Méthodes de segmentation et d'analyse automatique de textes thaï

Krit Kosawat

Université de Marne-La-Vallée

Institut d'électronique et d'informatique Gaspard-Monge

École Doctorale : Information Communication Modélisation et Simulation

Laboratoire d'Informatique

THÈSE

pour obtenir le grade de

Docteur de l'Université de Marne-La-Vallée

Discipline : Informatique

présentée et soutenue publiquement par

Krit KOSAWAT

le 8 septembre 2003

Méthodes de segmentation et d'analyse automatique de textes thaï

Automated methods of segmentation and analysis of Thai texts

Tome I : Mémoire

Directeurs de thèse
†Maurice GROSS
Éric LAPORTE

Jury :

Gilles	DELOUCHE	
Jacques	DÉSARMÉNIEN	
Franz	GUENTHNER	(*Rapporteur*)
Éric	LAPORTE	
Denis	MAUREL	(*Rapporteur*)

À la mémoire de Monsieur le Professeur Maurice GROSS

Remerciements

J'aimerais dédier cette thèse à Maurice GROSS, mon premier directeur de thèse. Ça a été un très grand honneur pour moi de pouvoir partager une passion avec un très grand scientifique. Ses critiques et conseils, toujours justes et précieux, m'ont souvent ouvert d'intéressantes voies de recherche.

Je remercie sincèrement Éric LAPORTE d'avoir accepté de diriger la suite de mon travail et de m'avoir guidé dans la bonne direction. Je dois beaucoup à sa méthode de travail et à son énergie inépuisable.

Ce travail n'aurait pas été achevé sans l'aide immense de Max SILBERZTEIN, qui m'a encouragé à aller toujours plus loin et est toujours prêt à réimplémenter son programme pour prendre en compte les difficultés de la langue thaï.

Je remercie vivement mes cinq relecteurs : Matthieu CONSTANT, Christian LECLÈRE, Claude MARTINEAU, Sébastien PAUMIER et Antoinette RENOUF, jamais fatigués de relire et de recorriger ma rédaction et de faire toujours de bons commentaires.

Je dois beaucoup à Takuya NAKAMURA pour ses connaissances en linguistique et surtout en bibliographie, à Chanaya DARNSAWASDI et Aurapin SIRIBOONMA pour la linguistique thaï, à Chamaiporn PRASERTKITTIKUL et Mayuree VITHAYAVEROJ pour la transcription phonétique, à Surasak NA NAKORN pour la traduction de termes techniques et à Cédrick FAIRON pour les expressions rationnelles. Je remercie également Teresa GOMEZ-DIAZ pour la version électronique de cette thèse et Anastasia YANNACOPOULOU qui m'a beaucoup facilité les démarches administratives.

Je n'oublierai jamais « la vie en rouge » (plutôt Bordeaux) passée au LADL avec Nathalie BELY, Blandine COURTOIS, Maurice GROSS, Christian LECLÈRE, Marianne MATHIS, Annie MEUNIER, Mireille PIOT, Philippe VASSEUX et notre visiteur régulier André DUGAS (lui, c'est plutôt Bourgogne). Je remercie également tous les membres de l'IGM pour leur accueil chaleureux.

Mes remerciements s'adressent aussi à mes rapporteurs : Franz GUENTHNER et Denis MAUREL, et également à Gilles DELOUCHE et Jacques DÉSARMÉNIEN qui m'ont fait l'honneur de faire partie de mon jury.

Enfin, je tiens à remercier le Bureau du Conseiller d'Éducation auprès de l'Ambassade Royale de Thaïlande et le National Electronic and Computer Technology Center pour la bourse d'études dont j'ai bénéficié.

Résumé

Ce travail de thèse a pour objectif de concevoir et réaliser un module informatico-linguistique apte à effectuer des analyses automatiques de textes thaï sous le système INTEX©.

Basé fondamentalement sur les langues indo-européennes écrites avec l'alphabet latin, INTEX© rencontre quelques difficultés pour travailler sur une langue très différente comme le thaï. Le problème crucial est la segmentation en mots et en phrases, étant donné que le thaï n'a pas de séparateur de mot : une phrase est écrite en une séquence de lettres continues, et les séparateurs de phrase sont fréquemment ambigus.

Aussi avons-nous développé et évalué deux méthodes de segmentation en mots, par expressions rationnelles et par transducteurs à nombre fini d'états, qui découpent respectivement des textes thaï en lettres et en syllabes. Nous avons également créé les dictionnaires électroniques du thaï qui servent à la fois à reconnaître les mots à partir des lettres ou des syllabes et à les étiqueter avec les codes syntaxiques et sémantiques. Deux méthodes de segmentation en phrases thaï, par la ponctuation et par mots-clés, sont également proposées et évaluées.

Nous montrons enfin que, grâce à notre travail, INTEX© est capable d'analyser des documents thaï, malgré toutes les difficultés.

Mots-clés : segmentation en mots, segmentation en phrases, dictionnaire électronique, analyse automatique de textes, thaï, INTEX, automate, transducteur, expression rationnelle.

Abstract

The aim of this thesis is to design and implement a computational linguistic module for analysing Thai texts under the INTEX© system.

Based essentially on Indo-European languages written in the Latin alphabet, INTEX© encounters some difficulties when processing a very different language such as Thai. The crucial problem is word and sentence segmentation, since Thai has no word separator: a sentence is written as a continuous sequence of letters, and sentence separators are frequently ambiguous.

Accordingly, we have developed and evaluated two methods of word segmentation, firstly by using Regular Expressions and secondly Finite-State Transducers, which segment Thai texts into letters and syllables respectively. We have also created Thai Electronic Dictionaries, which are used to recognise words from letters or from syllables and, at the same time, to label them with syntactic and semantic tags. Two methods of Thai sentence segmentation, based on punctuation marks and keywords, are also proposed and evaluated.

Finally, we demonstrate that, as a result of our work, INTEX© is able to analyse Thai documents in spite of the difficulties involved.

Keywords : Word Segmentation, Sentence Segmentation, Electronic Dictionary, Texts Analysis, Thai, INTEX, Automata, Transducer, Regular Expression.

Table des matières

LEXIQUE 1

INTRODUCTION 3

0.1 Problématique 3

0.2 Objectif 4

0.3 État de l'art 4
0.3.1 Segmentation en syllabes 4
0.3.2 Segmentation en mots 5
0.3.3 Segmentation en phrases 7
0.3.4 Corpus étiqueté du thaï 8

0.4 Méthodologie 9
0.4.1 Environnement de développement 9
0.4.2 Format des dictionnaires 9
0.4.3 Mode de segmentation 9
0.4.4 Modification des textes et des dictionnaires 10

0.5 Structure de la thèse 10

CHAPITRE 1 SYSTÈME INTEX 11

1.1 Théories de base 12
1.1.1 Expressions rationnelles 12
1.1.2 Automates et Transducteurs 14
1.1.3 Transducteurs avec variables (Enhanced Transducers) 16

1.2 Éléments essentiels d'INTEX 16
1.2.1 Fichier Alphabet 16
1.2.2 Modules de normalisation d'un texte 18
1.2.3 Ressources lexicales 22

1.3 Fonctionnalités principales d'INTEX 32
1.3.1 Analyse lexicale 32
1.3.2 Recherche de motifs 34
1.3.3 Levée d'ambiguïtés lexicales 38
1.3.4 Analyse syntaxique 40

1.4 Ajout de nouvelles langues dans INTEX 40
1.4.1 Critères de Windows 41
1.4.2 Critères d'INTEX 41
1.4.3 Exemple du thaï (langue à 1 octet) 43
1.4.4 Exemple du coréen (langue à 2 octets) 45

1.5 Problèmes du thaï dans INTEX 49

CHAPITRE 2 SYSTÈME D'ÉCRITURE DU THAÏ 51

2.1 Alphabet thaï 52
2.1.1 Consonnes 52
2.1.2 Voyelles 55
2.1.3 Signes diacritiques 57
2.1.4 Signes pāli-sanskrits 58
2.1.5 Chiffres 59
2.1.6 Signes de ponctuation 59

2.2 Syllabes 60
2.2.1 Composition des syllabes 60
2.2.2 Anomalies des syllabes thaï 62
2.2.3 Ordre de constitution syllabique 67

2.3 Mots 68
2.3.1 Mots simples 68
2.3.2 Mots composés 70
2.3.3 Ambiguïtés de découpage des mots thaï 71

CHAPITRE 3 DICTIONNAIRES ÉLECTRONIQUES DU THAÏ 73

3.1 Sources lexicales 73

3.2 Structure des entrées 73
3.2.1 Définition des mots simples et des mots composés 74
3.2.2 Informations lexicales 74

3.3 DELAS du thaï 77
3.3.1 Entrées de mots uniques 77
3.3.2 Entrées de mots homographes 78
3.3.3 Variantes orthographiques 78
3.3.4 Ordre alphabétique 79

3.4 DELAF du thaï 81
3.4.1 Structure des entrées 81
3.4.2 Ordre alphabétique 81
3.4.3 Liens entre variantes orthographiques 82
3.4.4 Dictionnaire des chiffres thaï 82

3.5 DELAC du thaï 84
3.5.1 Structure des entrées 84
3.5.2 Ordre alphabétique 84

3.6 DELACF du thaï 84
3.6.1 Structure des entrées 84
3.6.2 Ordre alphabétique 85

3.7 Problème d'application des dictionnaires 85

CHAPITRE 4 MÉTHODES DE SEGMENTATION 89

4.1 Segmentation en mots 89
4.1.1 Méthode par caractères 89
4.1.2 Méthode par syllabes 100
4.1.3 Conclusion 134

4.2 Segmentation en phrases 135
4.2.1 Méthode par la ponctuation 135
4.2.2 Méthode par mots-clés 136
4.2.3 Conclusion 138

4.3 Application des graphes de segmentation 138
4.3.1 Mode « Graphique » 138
4.3.2 Mode « Console » 139

CHAPITRE 5 ÉVALUATION ET COMPARAISON 141

5.1 Méthode d'évaluation 141
5.1.1 Outil d'évaluation 141
5.1.2 Évaluation sur les mots 142
5.1.3 Évaluation sur les phrases 142

5.2 Présentation des résultats 143
5.2.1 Résultats sur les mots 143
5.2.2 Bilan sur les mots 147
5.2.3 Résultats sur les phrases 148
5.2.4 Bilan sur les phrases 148

5.3 Conclusion 150

CHAPITRE 6 ANALYSE AUTOMATIQUE DE TEXTES THAÏ 151

6.1 Analyse morphologique 151
6.1.1 Analyse des affixes 151
6.1.2 Analyse des phonèmes muets 152

6.2 Analyse lexicale 153
6.2.1 Analyse des mots simples et composés 153
6.2.2 Analyse des expressions figées 154

6.3 Identification d'expressions par des graphes 157
6.3.1 Nombres décimaux 157
6.3.2 Expressions numériques 157
6.3.3 Expressions de date 161

6.4 Analyse syntaxique 165
6.4.1 Transducteurs du texte 165
6.4.2 Réduction des transducteurs du texte 167

6.5 Grammaires locales de levée d'ambiguïtés 170

CONCLUSION ET PERSPECTIVES **179**

BIBLIOGRAPHIE **181**

Corpus **192**

ANNEXE I FICHIERS ALPHABET **193**

ANNEXE II DICTIONNAIRES ÉLECTRONIQUES DU THAÏ **195**

ANNEXE III DICTIONNAIRES P0 ..... **373**

ANNEXE IV GRAPHES « REPLACE 1 » **385**

ANNEXE V DICTIONNAIRES P1 ..... **401**

ANNEXE VI GRAPHES « REPLACE 2 » **407**

ANNEXE VII DICTIONNAIRES P2 ..... **413**

ANNEXE VIII GRAPHES « REPLACE 3 » **419**

ANNEXE IX DICTIONNAIRES P3 ..... **425**

ANNEXE X GRAPHES « SENTENCE » **431**

ANNEXE XI CORPUS D'ANALYSE **443**

ANNEXE XII RÉSULTATS DES ANALYSES **495**

Liste des tableaux

Tableau 1.1 Symboles formels d'INTEX 13
Tableau 1.2 Dictionnaire de mots composés non-ambigus de l'anglais 20
Tableau 1.3 Catégories du DELAS français 24
Tableau 1.4 Codes syntactico-sémantiques du français 24
Tableau 1.5 Codes flexionnels du DELAF français 27
Tableau 1.6 Symboles lexicaux disponibles après l'application des ressources lexicales 34
Tableau 1.7 Microsoft Windows Codepage 1252 (Latin I) 42
Tableau 1.8 Microsoft Windows Codepage 874 (Thaï) 43
Tableau 1.9 Microsoft Windows Codepage 949 (Coréen) 46
Tableau 1.10 Caractères coréens qui commencent par 0xB0 47
Tableau 2.1 Consonnes du thaï 54
Tableau 2.2 Caractères vocaliques du thaï 55
Tableau 2.3 Voyelles du thaï 57
Tableau 2.4 Marques de tons du thaï 57
Tableau 2.5 Signes de phonème spécial du thaï 58
Tableau 2.6 Signes pāli-sanskrits 58
Tableau 2.7 Chiffres thaï 59
Tableau 2.8 Signes de ponctuation du thaï 59
Tableau 2.9 Syllabe à 3 éléments 60
Tableau 2.10 Syllabe à 4 éléments (avec consonne finale) 61
Tableau 2.11 Syllabe à 4 éléments (avec phonème muet) 61
Tableau 2.12 Syllabe à 5 éléments 61
Tableau 2.13 Formes syllabiques avec et sans consonne finale 66
Tableau 3.1 Codes des catégories des mots thaï 75
Tableau 3.2 Codes des informations sur les pronoms 76
Tableau 3.3 Codes des informations syntactico-sémantiques 76
Tableau 3.4 Codes des informations dialectologiques 77
Tableau 3.5 Table des caractères du thaï 80
Tableau 4.1 Sélection des zones de dictionnaires 99
Tableau 4.2 Nombre de lexèmes des Dictionnaires P0 et P1 119
Tableau 4.3 Nombre de lexèmes des Dictionnaires P0, P1 et P2 127
Tableau 4.4 Nombre de lexèmes des Dictionnaires P0, P1, P2 et P3 133
Tableau 5.1 Évaluation sur la reconnaissance des mots dans le Texte N°1 143
Tableau 5.2 Évaluation sur la reconnaissance des mots dans le Texte N°2 143
Tableau 5.3 Évaluation sur la reconnaissance des mots dans le Texte N°3 144
Tableau 5.4 Évaluation sur la reconnaissance des mots dans le Texte N°4 144
Tableau 5.5 Évaluation sur la reconnaissance des mots dans le Texte N°5 145
Tableau 5.6 Évaluation sur la reconnaissance des mots dans le Texte N°6 145
Tableau 5.7 Évaluation sur la reconnaissance des mots dans le Texte N°7 145
Tableau 5.8 Évaluation sur la reconnaissance des mots dans le Texte N°8 146
Tableau 5.9 Évaluation sur la reconnaissance des mots dans le Texte N°9 146
Tableau 5.10 Évaluation sur la reconnaissance des mots dans le Texte N°10 147
Tableau 5.11 Bilan sur les méthodes de segmentation en mots 147
Tableau 5.12 Évaluation sur l'insertion de séparateurs de phrases dans 10 textes 148

Tableau 5.13 Évaluation des méthodes concurrentes..... 149
Tableau 5.14 Bilan sur les méthodes de segmentation en phrases..... 150
Tableau 6.1 Analyse des mots simples et composés..... 153
Tableau 6.2 Catégorie grammaticale du corpus « SiPhanDin-P3 »..... 153

Liste des figures

Figure 1.1 Automate traditionnel (à gauche) et graphe d'INTEX (à droite) *14*
Figure 1.2 Réseau de transitions récursif *15*
Figure 1.3 Sous-graphe « Dnum [2,99] » *15*
Figure 1.4 Transducteur avec variables *16*
Figure 1.5 Graphe « Sentence » du français *19*
Figure 1.6 Graphe « Replace » du français *21*
Figure 1.7 Sous-graphe « Elisions » du français *21*
Figure 1.8 Sous-graphe « Contractions » du français *22*
Figure 1.9 Graphe du DELAS sur « france » *25*
Figure 1.10 Graphe du DELAF sur « tsar » *28*
Figure 1.11 Transducteur de flexion « N32 » *29*
Figure 1.12 Transducteur de flexion « N4 » *30*
Figure 1.13 Graphe du DELACF sur « roman policier » *31*
Figure 1.14 Graphe du DELAE sur « perdre…la raison » *32*
Figure 1.15 Vocabulaire du texte *34*
Figure 1.16 Transducteur d'un groupe nominal humain *36*
Figure 1.17 Résultat de l'option « Merge with input text » *36*
Figure 1.18 Résultat de l'option « Replace recognized sequences » *36*
Figure 1.19 Grammaire locale de levée d'ambiguïtés *39*
Figure 1.20 Texte après l'étiquetage linéaire *39*
Figure 1.21 Structure de la phrase « Il couvre une pomme de terre cuite. » *40*
Figure 2.1 Position des caractères d'une syllabe *67*
Figure 2.2 Représentation informatique d'une syllabe *67*
Figure 2.3 Position des caractères d'un phonème muet *68*
Figure 2.4 Représentation informatique d'un phonème muet *68*
Figure 3.1 DELAF de chiffres thaï *82*
Figure 3.2 Sous-graphe « TNB.grf » *83*
Figure 3.3 Liste de lexèmes provenant d'un texte français *85*
Figure 3.4 Liste de lexèmes provenant d'un texte thaï *86*
Figure 3.5 Application des dictionnaires sur un texte thaï manuellement découpé *87*
Figure 4.1 Forme syllabique modifiée par la méthode par caractères *90*
Figure 4.2 Phonème muet modifié par la méthode par caractères *90*
Figure 4.3 Corpus P0 et ses lexèmes *93*
Figure 4.4 Corpus P0 avec les espaces blancs cachés *94*
Figure 4.5 Vocabulaire du texte d'un Corpus P0 ..... *96*
Figure 4.6 Regroupement des caractères inséparables *101*
Figure 4.7 Regroupement des caractères inséparables dans un phonème muet *101*
Figure 4.8 Graphe « Consonant » *102*
Figure 4.9 Graphe « Consonant-1 » *103*
Figure 4.10 Graphe « Consonant-2 » *103*
Figure 4.11 Graphe « Consonant-3 » *104*
Figure 4.12 Graphe « Speller » *104*
Figure 4.13 Graphe « Speller-1 » *105*

Figure 4.14 Graphe « Tone » *105*
Figure 4.15 Graphe « Tone-1 » *105*
Figure 4.16 Graphe « Replace 1.1 » *106*
Figure 4.17 Graphe « Replace 1.2 » *106*
Figure 4.18 Graphe « Replace 1.3 » *107*
Figure 4.19 Graphe « Replace 1.4 » *107*
Figure 4.20 Graphe « Replace 1.5 » *107*
Figure 4.21 Graphe « Replace 1.6 » *108*
Figure 4.22 Graphe « Replace 1.7 » *108*
Figure 4.23 Graphe « Replace 1.8 » *108*
Figure 4.24 Graphe « Replace 1.9 » *109*
Figure 4.25 Graphe « Replace 1.10 » *109*
Figure 4.26 Graphe « Replace 1.11 » *110*
Figure 4.27 Graphe « Replace 1.12 » *110*
Figure 4.28 Graphe « Replace 1.13 » *111*
Figure 4.29 Graphe « Replace 1.14 » *111*
Figure 4.30 Graphe « Replace 1.15 » *112*
Figure 4.31 Graphe « Replace 1.16 » *112*
Figure 4.32 Graphe « Replace 1.17 » *113*
Figure 4.33 Graphe « Replace 1.18 » *113*
Figure 4.34 Graphe « Replace 1.19 » *113*
Figure 4.35 Graphe « Replace 1.20 » *114*
Figure 4.36 Graphe « Replace 1.21 » *114*
Figure 4.37 Graphe « Replace 1.22 » *114*
Figure 4.38 Graphe « Replace 1.23 » *115*
Figure 4.39 Graphe « Replace 1.24 » *115*
Figure 4.40 Graphe « Replace 1.25 » *116*
Figure 4.41 Graphe « Replace 1.26 » *116*
Figure 4.42 Graphe « Replace 1.27 » *116*
Figure 4.43 Graphe « Replace 1.28 » *117*
Figure 4.44 Graphe « Replace 1.29 » *117*
Figure 4.45 Corpus P1 et ses lexèmes *118*
Figure 4.46 Corpus P1 avec les espaces blancs cachés *118*
Figure 4.47 Vocabulaire du texte d'un Corpus P1 ..... *121*
Figure 4.48 Graphe « Karant » *122*
Figure 4.49 Graphe « Replace 2.1 » *123*
Figure 4.50 Graphe « Replace 2.2 » *123*
Figure 4.51 Graphe « Replace 2.3 » *124*
Figure 4.52 Graphe « Replace 2.4 » *124*
Figure 4.53 Graphe « Replace 2.5 » *125*
Figure 4.54 Graphe « Replace 2.6 » *125*
Figure 4.55 Corpus P2 et le lexème regroupé par « Replace 2 » *126*
Figure 4.56 Vocabulaire du texte d'un Corpus P2 ..... *127*
Figure 4.57 Graphe « Prefix-Cha » *128*
Figure 4.58 Graphe « Suffix-Syl » *129*
Figure 4.59 Graphe « Consonant-3-withTone » *129*
Figure 4.60 Graphe « Replace 3.1 » *130*
Figure 4.61 Graphe « Replace 3.2 » *130*
Figure 4.62 Graphe « Replace 3.3 » *130*
Figure 4.63 Graphe « Replace 3.4 » *131*

Figure 4.64 Graphe « Replace 3.5 » *131*
Figure 4.65 Corpus P3 et les lexèmes regroupés par « Replace 3 » *132*
Figure 4.66 Vocabulaire du texte d'un Corpus P3 ..... *134*
Figure 4.67 Graphe « Sentence-1 » *136*
Figure 4.68 Graphe « Sentence-2 » *137*
Figure 4.69 Graphe « Sentence-3 » *137*
Figure 6.1 Expression figée « tháŋ…kàp… » *154*
Figure 6.2 Expression figée « tháŋ…lé… » *154*
Figure 6.3 Expression figée « tháŋ…tháŋ… » *154*
Figure 6.4 Expression figée « …daj…nùŋ » idéale *155*
Figure 6.5 Expression figée « …daj…nùŋ » *155*
Figure 6.6 Expression figée « …nùŋ…daj » idéale *155*
Figure 6.7 Expression figée « …nùŋ…daj » *156*
Figure 6.8 Expression figée « …léːw…lâw » idéale *156*
Figure 6.9 Expression figée « …léːw…lâw » *156*
Figure 6.10 Expression des nombres décimaux *157*
Figure 6.11 Graphe « Tnum1-9 » *157*
Figure 6.12 Graphe « Tnum10-99 » *158*
Figure 6.13 Graphe « Tnum3-9 » *158*
Figure 6.14 Graphe « Tnum100-999 » *158*
Figure 6.15 Graphe « Tnum1000-9999 » *159*
Figure 6.16 Graphe « Tnum10000-99999 » *159*
Figure 6.17 Graphe « Tnum100000-999999 » *159*
Figure 6.18 Graphe « TnumMillion » *160*
Figure 6.19 Expression numérique en thaï « RealNumberExpression » *160*
Figure 6.20 Expression de date en thaï *161*
Figure 6.21 Expression de date « ModernDate » *161*
Figure 6.22 Graphe « Day » *162*
Figure 6.23 Graphe « Tnum1-31 » *162*
Figure 6.24 Graphe « Month » *162*
Figure 6.25 Graphe « Year » *163*
Figure 6.26 Graphe « YearUnit » *163*
Figure 6.27 Graphe « Tnum1-9999 » *163*
Figure 6.28 Expression de date « TraditionalDate » *164*
Figure 6.29 Graphe « ThaiMonth » *164*
Figure 6.30 Graphe « ThaiYear » *164*
Figure 6.31 Expression de date « NumericalDate » *165*
Figure 6.32 Résultat de la recherche des expressions de date *165*
Figure 6.33 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » en version P0 ..... *166*
Figure 6.34 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » en version P3 ..... *166*
Figure 6.35 Transducteur du texte en version P3 appliqué uniquement avec les dictionnaires z1 ..... *167*
Figure 6.36 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » version P0 réduit *167*
Figure 6.37 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » version P3 réduit *168*
Figure 6.38 Transducteur du texte en version P3 appliqué uniquement avec les dictionnaires z1 et réduit *168*
Figure 6.39 Transducteur du texte « ʔìːkphiaːŋwandiaːw » en version P0 ..... *169*
Figure 6.40 Transducteur du texte « ʔìːkphiaːŋwandiaːw » en version P3 ..... *169*
Figure 6.41 Transducteur du texte « ʔìːkphiaːŋwandiaːw » version P0 réduit *169*
Figure 6.42 Grammaire locale de la phrase « chănmaːrɔːkràːpphrásŏŋ » *170*

Figure 6.43 Transducteur du texte « chănma:rɔ:krà:pphrásŏŋ » désambiguïsé *170*
Figure 6.44 Grammaire locale de l'interjection..... *171*
Figure 6.45 Transducteur du texte « thŏ:ꜜ »..... *171*
Figure 6.46 Transducteur du texte « thŏ:ꜜ » désambiguïsé..... *171*
Figure 6.47 Transducteur du texte « mê:câwwó:jꜜ »..... *171*
Figure 6.48 Transducteur du texte « mê:câwwó:jꜜ » désambiguïsé..... *172*
Figure 6.49 Grammaire locale du verbe auxiliaire..... *172*
Figure 6.50 Transducteur du texte « thɣ:càtɔ̂:ŋma: »..... *172*
Figure 6.51 Transducteur du texte « thɣ:càtɔ̂:ŋma: » désambiguïsé..... *173*
Figure 6.52 Transducteur du texte « thɣ:càtɔ̂:ŋthô:t »..... *173*
Figure 6.53 Transducteur du texte « thɣ:càtɔ̂:ŋthô:t » retenu par la grammaire locale..... *174*
Figure 6.54 Transducteur du texte « thɣ:càtɔ̂:ŋthô:t » ignoré par la grammaire locale..... *174*
Figure 6.55 Transducteur du texte « ta:klom » *174*
Figure 6.56 Grammaire locale de la séquence « ta:klom »..... *175*
Figure 6.57 Transducteur du texte « ta:klomsŭa:j »..... *175*
Figure 6.58 Transducteur du texte « ta:klomsŭa:j » désambiguïsé..... *176*
Figure 6.59 Transducteur du texte « ta:klomto: » *176*
Figure 6.60 Transducteur du texte « ta:klomto: » désambiguïsé..... *176*
Figure 6.61 Transducteur du texte « ta:klomjen » *177*
Figure 6.62 Transducteur du texte « ta:klomjen » désambiguïsé..... *177*
Figure 6.63 Transducteur du texte « ta:klom » partiellement désambiguïsé *177*

Liste des algorithmes

Algorithme 3.1 Module de tri des mots thaï *80*
Algorithme 4.1 Programme de modification des textes « PrepareText.pl » *91*
Algorithme 4.2 Règles de modification des textes et des dictionnaires « GramText.txt » *91*
Algorithme 4.3 Script de segmentation « Console.bat » *139*

LEXIQUE

Lettres	Caractères recensés dans le fichier Alphabet (e.g. : a, b, c)
Séparateurs	Caractères hors du fichier Alphabet sauf les chiffres et les blancs (e.g. : &, !, ?)
Chiffres (digits)	Chiffres arabes de 0 à 9
Blancs	Espace, caractère de tabulation et changement de ligne/paragraphe
Lexèmes (tokens)	Objets atomiques INTEX© classés en quatre types : formes simples, étiquettes, chiffres et séparateurs
Formes Simples	Séquences de lettres entre deux séparateurs (e.g. : aaa, xyz)
Étiquettes (tags)	Informations linguistiques écrites entre accolades (e.g. : {aujourd'hui,.ADV}
Mots Simples	Formes simples qui correspondent à une ou plusieurs entrées lexicales dans un dictionnaire de mots simples (e.g. : pomme, table)
Mots Composés	Séquences de formes simples et de séparateurs qui correspondent à une ou plusieurs entrées lexicales dans un dictionnaire de mots composés (e.g. : pomme de terre, aujourd'hui)
Expressions Figées	Séquences potentiellement discontinues de formes simples qui correspondent à des entrées lexicales dans une grammaire lexicale d'expressions figées (e.g. : ne…pas, prendre…en compte)
LADL	Laboratoire d'Automatique Documentaire et Linguistique
DELA	Dictionnaire Électronique du LADL

DELAS	DELA des Mots Simples
DELAF	DELAS des Formes Fléchies
DELAC	DELA des Mots Composés
DELACF	DELAC des Formes Fléchies
DELAE	DELA des Expressions Figées

INTRODUCTION

Ces dernières années, la recherche en linguistique a considérablement progressé grâce à des outils informatiques. D'autant plus que les ordinateurs à ce jour ont beaucoup évolué : ils sont capables de traiter aisément des documents de taille importante en très peu de temps. Les linguistes aujourd'hui peuvent en bénéficier de manière fiable pour faciliter leurs travaux répétitifs ou quantitatifs en analyses de textes. INTEX© est un des outils dédiés à ces tâches. Intégrant plusieurs fonctionnalités destinées aux linguistes, il est l'outil le plus utilisé dans la communauté RELEX (réseau européen des lexiques-grammaires) qui comprend une dizaine d'équipes multinationales. Chaque équipe dans la communauté participe à ce programme en travaillant sur la description de sa langue maternelle, tant au niveau *dictionnaires électroniques* qu'au niveau *lexiques-grammaires* et toutes les équipes utilisent des méthodes semblables afin de pouvoir comparer leurs problèmes et concevoir un jour une standardisation pour les parties communes de toutes les langues. La communauté s'étend actuellement à des langues non européennes comme le coréen, par exemple.

Ce travail de thèse a permis d'intégrer le thaï dans la communauté. Nous proposons une description de la langue selon le format DELA, et donnons les éléments nécessaires au bon fonctionnement d'un programme qui propose des solutions, réalisables et fiables, aux problèmes de la reconnaissance des mots et des phrases thaï.

0.1 Problématique

Le thaï est une langue asiatique complètement différente des langues européennes. Les problèmes rencontrés devraient être différents de ceux rencontrés dans la communauté. On

peut mentionner ici deux problèmes majeurs : l'absence de séparateur de mots et l'ambiguïté du séparateur de phrases. En effet, un texte thaï ressemble à une suite ininterrompue de lettres, au mieux coupée à la fin de la phrase par un blanc qui est équivoque avec d'autres signes de ponctuation. Considérons un exemple de texte thaï ci-dessous :

จริงหรอกครับที่เราอยู่กับเทคโนโลยีมาตั้งแต่สมัยหิน ขาดเสียซึ่งเทคโนโลยี เราก็คงมีชีวิตไม่ต่างจากสัตว์มากนัก

(*Il est vrai que nous vivons avec la technologie depuis l'âge de pierre. Sans la technologie, nous ne serions pas très éloignés des animaux.*)

Ce texte comprend trois séquences de lettres et deux blancs qui n'ont pas la même fonction : le premier est un séparateur de phrases tandis que le deuxième est équivalent à une virgule. Les limites de phrases sont donc ambiguës. De plus, les mots sont soudés les uns aux autres, ce qui pose un autre problème lors de leur reconnaissance : les limites de mots sont également ambiguës car plusieurs découpages sont a priori possibles. Un système d'analyse de textes thaï doit d'abord être capable de traiter ce genre de problèmes.

