonétique*. – Institut de phonétique université de Nancy II, 1984.

[129] Marcello (F.) et DeMori (R.). – *Spoken Dialogs with Computers*, chap. Language Modeling. – Academic Press Edition, 1997.

[130] Mari (J.F.). – Perception de signaux complexes et interaction homme-machine. *Habilitation à diriger des recherches, Université de Nancy I*, 1996.

[131] Mari (J.F.), Fohr (D.) et Junqua (J.C.). – A second-order hmm for high performance word and phoneme-based continuous speech recognition. *In: Proceeding of the International Conference on Acoustics, Speech and Signal Processing*. – 1996.

[132] Matsunaga (S.), Yamada (T.) et Shikano (K.). – Task adaptation in stochastic language models for continuous speech recognition. *In: Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 165–168. – April 1994.

[133] Meziane (A.), Jacob (B.) et André-Obrecht (R.). – Modélisation de la durée des sons dans un système de reconnaissance automatique de la parole. *In: Actes des premières JST Francil 1997*, pp. 47–51. – 1997.

[134] Miller (J.W.) et Alleva (F.). – Evaluation of a language model using a clustered model backoff. *In: Proceeding of International Conference on Spoken Language Processing*, pp. 390–393. – 1996.

[135] Mohri (M.) et Riley (M.). – Weighted determinization and minimization for large vocabulary speech recognition. *In: Proceeding of European Conference on Speech Communication and Technology*, pp. 131–134. – 1997.

[136] Moisa (L.) et Giachin (E.). – Automatic clustering of words for probabilistic language models. *In: Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1249–1252. – Madrid, Spain, 1995.

[137] Mood (A.M.), Graybill (F.A.) et Boes (D.C.). – *Introduction to the Theory of Statistics*. – Singapore, McGrawHill, 1974.

[138] Myers (C.S.) et Rabiner (L.R.). – A level building dynamic time warping algorithm for connected word recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 2, n29, 1981, pp. 284–297.

[139] Nadas (A.). – On turing's formula for word probabilities. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 32, 1985, pp. 1414–1416.

[140] Newell (A.). – A tutorial on speech understanding systems. *In : speech recognition: invited papers of the IEEE symposium*, éd. par Reddy (DR.). Academic press. – New york, 1975.

[141] Ney (H.). – The use of a one-stage dynamic programming algorithm for connected word recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 2, n 32, 1984, pp. 263–271.

[142] Ney (H.). – Dynamic programming parsing for context-free grammars in continuous speech recognition. *IEEE Transactions on Signal Processing*, vol. 2, n 39, 1991, pp. 336–340.

[143] Ney (H.), Essen (U.) et Kneser (R.). – On structuring probabilistic dependences in stochastic language modelling. *Computer Speech and Language*, vol. 8, 1994, pp. 1–38.

[144] Niesler (T.R.) et Woodland (P.C.). – Variable-length category n-gram language models. *Computer Speech and Language*, vol. 13, n 1, January 1999, pp. 99–124.

[145] O'Boyle (P.), Owens (M.) et Smith (F.J.). – A weighted average n-gram model of natural language. *Computer Speech and Language*, vol. 8, n 4, October 1994, pp. 337–349.

[146] O'Boyle (P.), Owens (M.) et Smith (F.J.). – A weighted average n-gram model of natural language. *Computer Speech and Language*, no8, 1994, pp. 337–349.

[147] Ortmanns (S.) et Ney (H.). – A word graph algorithm for large vocabulary continuous speech recognition. *Computer Speech and Language*, vol. 11, 1997, pp. 43–72.

[148] Paescler (A.) et Ney (H.). – Continuous speech recognition using a stochastic language model. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 699–702. – May 1989.

[149] Pallett (D.). – *NIST Spoken Natural Language Processing Group: Benchmark Tests Description and Public Software*. – Rapport technique, http://www.itl.nist.gov/div894/894.01, 1997.

[150] Pallett (D.), Fiscus (J.), Fisher (W.), Garofolo (J.), Lund (B.) et Prysbocki (M.). – 1993 benchmark tests fot the arpa spoken language program. *In : Proc. ARPA*, pp. 49–79. – 1994.

[151] Pastor (J.), San-Segundo (R.) et Pardo (J.M.). – An asymmetric stochastic language model based on multi-tagged words. *In : Proceeding of International Conference on Spoken Language Processing*. – 1998.

[152] Paul (D.B.). – Algorithms for an optimal a^* search and linearizing the search in the stack decoder. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 693–696. – Toronto, 1991.

[153] Paul (D.B.). – An efficient a^* stack decoder algorithm for continuous speech recognition with a stochastic model. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 25–28. – 1992.

[154] Pican (N.), Mari (J.F.) et Fohr (D.). – Continuous speech recognition using a context sensitive ann and hmm2s. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 95–98. – Rhodes, Greece, 1997.

[155] P.Kenny. – a^* admissible heuristics for rapid lexical access. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*. – Toronto, 1991.