0.2 Objectif

L'objectif de cette thèse est de constituer des algorithmes de segmentation en mots et en phrases, compatibles avec INTEX©, et de construire des dictionnaires électroniques du thaï, ainsi que tous les éléments nécessaires pour concevoir et réaliser un outil informatico-linguistique apte à effectuer des analyses automatiques de textes thaï.

0.3 État de l'art

La recherche sur la segmentation de textes thaï a commencé il y a une vingtaine d'années ; d'abord, pour les éditeurs de texte, et ensuite, pour les autres applications en traitement des langues naturelles.

0.3.1 Segmentation en syllabes

La segmentation de première génération cherche seulement à reconnaître la limite des syllabes ou des mots afin de savoir où commence une nouvelle ligne dans un éditeur de texte lorsqu'une séquence de caractères dépasse la largeur de page. La précision dans ce cas n'est pas cruciale, car on ne rencontre ce problème qu'une seule fois à la fin de chaque ligne.

0.3.1.1 Méthode par règles

Les syllabes thaï ont une composition assez spécifique. Yupin THAIRATANANOND (1981) décrit plusieurs règles de formation syllabique conformes à la grammaire thaï traditionnelle en classant les caractères thaï en 5 groupes, d'après leur fonction. Son programme est développé avec le langage PL/I et examine les textes de la droite vers la gauche. Les syllabes exceptionnelles ne se conformant pas aux règles décrites sont stockées dans un fichier distinct. Cette méthode donne une précision de la segmentation en syllabes de plus de 85%.

Surin CHARNYAPORNPONG (1983) propose, à partir de l'analyse des syllabes thaï, des règles qui déterminent les frontières gauche et droite des syllabes avec une précision de 96%.

0.3.1.2 Méthode par dictionnaire

La méthode par règles comprend de nombreuses exceptions et elle est difficile à implémenter. Pour améliorer la précision de la segmentation en syllabes et surtout la vitesse d'exécution, Yuen POOVARAWAN et Wiwat IMARROM (1986) proposent un dictionnaire dans lequel 5 400 syllabes courantes sont stockées. Le programme parcourt le texte de gauche à droite et compare chaque séquence de lettres avec chaque syllabe du dictionnaire, en favorisant la syllabe la plus longue. Lorsqu'une syllabe correspond, il fait une marque dans le texte à la fin de cette syllabe, qui sera le début de la comparaison suivante. Cette méthode donne une précision de 99% sur des documents issus de quotidiens, mais le résultat diminue beaucoup lors de l'analyse des documents techniques comprenant de nombreuses syllabes inconnues.

Duangkaew SAWAMIPAK (1990) améliore l'approche ci-dessus en proposant une méthode hybride qui utilise également un dictionnaire de syllabes, accompagné de 43 règles de composition syllabique sous forme d'expressions rationnelles. Cette méthode peut améliorer la précision (jusqu'à 99,7%) même si le texte comprend beaucoup de syllabes hors du dictionnaire.

0.3.2 Segmentation en mots

Certaines recherches en traitement des langues naturelles telles que l'étiquetage de parties de discours, la traduction automatique, la recherche d'informations, etc. exigent la segmentation en mots et non en syllabes. C'est pourquoi les méthodes de segmentation de deuxième génération travaillent toutes sur les mots.

0.3.2.1 Méthode par dictionnaire

Puisqu'il n'y a pas de règles de constitution des mots thaï, la segmentation de texte en mots doit être faite par consultation d'un dictionnaire qui contient, dans ce cas, les mots au lieu des syllabes. Le principe est le même que pour les syllabes, c'est-à-dire que dans une séquence de caractères du texte, la fin d'un mot segmenté sera le début du mot suivant. Les segmentations seront donc faites l'une après l'autre, en flot linéaire. Le problème est de savoir comment choisir la bonne segmentation lorsque le texte peut être découpé de plusieurs façons. Par exemple, la séquence « ตากลม » [ta:klom] peut être segmentée en « ตา/กลม » [ta:/klom] ou en « ตาก/ลม » [ta:k/lom].

Ruttikorn VARAKULSIRIPUNTH *et al*. (1989) proposent la méthode « Longest Word Mapping » qui fait une comparaison de gauche à droite en donnant la priorité aux mots les plus longs. Cette méthode donne une précision bien meilleure qu'en donnant la priorité aux mots les plus courts.

Samphan RARUENROM (1991) utilise la même méthode avec la possibilité de recul « Back Tracking » lorsque le mot le plus long ne permet pas de segmenter le texte jusqu'à la fin de la séquence.

Somprathana RATTHAYANOND (1992) préfère présenter toutes les possibilités de découpage mais ne se prononce pas sur la bonne segmentation.

Virach SORNLERTLAMVANICH (1993) choisit la méthode « Maximal Matching » : la segmentation qui permet d'obtenir le nombre minimal de mots.

Or, le découpage en flot linéaire ne permet jamais de réussir la segmentation en mots à 100% parce que les textes sont souvent ambigus : sans aucune information sémantique, on ne peut pas déterminer les bons segments. De plus, le résultat dépend du degré de complétude du dictionnaire utilisé ; en effet, un mot inconnu peut facilement bloquer le système ou provoquer une erreur.

0.3.2.2 Méthode par apprentissage

De nos jours, les ordinateurs sont assez puissants pour faire de grands calculs. Lorsque de grands corpus préalablement segmentés et même entièrement étiquetés sont disponibles, on peut utiliser des techniques d'apprentissage issues de la recherche en intelligence artificielle pour que le système apprenne lui-même comment segmenter un texte.

Asanee KAWTRAKUL *et al*. (1995) utilisent un dictionnaire pour segmenter et étiqueter un texte en présentant toutes les segmentations possibles. Ensuite, les auteurs appliquent le

modèle statistique de Markov pour calculer la probabilité de chaque segment, en utilisant des informations apprises sur un corpus d'entraînement, afin de choisir la meilleure segmentation.

Surapant MEKNAVIN *et al*. (1997) utilisent deux algorithmes d'apprentissage (sur un corpus d'entraînement déjà segmenté), « RIPPER » et « Winnow », qui créent des règles de segmentation à partir des informations extraites des mots et de leurs contextes en calculant la probabilité par le modèle « Trigram » sur les parties de discours.

Chompunuch KOOPTIWOOT (1999) s'intéresse seulement aux séquences ambiguës en adoptant un système de programmation logique inductive, nommé « FOIL », qui tire également ses informations d'un corpus d'entraînement préalablement segmenté.

Néanmoins, la méthode par apprentissage a besoin de grands corpus d'entraînement, de préférence dans le même domaine que les corpus à segmenter, pour que les résultats soient satisfaisants.

D'autres langues asiatiques telles que le chinois et le japonais n'ont pas non plus de séparateur de mots. La segmentation en mots dans ces langues est également un sujet de recherche important. Richard SPROAT *et al*. (1996) proposent une méthode combinée d'informations lexicales et statistiques pour segmenter les séquences de mots chinois, tandis que Masaaki NAGATA (1999) utilise des techniques d'apprentissage pour segmenter les séquences de mots japonais.

0.3.3 Segmentation en phrases

Un des problèmes cruciaux dans un système d'analyse de textes thaï est de segmenter un paragraphe en phrases. Habituellement, un espace blanc est inséré entre deux phrases thaï. Mais d'après les études de Kobkool THAWARANON (1978), il est également inséré entre les groupes ou propositions dans une phrase, entre les mots d'une énumération, entre les prénoms et noms de famille, devant et derrière les chiffres et certains signes de ponctuation, etc. Faute de caractère majuscule qui commence un nom propre ou une nouvelle phrase, la segmentation en phrases pour le thaï n'est pas évidente.

0.3.3.1 Méthode par règles statistiques

Sungkornsarun LONGCHUPOLE (1995) essaie d'informatiser ce problème en proposant une méthode d'extraction des phrases d'un paragraphe. Cette méthode découpe un paragraphe en plusieurs mots et compte le nombre de verbes principaux afin d'estimer le nombre de phrases. Cependant, il faut reconnaître que rien ne distingue un verbe principal d'un autre

verbe. Les conjonctions sont aussi identifiées comme des frontières probables de phrases. La précision de cette approche est de 81,2% mais elle requiert d'analyser un paragraphe entier, ce qui limite la taille des paragraphes utilisables pour cette analyse.

0.3.3.2 Méthode par apprentissage

Pradit MITTRAPIYANURUK et Virach SORNLERTLAMVANICH (2000) proposent la méthode « POS Trigram » qui tire des informations du corpus pré-étiqueté « ORCHID » (*cf. §0.3.4*) et classifie tous les espaces blancs d'un texte en deux types : NSBS (non-sentence-break space) et SBS (sentence-break space) en calculant la probabilité sur les parties de discours du contexte devant et derrière chaque espace par l'algorithme « Viterbi ».

Paisarn CHAROENPORNSAWAT et Virach SORNLERTLAMVANICH (2001) utilisent la même méthode, mais avec l'algorithme d'apprentissage « Winnow », et fournissent un meilleur résultat que le « POS Trigram ».

La segmentation en phrases ne pose aucun problème en chinois et japonais modernes, qui emploient un petit cercle non-ambigu « 。 » pour indiquer la fin des phrases. En revanche, dans les langues occidentales, le problème existe parce qu'un point « . » peut signifier autre chose que la fin de phrase. Max SILBERZTEIN (1993) décrit plusieurs types d'utilisation du point en français, Gregory GREFENSTETTE et Pasi TAPANAINEN (1994) font de même en anglais, tandis qu'André DUGAS (1997) décrit tous les types d'utilisation de tous les signes de ponctuation en français. David D. PALMER et Marti A. HEARST (1997) proposent le système « Satz » qui utilise deux techniques d'apprentissage, réseau de neurone et arbre de décision, pour décrire la distribution des parties du discours à partir des mots figurant devant et derrière chaque signe de ponctuation dans un corpus d'entraînement. Ils montrent que leur système segmente aussi bien les phrases en anglais qu'en français et en allemand. Nathalie FRIBURGER *et al*. (2000) proposent une amélioration des graphes de segmentation en phrases pour le français sous le système INTEX©.

0.3.4 Corpus étiqueté du thaï

En 1997, Virach SORNLERTLAMVANICH *et al*. (1997) ont achevé le projet ORCHID : la construction d'un corpus thaï entièrement segmenté en phrases, en mots et totalement étiqueté par parties du discours. La segmentation en phrases se fait manuellement par des linguistes tandis que la segmentation en mots, ainsi que l'étiquetage, sont automatiquement faits par le modèle statistique « Trigram » et manuellement corrigés par des linguistes. Comprenant 10 864 phrases dont environ 400 000 mots, tous étiquetés avec 47 catégories grammaticales

différentes, ce corpus sert principalement de corpus d'entraînement pour toutes les méthodes de segmentation qui utilisent une technique d'apprentissage.

0.4 Méthodologie

La méthodologie adoptée dans notre travail est décrite ci-dessous :

0.4.1 Environnement de développement

Nous avons choisi INTEX© comme environnement de développement pour les raisons suivantes :

- C'est un système ouvert : on peut facilement ajouter de nouvelles langues ; les utilisateurs sont libres de créer leurs propres ressources.
- Il est compatible avec deux types de ressources : textes et automates.
- Les résultats de l'analyse sont présentés sous forme d'automates, ce qui les rend très compréhensibles, surtout en cas d'ambiguïté car il est capable d'exprimer toutes les possibilités sous forme de plusieurs chemins en parallèles. Cette solution convient très bien aux systèmes d'analyse de textes qui ne doivent laisser échapper aucune information.
- La vitesse d'exécution est très bonne, même en traitant des corpus de taille importante.
- C'est la plate-forme la plus utilisée dans la communauté RELEX. En effet, il est pratiquement plus avantageux de travailler avec la même plate-forme que les autres équipes dans la communauté afin de pouvoir comparer nos travaux et concevoir une standardisation.

0.4.2 Format des dictionnaires

La description de la langue thaï est réalisée au moyen de dictionnaires. Le format choisi est le format DELA, défini au LADL, pour des raisons pratiques : primo, c'est la formalisation des dictionnaires, commune à toute la communauté RELEX ; secundo, c'est le format adopté par le système INTEX©.

0.4.3 Mode de segmentation

La plupart des méthodes de segmentation en mots dans l'état de l'art utilisent le mode de découpage en flot linéaire qui produit un seul chemin sans concurrent dans le résultat. Ce

mode de segmentation ne nous convient pas parce qu'il peut segmenter à tort en omettant des mots corrects lorsque la séquence à segmenter est ambiguë ou comprend des mots inconnus. Étant donné qu'aucune méthode de segmentation n'est parfaite, nous préférons conserver toutes les possibilités de découpage, d'autant plus que notre système est capable de présenter tous les résultats sous forme de plusieurs chemins en parallèles.

0.4.4 Modification des textes et des dictionnaires

Puisqu'INTEX© ne prend pas en compte pour l'instant les langues sans séparateur, la notion de lexème (token) est définie comme une suite de lettres entre deux séparateurs, ce qui ne convient pas à la langue thaï. Pour surmonter ce problème de compatibilité, nous devons adapter la langue au programme, en ajoutant des séparateurs de mots dans nos textes et dictionnaires.

0.5 Structure de la thèse

Cette thèse est organisée en 6 chapitres :

Le Chapitre 1 présente le système INTEX©, ses composants et leurs fonctionnements. Nous expliquons également comment ajouter de nouvelles langues au système.

Le Chapitre 2 explique le système d'écriture du thaï : l'alphabet, la composition des syllabes et des mots.

Le Chapitre 3 décrit la procédure de construction des dictionnaires électroniques du thaï : DELAS, DELAF, DELAC et DELACF.

Le Chapitre 4 propose des méthodes de segmentation des textes thaï en mots et en phrases, compatibles avec le système INTEX©.

Le Chapitre 5 est consacré à l'évaluation de nos méthodes de segmentation et à la comparaison avec des méthodes concurrentes.

Le Chapitre 6 expose enfin différents types d'analyse automatique de textes thaï pouvant être réalisés avec notre système.

Le second tome regroupe les annexes : les fichiers alphabet, les dictionnaires, les graphes de segmentation en mots et en phrases, ainsi que des résultats de l'analyse de textes.

CHAPITRE 1 SYSTÈME INTEX

INTEX© est un environnement de développement permettant de créer des ressources linguistiques à large couverture et de les appliquer sur de vastes corpus en temps réel. Ces ressources peuvent prendre la forme de dictionnaires électroniques, bibliothèques de graphes et bases de données morpho-syntaxiques du type lexique-grammaire.

Il a été créé par Max SILBERZTEIN[1] dans le but d'avoir une plate-forme de description de langues au moyen de dictionnaires à large couverture selon la théorie des grammaires transformationnelles de Zellig HARRIS[2] et la méthodologie développée par Maurice GROSS[3]. Cette plate-forme peut lire des dictionnaires électroniques au format DELA, le format développé au Laboratoire d'Automatique Documentaire et Linguistique (LADL), comportant pour chaque entrée une série d'informations morphologiques, syntaxiques et, dans certains cas, sémantiques. Elle permet aux utilisateurs d'exploiter soit leurs propres ressources, soit de vastes bases de données lexicales composées, en particulier, des dictionnaires du français développés au LADL et des ressources linguistiques utilisant le même formalisme et développées pour d'autres langues au sein du réseau européen de laboratoires RELEX (allemand, anglais, bulgare, coréen, espagnol, grec, italien, norvégien, polonais, portugais, serbo-croate, slovaque, etc.).

INTEX est un système ouvert dans le sens où il permet aux utilisateurs d'ajouter leurs propres ressources au système et même de réutiliser séparément, dans leurs propres applications, les différents programmes dont il se compose. La ressource du type grammaire

[1] SILBERZTEIN 1993, 2000
[2] HARRIS 1970
[3] GROSS 1975

locale, habituellement créée par l'éditeur de graphe d'INTEX, peut être également réemployée dans d'autres grammaires locales. Typiquement, les utilisateurs construisent des graphes élémentaires qui sont équivalents à des transducteurs à nombre fini d'états, et réemploient ces graphes dans d'autres graphes de plus en plus complexes.

Une autre caractéristique d'INTEX est que les objets traités (grammaires, dictionnaires et textes) sont représentés de façon interne par des transducteurs à nombre fini d'états. En conséquence, toutes les fonctionnalités du système se ramènent à un nombre limité d'opérations sur des transducteurs. Par exemple, appliquer une grammaire à un texte revient en gros à construire l'union des transducteurs élémentaires, la déterminiser, puis à calculer l'intersection du résultat avec le transducteur du texte. Cette architecture permet d'utiliser des algorithmes de manière efficace notamment en terme de rapidité.

INTEX est utilisé dans plusieurs centres de recherches universitaires ou privés comme outil de développement linguistique, moteur de recherche, aide à l'enseignement des langues, outil d'extraction terminologique, et pour enseigner l'informatique linguistique.

1.1 Théories de base

INTEX est basé sur les théories suivantes :

1.1.1 Expressions rationnelles

Les expressions rationnelles sont des expressions logiques qui permettent de vérifier qu'une chaîne correspond à un format particulier, défini par l'expression. Elles permettent également, et par voie de conséquence, d'isoler des motifs particuliers au sein d'une chaîne de caractères. Elles existent sous plusieurs formes, et ont commencé à être développées et utilisées sur des systèmes UNIX® avec des logiciels comme « **grep** ».

INTEX utilise principalement les expressions rationnelles pour rechercher un ou des motifs dans un texte : c'est la fonctionnalité « Locate Pattern » ; mais elles sont un peu différentes des expressions rationnelles traditionnelles :

- Le symbole « | » (ou) est remplacé par « + ». Ainsi, l'expression « `jamais+toujours` » localise toutes les occurrences du mot « jamais » ou du mot « toujours ».

- Les expressions rationnelles traditionnelles fonctionnent par caractère tandis que celles d'INTEX fonctionnent par « lexème[4] ». Ainsi l'expression « `(c+d+l)e` » (équivalente à l'expression rationnelle traditionnelle « `[cdl]e` » pour retrouver les mots « ce », « de » ou « le ») ne permet pas de localiser ces mots dans INTEX. Il faut écrire « `ce+de+le` ».
- Un mot écrit en minuscules reconnaît toutes ses variantes, avec minuscules ou majuscules. En revanche, un mot qui contient au moins une lettre majuscule ne reconnaît pas ses variantes écrites en minuscules.
- Les symboles formels sont écrits entre angles. Ils font référence aux formes des unités linguistiques suivantes :

Symbole formel	Signification
<MOT>	Séquence de lettres[5]
<MIN>	Séquence de lettres minuscules[6]
<MAJ>	Séquence de lettres majuscules[7]
<PRE>	Séquence d'une lettre majuscule suivie de lettres minuscules
<NB>	Séquence de chiffres arabes
<PNC>	Caractère séparateur[8]
<^>	Début d'unité de traitement linguistique[9]
<$>	Fin d'unité de traitement linguistique[10]
<L>[11]	Lettre
<U>[12]	Lettre majuscule
<W>[13]	Lettre minuscule
<E>	Symbole vide

Tableau 1.1 Symboles formels d'INTEX

[4] Voir Lexique.
[5] Les lettres sont les caractères recensés dans le fichier Alphabet de la langue courante.
[6] *Idem*
[7] *Idem*
[8] Les séparateurs sont tous les caractères qui ne sont ni lettres, ni chiffres, ni blancs.
[9] En morphologie, l'unité de traitement linguistique sera le mot. En syntaxe, elle sera la phrase si le texte a été segmenté, le paragraphe sinon.
[10] *Idem*
[11] Ce symbole n'est utilisé qu'en morphologie.
[12] *Idem*
[13] *Idem*

- Le blanc ainsi que d'autres caractères d'espacement (tabulation, changement de ligne) sont facultatifs. En général, il n'y a pas lieu de rechercher des blancs.
- Le caractère « # » est utilisé pour interdire l'apparition du blanc.
- Les caractères de protection : le caractère « \ » est utilisé pour protéger un seul caractère. Si l'on veut protéger une séquence de caractères, il faut la mettre entre guillemets « " " ».
- Le symbole vide (epsilon) représenté par `<E>`, est utilisé en général pour noter un élément facultatif ou élidé.
- L'opérateur de Kleene « * » est utilisé pour indiquer un nombre quelconque d'occurrences (y compris zéro).

1.1.2 Automates et Transducteurs

Les automates ou plutôt les automates à nombre fini d'états sont une autre forme de représentation équivalente aux expressions rationnelles mais avec plus de lisibilité, surtout lorsque les séquences de mots à décrire deviennent plus complexes. Or, la représentation des automates d'INTEX (les graphes d'INTEX) est un peu différente des automates traditionnels car les transitions des automates traditionnels sont représentées chez INTEX dans des boîtes. De plus, les nœuds ne sont pas explicitement représentés chez INTEX. Cependant, les deux formes sont totalement équivalentes. La figure suivante montre les deux représentations.

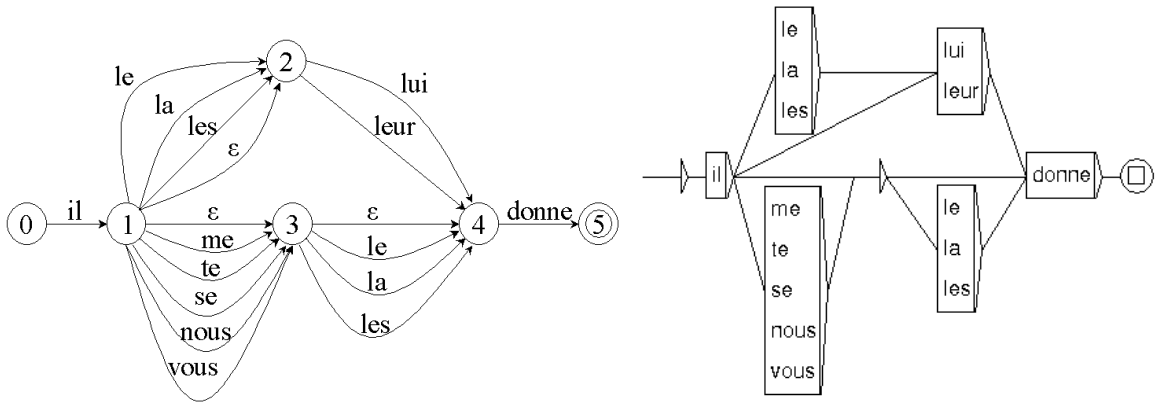

Figure 1.1 Automate traditionnel (à gauche) et graphe d'INTEX (à droite)

Un automate peut faire appel à un autre automate indiqué par une boîte grise. On appelle ce type d'automate « réseau de transitions récursif » ou en anglais « Recursive Transition Network » (RTN).

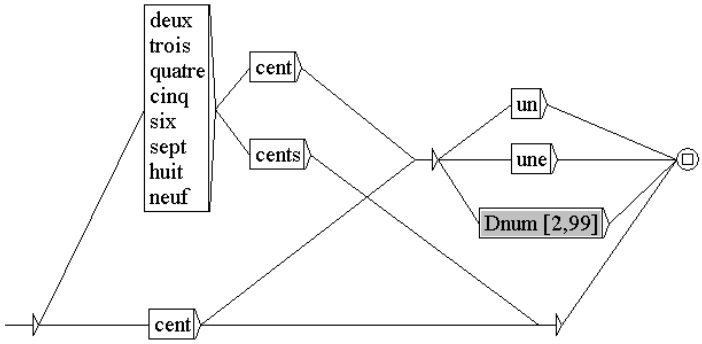

Figure 1.2 Réseau de transitions récursif

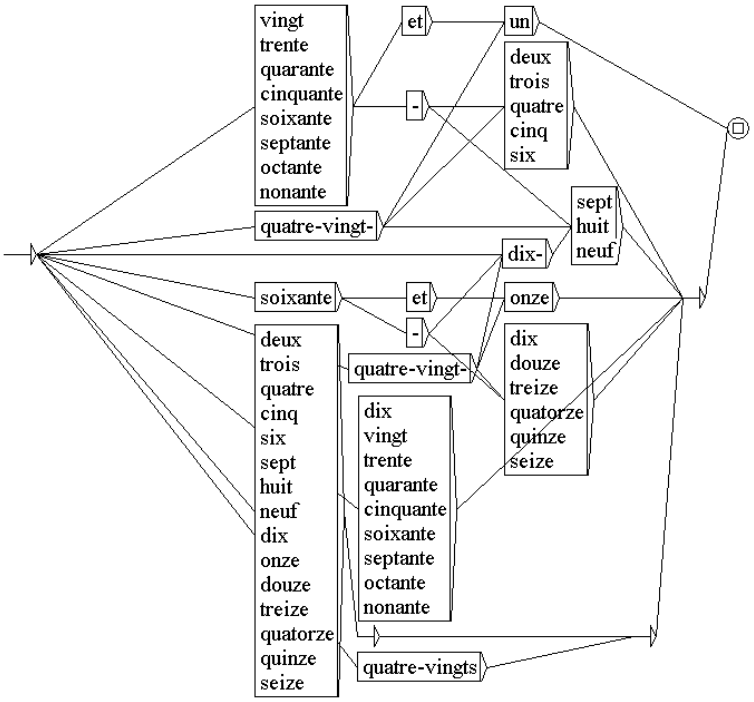

Figure 1.3 Sous-graphe « Dnum [2,99] »

Les transducteurs sont un type d'automates qui produisent des résultats en sortie. Ils seront utilisés dans deux modes : le mode « remplacement » (Replace), les séquences reconnues par le transducteur seront remplacées par les séquences produites ; le mode « insertion » (Merge), les séquences produites par le transducteur seront insérées dans le texte.

Il y a 5 règles à retenir lorsqu'on dessine les graphes d'INTEX :

- Les graphes sont toujours appliqués en avançant.
- Les séquences les plus à gauche sont prioritaires.
- Les séquences les plus longues sont prioritaires.
- Les graphes ne peuvent pas contenir des ordres contradictoires.
- Un transducteur ne peut pas reconnaître le mot vide.

1.1.3 Transducteurs avec variables (Enhanced Transducers)

Ce sont des transducteurs finis qui utilisent des variables internes pour ranger des parties de séquences reconnues. Le principe de fonctionnement de ces transducteurs est équivalent à la commande UNIX « **sed** ».

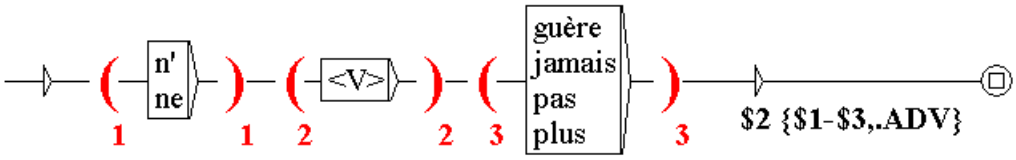

Figure 1.4 Transducteur avec variables

Le graphe ci-dessus permet de mémoriser des éléments entre parenthèses et de les associer à des variables numérotées, puis de les appeler en sortie avec le signe « $ ».

1.2 Éléments essentiels d'INTEX

Avant de pouvoir utiliser INTEX, il faut créer les 3 composants de base suivants ; tous les trois dépendent de la langue des textes :

1.2.1 Fichier Alphabet

C'est un fichier texte de type Windows ANSI (ASCII étendu à 8 bits), dans lequel chaque lettre est recensée et décrite. Ce fichier est lu par la quasi-totalité des programmes de façon transparente. Il est défini dans le dossier de la langue courante. Chaque langue doit posséder son propre fichier Alphabet qui se divise, en principe, en 2 parties : en-tête et lettres.

1.2.1.1 En-tête

Le fichier Alphabet commence par trois lignes facultatives : la première définit le nom et la taille de la police de caractères utilisée pour afficher les textes (en général, une police à largeur proportionnelle) ; la deuxième définit ceux pour afficher les concordances et dictionnaires (en général, une police à taille fixe). Si ces deux lignes ne sont pas présentes, INTEX utilisera la police par défaut de Windows. Si une seule ligne est présente, INTEX utilise la même police pour afficher les textes, concordances et dictionnaires. Ces deux lignes doivent impérativement commencer par le caractère spécial « # », par exemple :

```
#"Times New Roman" 12
#"Courier" 12
```

La troisième ligne définit le caractère à cacher lorsqu'on désactive le « Display Tags » dans les concordances ou dans le texte découpé (.snt). C'est très utile pour certaines langues asiatiques qui n'ont pas de séparateur de mots parce que cela permet de cacher le séparateur artificiel que l'on ajoute dans le texte (*cf. §1.4.3.2*). Ainsi, le texte peut retrouver sa forme originale (*cf. §4.1.1.3.2*). Cette ligne doit impérativement commencer par « `#ASIAN` », suivi d'un blanc et du caractère à cacher. L'exemple suivant permet de cacher le caractère « _ » :

```
#ASIAN _
```

Il n'est pas possible, pour l'instant, de cacher plus d'un caractère.

1.2.1.2 Lettres

On définit ensuite les caractères qu'on veut considérer comme « lettres ». Chaque lettre est décrite par 2 à 3 colonnes. Si la lettre n'a pas d'accent, on écrit la lettre majuscule suivie de la même lettre en minuscule, par exemple :

```
Aa
Bb
⋮
Zz
```

Si la lettre est accentuée, on écrit la lettre en majuscule sans accent, puis la lettre majuscule accentuée, puis la lettre minuscule accentuée, par exemple :

```
Aa
AÀà
AÂâ
AÄä
Bb
⋮
Zz
```

Si les lettres majuscules sont toujours accentuées dans les textes traités, alors il suffit d'entrer chaque lettre accentuée avec sa majuscule accentuée, par exemple :

```
Aa
Àà
Ââ
Ää
Bb
⋮
Zz
```

Si les lettres majuscules ne sont jamais accentuées dans les textes traités, il suffit d'entrer chaque lettre accentuée ou non avec sa majuscule non accentuée, par exemple :

```
Aa
Aà
Aâ
Aä
Bb
⋮
Zz
```

L'ordre dans lequel les lettres sont recensées dans le fichier Alphabet est important car il est utilisé par le programme de tri d'INTEX. Il est à noter que l'ordre vertical (du haut en bas) est prioritaire par rapport à l'ordre horizontal (de gauche à droite).

1.2.1.3 Séparateurs

Les séparateurs sont les caractères utilisés pour délimiter les séquences de lettres. Tous les caractères qui ne sont pas écrits dans le fichier Alphabet sont considérés comme « séparateurs ». Ils ne peuvent donc pas apparaître dans un mot simple ni commencer un mot composé.

1.2.1.4 Caractères réservés

Certains caractères ont un usage particulier pour INTEX, il ne faut jamais les inscrire dans le fichier Alphabet. Ces caractères sont :

```
!  "  #  $  (  )  *  +  /  :  <  >  \  ^  {  }
```

Comme ces caractères peuvent apparaître dans les textes (en tant que séparateurs), il est possible de les localiser par les expressions rationnelles ou par les graphes d'INTEX en utilisant les caractères de protection (*cf. §1.1.1*).

1.2.2 Modules de normalisation d'un texte

La normalisation consiste à transformer un fichier texte du format Windows ANSI (.txt) vers le format INTEX (.snt). Il s'agit d'une pré-analyse linguistique dont le but est de segmenter le texte en unités textuelles de traitement (en général, les phrases) et de normaliser l'orthographe de certaines formes sans utiliser le module lexical. La normalisation consiste en trois phases :

1.2.2.1 Segmentation du texte en phrases

C'est une phase obligatoire si l'on veut faire plus tard l'analyse syntaxique (*cf. §1.3.4*). Il s'agit de créer un transducteur pour découper le texte en unités plus petites. Le transducteur, appliqué en mode insertion, introduit la marque de délimitation de phrases « {S} » dans le texte. Le résultat obtenu est un fichier de texte découpé (.snt).

La figure suivante montre le graphe « Sentence.grf » qui sert à segmenter les phrases en français avec une précision de 99%. Par exemple, si l'on trouve un point d'interrogation suivi d'un mot qui commence par une majuscule, la marque de délimitation de phrases sera insérée entre les deux.

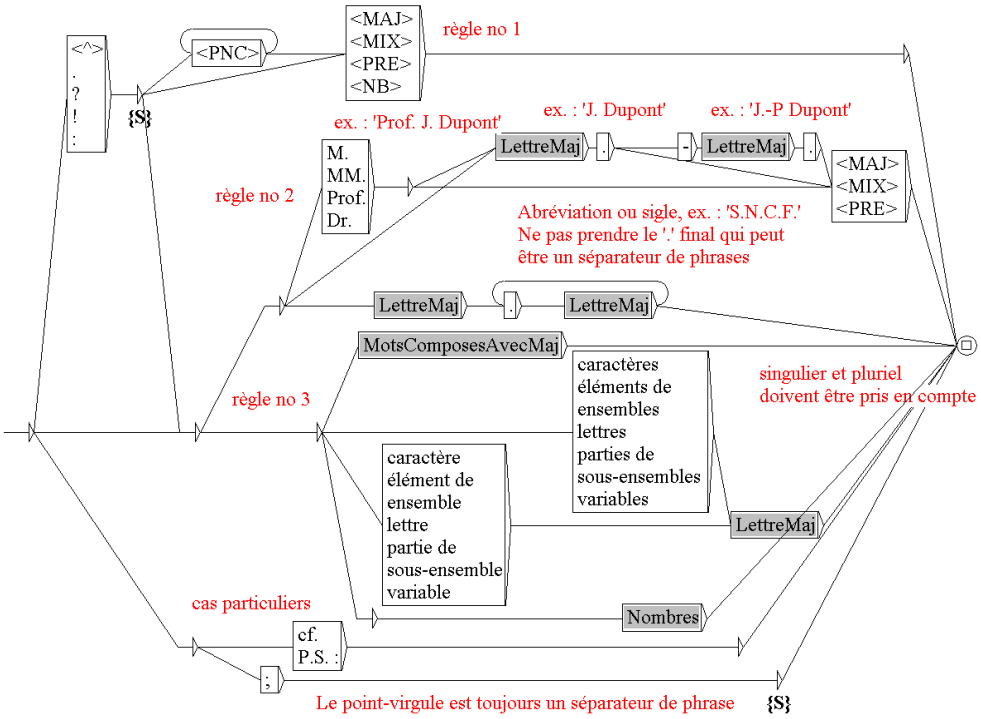

Figure 1.5 Graphe « Sentence » du français

Chaque langue a ses propres caractéristiques et nécessite l'utilisation d'un transducteur approprié.

1.2.2.2 Étiquetage des mots composés non-ambigus

La seconde phase est facultative et consiste à repérer et à étiqueter les mots composés non-ambigus ou qui contiennent des constituants non-autonomes. Par exemple, le mot « aujourd'hui » est un mot composé pour INTEX parce qu'il contient le séparateur « ' ». Par contre, les mots « aujourd » et « hui »[14] n'apparaissent dans aucun autre contexte, ce sont des

[14] Si l'on excepte une forme du verbe rarissime « huir ».

formes non-autonomes. Pour éviter que le système ne traite ces formes comme des lexèmes, nous pouvons remplacer le mot « aujourd'hui » dans le texte par l'entrée lexicale {aujourd'hui,.ADV}. Cette opération se fera par une simple consultation du dictionnaire de formes non-ambiguës. Le tableau suivant montre le dictionnaire de mots composés non-ambigus créé pour l'anglais.

a priori,.ADV
as soon as possible,.ADV
CD-ROM,.N+Conc:s
fast-food,.N
good-bye,.N:s
high-fat,.A
low-fat,.A
neo-nazis,neo-nazi.N+Hum:p
neo-nazi,.N+Hum:s
rendez-vous,.N:s:p
U.S.,United States of America.N+HumColl+Loc:p
U.S.A.,United States of America.N+HumColl+Loc:p
United States,United States of America.N+HumColl+Loc:p
United States of America,.N+HumColl+Loc:p

Tableau 1.2 Dictionnaire de mots composés non-ambigus de l'anglais

Cette méthode présente quelques avantages :

- Lorsqu'on supprime les formes non-autonomes, on est moins gêné par le bruit.
- Le système a moins de mots à consulter dans les dictionnaires et il est plus rapide.
- Il est inutile de coder les mots non-autonomes dans les dictionnaires de mots simples.

Néanmoins, elle présente aussi des désavantages :

- On ne peut plus localiser les formes comme « aujourd », ni « hui ».
- Lorsqu'on veut effectuer une recherche sur « aujourd'hui », il faut désormais taper {aujourd'hui} ou <aujourd'hui>, sinon on ne le retrouve jamais.
- Les entrées lexicales des mots composés non-ambigus apparaissent dans un dictionnaire séparé.

Donc, c'est à chaque utilisateur de décider s'il veut appliquer cette phase ou non. Il est à noter qu'il n'est pas interdit d'appliquer cette méthode aux mots simples non-ambigus.

1.2.2.3 Réécriture de formes déviantes

La troisième phase consiste à repérer et à réécrire certaines séquences déviantes non-ambiguës en appliquant des transducteurs en mode remplacement pour réécrire des séquences de formes déviantes sous une forme plus normalisée. Par exemple, en français, le mot « au » est une forme déviante des mots « à le ». Plutôt que d'alourdir le dictionnaire ou la grammaire pour tenir compte de ce mot, il est bien plus simple de décontracter « au » en « `{à,.PREP}` `{le,.DET:ms}` ». Il faut pourtant faire attention de n'effectuer ce type de substitutions que pour des séquences non-ambiguës.

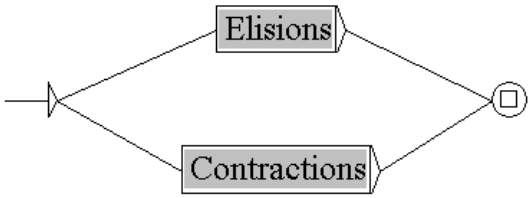

Figure 1.6 Graphe « Replace » du français

Le graphe « Replace.grf » sert à réécrire des formes déviantes en français. C'est, en effet, un RTN qui fait appel à deux sous-graphes : « Elisions.grf » et « Contractions.grf ».

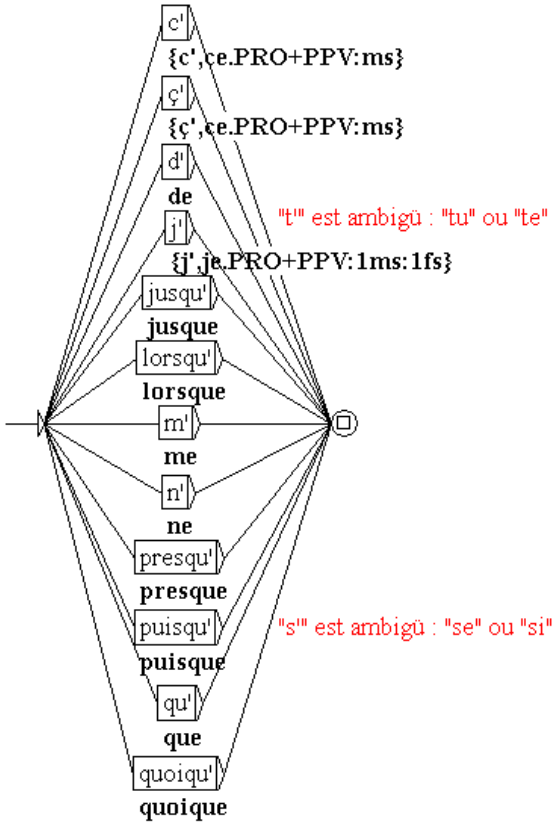

Figure 1.7 Sous-graphe « Elisions » du français

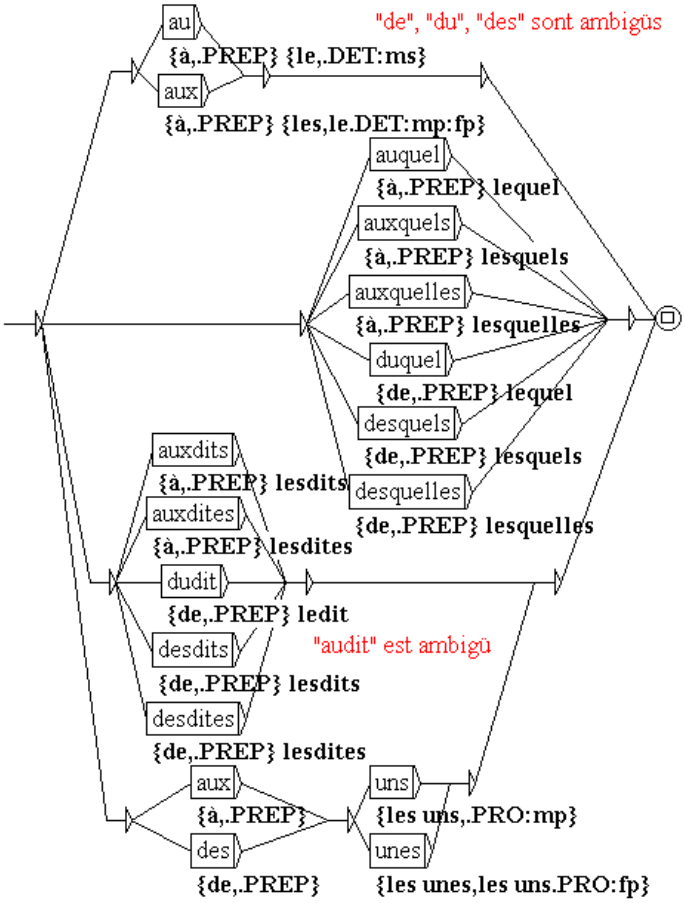

Figure 1.8 Sous-graphe « Contractions » du français

Les utilisateurs sont libres de modifier ces graphes. Ils peuvent aussi négliger cette troisième phase si cela ne leur convient pas.

La différence entre la seconde et la troisième phase est que la seconde phase nous permet de remplacer une séquence de plusieurs formes simples par une entrée lexicale, tandis que la troisième phase nous permet de remplacer une séquence d'une ou plusieurs formes simples par une séquence d'une ou plusieurs formes simples, ou entrées lexicales. La troisième phase est donc plus puissante mais certainement moins rapide.

1.2.3 Ressources lexicales

Ce sont les éléments les plus importants pour effectuer une analyse lexicale. Elles recensent et décrivent le vocabulaire courant de la langue, du point de vue morphologique, orthographique, syntaxique et parfois sémantique en utilisant le format développé au LADL.

Nous pouvons créer des ressources lexicales sous deux formes :

- Dictionnaires électroniques : des fichiers texte au format Windows ANSI. Ils ont une extension « .dic » et sont éditables par un programme de traitement de texte.

- Transducteurs lexicaux : des fichiers (avec une extension « .fst ») obtenus soit à partir d'un graphe d'INTEX, soit à partir de tables de lexique-grammaire[15].

Le système de dictionnaires du LADL (DELA) est fondé sur la définition formelle[16] suivante du mot :

Un mot simple est une séquence de lettres comprises entre deux séparateurs consécutifs.

Un mot composé est une séquence qui inclut au moins deux mots simples et au moins un séparateur.

Les mots simples sont recensés dans le Dictionnaire Électronique du LADL pour les mots Simples (DELAS) ; les mots composés sont recensés dans le Dictionnaire Électronique du LADL pour les mots Composés (DELAC). La séparation entre mots simples et mots composés est purement orthographique. Par exemple, « deltaplane » est un mot simple représenté dans le DELAS, tandis que sa variante orthographique « delta-plane » est un mot composé représenté dans le DELAC.

1.2.3.1 DELAS (DELA pour les mots simples)

C'est une base de données orthographiques et morphologiques qui recense et décrit tous les mots simples de la langue, c'est-à-dire, les mots qui ne comportent aucun séparateur (en particulier pas de trait d'union, ni apostrophe, ni espace blanc).

Les entrées du DELAS sont les mots simples sous leur forme canonique (en français : l'infinitif des verbes, le masculin singulier pour les noms et adjectifs). Ils sont suivis d'une virgule, d'un nom de catégorie en majuscules, puis éventuellement d'un numéro de classe flexionnelle, d'une ou plusieurs informations syntactico-sémantiques introduites par le caractère « + » ou d'un commentaire introduit par le caractère « / ». Par exemple :

```
aider,V3+t
andalou,A33/Andalousie
balle,N21+Conc
de,PREP
```

Le Tableau 1.3 présente les codes de catégories utilisés dans le DELAS du français et le Tableau 1.4 présente leurs codes syntactico-sémantiques.

[15] CONSTANT 2002 ; PAUMIER 2001 ; ROCHE 1993, 1999
[16] SILBERZTEIN 1993, p. 42

Code	Signification
A	Adjectif
ADV	Adverbe
CONJC	Conjonction de coordination
CONJS	Conjonction de subordination
DET	Déterminant
INTJ	Interjection
N	Nom (substantif)
PREP	Préposition
PRO	Pronom
V	Verbe
X	Constituant non-autonome de mot composé

Tableau 1.3 Catégories du DELAS français

Code	Signification
Abst	(Nom) Abstrait
Anl	(Nom) Animal
AnlColl	(Nom) Animal Collectif
Conc	(Nom) Concret
ConcColl	(Nom) Concret Collectif
Hum	(Nom) Humain
HumColl	(Nom) Humain Collectif
t	(Verbe) transitif
i	(Verbe) intransitif
en	(Verbe) particule PPV obligatoire
se	(Verbe) pronominal
ne	(Verbe) négation obligatoire
aux	(Verbe) auxiliaire
z1, z2, z3	(Vocabulaire) de base, standard, technique
1, 2, 3…	(Table) 1, 2, 3…

Tableau 1.4 Codes syntactico-sémantiques du français

Nous pouvons également générer un DELAS à partir d'un graphe d'INTEX. M. SILBERZTEIN[17] a établi le graphe de la famille dérivationnelle des mots construits sur « france » selon l'exemple de M. GROSS[18].

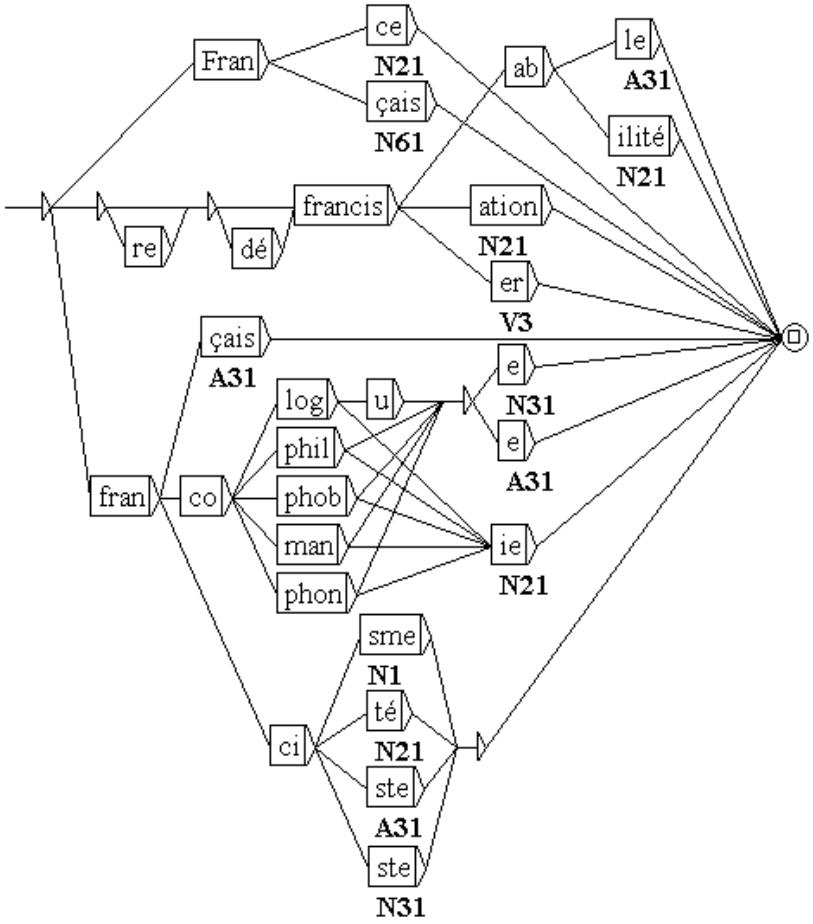

Figure 1.9 Graphe du DELAS sur « france »

Après avoir appliqué l'outil d'INTEX « Generate Language » au graphe ci-dessus en choisissant l'option « Simple word lexical entries » pour que le graphe travaille en mode « morphologie » (il parcourt les chemins et génère des mots en soudant tous les caractères de chaque élément), nous obtenons une liste équivalente à celle de la méthode manuelle précédemment expliquée :

```
francisation,N21
franciser,V3
français,A31
France,N21
Français,N61
```

[17] SILBERZTEIN 1993, p. 57
[18] GROSS 1989b

```
francité,N21
francisme,N1
franciste,N31
franciste,A31
francisable,A31
francisabilité,N21
refrancisation,N21
refranciser,V3
défrancisation,N21
défranciser,V3
défrancisable,A31
défrancisabilité,N21
refrancisable,A31
refrancisabilité,N21
redéfrancisation,N21
redéfranciser,V3
francologie,N21
francomane,N31
francomane,A31
francomanie,N21
francophile,N31
francophile,A31
francophilie,N21
francophobe,N31
francophobe,A31
francophobie,N21
francophone,N31
francophone,A31
francophonie,N21
francologue,N31
francologue,A31
redéfrancisable,A31
redéfrancisabilité,N21
```

Étant donné que le DELAS ne contient que la forme canonique des mots alors que les formes fléchies apparaissent dans le texte, il ne sert pas à faire l'analyse lexicale mais à créer le DELAF.

1.2.3.2 DELAF (DELAS des formes fléchies)

Les mots simples sont en effet fléchis dans les textes. Par exemple, en français, les noms et adjectifs peuvent apparaître au féminin et/ou au pluriel, les verbes sont généralement conjugués. Pour pouvoir traiter des textes écrits dans une langue donnée, INTEX a besoin d'un dictionnaire qui recense et décrit toutes les formes fléchies de la langue.

Le dictionnaire DELAF contient la totalité des formes fléchies issues de la base des mots du DELAS. Cet ensemble de formes peut être engendré de 3 façons :

1. Manuellement avec l'éditeur de texte, en respectant le format suivant :

Chaque ligne du DELAF doit contenir une forme fléchie, suivie d'une virgule, d'une forme canonique (éventuellement vide si elle est identique à la forme fléchie), d'un point, d'un nom de catégorie en majuscules, puis éventuellement d'une ou plusieurs informations syntactico-sémantiques introduites par le caractère « + », d'une ou plusieurs séries d'informations flexionnelles introduites par le caractère « : » ou d'un commentaire introduit par le caractère « / ». Par exemple :

```
eskuara,.N+z3:ms/langue basque
estimable,.A+z1:ms:fs
estimables,estimable.A+z1:mp:fp
```

Ensuite, sauvegarder la liste avec l'extension « .dic » dans le répertoire « Delaf » de la langue courante. Elle est utilisable directement.

Le Tableau 1.5 présente les codes flexionnels du DELAF français.

Code	Signification
m	Masculin
f	Féminin
s	Singulier
p	Pluriel
1, 2, 3	$1^{ère}$, $2^{ème}$, $3^{ème}$ personne
P	Présent de l'indicatif
I	Imparfait de l'indicatif
S	Présent du subjonctif
T	Imparfait du subjonctif
Y	Présent de l'impératif
C	Présent du conditionnel
J	Passé simple
W	Infinitif
G	Participe présent
K	Participe passé
F	Futur

Tableau 1.5 Codes flexionnels du DELAF français

Lorsqu'un dictionnaire devient trop grand, il est plus avantageux de le compresser en transducteur avec l'outil « Compress into FST ». Nous obtenons par la suite 2 fichiers : un fichier « .bin » qui contient le transducteur proprement dit, et un fichier « .inf » qui contient le vocabulaire du transducteur, autrement dit, l'ensemble des informations lexicales que nous trouvons dans le dictionnaire. Le fichier « .inf » est éditable, ce qui nous permet de modifier les codes à volonté.

2. Nous pouvons également créer un DELAF à partir d'un graphe d'INTEX :

   Le graphe suivant décrit les variantes orthographiques du mot « tsar ».

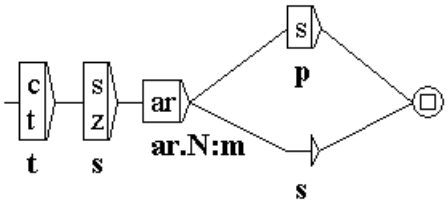

Figure 1.10 Graphe du DELAF sur « tsar »

Nous pouvons utiliser le graphe ci-dessus de deux façons : soit le compiler en transducteur et sauvegarder le fichier résultat « .fst » dans le répertoire « Delaf » ; soit générer une liste des mots fléchis par « Generate Language » avec l'option « Simple word lexical entries », ce qui nous donne la liste suivante :

```
czar,tsar.N:ms
czars,tsar.N:mp
csar,tsar.N:ms
csars,tsar.N:mp
tzar,tsar.N:ms
tzars,tsar.N:mp
tsar,tsar.N:ms
tsars,tsar.N:mp
```

Ensuite, sauvegarder cette liste dans le répertoire « Delaf » avec l'extension « .dic ».

3. Flexion automatique d'un dictionnaire DELAF à partir d'un DELAS :

Grâce aux informations de classes flexionnelles attachées aux codes de catégories dans chaque entrée du DELAS, nous savons comment générer l'ensemble des formes fléchies correspondant à un mot donné. Chaque code fait référence à un transducteur du même nom

qui contient une description de la morphologie flexionnelle de ce code. Tous les mots qui se fléchissent de la même façon sont associés au même transducteur. Par exemple :

```
cousin,N32+Hum
voisin,N32+Hum
```

Le code N32 fait appel au transducteur « N32.fst » représenté par le graphe suivant :

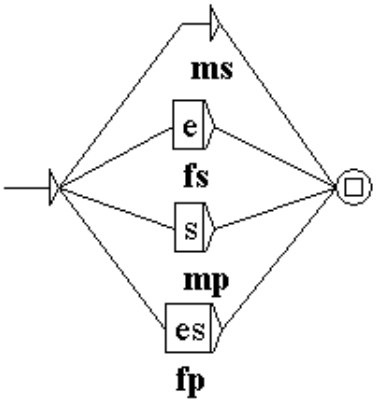

Figure 1.11 Transducteur de flexion « N32 »

Après avoir appliqué « Inflect Dictionary » en choisissant le type « DELAS » aux entrées ci-dessus, nous obtenons :

```
cousin,cousin.N32+Hum:ms
cousine,cousin.N32+Hum:fs
cousins,cousin.N32+Hum:mp
cousines,cousin.N32+Hum:fp
voisin,voisin.N32+Hum:ms
voisine,voisin.N32+Hum:fs
voisins,voisin.N32+Hum:mp
voisines,voisin.N32+Hum:fp
```

Parfois, nous avons besoin d'effacer une ou plusieurs des dernières lettres du lemme avant de pouvoir ajouter un suffixe. Ce qui arrive dans le cas du mot :

```
cheval,N4+Anl
```

Grâce à l'opérateur d'effacement « `L` » (Left) qui supprime un caractère vers la gauche, nous pouvons le faire comme le montre la figure suivante :

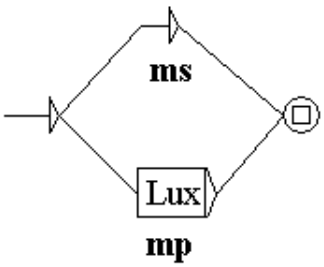

Figure 1.12 Transducteur de flexion « N4 »

Ce qui nous donne le résultat :

```
cheval,cheval.N4+Anl:ms
chevaux,cheval.N4+Anl:mp
```

Nous le sauvegardons ensuite dans le répertoire « Delaf » avec l'extension « .dic ».

1.2.3.3 DELAC (DELA pour les mots composés)

Le dictionnaire DELAC est analogue au dictionnaire DELAS. La différence est que les entrées lexicales du DELAC peuvent contenir des séparateurs.

Chaque ligne du DELAC doit contenir un texte (commençant impérativement par une lettre et pouvant éventuellement contenir des séparateurs autres que la virgule « , »), suivi d'une virgule, d'un nom de catégorie en majuscules, puis éventuellement d'une ou plusieurs informations syntactico-sémantiques introduites par le caractère « + » ou d'un commentaire introduit par le caractère « / ». Par exemple :

```
carte bancaire,N+Conc
carte-mère,N+Conc/informatique
```

Comme le DELAS, le DELAC ne sert pas à faire l'analyse lexicale. La fonction de flexion automatique du DELACF à partir du DELAC[19] n'étant pas encore intégrée dans INTEX, il est nécessaire de créer directement un DELACF.

1.2.3.4 DELACF (DELAC des formes fléchies)

Dans les textes, les mots composés, comme les mots simples, apparaissent fléchis. L'identification automatique des mots composés se fait grâce au dictionnaire DELACF qui relie les formes fléchies des mots composés avec leur forme canonique. Cette dernière constitue l'entrée lexicale du DELAC.

Le dictionnaire DELACF est analogue au dictionnaire DELAF. La différence est que les entrées lexicales et/ou les lemmes du DELACF peuvent contenir des séparateurs.

[19] CHROBOT 1998 ; SAVARY 2000, p. 63-89

Chaque ligne du DELACF doit contenir un texte (commençant impérativement par une lettre et pouvant éventuellement contenir des séparateurs autres que la virgule « , »), suivi d'une virgule, d'un texte (éventuellement vide s'il est identique au texte précédent ; ce texte peut éventuellement contenir des séparateurs autres que le point « . »), d'un point, d'un nom de catégorie en majuscules, puis éventuellement d'une ou plusieurs informations syntactico-sémantiques introduites par le caractère « + », d'une ou plusieurs séries d'informations flexionnelles introduites par le caractère « : » ou d'un commentaire introduit par le caractère « / ».

D'après cette définition, un mot simple peut aussi apparaître comme une forme fléchie ou comme une forme canonique du DELACF ; par exemple :

```
roman policier,polar.N+Conc:ms
U.S.A.,Etats-Unis d'Amérique.N+Géo+Pays:mp
USA,Etats-Unis d'Amérique.N+Géo+Pays:mp
```

Avec un éditeur de texte, nous pouvons créer un dictionnaire DELACF comme l'exemple ci-dessus, puis l'enregistrer dans le répertoire « Delacf » de la langue courante avec l'extension « .dic ». Il est utilisable directement.

Nous pouvons également créer un DELACF par un graphe d'INTEX comme la figure ci-dessous. Il suffit ensuite de le compiler en fichier « .fst » et le sauvegarder dans le répertoire « Delacf » de la langue courante.

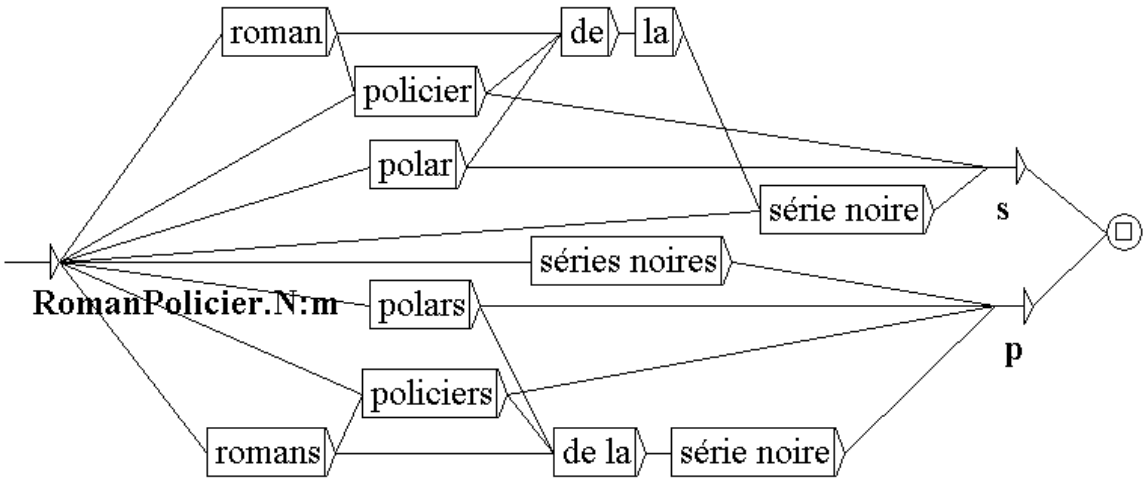

Figure 1.13 Graphe du DELACF sur « roman policier »

1.2.3.5 DELAE (DELA des expressions figées)

Nous appelons « expressions figées » des unités lexicales qui sont construites à partir de plus d'un mot simple (comme les mots composés), mais qui ne sont pas des séquences obligatoirement connexes. Par exemple, « *perdre…la raison* » dans la phrase :

Luc perd à nouveau la raison.

Le mot « à nouveau » a été inséré dans la phrase entre « perdre » et « la raison », ce qui empêche la reconnaissance de l'expression « perdre la raison » par le DELACF.

Pour remédier à ce problème, nous devons employer le transducteur avec variables qui accepte une insertion à l'intérieur de l'expression comme la figure suivante :

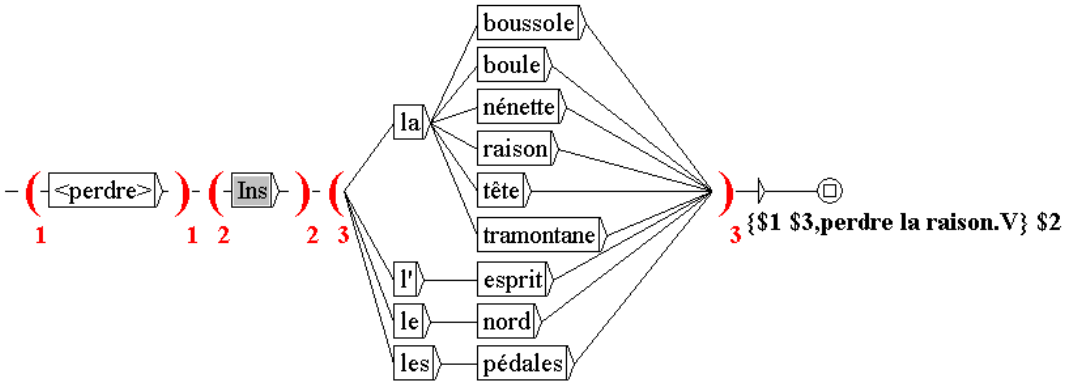

Figure 1.14 Graphe du DELAE sur « perdre…la raison »

Après avoir dessiné un graphe DELAE, nous devons le compiler en « .fst » et le sauvegarder dans le répertoire « Delae » de la langue courante avant de pouvoir l'utiliser.