[156] Pérennou (G.). – Bdlex : A data and cognition base of spoken french. *In : Research and Development in Language Processing, J.P. Haton et G. Pérennou eds INRIA*. – 1987.

[157] Printz (H.). – Fast computation of maximum entropy / minimum divergence feature

gain. *In: International Conference on Spoken and Language Processing*, pp. 2083–2086. – Sydney, Australia, 1998.

[158] Rabiner (L.R.) et Juang (B.). – A tutorial on hidden markov models and selected applications in speech recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, vol. 77, n2, February 1989, pp. 257–286.

[159] Rabiner (L.R.) et Juang (B.H.). – Fundamentals of speech recognition. *PTR Prentice Hall*, 1993.

[160] Rabiner (L.R.), Levinson (S.E.) et Sondhi (M.M.). – On the application of vector quantization and hidden markov markov models to speaker independent, isolated word recognition et an introduction to the application of the theory of probabilistic functions of a markov process to automatic speech recognition. *B.S.T.J*, vol. 62, n4, Avril 1983, pp. 1035–1074 et 1075–1105.

[161] Reddy (D.R.). – Speech recognition by machine: a review. *IEEE Proceedings*, vol. 64, n3, April 1976, pp. 502–531.

[162] Renals (S.), Morgan (N.) et Bourlard (H.). – *Probability estimation by feed-forward networks in continuous speech recognition.* – Rapport technique, Berkeley, International Computer Science Institute, 1991.

[163] Riccardi (G.), Pieraccini (R.) et Bocchieri (E.). – Stochastic automata for language modeling. *Computer Speech and Language*, vol. 10, n4, October 1996, pp. 265–293.

[164] Ries (K.), Buo (F.D.) et Waibel (A.). – Class phrase models for language modeling. *In: Proceeding of International Conference on Spoken Language Processing*, pp. 398–401. – 1996.

[165] Rosenfeld (R.). – *Adaptive Statistical Language Modeling: A Maximum Entropy Approach.* – Pittsburgh, PA 15213, Thèse de PhD, School of Computer Science Carnegie Mellon University, April 1994.

[166] Rosenfeld (R.). – A maximum entropy approach to adaptive statistical language modelling. *Computer Speech and Language*, vol. 10, n3, July 1996, pp. 188–228.

[167] Rosenfeld (R.) et Huang (X.). – Improvement in stochastic language modeling. *In: Proceeding of the DARPA Workshop on Speech and Natural Language.* pp. 107–111. – San Mateo, CA, February 1992.

[168] Rousselot (F.). – *Réalisation d'un programme comprenant des textes en utilisant un formalisme unique pour représenter toutes les connaissances nécessaires. Thèse d'état.* – Thèse de PhD, Université Pierre et Marie Curie, 1984.

[169] Roxane (L.). – *Au sujet des algorithmes de recherche des systèmes de reconnaissance de la parole à grands vocabulaires.* – Thèse de PhD, School of Computer Science Université McGill, Montréal, 1995.

[170] Sabah (G.). – *L'intelligence artificielle et le langage.* – Paris, Hermes, 1988.

[171] Sakoe (H.). – Two level dp matching - a dynamic time warping based pattern matching algorithm for continuous speech recognition. *IEEE Transactions of the IECE of Japan*, vol. 3, 1979.

[172] Savariaux (C.), Farhat (A.), Héon (M.), O'Shaughnessy (D.) et Lee (C.Z.). – Nouvelles avancées en reconnaissance de la parole continue grand vocabulaire du français basées sur le système de reconnaissance de l'inrs-télécommunications. *In: Actes des premières JST Francil 1997*, pp. 31–34. – Avignon, France, April 1997.

[173] Schukat-Talamazzini (E.G.), Hendrych (R.), Kompe (R.) et Niemann (H.). – Permugram

language models. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1773–1776. – Madrid, Spain, 1995.

[174] Schwartz (R.) et Chow (Y.L.). – Efficient and exact procedure for finding the n most likely sentence hypotheses. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 81–84. – April 1990.

[175] Seneff (S.). – The use of linguistic hierarchies in speech understanding. *In : Proceeding of International Conference on Spoken Language Processing.* – Sydney, Australie, 1998. Keynote Speech.

[176] Serres (M.). – *Hermès I la communication.* – édition de minuit, 1969.

[177] Shannon (C.E.). – A mathematical theory of communication. *Bell System Technical Journal*, vol. 27, 1948, pp. 379–423 (Part I) and 623–656 (Part II).

[178] Shannon (C.E.). – Prediction and entropy of printed english. *Bell System Technical Journal*, vol. 30, 1951, pp. 50–64.

[179] Shieber (S.M.). – An introduction to unification-based approaches to grammar. *In : CSLI Lecture Notes 4.* – Chicago U. Press, 1986.

[180] Simons (M.), Ney (H.) et Martin (S.C.). – Distant bigram language modelling using maximum entropy. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. II 787–790. – 1997.