Le DELAE n'existe que sous forme de transducteur. La forme « .dic » n'existe pas.

1.3 Fonctionnalités principales d'INTEX

On emploie habituellement INTEX avec l'un des objectifs suivants :

1.3.1 Analyse lexicale

Il s'agit d'appliquer des ressources lexicales (DELAF, DELACF et DELAE) à un texte pour y identifier et étiqueter les mots simples, les mots composés et les expressions figées par les informations lexicales correspondantes. L'opération s'effectue par union des transducteurs de chaque dictionnaire (le résultat est un transducteur), et projection de ce transducteur sur le texte. Cette méthode présente des avantages considérables, notamment en termes de rapidité d'exécution.

1.3.1.1 Jeu de priorités entre ressources lexicales

Nous avons la possibilité d'assigner un niveau de priorité à chaque ressource lexicale. Les ressources dont les noms se terminent par le signe « - » (e.g. : delaf1-.dic) ont la priorité maximale, par le signe « + » (e.g. : Npropre+.fst), la priorité minimale et par aucun signe (e.g. : delacf.dic), la priorité normale. Les ressources de priorité maximale seront appliquées en premier, suivies de celles de priorité normale et enfin celles de priorité minimale.

Les mots simples reconnus dans le texte par un DELAF de priorité supérieure seront ignorés lors de l'application d'autres DELAF de priorité inférieure. Cela permet de réduire le nombre d'occurrences reconnus par les dictionnaires en ignorant des termes peu fréquents.

Quant aux mots composés, si un DELACF est prioritaire par rapport aux DELAF, les mots composés reconnus ne seront plus décomposés par INTEX en séquences de formes simples. Par exemple, si la forme suivante est une entrée d'un DELACF prioritaire :

```
sous-marin nucléaire d'attaque,.N+Conc:ms
```

Alors, la séquence « sous-marin nucléaire d'attaque » dans le texte sera traitée comme un seul mot non-ambigu et les formes simples « sous », « marin », « nucléaire », « d » et « attaque » seront ignorées dans cet séquence.

Si plusieurs formes de mots composés de longueurs différentes sont reconnues à la même position par un DELACF prioritaire, alors seules les formes les plus longues seront prises en compte par le système. Par exemple, si les deux formes suivantes sont des entrées d'un DELACF prioritaire :

```
dans l'intimité,.ADV+PDET
dans l'intimité la plus stricte,.ADV+PCDC
```

Alors, la séquence « dans l'intimité la plus stricte » dans le texte sera traité comme un seul adverbe PCDC sans concurrence avec l'adverbe PDET suivi des trois mots simples « la », « plus » et « stricte ».

1.3.1.2 Vocabulaire du texte

L'application des ressources lexicales produit le « vocabulaire du texte », c'est-à-dire l'ensemble des unités linguistiques qui sont recensées dans les ressources lexicales sélectionnées et qui ont été trouvées dans le texte. Ce vocabulaire est représenté par trois des fenêtres de la figure suivante : DLF pour les mots simples, DLC pour les mots composés et DLE pour les expressions figées. La quatrième fenêtre (ERR) contient les mots inconnus.

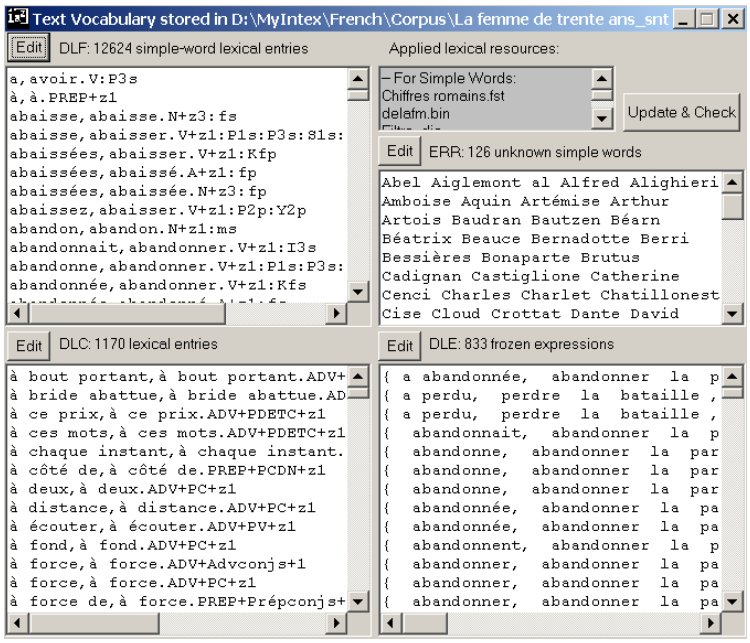

Figure 1.15 Vocabulaire du texte

Le vocabulaire du texte contient les ressources lexicales qui vont être utilisées par les modules grammaticaux d'INTEX : recherche de motifs, levée d'ambiguïtés et analyse syntaxique. Elles seront utilisées sous la forme des symboles lexicaux suivants :

Symbole lexical	Signification
<DIC>	Mot dans le vocabulaire du texte
<lemme>	Mot (dans le vocabulaire du texte) associé au lemme donné (e.g. : <être>, <cousin germain>)
<code[20]*>*	Mot (dans le vocabulaire du texte) associé au code donné (e.g. : <ADV>, <N+Hum>)
<lemme.code>	Mot (dans le vocabulaire du texte) associé au lemme et au code donnés (e.g. : <avoir.V:P3s>, <jeune fille.N+Hum:fp>)

Tableau 1.6 Symboles lexicaux disponibles après l'application des ressources lexicales

1.3.2 Recherche de motifs

C'est probablement la fonction d'INTEX la plus utilisée. Cette fonction nous permet de localiser les mots simples, les mots composés et les expressions figées dans le texte et de les représenter avec leurs contextes. Elle permet également de localiser des motifs définis par l'utilisateur soit avec des expressions rationnelles, soit avec des graphes.

[20] Les codes dans les Tableau 1.3, Tableau 1.4 et Tableau 1.5

1.3.2.1 Recherche de motifs par expressions rationnelles

Nous avons la possibilité de rechercher un motif en tapant directement le mot à localiser (e.g. : `toujours`) ou en utilisant les symboles formels du Tableau 1.1 (e.g. : `<PRE>`) sauf les symboles `<L>`, `<U>` et `<W>` qui ne sont actifs qu'en mode morphologie. Nous rappelons que les expressions rationnelles d'INTEX fonctionnent en mode lexème (*cf. §1.1.1*).

Nous pouvons également enchaîner plusieurs motifs, par exemple :

```
<PRE> est toujours <MOT><PNC>
```

Cette expression recherche des textes qui comprennent un mot commençant par une majuscule, suivi du mot « est », suivi du mot « toujours », suivi de n'importe quel mot et terminé par un séparateur ; par exemple : « *Luc est toujours là.* ».

Nous pouvons préciser l'ordre des priorités avec les parenthèses, par exemple :

```
est (jamais+toujours)
```

Ceci précise que nous voulons rechercher le mot « est » suivi du mot « jamais » ou du mot « toujours », ce qui est équivalent à :

```
(est jamais)+(est toujours)
```

Après l'application des ressources lexicales au texte, nous bénéficierons en plus des symboles lexicaux du Tableau 1.6, par exemple :

```
<N:ms+Hum> <être> (jamais+toujours) <DIC> <PNC>
```

Cette expression va retrouver des séquences qui commencent par un nom humain masculin singulier, suivi de n'importe quelle forme fléchie associée au lemme « être », suivie du mot « jamais » ou « toujours », suivi de n'importe quel mot dans le vocabulaire du texte, suivi d'un séparateur.

Afin d'exprimer la négation, nous pouvons préfixer les symboles lexicaux par le caractère « ! » ou utiliser le caractère spécial « - » au lieu du « + » avant les codes syntactico-sémantiques, par exemple :

```
<N-Hum> <!être> <!DIC>
```

Cela localise des séquences qui commencent par un nom non-humain, suivi de n'importe quelle forme fléchie qui n'est pas associée au lemme « être », suivi de n'importe quel mot hors du vocabulaire du texte.

1.3.2.2 Recherche de motifs par graphes

Tout ce que nous pouvons faire avec des expressions rationnelles, nous pouvons également le faire avec des graphes mais l'inverse n'est pas vrai parce que les graphes sont plus puissants : ils peuvent produire des résultats (transducteur), faire appel à d'autres graphes (RTN) et réarranger l'ordre de parties de séquences reconnues (transducteur avec variables). Par exemple :

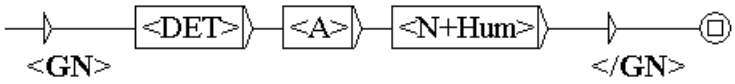

Figure 1.16 Transducteur d'un groupe nominal humain

La figure ci-dessus est un transducteur qui reconnaît un groupe nominal humain. Il permet également de modifier le texte : avec l'option « Merge with input text », les séquences produites par le transducteur sont insérées en certains points du texte (*voir Figure 1.17*) ; l'option « Replace recognized sequences » remplace les séquences reconnues par les séquences produites (*voir Figure 1.18*).

d de serpent, et remua dans le coeur de <GN>cette singulière jeune fille</GN> un monde de pensé
ns le silence comme un chant de oiseau. <GN>Le beau jeune homme</GN>, blond comme lui, le faisa
ire, comme si ce bonheur était le mien. <GN>Le beau jeune homme</GN> entendit sonner neuf heure
éanmoins elle le redoutait, semblable à <GN>le malheureux condamné à mort</GN> qui voudrait en
e sorte de anxiété pudique. - Ainsi, <GN>mon bon petit ange</GN>, reprit sa tante, le mariag
'onde s'écarta en mille jets bruns sous <GN>sa jolie tête blonde</GN>. j' entendis les cris aig
ait refléter le gai bonheur du paysage. <GN>Un beau jeune homme</GN> posait à terre le plus jol
nterie. L'actualité leur déplaît. Quand <GN>une vieille femme de chambre</GN> vint annoncer à l

Figure 1.17 Résultat de l'option « Merge with input text »

ns le silence comme un chant de oiseau. <GN></GN>, blond comme lui, le faisait danser dans ses
ire, comme si ce bonheur était le mien. <GN></GN> entendit sonner neuf heures. Après avoir tend
'onde s'écarta en mille jets bruns sous <GN></GN>. j' entendis les cris aigus du pauvre petit;
ait refléter le gai bonheur du paysage. <GN></GN> posait à terre le plus joli garçon que il fût
éanmoins elle le redoutait, semblable à <GN></GN> qui voudrait en avoir fini avec la vie, et qu
e sorte de anxiété pudique. - Ainsi, <GN></GN>, reprit sa tante, le mariage ne a été jusqu'à
d de serpent, et remua dans le coeur de <GN></GN> un monde de pensées encore endormi chez elle.
nterie. L'actualité leur déplaît. Quand <GN></GN> vint annoncer à la comtesse (car elle devait

Figure 1.18 Résultat de l'option « Replace recognized sequences »

1.3.2.3 Recherche morphologique

Même si la recherche de motifs fonctionne par défaut en mode lexème, nous pouvons toutefois obtenir qu'elle marche en mode morphologie (avec quelques limitations) par l'emploi de guillemets « " " », par exemple :

```
"re"<MOT>
```

Ceci recherche tous les mots qui commencent par « re ». Cette expression fonctionne parce que comme la forme « re » est entre guillemets, INTEX va rechercher exactement cette forme, tandis que sans guillemets, INTEX aurait attendu un délimiteur après cette forme pour la reconnaître, par exemple :

```
re<MOT>
```

est équivalent à :

```
"re"<$><MOT><$>
```

qui recherche la forme « re », suivie d'une fin d'unité (habituellement un blanc), suivie d'un mot, suivie d'une autre fin d'unité.

Cependant, il y a quelques restrictions :

- On ne peut mettre entre guillemets que les caractères, les symboles entre angles ne sont pas acceptés. Par exemple : « `"<A:fs>"ment` » n'est pas valide.
- Il faut que la séquence entre guillemets commence toujours le motif. Par exemple : « `"in"<MIN>` » est valide mais pas « `<PRE>"tion"` ».
- Les caractères entre guillemets seront considérés tels quels, c'est-à-dire que la minuscule ne reconnaît plus la majuscule. Par exemple : « `"re"fait` » ne reconnaît pas « Refait ».
- On peut enchaîner plusieurs séquences entre guillemets à condition qu'elles soient contiguës. Par exemple : « `"re"("con"+"deve")nu` » localise toutes les occurrences du mot « reconnu » ou du mot « redevenu ».
- Si l'on veut employer un symbole lexical du Tableau 1.6, il faut que le mot correspondant existe dans le texte, il ne suffit pas qu'il figure dans le dictionnaire de la langue. Par exemple : si l'on veut localiser toutes les formes fléchies du mot « reconnaître » en utilisant l'expression « `"re"<connaître>` », mais qu'aucune forme fléchie du mot « connaître » n'existe dans le texte, dans ce cas, on n'obtient aucun résultat même si les formes fléchies du mot « reconnaître » existent dans le texte et que « connaître » figure dans le dictionnaire de la langue.

1.3.3 Levée d'ambiguïtés lexicales

Une forme (simple ou composée) est considérée comme ambiguë si elle est associée à plusieurs informations lexicales différentes. Après avoir appliqué les ressources lexicales à un texte, de nombreuses formes simples et composées peuvent être associées à plusieurs informations linguistiques différentes. Par exemple : la forme « place » est associée au nom féminin singulier et à la forme conjuguée du verbe « placer » ; la séquence « carte bleue » correspond soit au mot simple « carte » suivi du mot simple « bleue » (une carte de couleur bleue), soit au mot composé du DELACF (une carte bancaire).

En principe, il n'est pas possible de lever toutes les ambiguïtés lexicales, INTEX fournit néanmoins quelques mécanismes pour en lever certaines :

1.3.3.1 Levée d'ambiguïtés par dictionnaire

Certains mots (simples ou composés) ne sont jamais ambigus dans un texte donné. Mais si nous n'avons pas appliqué la normalisation (*cf. §1.2.2.2*), ni le jeu des priorités entre ressources lexicales (*cf. §1.3.1.1*) pour une raison quelconque, nous avons encore une possibilité de lever des ambiguïtés dans cette phase par le dictionnaire « Disamb.dic ». Par exemple, le mot « par » peut être soit une préposition, soit un nom (terme de golf). Si nous sommes sûrs que le texte ne parle pas de golf, nous pouvons négliger la deuxième définition en ne gardant que la première dans « Disamb.dic ».

1.3.3.2 Levée d'ambiguïtés par grammaires locales

Une grammaire locale de levée d'ambiguïtés est un transducteur qui reconnaît des séquences dans certains contextes particuliers et associe des contraintes lexicales aux séquences reconnues pour éliminer des hypothèses invalides. Par exemple, la séquence « C'est » est ambiguë car après la consultation des ressources lexicales, nous obtenons dans le vocabulaire du texte :

```
C,C.DET+CR=100
c,c.N+z1:ms
c,ce.PRO+z1:ms
est,est.A+z1:ms:fs:mp:fp
est,est.N+z1:ms
est,être.V+z1:P3s
```

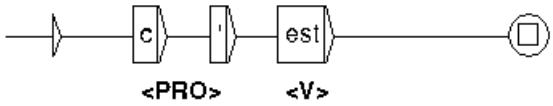

Figure 1.19 Grammaire locale de levée d'ambiguïtés

Grâce au transducteur de la figure ci-dessus, la séquence reconnue est forcée par les contraintes lexicales : <PRO> pour « c » et <V> pour « est ». Seules les entrées lexicales compatibles suivantes sont alors prises en compte. La séquence est donc totalement désambiguïsée.

```
c,ce.PRO+z1:ms
est,être.V+z1:P3s
```

1.3.3.3 Étiquetage linéaire d'un texte

L'étiquetage linéaire d'un texte consiste à remplacer dans le texte toutes les formes non-ambiguës par les étiquettes correspondantes, écrites entre accolades.

La forme non-ambiguë est soit l'unique étiquette trouvée dans les dictionnaires, soit la forme recensée dans le dictionnaire « Disamb.dic », soit la séquence désambiguïsée par la grammaire locale.

On a aussi la possibilité d'étiqueter les mots composés ambigus du type « carte bleue » en donnant la priorité à l'étiquette de mot composé. Cela donne souvent des résultats satisfaisants mais pas toujours, par exemple : « *Le service d'accueil s'est peu à peu transformé en permanence.* », la séquence « *en permanence* » sera étiquetée à tort comme un mot composé.

Après l'étiquetage, le texte devient un flot linéaire de lexèmes qui sont soit des formes, soit des entrées lexicales.

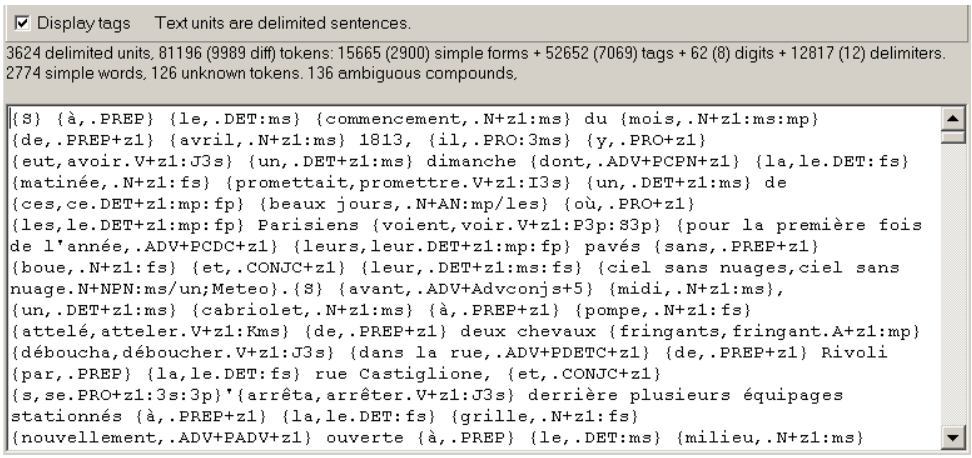

Figure 1.20 Texte après l'étiquetage linéaire

1.3.4 Analyse syntaxique

Il s'agit de représenter toutes les structures possibles des phrases sous la forme d'un transducteur du texte, dans lequel chaque unité linguistique est représentée par une étiquette et chaque chemin du nœud initial vers le nœud terminal correspond à une lecture possible de chaque phrase. Par exemple, la phrase « *Il couvre une pomme de terre cuite.* » a la structure suivante :

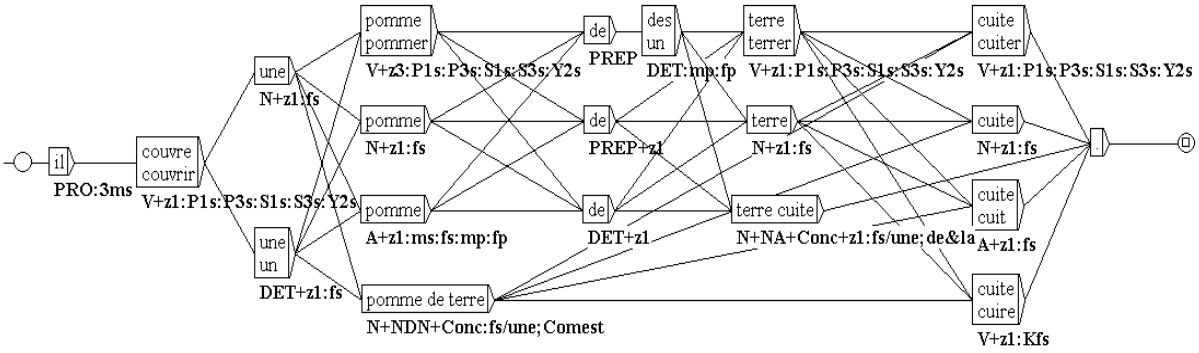

Figure 1.21 Structure de la phrase « Il couvre une pomme de terre cuite. »

La construction du transducteur du texte prend en compte le vocabulaire du texte, c'est-à-dire qu'il représente les mots simples aussi bien que les mots composés et les expressions figées. Le dictionnaire « Disamb.dic » est également disponible comme option qui permet de réduire le nombre de chemins inutiles.

Une nouvelle option intéressante dans le menu de la construction du transducteur du texte est « Remove .Xxx lexical items » qui permet de ne pas présenter, dans le résultat, les mots dont les codes grammaticaux commencent par X (les constituants non-autonomes et les mots inconnus, par exemple). Cette option peut réduire énormément la complexité du graphe, notamment dans la langue thaï que nous étudierons plus tard (*cf. §6.4.2*).

1.4 Ajout de nouvelles langues dans INTEX

Depuis la version 4, INTEX tourne sous le système d'exploitation Windows®. Il a été testé sous Windows 95, 98, ME, NT et 2000.

Pour ajouter une nouvelle langue dans INTEX, il faut satisfaire à la fois aux critères de Windows et aux critères d'INTEX.

1.4.1 Critères de Windows

Il faut installer :

- Une police du type ASCII étendu (Windows ANSI) de la langue désirée dans le répertoire « Fonts » du système, afin de pouvoir afficher correctement le résultat à l'écran et à l'imprimante dans cette langue.
- Une méthode d'entrée du clavier de la langue désirée, afin de pouvoir taper des caractères dans cette langue. Windows intègre préalablement beaucoup de langues dans son système, il suffit de les activer dans le paramétrage du système.

Il existe aussi une version localisée de Windows dans plusieurs langues, dans laquelle les polices et le clavier sont installés et configurés convenablement pour chaque langue locale.

1.4.2 Critères d'INTEX

Il faut créer :

- Un dossier de la langue désirée ; INTEX a besoin d'un répertoire privé pour chaque langue dans lequel il y a des sous-répertoires pour les corpus, les différents types de dictionnaires, les graphes, etc. La méthode la plus simple est de dupliquer un dossier déjà existant (français ou anglais, par exemple), de le renommer dans la langue désirée et de vider les contenus du répertoire et de ses sous-répertoires.
- Un fichier Alphabet ; le fichier qui recense et décrit les lettres de la langue comme expliqué au §1.2.1. Ce fichier doit être dans le dossier de la langue désirée.

Pour pouvoir bien définir le fichier Alphabet, il faut d'abord comprendre les codages de caractères. En réalité, il existe plusieurs types de codages de caractères dans le monde informatique. Les codages de l'ancienne génération (ASCII, EBCDIC, par exemple) utilisent 7 bits. En conséquence, ils ne peuvent accueillir que $2^7 = 128$ caractères dont une grande partie est occupée par les caractères de contrôle, les caractères latins majuscules, minuscules, les chiffres et les signes de ponctuation. Il n'y a même pas de place pour les caractères accentués. Les codages de la génération suivante (ASCII étendu, EBCDIC étendu, etc.) utilisent 8 bits, c'est-à-dire qu'ils peuvent accepter jusqu'à $2^8 = 256$ caractères. Les caractères latins accentués sont désormais codés. Ce sont les codages les plus répandus même actuellement, plus particulièrement, l'ASCII étendu qui devient le codage standard de Windows. C'est la raison pour laquelle INTEX adopte ce type de codage. Le Tableau 1.7 montre les caractères latins utilisés sous Windows.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC		‚ 201A	ƒ 0192	„ 201E	… 2026	† 2020	‡ 2021	ˆ 02C6	‰ 2030	Š 0160	‹ 2039	Œ 0152		Ž 017D	
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014	˜ 02DC	™ 2122	š 0161	› 203A	œ 0153		ž 017E	Ÿ 0178
A0	NBSP 00A0	¡ 00A1	¢ 00A2	£ 00A3	¤ 00A4	¥ 00A5	¦ 00A6	§ 00A7	¨ 00A8	© 00A9	ª 00AA	« 00AB	¬ 00AC	- 00AD	® 00AE	¯ 00AF
B0	° 00B0	± 00B1	² 00B2	³ 00B3	´ 00B4	µ 00B5	¶ 00B6	· 00B7	¸ 00B8	¹ 00B9	º 00BA	» 00BB	¼ 00BC	½ 00BD	¾ 00BE	¿ 00BF
C0	À 00C0	Á 00C1	Â 00C2	Ã 00C3	Ä 00C4	Å 00C5	Æ 00C6	Ç 00C7	È 00C8	É 00C9	Ê 00CA	Ë 00CB	Ì 00CC	Í 00CD	Î 00CE	Ï 00CF
D0	Ð 00D0	Ñ 00D1	Ò 00D2	Ó 00D3	Ô 00D4	Õ 00D5	Ö 00D6	× 00D7	Ø 00D8	Ù 00D9	Ú 00DA	Û 00DB	Ü 00DC	Ý 00DD	Þ 00DE	ß 00DF
E0	à 00E0	á 00E1	â 00E2	ã 00E3	ä 00E4	å 00E5	æ 00E6	ç 00E7	è 00E8	é 00E9	ê 00EA	ë 00EB	ì 00EC	í 00ED	î 00EE	ï 00EF
F0	ð 00F0	ñ 00F1	ò 00F2	ó 00F3	ô 00F4	õ 00F5	ö 00F6	÷ 00F7	ø 00F8	ù 00F9	ú 00FA	û 00FB	ü 00FC	ý 00FD	þ 00FE	ÿ 00FF

Tableau 1.7 Microsoft Windows Codepage 1252 (Latin I)

Le codage le plus récent est UNICODE qui utilise 16 bits et peut gérer jusqu'à 65 536 caractères (2^{16}). Ce codage est très intéressant parce qu'il est assez grand pour accepter presque toutes les langues du monde. Malheureusement, INTEX n'accepte pas encore ce type de codage.

La plupart des versions de Windows utilisent toujours l'ASCII étendu à 8 bits. Cependant, ses 256 places ne suffisent pas pour ajouter des caractères étrangers, elles ne suffisent même pas pour des langues à grand alphabet comme le chinois, le japonais ou le coréen qui possèdent des milliers de caractères. Nous montrerons comment Windows gère ce problème à travers 2 exemples : le thaï (langue à petit alphabet) et le coréen (langue à grand alphabet). Nous expliquerons également comment définir le fichier Alphabet dans chaque cas.

1.4.3 Exemple du thaï (langue à 1 octet)

Le nombre des caractères thaï est relativement petit (87 seulement, y compris les 10 chiffres thaï). Un octet (8 bits) qui offre 256 places, est largement suffisant. Néanmoins, la table de caractères du Tableau 1.7 est déjà complète, il faut donc supprimer certains caractères parmi les moins courants afin d'acquérir assez de places pour les 87 remplaçants du thaï.

1.4.3.1 Table des caractères thaï

Étant donné que les caractères latins accentués (en bas du Tableau 1.7) sont très peu utilisés (voire pas du tout) dans la langue thaï, ils sont remplacés par les caractères thaï comme le montre le Tableau 1.8.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	\ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80	€ 20AC					… 2026										
90		‘ 2018	’ 2019	“ 201C	” 201D	• 2022	– 2013	— 2014								
A0	NBSP 00A0	ก 0E01	ข 0E02	ฃ 0E03	ค 0E04	ฅ 0E05	ฆ 0E06	ง 0E07	จ 0E08	ฉ 0E09	ช 0E0A	ซ 0E0B	ฌ 0E0C	ญ 0E0D	ฎ 0E0E	ฏ 0E0F
B0	ฐ 0E10	ฑ 0E11	ฒ 0E12	ณ 0E13	ด 0E14	ต 0E15	ถ 0E16	ท 0E17	ธ 0E18	น 0E19	บ 0E1A	ป 0E1B	ผ 0E1C	ฝ 0E1D	พ 0E1E	ฟ 0E1F
C0	ภ 0E20	ม 0E21	ย 0E22	ร 0E23	ฤ 0E24	ล 0E25	ฦ 0E26	ว 0E27	ศ 0E28	ษ 0E29	ส 0E2A	ห 0E2B	ฬ 0E2C	อ 0E2D	ฮ 0E2E	ฯ 0E2F
D0	ะ 0E30	ั 0E31	า 0E32	ำ 0E33	ิ 0E34	ี 0E35	ึ 0E36	ื 0E37	ุ 0E38	ู 0E39	ฺ 0E3A					฿ 0E3F
E0	เ 0E40	แ 0E41	โ 0E42	ใ 0E43	ไ 0E44	ๅ 0E45	ๆ 0E46	็ 0E47	่ 0E48	้ 0E49	๊ 0E4A	๋ 0E4B	์ 0E4C	ํ 0E4D	๎ 0E4E	๏ 0E4F
F0	๐ 0E50	๑ 0E51	๒ 0E52	๓ 0E53	๔ 0E54	๕ 0E55	๖ 0E56	๗ 0E57	๘ 0E58	๙ 0E59	๚ 0E5A	๛ 0E5B				

Tableau 1.8 Microsoft Windows Codepage 874 (Thaï)

Le seul désavantage de cette méthode est que l'on ne peut pas écrire les caractères latins accentués et les caractères thaï avec la même police. Ceci empêche, par exemple, de faire de la traduction automatique du français vers le thaï, et vice versa, dans INTEX.

1.4.3.2 Fichier Alphabet du thaï dans INTEX

Étant donné que le thaï utilise maintenant le même codage que le français, nous pouvons définir le fichier Alphabet de la même façon.

Commençons par l'en-tête, nous définissons les polices thaï à utiliser et leur taille, puis, le caractère à cacher. Nous choisissons le blanc (0x20) comme séparateur artificiel des mots thaï parce qu'il sera transparent dans les transducteurs du texte (*cf. §6.4.1*) et nous mettons **2 blancs** derrière « ASIAN » : le deuxième est celui à cacher, le premier sert à séparer les deux termes (*cf. §1.2.1.1*).

```
#"DB ThaiText" 16
#"DB ThaiTextFixed" 16
#ASIAN
```

Nous décrivons ensuite les caractères que nous voulons considérer comme « lettres » en commençant par les caractères latins. Ceci est très important, d'une part parce que des mots anglais apparaissent souvent dans le texte thaï, d'autre part parce que nos dictionnaires sont codés avec les parties du discours et les codes syntactico-sémantiques en alphabet latin (ADV+z1, PREP, etc.). Si nous excluons ces caractères, nous ne pouvons pas employer les symboles lexicaux dans INTEX.

Comme les caractères latins accentués n'existent plus dans le Tableau 1.8, nous décrivons seulement les caractères sans accents.

```
Aa
Bb
Cc
⋮
Zz
```

Ensuite, nous définissons les caractères thaï en excluant les signes de ponctuation. Étant donné que l'alphabet thaï ne connaît pas la distinction entre majuscules et minuscules, nous écrivons deux fois le même caractère dans la même ligne. Nous incluons les chiffres thaï à la fin du fichier car nous voulons les considérer comme lettres afin de pouvoir les reconnaître plus tard par le dictionnaire des chiffres thaï (*cf. §3.4.4*).

ก ก (consonnes)
ข ข ⋮
ฃ ฃ
⋮
ฮ ฮ
ะ ะ (voyelles)
ั ั ⋮
า า
⋮
. .
เ เ
แ แ
⋮
า า
็ ็ (diacritiques)
่ ่ ⋮
⋮
์ ์
๐ ๐ (chiffres)
๑ ๑ ⋮
๒ ๒
⋮
๙ ๙

Le fichier complet est détaillé dans l'Annexe I.A. Les caractères thaï sont expliqués au Chapitre 2.

1.4.4 Exemple du coréen (langue à 2 octets)

Une langue à grand alphabet comme le coréen ne peut pas utiliser la même méthode que le thaï car elle possède des milliers de caractères[21] et la table ASCII étendu à 256 places ne lui suffit même pas. Pourtant, à moins d'utiliser l'UNICODE qui possède 65 536 places, on représente le coréen dans le système Windows avec le « double ASCII », c'est-à-dire que deux codes ASCII (2 octets) se combinent pour coder un caractère coréen. Ce système s'appelle « WANSUNG » et demeure toujours le standard sous Windows coréen.

[21] Caractères syllabiques : le coréen a en fait deux niveaux d'alphabet. Du point de vue informatique (et graphique), les éléments de l'alphabet sont des syllabes. Il en existe bien sûr plusieurs milliers. Du point de vue phonétique (et graphique), chacun se décompose en plusieurs lettres ou phonèmes qui appartiennent à un alphabet d'une vingtaine d'éléments.

1.4.4.1 Tables des caractères coréens

Les caractères latins (non accentués), les chiffres arabes et les signes de ponctuation occupent toujours la première partie de la table. Les deux codes du WANSUNG sont appelés l'octet de tête et l'octet de queue. Le premier trouve sa place entre 0x81 et 0xFE (126 places ; *voir Tableau 1.9*) tandis que le deuxième se trouve dans une des 3 zones suivantes : 0x41 à 0x5A (26 places), 0x61 à 0x7A (26 places) et 0x81 à 0xFE (126 places). Si Windows trouve un code dans le champ de l'alphabet latin, il l'interprète comme tel. En revanche, à chaque fois qu'il trouve un code dans la zone des octets de tête et que le code suivant est dans une des trois zones des octets de queue, alors il interprète les deux codes ensemble en un caractère coréen. Le Tableau 1.10 montre tous les caractères dont l'octet de tête est 0xB0.

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
00	NUL 0000	STX 0001	SOT 0002	ETX 0003	EOT 0004	ENQ 0005	ACK 0006	BEL 0007	BS 0008	HT 0009	LF 000A	VT 000B	FF 000C	CR 000D	SO 000E	SI 000F
10	DLE 0010	DC1 0011	DC2 0012	DC3 0013	DC4 0014	NAK 0015	SYN 0016	ETB 0017	CAN 0018	EM 0019	SUB 001A	ESC 001B	FS 001C	GS 001D	RS 001E	US 001F
20	SP 0020	! 0021	" 0022	# 0023	$ 0024	% 0025	& 0026	' 0027	(0028	) 0029	* 002A	+ 002B	, 002C	- 002D	. 002E	/ 002F
30	0 0030	1 0031	2 0032	3 0033	4 0034	5 0035	6 0036	7 0037	8 0038	9 0039	: 003A	; 003B	< 003C	= 003D	> 003E	? 003F
40	@ 0040	A 0041	B 0042	C 0043	D 0044	E 0045	F 0046	G 0047	H 0048	I 0049	J 004A	K 004B	L 004C	M 004D	N 004E	O 004F
50	P 0050	Q 0051	R 0052	S 0053	T 0054	U 0055	V 0056	W 0057	X 0058	Y 0059	Z 005A	[005B	₩ 005C	] 005D	^ 005E	_ 005F
60	` 0060	a 0061	b 0062	c 0063	d 0064	e 0065	f 0066	g 0067	h 0068	i 0069	j 006A	k 006B	l 006C	m 006D	n 006E	o 006F
70	p 0070	q 0071	r 0072	s 0073	t 0074	u 0075	v 0076	w 0077	x 0078	y 0079	z 007A	{ 007B	| 007C	} 007D	~ 007E	DEL 007F
80		81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
F0	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	

Tableau 1.9 Microsoft Windows Codepage 949 (Coréen)

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F
40		컮 CE9A	컯 CE9B	컱 CE9C	컲 CE9D	컳 CE9E	컴 CE9F	컵 CEA2	컶 CEA6	컷 CEA7	컸 CEA8	컹 CEA9	컺 CEAA	컻 CEAB	컾 CEAE	컿 CEAF
50	캀 CEB0	캁 CEB1	캂 CEB2	캃 CEB3	캄 CEB4	캅 CEB5	캆 CEB6	캇 CEB7	캈 CEB8	캉 CEB9	캊 CEBA					
60		캏 CEBB	캑 CEBC	캒 CEBD	캓 CEBE	캔 CEBF	캕 CEC0	캖 CEC2	캗 CEC3	캘 CEC4	캙 CEC5	캚 CEC6	캛 CEC7	캜 CEC8	캝 CEC9	캞 CECA
70	캟 CECB	캠 CECC	캡 CECD	캢 CECE	캣 CECF	캤 CED0	캥 CED1	캦 CED2	캧 CED3	캨 CED4	캩 CED5					
80		캫 CED6	캬 CED7	캭 CED8	캮 CED9	캯 CEDA	캰 CEDB	캱 CEDC	캲 CEDD	캳 CEDE	캴 CEDF	캵 CEE0	캶 CEE1	캷 CEE2	캸 CEE3	컊 CEE6
90	컧 CEE7	컩 CEE9	컪 CEEA	컫 CEED	컬 CEEE	컭 CEEF	컮 CEF0	컯 CEF1	컰 CEF2	컱 CEF3	컲 CEF6	컳 CEFA	컴 CEFB	컵 CEFC	컶 CEFD	컷 CEFE
A0	켛 CEFF	가 AC00	각 AC01	간 AC04	갇 AC07	갈 AC08	갉 AC09	갊 AC0A	감 AC10	갑 AC11	값 AC12	갓 AC13	갔 AC14	강 AC15	갖 AC16	갗 AC17
B0	같 AC19	갚 AC1A	갛 AC1B	개 AC1C	객 AC1D	갠 AC20	갤 AC24	갬 AC2C	갭 AC2D	갯 AC2F	갰 AC30	갱 AC31	갸 AC38	갹 AC39	갼 AC3C	걀 AC40
C0	걋 AC4B	걍 AC4D	걔 AC54	걘 AC58	걜 AC5C	거 AC70	걱 AC71	건 AC74	걷 AC77	걸 AC78	걺 AC7A	검 AC80	겁 AC81	것 AC83	겄 AC84	겅 AC85
D0	겆 AC86	겉 AC89	겊 AC8A	겋 AC8B	게 AC8C	겐 AC90	겔 AC94	겜 AC9C	겝 AC9D	겟 AC9F	겠 ACA0	겡 ACA1	겨 ACA8	격 ACA9	겪 ACAA	견 ACAC
E0	겯 ACAF	결 ACB0	겸 ACB8	겹 ACB9	겻 ACBB	겼 ACBC	경 ACBD	곁 ACC1	계 ACC4	곈 ACC8	곌 ACCC	곕 ACD5	곗 ACD7	고 ACE0	곡 ACE1	곤 ACE4
F0	곧 ACE7	골 ACE8	곪 ACEA	곬 ACEC	곯 ACEF	곰 ACF0	곱 ACF1	곳 ACF3	공 ACF5	곶 ACF6	과 ACFC	곽 ACFD	관 AD00	괄 AD04	괆 AD06	

Tableau 1.10 Caractères coréens qui commencent par 0xB0

Grâce à cette méthode, le système peut supporter jusqu'à 22 428 caractères coréens (126x(26+26+126)).

Or, il ne suffit pas d'installer une police du type WANSUNG pour pouvoir utiliser correctement le coréen, Windows a aussi besoin d'autres programmes gestionnaires spécifiques qui savent traiter, par exemple, le déplacement du curseur, l'effacement d'un caractère (un ou deux octets à la fois ?) et surtout, la méthode de saisie au clavier qui est assez compliquée. Ces programmes gestionnaires existent sur le marché. Mais si le but est de vouloir travailler avec INTEX en coréen, la solution la plus simple semble être d'installer INTEX sous Windows version coréenne.

1.4.4.2 Fichier Alphabet du coréen dans INTEX

Le fichier Alphabet recense tous les caractères que nous voulons considérer comme lettres. Bien que le coréen possède des milliers de caractères, nous ne devons en déclarer qu'une centaine.

INTEX ignore complètement le codage WANSUNG et interprète un caractère WANSUNG comme deux caractères ASCII.[22] En conséquence, pour le fichier Alphabet, il suffit de déclarer tous les caractères ASCII pouvant apparaître dans les codes WANSUNG, c'est-à-dire, les codes 0x41 à 0x5A, 0x61 à 0x7A et 0x81 à 0xFE (seulement 178 caractères).

Commençons par l'en-tête, comme il n'y a pas de caractère à cacher, nous déclarons tout simplement :

```
#"GulimChe" 12
```

Ensuite, les caractères latins sont déclarés pour pouvoir utiliser les symboles lexicaux.

```
Aa
Bb
⋮
Zz
```

Pour les caractères coréens, puisque les codes 0x41-0x5A correspondant aux caractères « A » à « Z » et les codes 0x61-0x7A correspondant aux caractères « a » à « z », ont déjà été déclarés à l'étape précédente, nous pouvons passer à l'étape suivante.

Vu que chaque ligne de la déclaration doit comporter au moins 2 codes ASCII, nous déclarons les codes entre 0x81 et 0xFE en double comme pour le thaï.

```
□□      (0x81)(0x81)
,,      (0x82)(0x82)
ƒƒ      (0x83)(0x83)
„„      (0x84)(0x84)
⋮
ýý      (0xFD)(0xFD)
þþ      (0xFE)(0xFE)
```

D'après nos expériences, les caractères courants du coréen se trouvent uniquement entre 0xA1 et 0xFE et ils sont bien triés dans l'ordre alphabétique. En déclarant seulement les caractères de cette zone dans le fichier Alphabet (*voir Annexe I.B*), nous obtenons déjà des résultats très satisfaisants.

[22] Même si INTEX ne comprend pas le codage WANSUNG, l'affichage à l'écran est correct grâce aux programmes gestionnaires dans Windows coréen.

Il y a quand même quelques précautions à prendre afin de pouvoir bien travailler avec le coréen dans INTEX :

- Il existe aussi des caractères latins à 2 octets, codés avec la méthode WANSUNG. Il faut faire attention à ne pas les mélanger avec les caractères latins à 1 octet.
- Des signes de ponctuation à 2 octets existent également et il faut absolument les éviter car ils sont dans la zone WANSUNG. Tous les caractères dans la zone WANSUNG sont déclarés dans le fichier Alphabet en tant que « lettres ». De ce fait, les signes de ponctuation à 2 octets sont considérés comme lettres et pas comme séparateurs.
- Puisqu'INTEX considère un caractère coréen comme 2 caractères, si nous voulons, par exemple, supprimer un caractère coréen dans INTEX par l'opérateur d'effacement « `L` » (*cf. Figure 1.12*), il faut supprimer 2 caractères avec « `LL` ».

1.5 Problèmes du thaï dans INTEX

La langue thaï a une structure très différente des langues européennes. Même si nous pouvons déjà intégrer les caractères thaï dans INTEX, il reste encore beaucoup de problèmes importants à régler. Regardons l'extrait de texte thaï suivant :

โดยขณะนี้ นักลงทุนชาวญี่ปุ่นได้เตรียมจะออกมาลงทุนในต่างประเทศอีกระลอก เนื่องจากภาวะเศรษฐกิจภายในประเทศญี่ปุ่นประสบปัญหาค่อนข้างมาก ส่งผลกระทบต่อต้นทุนการผลิตที่มีแนวโน้มจะสูงกว่าการเข้าลงทุนในประเทศอื่นๆ ทั้งนี้ ประเทศไทยถือเป็นประเทศเป้าหมายหลักในภูมิภาคเอเชียอาคเนย์ของนักลงทุนญี่ปุ่นที่จะเข้ามาลงทุนมากที่สุดเป็นอันดับต้นๆ เนื่องจากประเทศไทยมีข้อดีต่างๆ เช่น ค่าจ้างแรงงานที่ไม่แพง ความมั่นคงทางการเมือง เป็นต้น อย่างไรก็ตาม ไทยเองก็มีคู่แข่งที่น่ากลัวคือประเทศจีน ดังนั้น หากไทยต้องการที่จะไม่ถูกแย่งตลาดไปยังจีนเป็นจำนวนมากนั้น จำเป็นต้องตื่นตัวและเตรียมพร้อมในการปรับปรุงเรื่องต่างๆที่จะเป็นการดึงดูดนักลงทุนอย่างเร่งด่วน

Certains caractères thaï s'écrivent au-dessus ou au-dessous des autres, mais cela ne pose aucun problème à INTEX parce que leurs codes de caractères en ASCII constituent un flot linéaire de gauche à droite.

Le vrai problème est que le texte thaï ressemble à une suite ininterrompue de lettres, au mieux découpée à la fin de phrase avec un blanc (jamais avec un point). En fait, le thaï n'a pas de séparateur de mots, les mots sont généralement soudés. De plus, le blanc qui sert à séparer des phrases est ambigu parce que les signes de ponctuation d'origine européenne

(telles que la virgule, le point-virgule, les deux-points, le point d'exclamation et le point d'interrogation) sont facultatifs en thaï et sont généralement remplacés par des blancs. D'autant plus que l'alphabet thaï n'a pas de majuscules qui commencent les nouvelles phrases comme en ont les langues européennes, la frontière de phrase thaï est visuellement indécise.

Les problèmes du thaï ressemblent beaucoup aux problèmes de la reconnaissance vocale. À partir de la voix capturée d'un locuteur, sa parole est transcrite en phonétique comme :

fʀɑ̃klɛ̃sekɔ̃teʒyskadis/alɑ̃dʀwaealɑ̃vɛʀ/epøʀesitelalfabɛdynsœltʀɛt/
ilɛmdesine/ɔpsɛʀvele∫ozotuʀdəlɥi/epaʀtaʒesezɛkspeʀjɑ̃s/
ilpʀɑ̃sekʀɛjɔ̃esɔ̃kaje/pɥiilsaswapuʀʀefle∫iʀ/
ilpɑ̃sdabɔʀovɑ̃dœʀdəglas/pɥialapistsiklabl/eɑ̃sɥitotɛʀɛ̃dəfut

À ce stade, on rencontre deux problèmes communs avec le thaï : les caractères sont collés ensemble et le signe « / » (pause) qui peut servir à découper la séquence en phrases est ambigu car il peut correspondre à plusieurs signes de ponctuation ou à aucune (cela peut être juste une pause de respiration du locuteur). La reconnaissance des mots et des phrases est donc indispensable pour transcrire la séquence phonétique ci-dessus comme suit :

Franklin sait compter jusqu'à dix à l'endroit et à l'envers et peut réciter l'alphabet d'une seule traite. Il aime dessiner, observer les choses autour de lui et partager ses expériences.[23] Il prend ses crayons et son cahier, puis il s'assoit pour réfléchir. Il pense d'abord au vendeur de glace, puis à la piste cyclable, et ensuite au terrain de foot.[24]

En outre, à cause des limites de la programmation, INTEX ne peut pas supporter plus de 512 caractères dans un lexème et une phrase ne peut pas accepter plus de 512 lexèmes. Nous devons absolument trouver des moyens pour segmenter des textes thaï en unités plus petites (*voir Chapitre 4*) avant de pouvoir procéder aux analyses des textes (*voir Chapitre 6*).

[23] BOURGEOIS et CLARK 2002, p. 1
[24] *Ibid*., p. 7

CHAPITRE 2 SYSTÈME D'ÉCRITURE DU THAÏ

Le thaï (siamois) est la langue officielle du royaume de Thaïlande. Il appartient à la famille taï, un sous-groupe de la famille kadaï (ou kam-taï). Un certain nombre de linguistes considèrent maintenant le kadaï, avec l'austronésien, comme une branche de l'austro-taï, bien que cette hypothèse demeure controversée.[1]

La recherche linguistique a identifié la zone près de la frontière nord du Viêt-nam et de la frontière sud-est de la Chine comme le point d'origine probable des langues taï. Aujourd'hui, la famille taï inclut les langues parlées en Assam[2], au Myanmar[3] du nord, en Thaïlande, au Laos, au Viêt-nam du nord et dans les provinces chinoises du Yunnan, du Guizhou et du Guangxi.

Les points communs entre les langues taï sont les suivants. Le morphème est monosyllabique. La plupart de ces langues, dites isolantes, ont six tons (sauf le thaï et le shan[4], qui ont cinq tons). L'ordre des constituants syntaxiques est semblable à celui du français. Il s'agit de langues centrifuges : le déterminant suit le déterminé. Les substantifs sont employés avec un classificateur. Il y a plusieurs dizaines de tels classificateurs en thaï et en lao, par exemple. Pas de marque de nombre, ni de genre grammatical. Pas de déclinaison ni de conjugaison. Au verbe peuvent s'adjoindre des marques d'aspect, exprimées par des morphèmes libres (isolés).[5]

---

[1] HUDAK 1987, p. 757
[2] Un état de l'Inde
[3] Ex-Birmanie
[4] Langue parlée au Myanmar du nord
[5] COYAUD 1997, p. 72

Quant à l'écriture du thaï, elle fut inventée en 1283 par le Roi Ramkhamhaeng le Grand du royaume Sukhothai, le premier royaume majeur thaïlandais, en adaptant l'écriture brâhmî (de l'Inde) par l'intermédiaire du khmer (du Cambodge) et du môn (de la Birmanie). Elle a subi de nombreux changements stylistiques depuis lors.

2.1 Alphabet thaï

Basé sur un système alphabétique, le thaï possède ses propres caractères dont certains s'écrivent au-dessus ou au-dessous des autres. Il s'écrit horizontalement de gauche à droite et principalement sans espace entre les mots. L'alphabet thaï ne distingue pas les majuscules des minuscules. Les caractères sont classés en 6 catégories que nous allons passer en revue.

2.1.1 Consonnes

Il existe 44 graphèmes consonantiques dont 2 sont obsolètes. Toutes les consonnes peuvent apparaître en position initiale de syllabe mais seulement 35 en position finale. Dans la position initiale, il n'existe que 21 phonèmes consonantiques parce qu'un même phonème peut correspondre à plusieurs graphèmes. De la même manière, dans la position finale, il n'en existe que 8. En effet, certaines consonnes peuvent se prononcer différemment selon qu'elles sont consonnes initiales ou consonnes finales. De plus, certaines consonnes peuvent se combiner avec une autre consonne pour former un agglomérat consonantique[6]. Les agglomérats consonantiques n'existent fréquemment que dans la position initiale des syllabes.

Graphème	Épelé[7]	Phonème		Agglomérat consonantique [Phonème]
		initial	final	
ก	kɔ: kàj (poulet)	k	k	กร [kr], กล [kl], กว [kw]
ข	khɔ̌: khàj (œuf)	kh	k	ขร [khr], ขล [khl], ขว [khw]
ฃ[8]	khɔ̌: khùat (bouteille)	kh	-	
ค	khɔ: khwa:j (buffle)	kh	k	คร [khr], คล [khl], คว [khw]
ฅ[9]	khɔ: khon (personne)	kh	-	
ฆ	khɔ: rákhaŋ (cloche)	kh	k	

[6] Un groupe de deux consonnes successives. [DUBOIS *et al*. 1994, p. 22]
[7] Nous faisons figurer entre parenthèses le nom usuel du graphème.
[8] Caractère obsolète
[9] *Idem*

ง	ŋɔː ŋuː (serpent)	ŋ	ŋ	
จ	cɔː caːn (assiette)	c	t	จร [c][10]
ฉ	chɔ̌ː chǐŋ (cymbale)	ch	-	
ช	chɔː cháːŋ (éléphant)	ch	t	ชร [chr][11]
ซ	sɔː sôː (chaîne)	s	t	ซร [s][12] / [sr][13]
ฌ	chɔː chɣː (arbre)	ch	-	
ญ	jɔː jǐŋ (femme)	j	n	
ฎ	dɔː chádaː (couronne)	d	t	
ฏ	tɔː pàtàk (pique)	t	t	
ฐ	thɔ̌ː thǎːn (socle)	th	t	
ฑ	thɔː montho: (géant)	th / d	t	
ฒ	thɔː phûːthâw (vieillard)	th	t	
ณ	nɔː neːn (novice)	n	n	
ด	dɔː dèk (enfant)	d	t	ดร [dr][14]
ต	tɔː tàw (tortue)	t	t	ตร [tr]
ถ	thɔ̌ː thŭŋ (sac)	th	t	
ท	thɔː tháhǎːn (soldat)	th	t	ทร [s][15] / [thr]
ธ	thɔː thoŋ (drapeau)	th	t	ธร[thr][16]
น	nɔː nŭː (souris)	n	n	

[10] /r/ n'est pas prononcé.
[11] Phonème rare du vocabulaire poétique
[12] /r/ n'est pas prononcé.
[13] Phonème rare du vocabulaire formel
[14] Phonème rare du vocabulaire d'origine européenne
[15] Prononciation spéciale
[16] Phonème rare du vocabulaire archaïque

บ	bɔ: bajmáj (feuille)	b	p	(บร[br], บล[bl])[17]
ป	pɔ: pla: (poisson)	p	p	ปร [pr], ปล [pl]
ผ	phɔ̌: phûŋ (abeille)	ph	-	ผล [phl]
ฝ	fɔ̌: fǎ: (couvercle)	f	-	
พ	phɔ: pha:n (coupe)	ph	p	พร [phr], พล [phl]
ฟ	fɔ: fan (dent)	f	p	(ฟร [fr], ฟล [fl])[18]
ภ	phɔ: sămphaw (jonque)	ph	p	ภร [phr]
ม	mɔ: má: (cheval)	m	m	
ย	jɔ: ják (démon)	j	j	
ร	rɔ: rɯa (bateau)	r	n	
ล	lɔ: liŋ (singe)	l	n	
ว	wɔ: wɛ̌:n (bague)	w	w	
ศ	sɔ̌: sǎ:la: (pavillon)	s	t	ศร [s][19]
ษ	sɔ̌: rɯ:sǐ: (ermite)	s	t	
ส	sɔ̌: sŭa (tigre)	s	t	สร [s][20] / [sr]
ห	hɔ̌: hĭ:p (caisse)	h	-	(หง [ŋ], หญ [j], หน [n], หม [m], หย [j], หร [r], หล [l], หว [w])[21]
ฬ	lɔ: cùla: (cerf-volant)	l	n	
อ	ʔɔ: ʔà:ŋ (cuvette)	ʔ[22]	-	อย [j][23]
ฮ	hɔ: nókhû:k (hibou)	h	-	

Tableau 2.1 Consonnes du thaï

[17] Phonème rare du vocabulaire d'origine européenne
[18] *Idem*
[19] /r/ n'est pas prononcé.
[20] *Idem*
[21] Agglomérats tonals : chacun se prononce comme sa deuxième consonne mais avec un changement de ton.
[22] Certains linguistes considèrent ce phonème comme muet.
[23] Agglomérat tonal : il se prononce comme sa deuxième consonne mais avec un changement de ton.

2.1.2 Voyelles

Il faut souvent une séquence de plusieurs caractères vocaliques pour représenter un phonème vocalique. La table de caractères ASCII du thaï compte 18 caractères vocaliques (*voir Tableau 2.2*). Ces caractères peuvent se combiner entre eux, et même avec certaines consonnes (ย, ว, อ), pour former 32 phonèmes vocaliques différents (*voir Tableau 2.3*). La plupart des caractères vocaliques, dits non-isolants, s'écrivent obligatoirement avec une consonne (ou un agglomérat consonantique). Cette dernière est marquée dans les tableaux par un tiret « - ».

Caractère vocalique	Épelé
-ะ	sàrà ʔà
ั-	máj hăn ʔakàːt
-า	sàrà ʔaː
ำ-	sàrà ʔam
ิ-	sàrà ʔì
ี-	sàrà ʔiː
ึ-	sàrà ʔɯ̀
ื-	sàrà ʔɯː
ุ-	sàrà ʔù
ู-	sàrà ʔuː
เ-	sàrà ʔeː
แ-	sàrà ʔɛː
โ-	sàrà ʔoː
ใ-	sàrà ʔaj máj múan
ไ-	sàrà ʔaj máj málaːj
-ๅ	lâk khâŋ jaːw
ฤ[24]	rɔː rɯ́
ฦ[25]	lɔː lɯ́

Tableau 2.2 Caractères vocaliques du thaï

[24] Caractère vocalique isolant
[25] Caractère vocalique isolant et obsolète

En prenant en compte la position des caractères vocaliques par rapport à la position des consonnes initiales, nous pouvons classer les caractères vocaliques thaï en 4 catégories :

- Voyelles antéposées : caractères vocaliques qui s'écrivent devant une consonne initiale : เ-, แ-, โ-, ใ-, ไ-
- Voyelles postposées : caractères vocaliques qui s'écrivent derrière une consonne initiale : -ะ, -า, -ำ, -ๅ
- Voyelles suscrites : caractères vocaliques qui s'écrivent au-dessus d'une consonne initiale ou de la deuxième consonne[26] d'un agglomérat consonantique : ◌ั, ◌ิ, ◌ี, ◌ึ, ◌ื
- Voyelles souscrites : caractères vocaliques qui s'écrivent au-dessous d'une consonne initiale ou de la deuxième consonne d'un agglomérat consonantique : ◌ุ, ◌ู

Les phonèmes vocaliques du thaï sont divisés en phonèmes courts et longs. Les deux types de phonèmes sont distingués par des formes différentes.

Voyelle courte	Phonème	Exemple	Voyelle longue	Phonème	Exemple
-ะ	a	กะ [kà]	-า	aː	กา [kaː]
◌ิ	i	กิ [kì]	◌ี	iː	กี [kiː]
◌ึ	ɯ	กึ [kɯ̀]	◌ื	ɯː	กือ[27] [kɯː]
◌ุ	u	กุ [kù]	◌ู	uː	กู [kuː]
เ-ะ	e	เกะ [kè]	เ-	eː	เก [keː]
แ-ะ	ɛ	แกะ [kɛ̀]	แ-	ɛː	แก [kɛː]
โ-ะ	o	โกะ [kò]	โ-	oː	โก [koː]
เ-าะ	ɔ	เกาะ [kɔ̀]	-อ	ɔː	กอ [kɔː]
เ-อะ	ɣ	เกอะ [kɣ̀]	เ-อ	ɣː	เกอ [kɣː]
เ◌ียะ	ia	เกียะ [kìa]	เ◌ีย	iaː	เกีย [kiaː]
เ◌ือะ	ɯa	เกือะ [kɯ̀a]	เ◌ือ	ɯaː	เกือ [kɯaː]
◌ัวะ	ua	กัวะ [kùa]	◌ัว	uaː	กัว [kuaː]
ฤ (voyelle isolante)	rɯ	กฤ [krɯ̀]	ฤๅ (voyelle isolante)	rɯː	กฤๅ [krɯː]

[26] À l'exception du mot « ศีรษะ » [sǐːsà] (tête) : la voyelle suscrite « ◌ี » [iː] s'écrit sur la première consonne de l'agglomérat consonantique « ศร ».

[27] En l'absence d'une consonne finale pour cette voyelle, on ajoute automatiquement la consonne « อ ».

ฦ[28] (voyelle isolante)	lɯ	กฦ [klɯ̀]	ฦๅ[29] (voyelle isolante)	lɯː	กฦๅ [klɯː]
-ำ	am	กำ [kam]			
ไ-	aj	ไก [kaj]			
ไ-	aj	ไก [kaj]			
เ-า	aw	เกา [kaw]			

Tableau 2.3 Voyelles du thaï

Ainsi, une voyelle peut s'écrire soit devant, soit derrière, soit au-dessus, soit au-dessous d'une consonne initiale ou d'un agglomérat consonantique initial. Il existe même des phonèmes vocaliques s'écrivant avec plusieurs caractères qui entourent une consonne initiale. Mais phonétiquement, le phonème vocalique se prononce toujours à la suite du phonème consonantique initial (*voir Exemples dans le Tableau 2.3*).

2.1.3 Signes diacritiques

2.1.3.1 Marques de tons

Les tons sont une des caractéristiques de la langue thaï. Il s'agit de traits distinctifs affectant des phonèmes. Les différents tons associés à un même mot peuvent correspondre à des sens différents. Les marques de tons s'écrivent au-dessus d'une consonne initiale ou de la deuxième consonne d'un agglomérat consonantique initial ou encore au-dessus d'une voyelle suscrite, si cette dernière existe. En thaï, il existe 5 tons dont 4 graphèmes :

Ton	Graphème	Épelé	Signe phonétique[30]	Exemple
moyen[31]	(aucun)	-	(aucun)	ปา [paː] (jeter)
bas[32]	่	máj ʔèk	̀	ป่า [pàː] (forêt)
descendant[33]	้	máj thoː	̂	ป้า [pâː] (tante)
haut[34]	๊	máj triː	́	ป๊า [páː] (père)
montant[35]	๋	máj càttàwaː	̌	ป๋า [pǎː] (père)

Tableau 2.4 Marques de tons du thaï

[28] Voyelle obsolète
[29] *Idem*
[30] Le signe phonétique tonal s'écrit sur le phonème vocalique.
[31] Se réalise à une hauteur d'intonation moyenne.
[32] Un peu plus bas que le ton moyen.
[33] Commence sa courbe mélodique plus haut que la hauteur moyenne, puis monte légèrement avant de redescendre.
[34] Commence sa courbe mélodique un peu plus haut que la hauteur moyenne, puis monte légèrement.
[35] Commence sa courbe mélodique un petit peu plus bas que le ton bas, puis descend légèrement avant de remonter.

2.1.3.2 Signes de phonème spécial

Il existe encore 2 signes diacritiques en thaï :

- Le signe de phonème court s'écrit au-dessus d'une consonne initiale ou de la deuxième consonne d'un agglomérat consonantique initial afin d'indiquer que la voyelle de la syllabe doit être prononcée brève même si elle est écrite longue.
- Le signe « thanthákhât », autrement dit, le signe qui tue, se met au-dessus d'un caractère pour le rendre muet. Il peut également tuer plusieurs caractères contigus. Dans ce cas, le signe s'écrit sur la dernière lettre de ces caractères. Les caractères qui deviennent muets, dits en thaï « tua karant », peuvent être d'une à trois consonnes, composées ou non avec les voyelles « ◌ิ » [i] ou « ◌ุ » [u]. Normalement, le signe de phonème muet ne change que la prononciation, pas le sens du mot. Cela permet de garder les formes originales des mots empruntés, par exemple, tout en adaptant leur prononciation aux habitudes thaïlandaises qui favorisent les mots courts.

Signe	Épelé	Signification	Exemple	
◌็	máj tàjkhú:	Phonème court	ลอก [lɔ̂ːk] (copier)	ล็อก [lɔ́k] (verrouiller)
◌์	thanthákhât	Phonème muet	สิทธิ [sìttí] ((un) droit)	สิทธิ์ [sìt] ((un) droit)

Tableau 2.5 Signes de phonème spécial du thaï

2.1.4 Signes pāli-sanskrits

Le thaï emprunta autrefois beaucoup de mots de l'Inde et la table ASCII du thaï code encore de nos jours 3 signes pāli-sanskrits. Ces signes sont, à présent, inusités sauf dans des documents anciens ou dans des textes de prières écrits en pāli ou en sanskrit.

Signe	Épelé	Utilisation	
		En pāli	En sanskrit
◌์	phinthú	Indiquer la consonne finale	Indiquer l'agglomérat consonantique
◌ं	níkkháhìt	Comme consonne finale [ŋ]	Comme consonne finale [m]
◌ฺ	ja:mákka:n	Indiquer l'agglomérat consonantique	Indiquer l'agglomérat consonantique

Tableau 2.6 Signes pāli-sanskrits

2.1.5 Chiffres

Les caractères de chiffres thaï sont utilisés moins fréquemment que les chiffres arabes dans la vie quotidienne, à l'exception des documents officiels. Ils vont de zéro jusqu'à neuf.