[181] Siohan (O.). – On the robustness of linear discriminant analysis as a preprocessing step for noisy speech recognition. *In : Proceeding of International Conference on Acoustics, Speech and Signal Processing*, pp. 125–128. – 1995.

[182] Smaïli (K.). – *Conception et réalisation d'une machine à dicter à entrée vocale destinée aux grands vocabulaires: Le système MAUD.* – Thèse de PhD, Université de Nancy I, 1991.

[183] Smaïli (K.), Charpillet (F.) et Haton (J.P.). – A new algorithm for automatic word classification based on an improved simulating annealing technique. *In : 5th International Conference in Cognitive Science and Natural Language Processing.* – 1996.

[184] Smaïli (K.), Zitouni (I.), Charpillet (F.) et Haton (J.P.). – An hybrid language model for a continuous dictation prototype. *In : Proceeding of the European Conference On Speech Communication and Technologie*, pp. 2755–2758. – Rhodes, Greece, 1997.

[185] Smaïli (K.), Brun (A.), Zitouni (I.) et Haton (J.P.). – Automatic and manual clustering for large vocabulary speech recognition: A comparative study. *In : European Conference on Speech Communication and Technology.* – Budapest, Hongrie, Septembre 1999.

[186] Soong (F.K.) et Huang (E.). – A tree-treillis bases fast search for finding the n best sentence hypotheses in continuous speech recognition. *In : Proceeding of the International Conference on Acoustics, Speech and Signal Processing*, pp. 705–708. – Toronto, Canada, May 1991.

[187] Spargins (J.). – A note on the iterative application of bayes' rule. *IEEE Transactions Information Theory*, 1965, pp. IT–11: 544–549.

[188] Steinbiss (V.). – Sentence-hypotheses generation in a continuous-speech recognition system. *In : Proceeding of European Conference on Speech Communication and Technology*, pp. 51–54. – Paris, 1989.

[189] Stone (M.). – Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society*, vol. 36, n2, 1974, pp. 111–147.

[190] Suhm (B.) et Waibel (A.). – Towards better language models for spontaneous speech. *In : Proceeding of International Conference on Spoken Language Processing*, pp. 831–834. – 1994.

[191] Tillmann (C.) et Ney (H.). – Selection criteria for word trigger pairs in language modeling. *In: Grammatical Inference: Learning Syntax from Sentences, Third International Colloquium, ICGI-96 (Lecture Notes in Artificial Intelligence 1147)*, éd. par Miclet (Laurent) et de la Higuera (Colin). pp. 98–106. – Montpellier, France, September 1996.

[192] Uebler (U.) et Niemann (H.). – Morphological modeling of word classes for language models. *In: Proceeding of International Conference on Spoken Language Processing*, pp. 1687–1690. – Sydney, Australia, 1998.

[193] Valverde-Albacete (F.) et Pardo (J.M.). – A hierarchical language model for csr. *In: Proceeding of International Conference on Spoken Language Processing.* – Sydney, Australia, 1998.

[194] Ward (W.). – Evaluation of the cmu atis system. *In: Proceeding of the DARPA Speech and Natural Language Workshop*, pp. 101–105. – February 1991.

[195] Weaver (W.) et Shannon (C.E.). – Théorie mathématique de la communication. *Les classiques des sciences humaines*, 1975.

[196] Witschel (P.). – Constructing linguistic oriented language models for large vocabulary speech recognition. *In: Proceeding of the European Conference On Speech Communication and Technologie*, pp. 1199–1202. – September 1993.

[197] Witten (I.H.) et Bell (T.C.). – The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions Information Theory*, vol. 34, n4, 1991, pp. 1085–1094.

[198] Yodo (N.), Shikano (K.) et Nakamura (S.). – Compression algorithm of trigram language models based on maximum likelihood estimation. *In: Proceeding of International Conference on Spoken Language Processing*, pp. 1683–1686. – Sydney, Australia, 1998.

[199] Young (S.). – A review of large-vocabulary continuous-speech recognition. *IEEE Signal Processing Magazine*, September 1996, pp. 45–57.

[200] Young (S.J.), Odell (J.J.) et Woodland (P.C.). – Tree-based state tying for high accuracy acoustic modeling. *In: 94 ARPA HLT Workshop*, pp. 307–312. – 1994.

[201] Zitouni (I.). – *Reconnaissance de grands vocabulaires en parole continue.* – Dea, Univesité Nancy I, 1996.

[202] Zitouni (I.), Mari (J.F.), Smaïli (K.) et Haton (J.P.). – Variable-length sequence language model for large vocabulary continuous dictation machine: The n-seqgram approach. *In: Proceeding of the European Conference On Speech Communication and Technologie.* – Septembre 1999.

[203] Zitouni (I.) et Smaïli (K.). – Apport d'une grammaire d'unification dans un système de dicté automatique. *In: Deuxièmes journées jeunes chercheurs en parole.* – La Rochelle, France, Novembre 1997.