Chiffre	Épelé	Valeur
๐	sǔ:n	0
๑	nʉ̀ŋ	1
๒	sɔ̌:ŋ	2
๓	sǎ:m	3
๔	sì:	4
๕	hâ:	5
๖	hòk	6
๗	cèt	7
๘	pɛ̀:t	8
๙	kâ:w	9

Tableau 2.7 Chiffres thaï

2.1.6 Signes de ponctuation

Il existe 6 signes de ponctuation thaï qu'il ne faut pas considérer comme des lettres. Ils sont donc exclus du fichier Alphabet d'INTEX (*cf. §1.2.1.3, §1.4.3.2*). Les 3 derniers sont désuets et ne sont trouvés que dans des documents anciens.

Signe de ponctuation	Épelé	Signification
ฯ	pajja:nnɔ́:j	Signe abréviatif
ๆ	máj jámók	Signe répétitif
฿	bà:t	Monnaie thaïlandaise
๏	fɔ:ŋman	Début de paragraphe
๚	ʔaŋkhânkhû:	Fin de chapitre
๛	kho:mû:t	Fin d'histoire

Tableau 2.8 Signes de ponctuation du thaï

2.2 Syllabes

Une syllabe thaï est un ensemble composé de consonnes, de voyelles et d'un ton. Elle se prononce d'une seule émission de voix. Étant donné que le thaï est une langue tonale, le ton est un élément indispensable dans une syllabe même si, parfois, il n'est pas explicitement écrit.

2.2.1 Composition des syllabes

Le noyau de la syllabe (à l'oral) est la voyelle. Malgré cela, la consonne initiale est primordiale (à l'écrit) parce que la plupart des voyelles thaï sont non-isolantes et doivent apparaître avec une consonne. À l'exception des voyelles isolantes, une syllabe thaï est composée des façons suivantes :

2.2.1.1 Syllabe à 3 éléments

C'est la syllabe la plus petite en thaï. Sa structure est :

Consonne initiale + Voyelle + Ton

Il faut noter que la consonne initiale peut être un agglomérat consonantique et que le ton ne s'écrit pas quand il s'agit du ton moyen. Par exemple :

Exemple	Consonne initiale	Voyelle	Ton
ปลา [plaː] (poisson)	ปล [pl]	-า [aː]	- (moyen)
กู้ [kûː] (emprunter)	ก [k]	◌ู [uː]	◌้ [◌̂]
โต๊ะ [tó] (table)	ต [t]	โ-ะ [o]	◌๊ [◌́]

Tableau 2.9 Syllabe à 3 éléments

2.2.1.2 Syllabe à 4 éléments

Il existe 2 types de syllabe à 4 éléments :

2.2.1.2.1 Avec consonne finale

La structure est comme la syllabe à 3 éléments, avec une consonne finale en plus. Les consonnes initiale et finale peuvent également être des agglomérats consonantiques.

Consonne initiale + Voyelle + Ton + Consonne finale

Par exemple :

Exemple	Consonne initiale	Voyelle	Ton[36]	Consonne finale
แปลก [plɛ̀ːk] (étrange)	ปล [pl]	แ- [ɛː]	- [̀]	ก [k]
บุตร [bùt] (enfant)	บ [b]	ุ [u]	- [̀]	ตร [t][37]
ข้าว [khâːw] (riz)	ข [kh]	-า [aː]	้- [̂]	ว [w]

Tableau 2.10 Syllabe à 4 éléments (avec consonne finale)

2.2.1.2.2 Avec phonème muet

Comme la syllabe à 3 éléments, plus un phonème muet :

Consonne initiale + Voyelle + Ton + Phonème muet

Exemple	Consonne initiale	Voyelle	Ton	Phonème muet
สีห์ [sǐː] (lion)	ส [s]	ี- [iː]	- [̌]	ห์ [-]
เล่ห์ [lêː] (ruse)	ล [l]	เ- [eː]	่- [̂]	ห์ [-]
เกียร์ [kia] (engrenage)	ก [k]	เ-ีย [ia]	- (moyen)	ร์ [-]

Tableau 2.11 Syllabe à 4 éléments (avec phonème muet)

2.2.1.3 Syllabe à 5 éléments

C'est la structure la plus grande, composée de tous les éléments. Les consonnes initiale et finale peuvent également être des agglomérats consonantiques.

Consonne initiale + Voyelle + Ton + Consonne finale + Phonème muet

Exemple	Consonne initiale	Voyelle	Ton	Consonne finale	Phonème muet
สิทธิ์ [sìt] (droit)	ส [s]	ิ- [i]	- [̀]	ท [t]	ธิ์ [-]
พราหมณ์ [phraːm] (brahmane)	พร [phr]	-า [aː]	- (moyen)	หม [m]	ณ์ [-]
ฟิล์ม[38] [fim] (film)	ฟ [f]	ิ- [i]	- (moyen)	ม [m]	ล์ [-]
ศาสตร์ [sàːt] (science)	ศ [s]	-า [aː]	- [̀]	ส [t]	ตร์ [-]

Tableau 2.12 Syllabe à 5 éléments

[36] Parfois, le phonème tonal ne correspond pas à son graphème. En effet, les consonnes (initiale et finale) et la voyelle peuvent modifier le phonème tonal de la syllabe qu'elles composent. Pour plus d'information, consulter un livre de la grammaire thaï ; par exemple : DELOUCHE 1994, p. LI. 5 ; SIRIBOONMA 1999, pp. 131-149 ; HUDAK 1987, pp. 761-766.

[37] Normalement en position finale, l'agglomérat consonantique se prononce comme une seule consonne.

[38] Il est possible que le phonème muet ne soit pas en dernière position de la syllabe.

2.2.2 Anomalies des syllabes thaï

Certaines syllabes ont une forme irrégulière qui fait exception aux règles de composition des syllabes. Il s'agit de la variation des formes vocaliques surtout lorsqu'elles prennent une consonne finale.

2.2.2.1 Forme supprimée

Il peut arriver que le graphème vocalique d'une syllabe ne soit pas explicitement écrit alors que le phonème est effectivement prononcé. Il s'agit des voyelles « -ะ » [a], « -อ » [ɔː] et « เ-ะ » [o].

2.2.2.1.1 Voyelle « -ะ » [a]

- Certaines syllabes comportent la voyelle [a] sans qu'elle soit écrite ; par exemple :

  « ณ » [n] qui se prononce [ná] et « ธ » [th] qui se prononce [thá].

2.2.2.1.2 Voyelle « -อ » [ɔː]

- Certaines syllabes comportent la voyelle [ɔː] sans qu'elle soit écrite ; par exemple :

  « บ » [b] qui se prononce [bɔː].

- Lorsque la voyelle [ɔː] prend le « ร » [r] comme consonne finale, le graphème vocalique est entièrement supprimé ; par exemple :

  « ก » [k] + « -อ » [ɔː] + « ร » [n][39] → « กร » [kɔːn]

2.2.2.1.3 Voyelle « เ-ะ » [o]

- Lorsque la voyelle [o] prend une consonne finale (sauf le « ร » [r]), le graphème vocalique est entièrement supprimé ; par exemple :

  « ก » [k] + « เ-ะ » [o] + « ด » [t] → « กด » [kòt]

2.2.2.2 Forme réduite

Certaines voyelles réduisent leurs graphèmes en présence d'une consonne finale.

2.2.2.2.1 Voyelle « เ-อ » [ɣː]

- Si la voyelle « เ-อ » [ɣː] s'écrit avec la consonne finale « ย » [j], le « อ » est supprimé ; par exemple :

  « ก » [k] + « เ-อ » [ɣː] + « ย » [j] → « เกย » [kɣːj]

[39] Quand le « ร » [r] est utilisé en tant que consonne finale, il se prononce [n] (*cf. Tableau 2.1*).

2.2.2.2.2 Voyelle « ◌ัว » [uaː]

- Si la voyelle « ◌ัว » [uaː] s'écrit avec une consonne finale, le « ◌ั » est supprimé :

  « ก » [k] + « ◌ัว » [uaː] + « น » [n] → « กวน » [kuaːn]

2.2.2.3 Forme modifiée

Certaines voyelles changent de forme écrite en présence d'une consonne finale.

2.2.2.3.1 Voyelle « -ะ » [a]

- Lorsque la voyelle « -ะ » [a] prend une consonne finale, le « -ะ » est graphiquement changé en « ◌ั » ; par exemple :

  « ก » [k] + « -ะ » [a] + « น » [n] → « กัน » [kan]

- Quand la consonne « ร » [r] ou l'agglomérat consonantique commençant par le « ร » sont écrits en tant que consonne finale de la voyelle « -ะ » [a], le « -ะ » est graphiquement changé en « ร ». Cela s'appelle en thaï « rɔːhăn ». Par exemple :

  « ส » [s] + « -ะ » [a] + « ร » [n] → « สรร » [săn]

  « ก » [k] + « -ะ » [a] + « รม » [m] → « กรรม » [kam]

2.2.2.3.2 Voyelle « เ-ะ » [e]

- Lorsque la voyelle « เ-ะ » [e] prend une consonne finale, le « -ะ » est graphiquement changé en « ◌็ » ; par exemple :

  « ก » [k] + « เ-ะ » [e] + « ง » [ŋ] → « เก็ง » [keŋ]

2.2.2.3.3 Voyelle « แ-ะ » [ɛ]

- Lorsque la voyelle « แ-ะ » [ɛ] prend une consonne finale, le « -ะ » est graphiquement changé en « ◌็ » ; par exemple :

  « ข » [kh] + « แ-ะ » [ɛ] + « ง » [ŋ] → « แข็ง » [khĕŋ]

2.2.2.3.4 Voyelle « เ-าะ » [ɔ]

- Lorsque la consonne initiale « ก » [k] prend la voyelle « เ-าะ » [ɔ] avec le ton descendant « ◌้ » [◌̂], la forme « เก้าะ » est complètement changée en « กิ๋ » [kɔ̂].

- Lorsque la voyelle « เ-าะ » [ɔ] prend une consonne finale, la forme est complètement changée en « ◌็อ » ; par exemple :

  « น » [n] + « เ-าะ » [ɔ] + « ต » [t] → « น็อต » [nɔ́t]

2.2.2.3.5 Voyelle « เ-อ » [ɣː]

- Lorsque la voyelle « เ-อ » [ɣː] prend une consonne finale (sauf le « ย » [j]), le « -อ » est graphiquement changé en « ◌ิ »[40] ; par exemple :

  « ก » [k] + « เ-อ » [ɣː] + « ด » [t] → « เกิด » [kɣ̀ːt]

2.2.2.3.6 Voyelle « ◌ัวะ » [ua]

- Lorsque la voyelle « ◌ัวะ » [ua] prend une consonne finale, le « -ะ » est graphiquement changé en « ◌็ » et le « ◌ั » est supprimé ; par exemple :

  « ก » [k] + « ◌ัวะ » [ua] + « ง » [ŋ] → « กั็วง » [kuaŋ]

2.2.2.4 Forme ajoutée

Parfois, il faut ajouter un élément dans la syllabe pour qu'elle soit complète. Il ne s'agit, dans ce cas, que d'une voyelle.

2.2.2.4.1 Voyelle « ◌ื » [ɯː]

- Lorsque la voyelle « ◌ื » [ɯː] ne prend aucune consonne finale, on ajoute automatiquement le « -อ » ; par exemple :

  « ม » [m] + « ◌ื » [ɯː] → « มือ » [mɯː]

Voici la table récapitulative des formes syllabiques du thaï (sans phonème muet) :

(c_i = consonne initiale ; c_f = consonne finale)

Composition	Forme					Exemple
	normale	supprimée	réduite	modifiée	ajoutée	
c_i + -ะ [a]	c_iะ					กะ [kà] (estimer)
		c_i				ณ [ná] (à)
c_i + -ะ [a] + c_f				c_iัc_f		กัน [kan] (empêcher)
c_i + -ะ [a] + ร [n]				c_iรร		สรร [săn] (choisir)
c_i + -ะ [a] + รc_f				c_iรรc_f		กรรม [kam] (action)
c_i + -า [aː]	c_iา					ปา [paː] (jeter)
c_i + -า [aː] + c_f	c_iาc_f					ปาก [pàːk] (bouche)

[40] À l'exception des mots « เทอญ » [tɤːn], « เทอม » [tɤːm] et « เยอว » [jɤːw] qui ne changent pas de forme.

c_i + ◌ิ [i]	c_iิ					ติ [tì] (blâmer)
c_i + ◌ิ [i] + c_f	c_iิc_f					ปิด [pìt] (fermer)
c_i + ◌ี [iː]	c_iี					สี [sǐː] (couleur)
c_i + ◌ี [iː] + c_f	c_iีc_f					ถีบ [thìːp] (pédaler)
c_i + ◌ึ [ɯ]	c_iึ					ครึ [khrɯ́] (démodé)
c_i + ◌ึ [ɯ] + c_f	c_iึc_f					ถึง [thɯ̌ŋ] (arriver)
c_i + ◌ื [ɯː]					c_iือ	มือ [mɯː] (main)
c_i + ◌ื [ɯː] + c_f	c_iืc_f					มืด [mɯ̂ːt] (nuit)
c_i + ◌ุ [u]	c_iุ					ดุ [dù] (sévère)
c_i + ◌ุ [u] + c_f	c_iุc_f					จุด [cùt] (point)
c_i + ◌ู [uː]	c_iู					รู [ruː] (trou)
c_i + ◌ู [uː] + c_f	c_iูc_f					พูด [phûːt] (parler)
c_i + เ-ะ [e]	เc_iะ					เละ [lé] (décomposé)
c_i + เ-ะ [e] + c_f				เc_i็c_f		เล็ก [lék] (petit)
c_i + เ- [eː]	เc_i					เท [theː] (verser)
c_i + เ- [eː] + c_f	เ$c_i c_f$					เลข [lêːk] (chiffre)
c_i + แ-ะ [ɛ]	แc_iะ					และ [lɛ́] (et)
c_i + แ-ะ [ɛ] + c_f				แc_i็c_f		แข็ง [khɛ̌ŋ] (dur)
c_i + แ- [ɛː]	แc_i					แพ [phɛː] (radeau)
c_i + แ- [ɛː] + c_f	แ$c_i c_f$					แพง [phɛːŋ] (cher)
c_i + โ-ะ [o]	โc_iะ					โต๊ะ [tó] (table)
c_i + โ-ะ [o] + c_f		$c_i c_f$				ตก [tòk] (tomber)
c_i + โ- [oː]	โc_i					โค [khoː] (vache)
c_i + โ- [oː] + c_f	โ$c_i c_f$					โคม [khoːm] (lampe)
c_i + เ-าะ [ɔ]	เc_iาะ					เกาะ [kɔ̀] (île)
c_i + เ-าะ [ɔ] + c_f				เc_i็อc_f		ล็อก [lɔ́k] (verrouiller)
c_i + -อ [ɔː]	c_iอ					ขอ [khɔ̌ː] (demander)
		c_i				บ่ [bɔ̀ː] (non)
c_i + -อ [ɔː] + c_f	c_iอc_f					นอน [nɔːn] (dormir)

c_i + -อ [ɔː] + ร [n]		c_iร				กร [kɔːn] (main)
c_i + เ-อะ [ɣ]	เc_iอะ					เลอะ [lɣ́] (sale)
c_i + เ-อะ [ɣ] + c_f	(*inexistant*)					(*inexistant*)
c_i + เ-อ [ɣː]	เc_iอ					เธอ [thɣː] (toi)
c_i + เ-อ [ɣː] + c_f	เc_iอc_f					เทอม [thɣːm] (session)
				เ$c_i$$c_f$		เกิด [kɣ̀ːt] (naître)
c_i + เ-อ [ɣː] + ย [j]			เc_iย			เลย [lɣːj] (dépasser)
c_i + เ-ีย [ia]	เc_iีย					เดียะ [dìa] (vif)
c_i + เ-ีย [ia] + c_f	(*inexistant*)					(*inexistant*)
c_i + เ-ีย [iaː]	เc_iีย					เสีย [sǐaː] (panne)
c_i + เ-ีย [iaː] + c_f	เc_iียc_f					เสียง [sǐaːŋ] (voix)
c_i + เ-ือะ [ɯa]	เc_iือะ					เกือะ [kɯ̀a] (*pas de sens*)
c_i + เ-ือะ [ɯa] + c_f	(*inexistant*)					(*inexistant*)
c_i + เ-ือ [ɯaː]	เc_iือ					เรือ [rɯaː] (bateau)
c_i + เ-ือ [ɯaː] + c_f	เc_iือc_f					เดือน [dɯaːn] (mois)
c_i + -ัวะ [ua]	c_iัวะ					จั๊วะ [cúa] (très blanc)
c_i + -ัวะ [ua] + c_f				c_iั๊วc_f		ก็วง [kuaŋ] (*pas de sens*)
c_i + -ัว [uaː]	c_iัว					บัว [buaː] (lotus)
c_i + -ัว [uaː] + c_f			c_iวc_f			บวม [buaːm] (gonfler)
c_i + -ำ [am]	c_iำ					ทำ [tham] (faire)
c_i + -ำ [am] + c_f	(*inexistant*)					(*inexistant*)
c_i + ใ- [aj]	ใc_i					ใคร [khraj] (qui)
c_i + ใ- [aj] + c_f	(*inexistant*)					(*inexistant*)
c_i + ไ- [aj]	ไc_i					ไป [paj] (aller)
c_i + ไ- [aj] + ย [j]	ไc_iย					ไทย [thaj] (thaï)
c_i + เ-า [aw]	เc_iา					เรา [raw] (nous)
c_i + เ-า [aw] + c_f	(*inexistant*)					(*inexistant*)

Tableau 2.13 Formes syllabiques avec et sans consonne finale

2.2.3 Ordre de constitution syllabique

Étant donné que certains caractères thaï s'écrivent au-dessus ou au-dessous des autres, une ligne du texte comprend alors 4 niveaux d'écriture. D'après §2.2.1 et §2.2.2, nous constatons qu'une syllabe thaï apparaît sous la forme définie par le schéma ci-dessous (les caractères entre crochets sont facultatifs). En conséquence, contrairement à l'oral, le noyau de la syllabe à l'écrit est la consonne initiale car elle est le seul élément obligatoire.[41]

$$\begin{matrix}[D]\\ [V_{su}]\\ [V_a]C_i[V_p][C_f][M]\\ [V_{so}]\end{matrix}$$

Figure 2.1 Position des caractères d'une syllabe[42]

C_i : Caractère(s) consonantique(s) initial(aux)
C_f : Caractère(s) consonantique(s) final(s)
V_a : Caractère vocalique antéposé
V_p : Caractère(s) vocalique(s) postposé(s)
V_{su} : Caractère vocalique suscrit
V_{so} : Caractère vocalique souscrit
D : Signe diacritique
M : Phonème muet

Contrairement à l'apparence graphique, la représentation informatique d'une syllabe est linéaire et est composée des caractères entrés l'un après l'autre au clavier. Les codes ASCII d'un texte sont enregistrés dans les fichiers ou en mémoire en respectant cet ordre.[43]

$$[V_a]C_i[V_s][D][V_p][C_f][M]$$

Figure 2.2 Représentation informatique d'une syllabe

V_s : Caractère vocalique suscrit ou souscrit[44]

[41] À l'exception des voyelles isolantes qui peuvent apparaître toutes seules.
[42] Le signe diacritique s'écrit au niveau de la voyelle suscrite si cette dernière est absente.
[43] Le programme de gestionnaire de Windows thaï n'autorise pas les frappes du clavier qui ne respectent pas cet ordre.
[44] Ces deux types de caractère vocalique ne coexistent jamais.

Quant au phonème muet, il peut apparaître sous la forme définie par le schéma suivant. Il est à noter qu'une consonne et le signe de phonème muet sont obligatoires.

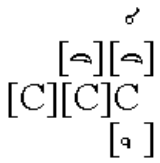

Figure 2.3 Position des caractères d'un phonème muet

C : Caractère consonantique
◌ : Caractère vocalique [i]
◌ : Caractère vocalique [u]
◌ : Signe de phonème muet

La représentation informatique est également linéaire et est organisée de la façon suivante :

[C][C][◌]C([◌]ou[45][◌])

Figure 2.4 Représentation informatique d'un phonème muet

2.3 Mots

Les mots thaï comportent une ou plusieurs syllabes. Linguistiquement parlant, ils sont divisés en mots simples et mots composés. Faute de séparateur de mots, les deux types ont toujours la même apparence, c'est-à-dire que tous les caractères sont collés ensemble. La distinction entre les deux se fait donc par l'analyse de leurs composants.

Étant invariables, les mots thaï n'ont aucune flexion : les verbes ne se conjuguent pas ; les noms et les adjectifs sont également invariables en genre et en nombre.

2.3.1 Mots simples

Un mot simple thaï ne peut pas se diviser en plusieurs monèmes. Même s'il s'agit d'un mot polysyllabique, chaque syllabe n'a aucun rapport avec le mot en question. Ce mot est une unité de sens qui ne peut être déduite des sens de ses syllabes ; par exemple :

[45] Ces deux types de caractère vocalique ne coexistent jamais.

กระถาง [kràthăːŋ] (pot de fleurs) → กระ [krà] (tortue)
ถาง [thăːŋ] (sarcler)
นาฬิกา [naːlíkaː] (horloge) → นา [naː] (champ)
ฬิ [lí] (*pas de sens*)
กา [kaː] (corbeau)

Les deux exemples ci-dessus, ne pouvant pas se diviser, ont chacun un seul monème et sont donc des mots simples. Cependant, la plupart des mots simples thaï sont monosyllabiques, par exemple :

ปลา [plaː] (poisson)
นอน [nɔːn] (dormir)
ดี [diː] (bon)
ไก่ [kàj] (poulet)
ลง [loŋ] (descendre)

Les mots simples polysyllabiques du thaï sont souvent des emprunts ayant des origines différentes telles que pālie, sanskrite, khmère, chinoise, européennes, etc. Par exemple :

ราชินี [raːchíniː] (reine)	(pāli)
ทฤษฎี [trítsàdiː] (théorie)	(sanskrit)
กังวล [kaŋwon] (s'inquiéter)	(khmer)
บะหมี่ [bàmìː] (nouille)	(chinois)
เทนนิส [thennít] (tennis)	(anglais)
โชเฟอร์ [choːfɤ̄ː] (chauffeur)	(français)

Néanmoins, il existe un certain nombre de mots simples polysyllabiques proprement thaï, même s'ils ne sont pas nombreux. Par exemple :

ตลาด [tàlàːt] (marché)
มะพร้าว [máphάːw] (coco)
สะใภ้ [sàpháj] (belle-fille)
กระทะ [kràthá] (poêle)
ทะเล [tháleː] (mer)

2.3.2 Mots composés

Les mots composés thaï comportent plusieurs mots simples (mot base[46] + complément(s)) et sont toujours polysyllabiques. Il existe 2 types de composition :

2.3.2.1 Composition thaïlandaise

Un mot composé est dit de composition thaïlandaise si ce mot comporte un mot base et des compléments modifieurs à sa droite uniquement (ordre progressif). Le mot base est donc le premier composant (à gauche) dans le mot composé. L'interprétation se fait alors de gauche à droite ; par exemple :

เครื่อง [**khrɯ̂a:ŋ**] (**machine**) + บิน [bin] (voler)

→ **เครื่อง**บิน [**khrɯ̂a:ŋ**bin] (avion)

ที่ [**thî:**] (**instrument**) + เปิด [pɣ̀:t] (ouvrir) + ขวด [khùa:t] (bouteille)

→ **ที่**เปิดขวด [**thî:**pɣ̀:tkhùa:t] (décapsuleur)

2.3.2.2 Composition pāli-sanskrite

Un mot composé est dit de composition pāli-sanskrite si ce mot comporte un mot base et des compléments modifieurs à sa gauche uniquement (ordre régressif). Le mot base est donc le dernier composant (à droite) dans le mot composé. L'interprétation se fait alors de droite à gauche. Il y a deux types de mots composés pāli-sanskrits :

2.3.2.2.1 Mots composés accolés

Les mots simples sont juste accolés ensemble. Quant à la prononciation, si le mot à gauche ne se termine pas par une voyelle, il faut prononcer comme s'il se terminait par la voyelle [a]. Par exemple :

รัฐ [rát] (Etat) + **ศาสตร์** [**sà:t**] (**science**)

→ รัฐ**ศาสตร์** [rátthà**sà:t**] (**science** politique)

จักษุ [càksù] (œil) + **แพทย์** [**phɛ̂:t**] (**médecin**)

→ จักษุ**แพทย์** [càksù**phɛ̂:t**] (ophtalmologue)

2.3.2.2.2 Mots composés avec adaptation morphologique

Lorsque le mot base commence par le phonème « อ » [ʔ], on observe parfois une adaptation morphologique entre la première syllabe du mot base et la dernière syllabe du complément (à sa gauche). Par exemple :

[46] Les mots bases dans nos exemples s'écrivent en gras.

ภัตต [pháttà] (nourriture) + **อาคาร** [**ʔaːkhaːn**] (**bâtiment**)

→ ภัตต**าคาร** [phátt**aːkhaːn**] (restaurant)

ราช [raːchá] (roi) + **โอวาท** [**ʔoːwâːt**] (**sermon**)

→ รา**โชวาท** [raːch**oːwâːt**] (**sermon** du roi)

2.3.3 Ambiguïtés de découpage des mots thaï

Faute de séparateurs, les mots thaï sont collés. Cela pose un problème de reconnaissance des mots, d'autant plus que les caractères consonantiques thaï sont ambigus parce qu'ils peuvent être des consonnes initiales, des consonnes finales ou un des éléments d'agglomérats. Nous rencontrons souvent des problèmes de découpage :

ตากลม ($C_1V_pC_2C_3C_4$)

Dans cet exemple, C_1 n'est pas ambiguë parce qu'elle est suivie d'une voyelle postposée, c'est sans doute une consonne initiale. Par contre, C_2, C_3, et C_4 le sont par manque de voyelle explicite. Les possibilités sont, par exemple :

- C_2, C_3, et C_4 sont toutes consonnes initiales formant trois syllabes avec chacune une voyelle de forme supprimée (voyelle [a] ou [ɔː]).
- C_2 est une consonne initiale, C_3 une consonne finale, les deux forment une syllabe avec la voyelle de forme supprimée [o] ; C_4 est la consonne initiale d'une autre syllabe formée avec une voyelle de forme supprimée [a] ou [ɔː].
- C_2 est la consonne initiale d'une syllabe formée avec une voyelle de forme supprimée [a] ou [ɔː] ; C_3 est une consonne initiale, C_4 une consonne finale, les deux dernières forment une autre syllabe avec la voyelle de forme supprimée [o].
- C_2 et C_3 forment l'agglomérat consonantique final de V_p ; C_4 est la consonne initiale d'une autre syllabe formée avec une voyelle de forme supprimée [a] ou [ɔː].
- C_2 et C_3 forment un agglomérat consonantique initial et C_4 est une consonne finale, les deux forment une syllabe avec la voyelle de forme supprimée [o].
- C_2 est la consonne finale de V_p alors que C_3 et C_4 respectivement en tant que consonnes initiale et finale, forment une autre syllabe avec la voyelle de forme supprimée [o].
- Etc.

Nous constatons que les possibilités sont nombreuses. Même si certaines n'ont aucun sens, les deux dernières sont bien réelles comme le montrent les deux analyses ci-dessous. (CC_i représente un agglomérat consonantique initial.)

ตา/กลม (C_iV_p/CC_iC_f) [taː/klom] → œil/rond
ตาก/ลม ($C_iV_pC_f$/C_iC_f) [taːk/lom] → sécher/vent (sécher au vent, prendre l'air)

Une séquence de mots plus longue est logiquement plus ambiguë, par exemple :

มารอกราบ ($C_1V_pC_2C_3C_4C_5V_pC_6$)

Cet exemple présente encore plus de possibilités de découpage, parmi lesquelles trois possèdent des sens. Nous rappelons que la consonne « อ » [ʔ] peut être utilisée en tant que voyelle « -อ » [ɔː].

มา/รอ/กราบ (C_iV_p/C_iV_p/$CC_iV_pC_f$) [maː/rɔː/kràːp] → venir/attendre/se prosterner
มา/รอก/ราบ (C_iV_p/$C_iV_pC_f$/$C_iV_pC_f$) [maː/rɔ̂ːk/râːp] → venir/poulie/plat
มาร/อก/ราบ ($C_iV_pC_f$/C_iC_f/$C_iV_pC_f$) [maːn/ʔòk/râːp] → monstre/poitrine/plat

À ce stade, nous ne pouvons pas déterminer la meilleure segmentation, cela dépend du contexte. Un bon système d'analyse des textes doit pouvoir donner toutes les possibilités de découpage, en attendant d'autres informations qui nous permettront de déterminer la segmentation correcte (*voir §6.5*).

CHAPITRE 3 DICTIONNAIRES ÉLECTRONIQUES DU THAÏ

Les dictionnaires électroniques sont les éléments les plus importants des systèmes d'analyse des textes. Ils recensent et décrivent le vocabulaire de la langue du point de vue morphologique, orthographique, syntaxique et parfois sémantique. Pour décrire le vocabulaire thaï dans INTEX, nous utilisons le même format que d'autres langues, le format DELA développé au LADL. Nous avons commencé par créer un DELAS contenant des mots simples et un DELAC contenant des mots composés.

3.1 Sources lexicales

Nous utilisons comme référence principale le « Dictionnaire de l'Institut Royal[1] 1996 »[2], le seul dictionnaire officiel en Thaïlande actuellement, complété par le « Dictionnaire de l'Étudiant 1991 »[3] et le « Dictionnaire de l'Étudiant 1999 »[4]. Nous avons au total 34 743 entrées classées selon leur partie du discours que nous détaillerons plus loin dans ce chapitre.

3.2 Structure des entrées

Le DELAS et le DELAC du thaï partagent la même structure des entrées, c'est-à-dire que chaque entrée commence par un mot (simple ou composé selon le cas), suivi d'une virgule et des informations lexicales.

---

[1] L'Institut Royal en Thaïlande joue le même rôle que l'Académie Française en France.
[2] ROYAL INSTITUTE 1996a
[3] YANAPRATHIP 1991
[4] HIRANYAKARN 1999

3.2.1 Définition des mots simples et des mots composés

La différence entre mots simples et mots composés dans INTEX est purement formelle. Contrairement à la définition linguistique (*cf. §2.3*), les mots simples et les mots composés du thaï sont définis dans INTEX de la façon suivante :

Les mots simples sont des séquences de lettres (recensées dans le fichier Alphabet du répertoire de la langue thaï) sans séparateur, ni chiffres (arabes ou thaï).

Les mots composés sont des séquences de lettres avec au moins un séparateur.

D'après cette définition, presque tous les mots thaï (34 393 entrées) sont donc des mots simples, faute de séparateur. Seuls les mots s'écrivant obligatoirement avec des signes de ponctuation telles que le signe abréviatif, le signe répétitif ou l'espace blanc, par exemple, sont considérés comme mots composés (350 entrées).

3.2.2 Informations lexicales

Selon le format DELA, les informations lexicales d'une entrée commencent par un code de catégorie grammaticale en lettres majuscules, suivi éventuellement d'un numéro de classe flexionnelle, d'une ou plusieurs informations syntactico-sémantiques introduites par le caractère « + », d'une ou plusieurs séries d'informations flexionnelles introduites par le caractère « : » ou d'un commentaire introduit par le caractère « / ».

3.2.2.1 Catégories grammaticales

L'Institut Royal a classé son vocabulaire en 8 catégories selon leur partie du discours : nom, pronom, verbe, adjectif-verbe, préposition, conjonction, nibat et interjection. Il est à noter que, conformément à la grammaire thaï, l'adjectif et l'adverbe sont classés dans la même catégorie appelée ici « adjectif-verbe » et que l'Institut Royal préfère coder un certain type de conjonctions (plus précisément, les conjonctions à effet stylistique) avec un code séparé, appelé « nibat ».

En plus des 8 parties du discours, l'Institut Royal a aussi recensé un certain nombre d'expressions idiomatiques dans son dictionnaire. Celles qui sont toujours utilisées en tant que noms, verbes ou adjectifs sont codées N, V ou ADJV respectivement. Cependant, celles qui n'ont aucun code spécifique sont codées « EXP » dans nos dictionnaires électroniques.

Le dictionnaire de l'Institut Royal a également recensé un certain nombre de constituants de mots composés. Les constituants pouvant apparaître à gauche des mots bases selon la composition pāli-sanskrite (*cf. §2.3.2.2*) sont codés comme « préfixe » dans nos dictionnaires tandis que les constituants pouvant apparaître à droite des mots bases selon la composition thaïlandaise (*cf. §2.3.2.1*) sont codés comme « suffixe ».

Nous avons donc au total 11 catégories grammaticales. Le tableau suivant montre tous les codes de catégories des mots thaï ainsi que l'effectif de chaque catégorie :

Code	Catégorie	Effectif
N	Nom	18 899
PRO	Pronom	95
V	Verbe	7 883
ADJV	Adjectif-verbe	6 532
PREP	Préposition	42
CONJ	Conjonction	91
NIBAT	Conjonction à effet stylistique	6
INTJ	Interjection	87
EXP	Expression	32
PFX	Préfixe	1 012
SFX	Suffixe	64

Tableau 3.1 Codes des catégories des mots thaï

3.2.2.2 Informations complémentaires sur les pronoms

Nous rappelons que les mots thaï n'ont pas de flexion. En conséquence, le numéro de code flexionnel n'existe pas. Cependant, nous pouvons tout de même coder la personne, le genre et le nombre dans les pronoms. Sachant que ce ne sont pas des informations sur les mots (signifiants) eux-mêmes mais sur leurs valeurs (signifiés). Par exemple, le mot « กระผม » [kràphŏm] ne provoque pas d'accord morphologique, mais comme c'est un pronom personnel de la première personne utilisé par un homme, il est codé « :1ms ».

Les codes suivants sont toujours écrits derrière les deux-points « : ».

Code de pronom	Signification
m	Masculin
f	Féminin
s	Singulier
p	Pluriel
1, 2, 3	1ère, 2ème, 3ème personne

Tableau 3.2 Codes des informations sur les pronoms

3.2.2.3 Informations syntactico-sémantiques

Les termes spécialisés et techniques sont classés en fonction d'informations syntaxiques ou sémantiques ou par domaines d'utilisation. Ces codes sont toujours employés avec le signe « + ».

Code	Signification	Code	Signification
Arch	Archaïque	Idiom	Expression idiomatique
Astrl	Astrologie	Law	Droit
Astrn	Astronomie	Light	Lumière
Aux	Auxiliaire	Math	Mathématiques
Chem	Chimie	Med	Médecine
Collq	Familier	Obs	Obsolète
Dial	Dialecte	Phys	Physique
Econ	Economie	Poet	Poésie
Elec	Electricité	Royal	Terme honorifique ou de cour
Form	Formel	Sci	Science
Geog	Géographie	Stat	Statistique
Gram	Grammaire	Val	Valeur

Tableau 3.3 Codes des informations syntactico-sémantiques

3.2.2.4 Informations dialectologiques

Les mots dialectaux sont divisés en plusieurs sous-catégories selon la région, la province ou la religion. Les codes ci-dessous sont employés avec le signe « + » derrière le code « Dial ».

Code de dialecte	Région	Province	Religion
[E]	Est		
[NE]	Nord-est		
[NW]	Nord-ouest		
[S]	Sud		
[Cb]		Chanthaburi	
[Cm]		Chiangmai	
[Kr]		Khorat	
[Pk]		Phuket	
[Pt]		Pattani	
[Ry]		Rayong	
[Ud]		Udonthani	
[Is]			Islam

Tableau 3.4 Codes des informations dialectologiques

3.3 DELAS du thaï

D'après la définition des mots simples (*cf. §3.2.1*), le DELAS du thaï ne contient que des séquences de lettres sans aucun séparateur.

3.3.1 Entrées de mots uniques

Les mots à entrée unique sont des mots sans homographes, et ne donnant pas lieu dans les dictionnaires classiques à plus d'une entrée. Les exemples ci-dessous illustrent la structure des entrées du DELAS pour ce type de mots :

ก้ง,ADJV+Dial+[NW]

กงเกวียนกำเกวียน,EXP+Idiom

กระผม,PRO:1ms

แกมมา,N+Phys

ไกล่เกลี่ย,V

3.3.2 Entrées de mots homographes

Lorsque plusieurs homographes partagent exactement les mêmes informations lexicales, ils ont une entrée unique, même si sémantiquement, il s'agit de mots différents. Par exemple, le mot « แสง »[sɛ̌ːŋ] qui est un nom avec 4 significations[5] : lumière, bijou, arme et pantalon, n'a qu'une entrée :

แสง,N

En revanche, les homographes ayant des informations lexicales différentes ont chacun une entrée séparée comme l'exemple du mot « ขัน » [khăn] ci-dessous :

ขัน,ADJV	(courageux, drôle)
ขัน,N	(cuvette)
ขัน,V	(tourner, crier, rire)

Par contre, si la différence ne concerne que les informations complémentaires sur les pronoms, alors une seule entrée suffit. Par exemple, le mot « เรา » [raw] est un pronom soit à la première personne du pluriel, soit à la deuxième personne du singulier, il a une entrée :

เรา,PRO:1p:2s

3.3.3 Variantes orthographiques

Un mot peut avoir plusieurs orthographes. En thaï, les variantes orthographiques ne sont pas nombreuses. Elles ont chacune une entrée distincte[6] dans le DELAS. Par exemple, le mot « หม้อห้อม » [mɔ̂ːhɔ̂ːm] (une sorte de chemise traditionnelle) a 3 variantes orthographiques, donc 3 entrées :

ม่อห้อม,N+Dial+[NW]

ม่อฮ่อม,N+Dial+[NW]

หม้อห้อม,N+Dial+[NW]

[5] Selon ROYAL INSTITUTE 1996a

[6] Toutes les variantes du même mot seront ultérieurement reliées dans le DELAF.

3.3.4 Ordre alphabétique

Les mots dans le DELAS sont triés suivant un ordre spécifique du thaï, c'est-à-dire, selon les règles définies par l'Institut Royal :

- Le tri se fait de gauche à droite, caractère par caractère. Ainsi une voyelle composée telle « เ-าะ » [ɔ] est considérée comme une suite de plusieurs caractères vocaliques à trier individuellement.
- Le tri se fait selon les graphies, pas les phonies. Ainsi une syllabe comprenant une voyelle graphiquement supprimée telle « กด » [kòt] est considérée comme une suite de caractères consonantiques sans voyelle.
- Tous les caractères consonantiques (0xA1-0xCE) sont avant tous les caractères vocaliques (0xD0-0xD9 et 0xE0-0xE5) sauf les caractères vocaliques « ฤ » (0xC4) et « ฦ » (0xC6) qui doivent être triés comme si c'étaient des consonnes et qui sont juste après les caractères consonantiques « ร » (0xC3) et « ล » (0xC5) respectivement (*voir Tableau 3.5*).
- Dans une syllabe, le caractère consonantique initial est pris en compte avant les caractères vocaliques. Ainsi les caractères vocaliques antéposés (เ-, แ-, โ-, ใ-, ไ-) (0xE0-0xE4), même s'ils apparaissent à gauche des caractères consonantiques initiaux, doivent être considérés comme s'ils s'écrivaient à droite de ces consonnes. L'exemple suivant montre une liste de mots thaï bien triée :

กก		→		(0xA1/0xA1)
เกก	(0xE0/0xA1/0xA1)	→	(กเก)	(0xA1/0xE0/0xA1)
โกง	(0xE2/0xA1/0xA7)	→	(กโง)	(0xA1/0xE2/0xA7)
ขด		→		(0xA2/0xB4)

- Les signes diacritiques (◌ิ, ◌ี, ◌ึ, ◌ื, ◌ั, ◌็, ◌่, ◌้, ◌๊, ◌๋) (0xE7-0xEC) sont ignorés dans le tri sauf dans le cas où tous les autres caractères des deux mots comparés seraient totalement identiques. L'exemple suivant montre une liste de mots thaï bien triée :

จง	→	(0xA8/0xA7)
จ่ง	→	(0xA8/(0xE8)/0xA7)
จ๋ง	→	(0xA8/(0xEB)/0xA7)
จงกล	→	(0xA8/0xA7/0xA1/0xC5)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
A	NBSP 0xA0	ก	ข	ฃ	ค	ฅ	ฆ	ง	จ	ฉ	ช	ซ	ฌ	ญ	ฎ	ฏ
B	ฐ	ฑ	ฒ	ณ	ด	ต	ถ	ท	ธ	น	บ	ป	ผ	ฝ	พ	ฟ
C	ภ	ม	ย	ร	ฤ	ล	ฦ	ว	ศ	ษ	ส	ห	ฬ	อ	ฮ	ฯ
D	ะ	ั	า	ำ	ิ	ี	ึ	ื	ุ	ู	ฺ					฿
E	เ	แ	โ	ใ	ไ	ๅ	ๆ	็	่	้	๊	๋	์	๎	๏	๐
F	๑	๒	๓	๔	๕	๖	๗	๘	๙	๚	๛	∽				

Tableau 3.5 Table des caractères du thaï

```
#define voyant(c) (0xE0<=(c) && (c)<=0xE4)  //Voyelles antéposées
#define diac(c) (0xE7<=(c) && (c)<=0xEC)    //Signes diacritiques
int thstrcmp (const char *mot1, const char *mot2)
{
        char c1, c2;
        int d = 0;
        for (;;) {
                while (*mot1 == *mot2 && *mot1 != 0) {
                        *mot1++;
                        *mot2++;        //avancer jusqu'on trouve les caractères non-égaux
                }
                c1 = *mot1;
                c2 = *mot2;
                if (diac(c1) && diac(c2)) {   //si les deux sont diacritiques
                        *mot1++;              //sauter-les
                        *mot2++;
                        if (d == 0) d = c1 - c2;      //enregistrer quand même la
                }                                     //différence de diacritiques
                else if (diac(c1)) {
                        *mot1++;
                        if (d == 0) d = 1;    //le mot avec diacritique doit être après
                }                             //le mot sans diacritique
                else if (diac(c2)) {
                        *mot2++;
                        if (d == 0) d = -1;
                }
                else break;
        }
        if (c1 == 0 && c2 == 0) return d;     //fin de 2 mots, retourner
                                              //la différence de diacritiques
        if (c1 == 0 || c2 == 0) return c1 - c2;       //fin d'un mot, retourner la
                                                      //différence de caractères
        if (voyant(c1) && voyant(c2)) {       //si les deux caractères sont voyelles antéposées
                if (mot1[1] != mot2[1])       //considérer les caractères suivants
                        return mot1[1] - mot2[1];
                else
                        return c1 - c2;
        }
        else if (voyant(c1)) {
                if (mot1[1] != c2)
                        return mot1[1] - c2;
                else
                        return 1;
        }
        else if (voyant(c2)) {
                if (c1 != mot2[1])
                        return c1 - mot2[1];
                else
                        return -1;
        }
        else return c1 - c2;          //cas général, retourner la différence des caractères
}
```

Algorithme 3.1 Module de tri des mots thaï

Nous utilisons le module écrit en langage C ci-dessus pour trier les mots thaï dans le DELAS. Le programme principal doit appeler ce module en lui envoyant 2 mots à comparer. Ce module lui retournera une valeur négative si le mot1 doit être avant le mot2, une valeur positive dans le cas contraire et zéro si les deux mots sont entièrement identiques.

Les 34 393 entrées du DELAS thaï sont présentées dans l'Annexe II.A.

3.4 DELAF du thaï

Dans les langues flexionnelles, le DELAF contient la totalité des formes fléchies et conjuguées issues de la base des mots du DELAS. Toutes les formes fléchies ou conjuguées sont reliées à leur forme canonique. Pour le thaï, grâce à son invariabilité, les entrées du DELAF sont identiques à celles du DELAS. Les formes canoniques sont en principe ignorées.

3.4.1 Structure des entrées

La structure des entrées du DELAF thaï est simple. Elle commence par un mot simple suivi d'une virgule, d'un point et ensuite, des informations lexicales. Pour créer un DELAF à partir d'un DELAS, il suffit de modifier les entrées du DELAS en ajoutant un point entre la virgule et les informations lexicales. Prenons l'exemple du §3.3.1, nous le modifions en :

กงเกวียนกำเกวียน,.EXP+Idiom
กระผม,.PRO:1ms
กั้ง,.ADJV+Dial+[NW]
แกมมา,.N+Phys
ไกล่เกลี่ย,.V

3.4.2 Ordre alphabétique

L'ordre alphabétique dans le DELAF thaï n'est pas similaire à celui du DELAS. En effet, INTEX utilise un module de tri unique[7] pour toutes les langues qui trie les entrées selon l'ordre défini dans le fichier Alphabet. Cette méthode n'est pas conforme aux règles de tri thaï, mais afin de garder la compatibilité avec d'autres langues, le module de tri thaï n'est pas pour l'instant intégré dans INTEX. En conséquence, les entrées du DELAF thaï sont triées de gauche à droite en fonction des codes ASCII des caractères sans tenir compte des cas exceptionnels. Prenons les exemples du §3.3.4, nous obtenons une liste triée par INTEX comme suit :

[7] Pour trier un dictionnaire dans INTEX, il suffit de choisir l'option « Sort Dictionary » dans le menu « DELA ».

กก → (0xA1/0xA1)
ขด → (0xA2/0xB4)
จง → (0xA8/0xA7)
จงกล → (0xA8/0xA7/0xA1/0xC5)
จ่ง → (0xA8/0xE8/0xA7)
จ๋ง → (0xA8/0xEB/0xA7)
เกก → (0xE0/0xA1/0xA1)
โกง → (0xE2/0xA1/0xA7)

3.4.3 Liens entre variantes orthographiques

Etant donné que le champ réservé pour la forme canonique de chaque entrée du DELAF est libre, nous pouvons en profiter pour relier les variantes orthographiques avec leur forme standard en mettant cette dernière comme un lemme. Prenons l'exemple du §3.3.3, nous relions les variantes orthographiques comme suit :

ม่อห้อม,หม้อห้อม.N+Dial+[NW]
ม่อฮ่อม,หม้อห้อม.N+Dial+[NW]
หม้อห้อม,.N+Dial+[NW]

L'avantage de cette méthode est que nous pouvons désormais rechercher toutes les variantes orthographiques dans un texte en écrivant un seul motif, c'est-à-dire en mettant la forme standard entre angles : <หม้อห้อม> trouvera les trois variantes ci-dessus.

Toutes les variantes orthographiques du DELAF thaï sont présentées dans l'Annexe II.B.

3.4.4 Dictionnaire des chiffres thaï

Comme un DELAF peut être créé à partir d'un graphe (*cf. §1.2.3.2*), voici un transducteur qui reconnaît les chiffres thaï et leur associe des informations lexicales : la catégorie grammaticale <TNB> et les valeurs équivalentes en chiffres arabes.

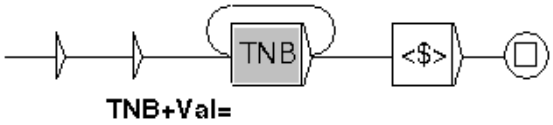

Figure 3.1 DELAF de chiffres thaï

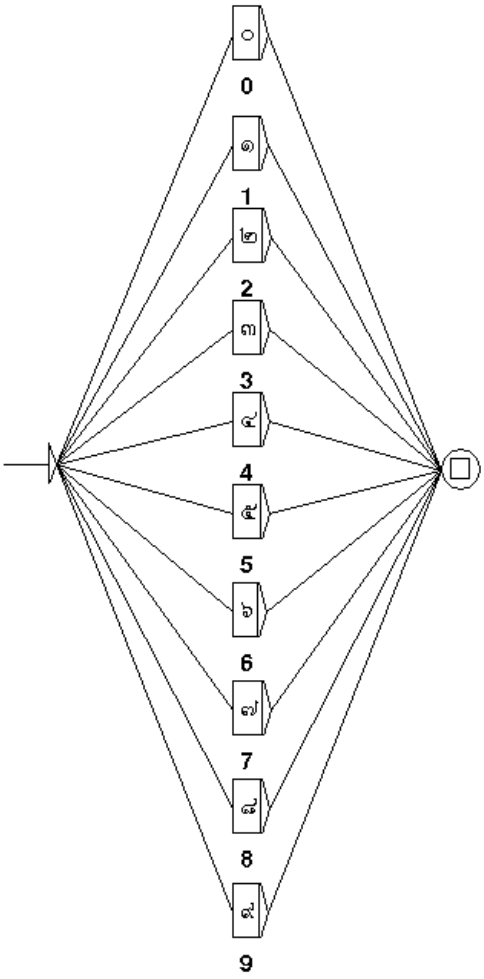

Figure 3.2 Sous-graphe « TNB.grf »

Les graphes du DELAF n'acceptent que les caractères définis en tant que « lettres ». C'est la raison pour laquelle nous devons définir les chiffres thaï dans le fichier Alphabet du thaï (*cf. §1.4.3.2*) afin de pouvoir les écrire dans le graphe « TNB » ci-dessus.

Après avoir compilé ces graphes en un fichier « .fst » et avoir mis ce dernier dans le répertoire « Delaf », il devient un dictionnaire DELAF directement apte à fonctionner et dont les entrées sont du type ci-dessous :

๑๒๓,๑๒๓.TNB+Val=123

๔๕๖๗๘๙๐,๔๕๖๗๘๙๐.TNB+Val=4567890

Créer le DELAF de chiffres thaï sous la forme d'un graphe présente un avantage considérable par rapport à la création avec un éditeur de texte. En effet, les séquences de chiffres sont productives : pour créer un dictionnaire-liste capable de reconnaître les chiffres de 1 à 1 000 000, par exemple, il nous faudrait taper un million d'entrées avec l'éditeur de texte tandis que les deux transducteurs montrés ci-dessus contiennent les mêmes informations.

3.5 DELAC du thaï

Le DELAC du thaï est analogue au DELAS sauf que ses entrées comprennent au moins un séparateur.

3.5.1 Structure des entrées

La structure des entrées du DELAC commence par un mot composé, suivi d'une virgule et des informations lexicales. Par exemple :

กะดอกๆ,ADJV+Dial+[S]
ขิงก็รา ข่าก็แรง,EXP+Idiom
จดๆ จ้องๆ,V
ดำๆ แดงๆ,N+Idiom
ฮึยๆ,INTJ

3.5.2 Ordre alphabétique

Nous préférons trier les entrées du DELAC selon les règles thaïlandaises expliquées au §3.3.4. Néanmoins, le module de tri des mots thaï (*cf. Algorithme 3.1*) ne tient pas compte des séparateurs, il peut ne pas placer les mots composés dans le bon ordre. Par exemple, le signe abréviatif (0xCF) et le signe répétitif (0xE6), qui peuvent apparaître dans les entrées, ont des codes ASCII plus grands que ceux des caractères consonantiques et des caractères vocaliques respectivement (*voir Tableau 3.5*). En revanche, pour le tri, ils doivent être pris en compte comme des signes diacritiques. Afin d'appliquer cette règle, il suffit de remplacer la deuxième ligne de l'Algorithme 3.1 par la ligne suivante :

```
#define diac(c) (c)==0xCF || (0xE6<=(c) && (c)<=0xEC)
```

Les 350 entrées du DELAC thaï sont triées et sont présentées dans l'Annexe II.C.

3.6 DELACF du thaï

Le DELACF du thaï est analogue au DELAF sauf que ses entrées comprennent au moins un séparateur comme celles du DELAC.

3.6.1 Structure des entrées

La structure des entrées du DELACF commence par un mot composé, suivi d'une virgule, d'un point et ensuite, des informations lexicales. Puisqu'il n'y a pas de forme canonique, ni de variantes orthographiques dans le DELACF thaï, le champ pour le lemme est

abandonné. En conséquence, pour créer un DELACF à partir d'un DELAC, il suffit d'ajouter un point entre la virgule et les informations lexicales. Prenons l'exemple du §3.5.1, nous créons un DELACF comme suit :

กะดอกๆ,.ADJV+Dial+[S]
ขิงก็รา ข่าก็แรง,.EXP+Idiom
จดๆ จ้องๆ,.V
ดำๆ แดงๆ,.N+Idiom
ฮึยๆ,.INTJ

3.6.2 Ordre alphabétique

À cause du problème de la compatibilité, nous continuons à utiliser la méthode de tri proposée par INTEX pour trier le DELACF thaï comme dans le cas du DELAF, c'est-à-dire, le tri de gauche à droite par les codes ASCII des caractères sans tenir compte des exceptions.

3.7 Problème d'application des dictionnaires

Les DELAF et DELACF que nous avons créés pour le thaï ne peuvent pas directement servir. En effet, afin de pouvoir appliquer les dictionnaires, les entrées du DELAF doivent correspondre à des lexèmes du texte, et les entrées du DELACF à des séquences de lexèmes, comme c'est le cas en français ci-dessous :

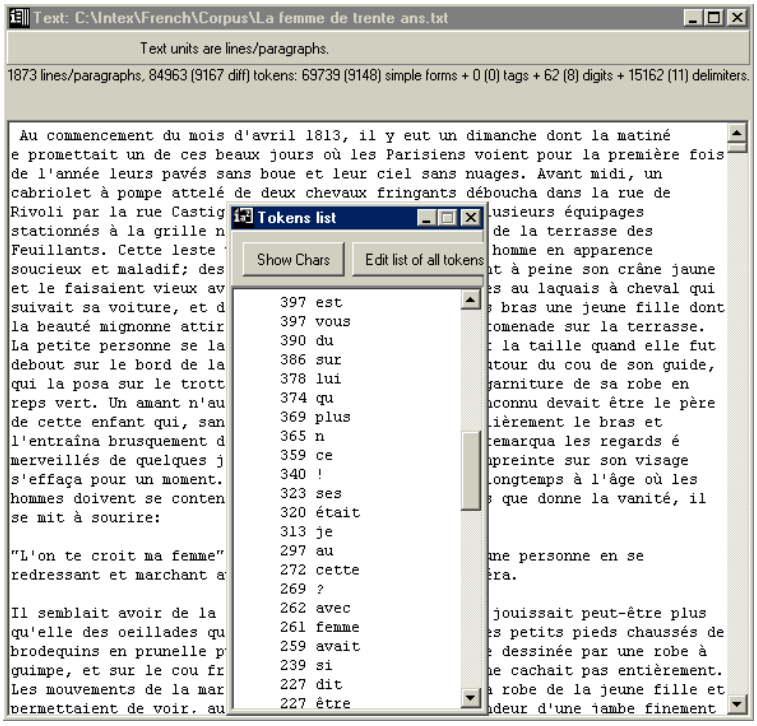

Figure 3.3 Liste de lexèmes provenant d'un texte français

Faute de séparateur de mots, les lexèmes délimités dans un texte thaï sont souvent des phrases ou des syntagmes et peuvent également être des séquences de mots ou des mots simples comme illustré dans la figure suivante. Ceci empêche l'application directe de nos dictionnaires DELAF et DELACF.

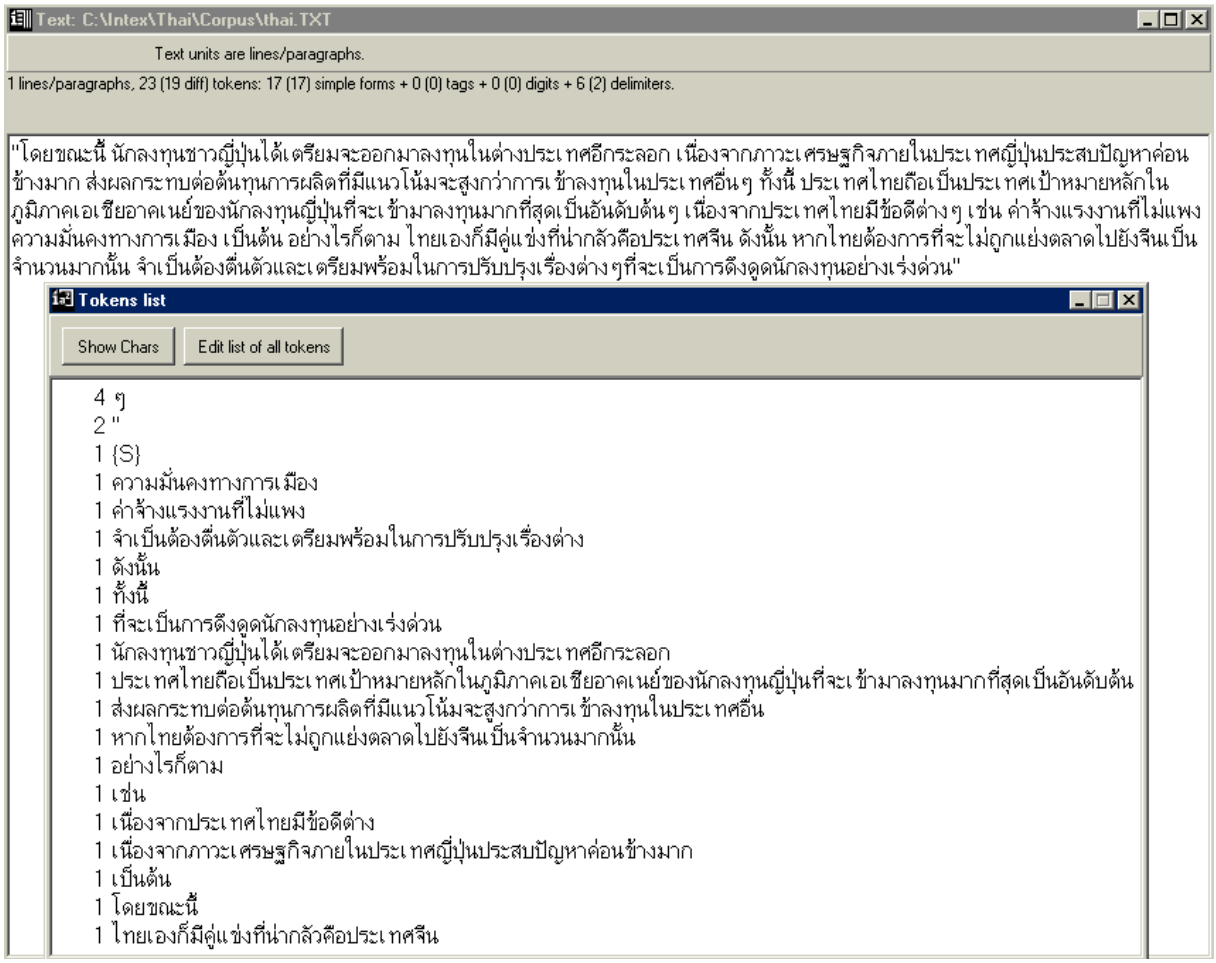

Figure 3.4 Liste de lexèmes provenant d'un texte thaï

Pour remédier à ce problème, il existe 3 méthodes :

- Ajouter des entrées dans les dictionnaires pour qu'ils couvrent toutes les formes pouvant apparaître comme lexèmes dans le texte. Cette méthode n'est pas du tout réalisable parce que les formes se combinent à l'infini et il est pratiquement impossible de les prévoir toutes.
- Découper les textes en lexèmes plus nombreux et plus petits correspondant aux entrées dans les dictionnaires. Cette méthode est la solution idéale car en faisant cela, les textes thaï ressembleraient aux textes français, ce qui autoriserait l'application directe des dictionnaires DELAF et DELACF comme le montre la

figure ci-dessous. Malheureusement, à cause des ambiguïtés de découpage mentionnées au §2.3.3, la segmentation automatique n'est pas entièrement réalisable, ce qui veut dire que les découpages corrects de mots demandent toujours l'aide d'opérations manuelles. Ces dernières sont de moins en moins faisables quand les textes deviennent de plus en plus grands.

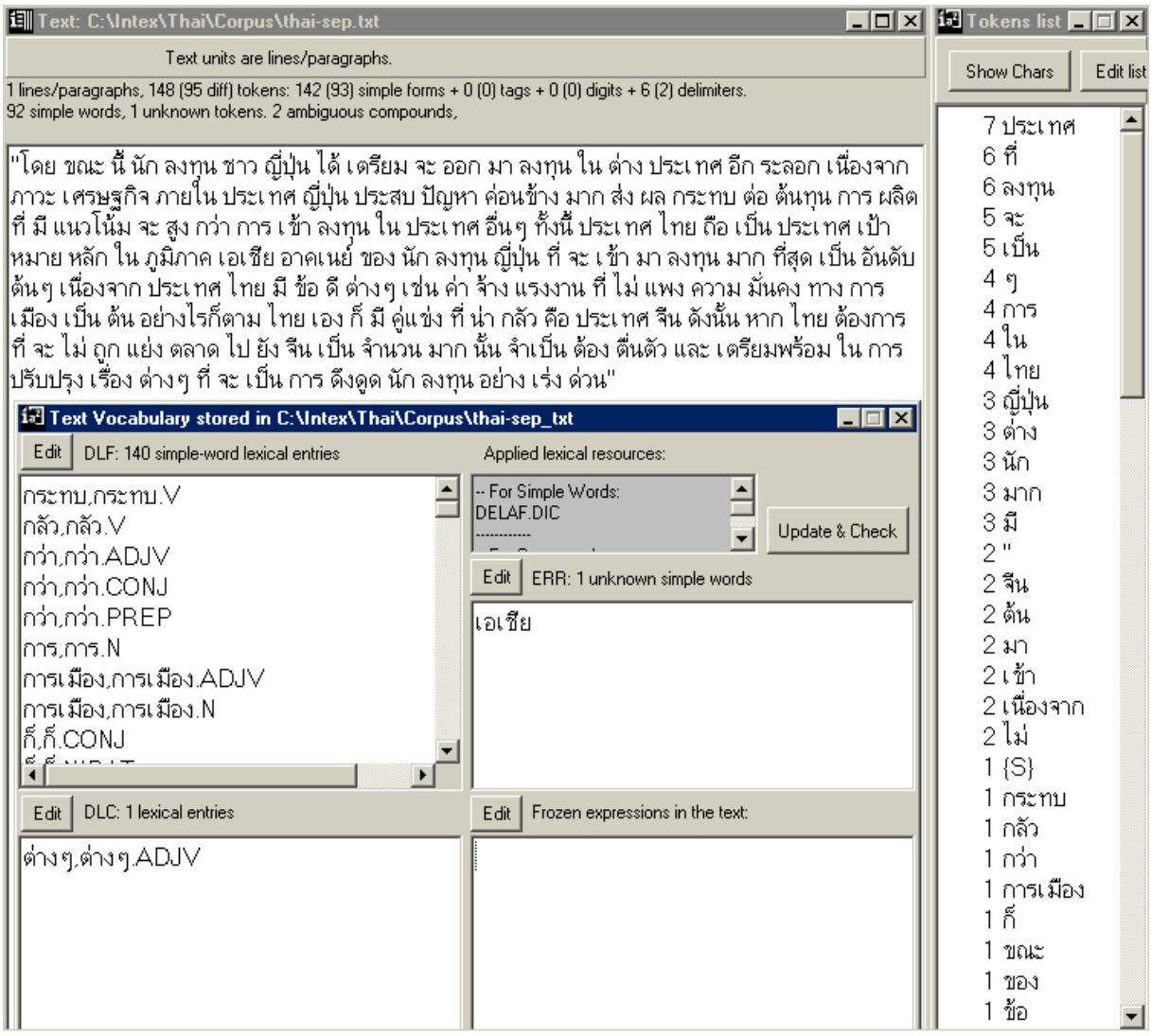

Figure 3.5 Application des dictionnaires sur un texte thaï manuellement découpé

- Modifier les textes et les dictionnaires de la même façon afin que les lexèmes soient les mêmes dans les textes et dans les entrées des dictionnaires, ce qui nous permettra ensuite de pouvoir appliquer les dictionnaires sur les textes. Cette méthode est un compromis entre les deux premières. Elle demande un travail des deux côtés (dictionnaires et textes). Par contre, elle est tout à fait réalisable.

Nous avons choisi la troisième méthode et nous développerons ce concept dans le prochain chapitre.

CHAPITRE 4 MÉTHODES DE SEGMENTATION

Le fait que le thaï soit une langue sans séparateur implique deux niveaux de segmentation : segmentation en mots et segmentation en phrases.

4.1 Segmentation en mots

En général, l'objectif de la segmentation en mots est de permettre au système de reconnaître les mots simples et composés dans les textes au moyen des dictionnaires. Cela est pour l'instant impossible du fait que l'application des dictionnaires d'INTEX ne fonctionne pas par caractère mais par lexème, ce qui veut dire que pour reconnaître un mot simple, il faut qu'il corresponde à la fois à un lexème et à une entrée du DELAF. Cette limitation fait obstacle au traitement des langues sans séparateur de mots telles que le chinois, le japonais et le thaï. Pour résoudre ce problème, nous avons choisi de segmenter les textes en lexèmes plus petits et de segmenter de la même façon les entrées des dictionnaires, afin que ces dernières aient les mêmes formes que les lexèmes des textes. Après cela, INTEX sera capable d'appliquer les dictionnaires sur les textes.

4.1.1 Méthode par caractères

C'est la méthode la plus simple car elle ne réclame aucune connaissance en linguistique.

4.1.1.1 Principe

En consultant le dictionnaire de mots composés du français, INTEX est capable de reconnaître les mots « pomme de terre » et « terre cuite » dans la séquence de lexèmes « pomme de terre cuite » et de les lister dans le vocabulaire du texte, comme suit :

```
pomme de terre,pomme de terre.N+NDN+Conc:fs/une;Comest
terre cuite,terre cuite.N+NA+Conc+z1:fs/une;de&la
```

La procédure de reconnaissance des mots composés en français fonctionne par lexèmes de telle manière que plusieurs lexèmes qui se suivent peuvent former un mot composé. Si l'on considère que chaque lexème ne contient qu'un seul caractère et que plusieurs lexèmes (d'un caractère) qui se suivent peuvent former un mot, INTEX est également capable de reconnaître les mots thaï. Pour cela, il suffit d'ajouter (automatiquement) un espace blanc entre chaque caractère thaï dans les textes ainsi que dans toutes les entrées des dictionnaires et de placer ces derniers dans le répertoire des dictionnaires de mots composés d'INTEX. Désormais, INTEX considère que tous les mots thaï sont des mots composés et que chaque caractère thaï est un mot simple.

En analysant la forme syllabique du thaï (*cf. Figure 2.2*), elle sera transformée de la manière suivante :

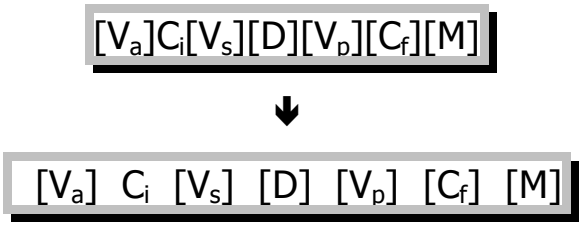

Figure 4.1 Forme syllabique modifiée par la méthode par caractères

V_a : Caractère vocalique antéposé
C_i : Caractère consonantique initial
V_s : Caractère vocalique suscrit ou souscrit
D : Signe diacritique
V_p : Caractère vocalique postposé
C_f : Caractère consonantique final
M : Phonème muet

Le phonème muet (*cf. Figure 2.4*) sera transformé de la même façon :

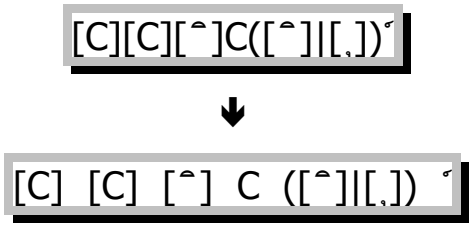

Figure 4.2 Phonème muet modifié par la méthode par caractères

C : Caractère consonantique
◌̂ : Caractère vocalique [i]
◌̯ : Caractère vocalique [u]
ˀ : Signe de phonème muet

4.1.1.2 Algorithme

La modification du texte et du dictionnaire sera facilement réalisée au moyen d'expressions rationnelles. Malheureusement, celles d'INTEX ne fonctionnent pas en mode caractère (*cf. §1.1.1*) et nous sommes donc obligé d'utiliser un programme externe pour effectuer ce travail. Nous avons choisi le langage PERL pour sa simplicité et son efficacité dans la manipulation d'expressions rationnelles, surtout en mode caractère, et nous avons implémenté le programme suivant :

```
$a = $ARGV[0]; #Fichier à segmenter
$b = $ARGV[1]; #Grammaire
$c = $ARGV[2]; #Fichier segmenté
#Ouvrir le fichier texte en lecture <
open(IN,"<$a")||die("Erreur à l'ouverture du Fichier à segmenter");
#Ouvrir la grammaire en lecture
open(GRAM,"<$b")||die("Erreur à l'ouverture de la Grammaire");
#Ouvrir un fichier en écriture >
open(OUT,">$c")||die("Erreur à l'ouverture du Fichier résultat");
#Lire la grammaire
while(<GRAM>){
        push(@gram,$_);
}
#Segmenter le texte
while(<IN>){
        $texte = $_;
        foreach $regle(@gram){
                chomp($regle);
                ($nb,$avant,$apres) = split(/\#/,$regle);
                print STDOUT "$avant---$apres\n";
                eval("\$texte =~ s/$avant/$apres/g");
        }
        print OUT "$texte";
}
close(IN);
close(GRAM);
close(OUT);
```

Algorithme 4.1 Programme de modification des textes « PrepareText.pl »

On exécute le programme ci-dessus en lui donnant 3 arguments : le premier est le fichier texte à segmenter, le deuxième, le fichier de règles à appliquer et le dernier, le fichier de résultat. L'idée de placer les règles dans un fichier distinct nous permet de les modifier à volonté sans toucher le programme principal. Les règles de modification seront appliquées l'une après l'autre dans l'ordre suivant :

```
R1#[ \t]+#
R2#([\([\{]) #$1
R3# (?=[ๆ\)\]\}\n])#
R4#([ก-ฮะ-ๅเ-็่-๋์]) (?=[ก-ฮะ-ๅเ-็่-๋์])#$1
R5#([\wก-ฮะ-ๅเ-็่-๋์๐-๙])(?=[ก-ฮะ-ๅเ-็่-๋์])#$1
R6#([ก-ฮะ-ๅเ-็่-๋์])(?=[\wก-ฮะ-ๅเ-็่-๋์๐-๙])#$1
```

Algorithme 4.2 Règles de modification des textes et des dictionnaires « GramText.txt »

Chaque règle est divisée par le signe « # » en 3 parties : le numéro de la règle, le motif à rechercher et le motif de remplacement. Les séquences correspondant au motif décrit dans la deuxième partie seront remplacées par le motif décrit dans la troisième partie.

Il est à noter qu'il n'est pas nécessaire d'ajouter un blanc à tous les caractères du fait que seules les séquences de caractères thaï posent le problème. Nous profitons par la même occasion de ce programme pour nettoyer les textes en supprimant les caractères superflus.

- **Règle N° 1** : Nettoyer le texte en remplaçant une suite de caractères d'espacement[1] par un seul espace blanc.
- **Règles N° 2 et 3** : Supprimer les espaces blancs derrière les parenthèses ouvrantes « (, [, { », devant les parenthèses fermantes «),], } », devant le signe répétitif « ๆ » et devant la nouvelle ligne[2] parce qu'ils sont superflus.
- **Règle N° 4** : Remplacer l'espace blanc (0x20) entre deux caractères thaï[3] par un blanc insécable (0xA0) ; cela permet de ne pas mélanger les blancs originaux du texte avec les blancs artificiels que nous allons ajouter dans les étapes suivantes.
- **Règles N° 5 et 6** : Ajouter un espace blanc (0x20) entre deux caractères dont l'un est caractère thaï ; l'autre caractère peut être un caractère thaï, un chiffre thaï, un caractère latin ou un chiffre arabe[4]. Ces règles permettent de ne segmenter que les séquences de caractères thaï. Ni les séquences de caractères latins, ni celles de chiffres (arabes ou thaï) ne seront modifiées parce qu'elles ne posent aucun problème lors de la reconnaissance des mots par dictionnaires.

4.1.1.3 Corpus P0

Après avoir appliqué cet algorithme, le texte thaï devient une suite alternée de caractères thaï et de blancs. Nous appelons ce type de texte « Corpus P0 » et nous les analysons uniquement avec les « Dictionnaires P0 », dictionnaires modifiés par la même méthode.

[1] Un caractère d'espacement est soit un espace blanc, soit une tabulation « \t »
[2] En PERL, le symbole « \n » représente une nouvelle ligne, autrement dit, un nouveau paragraphe.
[3] Les caractères thaï dans ce cas sont les caractères consonantiques, les caractères vocaliques, les signes diacritiques et les signes pāli-sanskrits (*cf. §2.1*).
[4] En PERL, le symbole « \w » représente un caractère parmi les caractères latins « A – Z », « a – z », les chiffres arabes « 0-9 » et le caractère « _ ».

4.1.1.3.1 Lexèmes du Corpus P0

Dans ce type de corpus, chacun des lexèmes du texte contient un seul caractère, ce qui fait que le nombre de formes simples comptées par le programme égale le nombre de lettres dans le texte. Par exemple, le texte ci-dessous est effectivement composé de 142 mots simples. La méthode par caractères a fait passer ce nombre à 608, soit 328%[5] d'augmentation.

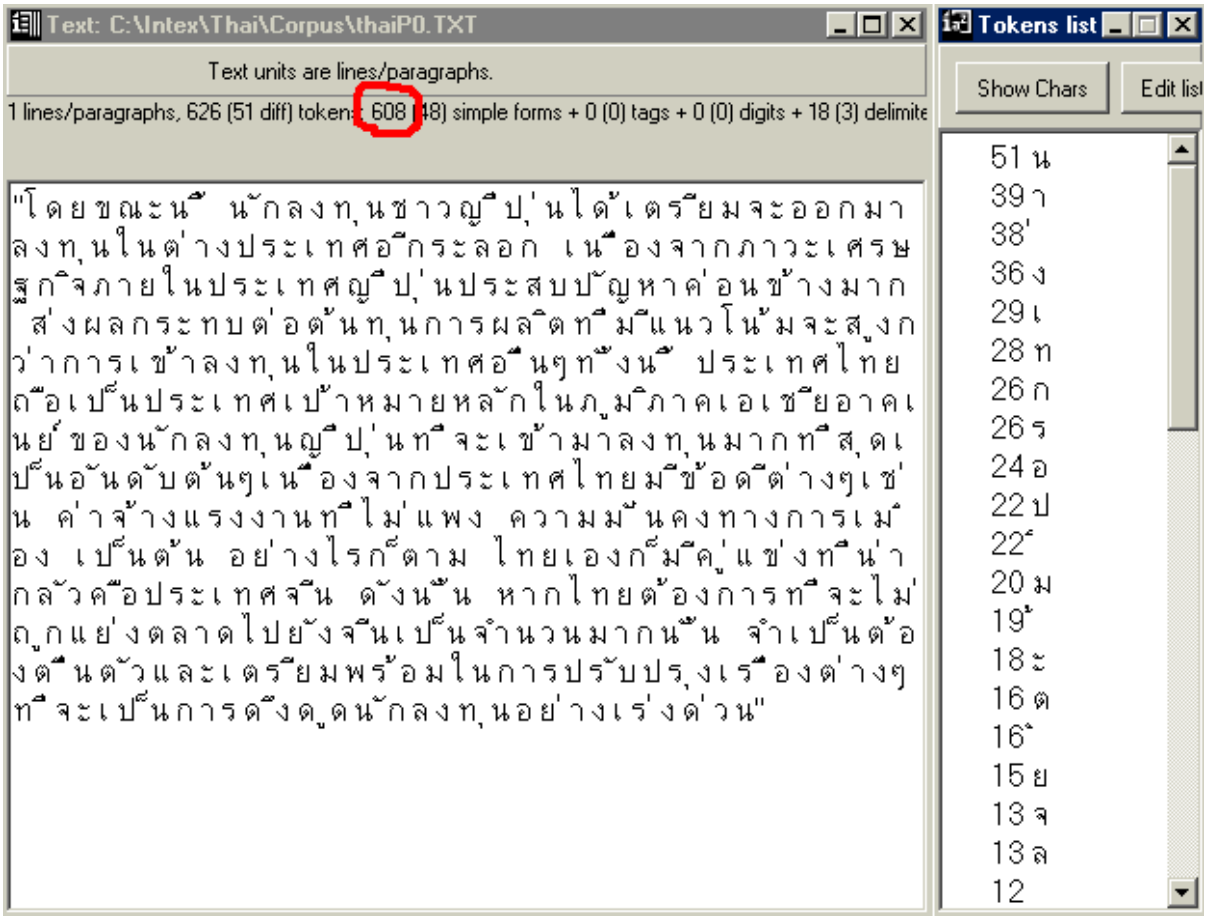

Figure 4.3 Corpus P0 et ses lexèmes

4.1.1.3.2 Cacher les espaces blancs

Grâce au caractère caché défini dans le fichier Alphabet (*cf. §1.4.3.2*), nous pouvons ne pas afficher les espaces blancs dans le corpus en désactivant l'option « Display tags ». On retrouve ainsi la forme originale du texte. Mais pour INTEX, chaque lexème contient toujours un seul caractère comme montré dans la figure suivante, et les blancs affichés dans le texte ne sont pas les espaces blancs (0x20) mais les blancs insécables (0xA0) (*cf. Règle N° 4 au §4.1.1.2*).

---

[5] Le taux de changement est calculé par : $\left(\frac{\textit{Nouvelle valeur} - \textit{Ancienne valeur}}{\textit{Ancienne valeur}}\right) \times 100\ \%$

Dans le cas ci-dessus, le taux d'augmentation = (608-142)x100/142 = 328,17%

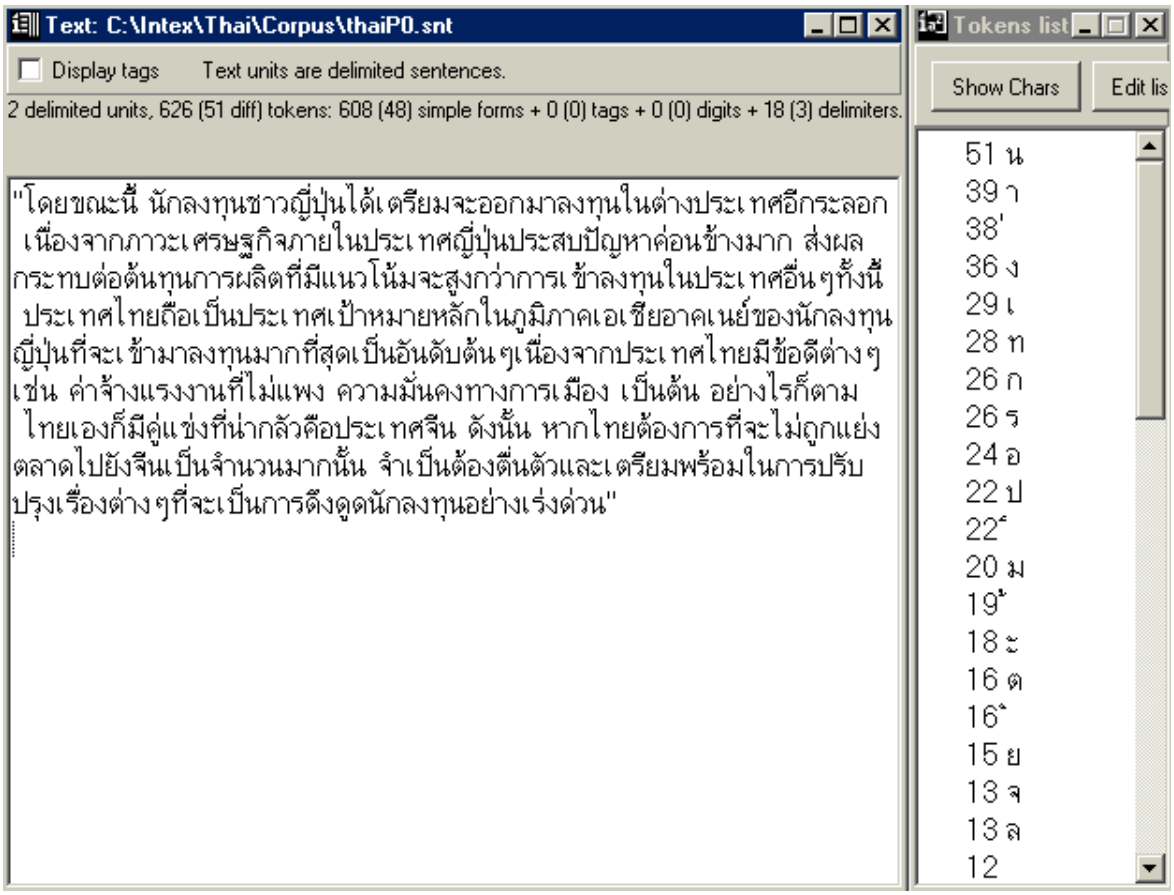

Figure 4.4 Corpus P0 avec les espaces blancs cachés

4.1.1.4 Dictionnaires P0

Afin que les entrées des dictionnaires correspondent aux lexèmes du Corpus P0, nous modifions les dictionnaires de la même façon que le corpus. Nous donnons ci-dessous un extrait du dictionnaire DELAF thaï modifié (version P0). Nous remarquons que seules les séquences de caractères thaï sont segmentées (*cf. Règles N° 5 et 6 au §4.1.1.2*) :

ขึ้ น ชี' อ,.V
ค อ ย ล์,.N
จุ ฬ า,.N
ฉ ก า จ ฉ ก ร ร จ์,.ADJV
ช า ติ พั น ธุ์,.N
ช' ว ง สิ ท ธิ์,.V+Law
อั ฐิ,.N
เ ก ร า ะ,.N
เ จ ริ ญ,.V
เ สิ ร์ จ,.N
ใ ค ร,.PRO

4.1.1.4.1 Nombre de lexèmes

Le nombre total de lexèmes des Dictionnaires P0 s'élève à 226 977 à comparer à 34 862 lexèmes dans les DELAS et DELAC, soit 551%[6] d'augmentation, due à la méthode par caractères.

4.1.1.4.2 Transfert du DELAF au DELACF

Après avoir inséré les espaces blancs dans les entrées des dictionnaires, les mots simples deviennent des mots composés et doivent être transférés dans le dossier DELACF. Seules les 6 entrées suivantes contiennent un caractère unique et restent dans le DELAF :

ฉ,.ADJV

ฌ,.PREP

ธ,.PRO+Poet:3s

น,.N

บ,.ADJV

ฤ,.ADJV

Le DELAF P0 compte ainsi 6 entrées et le DELACF P0, 34 737 entrées.

4.1.1.4.3 Lemmes

Nous profitons du champ libre des formes canoniques pour y mettre la forme originale des mots segmentés. Cela nous permet d'utiliser les symboles lexicaux de manière naturelle (e.g. : <จุฬา>) sans devoir taper des blancs.

ขึ้ น ชื่ อ,ขึ้นชื่อ.V

ค อ ย ล์,คอยล์.N

จุ ฬ า,จุฬา.N

ฉ ก า จ ฉ ก ร ร จ์,ฉกาจฉกรรจ์.ADJV

ช า ติ พั น ธุ์,ชาติพันธุ์.N

ช่ ว ง สิ ท ธิ์,ช่วงสิทธิ์.V+Law

อั ฐิ,อัฐิ.N

เ ก ร า ะ,เกราะ.N

เ จ ริ ญ,เจริญ.V

เ สิ ร์ จ,เสิร์จ.N

ใ ค ร,ใคร.PRO

[6] (226977-34862)x100/34862 = 551,07%

Les variantes orthographiques sont toujours reliées à leur forme standard (*cf. §3.4.3*) écrite sans blancs :

ม' อ ห้ อ ม,หม้อห้อม.N+Dial+[NW]
ม' อ ฮ' อ ม,หม้อห้อม.N+Dial+[NW]
ห ม้ อ ห้ อ ม,หม้อห้อม.N+Dial+[NW]

4.1.1.5 Application des Dictionnaires P0

Maintenant, nous pouvons appliquer les Dictionnaires P0 sur le Corpus P0 pour reconnaître les mots thaï et obtenir le vocabulaire du texte comme la figure ci-dessous. Tous les mots contenant plus d'un caractère sont reconnus en tant que mots composés : par exemple, dans notre corpus d'essai, nous reconnaissons seulement 3 mots simples contre 403 mots composés. Notre corpus d'essai contient en fait 142 mots qui sont tous reconnus par nos dictionnaires, c'est-à-dire que le taux de silence est de 0%. En revanche, le taux de bruit s'élève à 65%[7] par rapport au nombre total de mots reconnus.

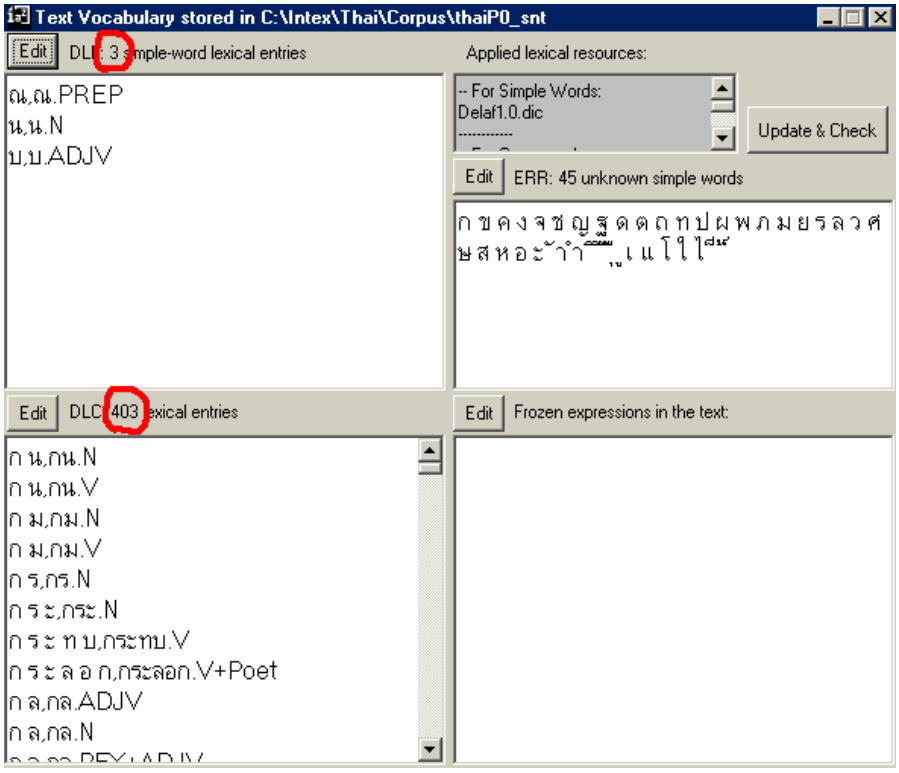

Figure 4.5 Vocabulaire du texte d'un Corpus P0

[7] Le taux de bruit est calculé par : $\left(\frac{Nombre\ de\ mots\ reconnus - Nombre\ de\ mots\ exacts}{Nombre\ de\ mots\ reconnus}\right) \times 100\ \%$

Dans le cas ci-dessus, le taux de bruit = (406-142)x100/406 = 65,02%

4.1.1.6 Problème du bruit

Même si la méthode par caractères nous permet de reconnaître tous les mots simples et composés du thaï, elle reconnaît également un certain nombre de mots orthographiquement conformes aux entrées des dictionnaires mais qui n'existent pas réellement dans le corpus. Ces mots-là sont considérés comme du « bruit ». Par exemple, la séquence « จะเข้ามา » est composée des 3 mots simples « จะ », « เข้า » et « มา ». Avec la méthode par caractères, cette séquence est transformée en « จ ะ เ ข ้ า ม า », ce qui permet aux dictionnaires de reconnaître, à l'intérieur de la séquence, les mots suivants :

ข ้ า,ข้า.N
ข ้ า,ข้า.PRO:1s
ข ้ า ม,ข้าม.ADJV
ข ้ า ม,ข้าม.V
จ ะ,จะ.V+Aux ⇨ *segmentation correcte*
จ ะ เ ข ้,จะเข้.N
ม า,มา.ADJV
ม า,มา.N } *segmentation correcte*
ม า,มา.V
เ ข,เข.ADJV
เ ข ้,เข้.N+Dial+[S]
เ ข ้ า,เข้า.ADJV
เ ข ้ า,เข้า.N+Arch } *segmentation correcte*
เ ข ้ า,เข้า.V

Dans cet exemple, INTEX reconnaît 14 mots orthographiquement conformes aux entrées des dictionnaires alors qu'effectivement, il n'en existe que 3 ; les 11 autres sont du bruit : 4 au niveau lexical (mots à plusieurs parties de discours) et 7 au niveau de la segmentation qui ont deux origines suivantes :

4.1.1.6.1 Détection de parties des mots

Nous pouvons fréquemment trouver, avec la méthode par caractères, des mots inclus dans d'autres. Par exemple, dans le mot « anaconda », INTEX peut détecter avec cette méthode les mots suivants :

```
a,.N+z1:ms:mp
a,.XI+z1
a,avoir.V+z1:P3s
```

```
acon,.N+z3:ms
an,.N+z1:ms
ana,.N+z3:ms:mp
anaconda,.N+z3:ms
c,.N+z1:ms:mp
co,.PFX+z1
con,.A+z1:ms
con,.N+z1:ms
con,.XI+z1
d,.N+z1:ms:mp
da,.INTJ+z3
n,.N+z1:ms:mp
na,.INTJ+z1
o,.N+z1:ms:mp
on,.PRO+z1:3s
on,.XI+z1
onda,onder.V+z3:J3s
```

4.1.1.6.2 Détection de séquences enjambant plusieurs mots

Dans les langues sans séparateur, l'enchaînement des mots crée du bruit. Par exemple, quand les mots « un » et « anaconda » sont enchaînés (« unanaconda »), ils sont ensuite transformés par la méthode par caractères en « u n a n a c o n d a », ce qui permet à INTEX de reconnaître à tort, en raison de l'enchaînement, les mots suivants :

```
n a,na.INTJ+z1
n a n a,nana.N+z1:fs
```

4.1.1.7 Réorganisation des dictionnaires

Le bruit pourrait être réduit en réorganisant nos dictionnaires.

4.1.1.7.1 Séparation des dictionnaires d'affixes

Afin de réduire le bruit provenant de la détection de parties des mots, nous retirons les préfixes et les suffixes de nos Dictionnaires P0 et les plaçons dans deux dictionnaires séparés « Prefix-P0.dic » (1 012 entrées) et « Suffix-P0.dic » (64 entrées). Cette séparation nous permet de travailler plus efficacement en analyse lexicale sans être trop dérangé par le bruit et nous n'appliquons les dictionnaires d'affixes que lorsque nous voulons vraiment faire de l'analyse morphologique (*voir §6.1.1*).

D'après nos expériences, nous constatons une réduction du bruit d'environ 8% par cette séparation.

Le dictionnaire « Prefix-P0 » est présenté dans l'Annexe III.A, le « Suffix-P0 » dans l'Annexe III.B.

4.1.1.7.2 Séparation des termes rares

En analysant le vocabulaire du texte, on s'aperçoit que l'on est souvent gêné par des termes rares ou même rarissimes. Pour éviter cela, il vaut mieux séparer ces termes en les plaçant dans des dictionnaires différents. Nous employons la même technique que celle du français, c'est-à-dire, les codes de niveaux de langue[8] : z1, z2 et z3. Les mots courants sont placés dans les dictionnaires z1, les mots rares dans les dictionnaires z2 et les mots rarissimes dans les dictionnaires z3. Et nous n'appliquons les dictionnaires z2 et z3 que lorsque la nature du texte le justifie.

La définition des zones est un travail délicat et difficile qui demande une étude approfondie qui sortirait du cadre de ce travail. Nous avons défini les zones, pour l'instant, en n'utilisant comme critères de sélection que les codes d'usage du Tableau 3.3 : les mots tombés en désuétude sont dans la zone 3 (639 entrées) ; les mots à usage spécifique sont dans la zone 2 (2 242 entrées) ; les autres mots, en attendant une étude plus concrète, restent dans la zone 1 (30 786 entrées).

Dictionnaire		**Signification**
z2	**z3**	
	Arch	Archaïque
Dial		Dialecte
Form		Formel
	Obs	Obsolète
Poet		Poésie
Royal		Terme de cour

Tableau 4.1 Sélection des zones de dictionnaires

D'après nos expériences, le taux de bruit diminue d'environ 2% en n'appliquant pas les dictionnaires z3 et d'environ 8% en n'appliquant que les dictionnaires z1. De plus, en combinant avec la séparation des affixes (*cf. §4.1.1.7.1*), le taux de réduction du bruit peut s'élever jusqu'à 15% environ.

Des extraits des Dictionnaires P0 sont présentés dans l'Annexe III.

[8] GARRIGUES 1993

4.1.2 Méthode par syllabes

Malgré son efficacité en reconnaissance lexicale, la méthode par caractères crée un bruit excessif. Bien que ce dernier soit réduit par la réorganisation des dictionnaires, il en reste encore trop. C'est pour cette raison que nous proposons une nouvelle méthode de segmentation qui exploite des connaissances linguistiques sur le thaï, plus précisément, sur les séquences de caractères possibles en thaï.

Selon le système d'écriture du thaï (*cf. Chapitre 2*), certains caractères ou séquences de caractères ne sont pas autonomes, c'est-à-dire qu'ils appartiennent toujours au même mot que le noyau de la syllabe dont ils font partie. En conséquence, il n'est pas nécessaire d'insérer un séparateur entre ces caractères et le noyau. Ainsi, les segments définis par cette méthode sont plus grands que ceux obtenus par la méthode par caractères. Cela permet aux dictionnaires de générer moins de bruit tout en gardant le taux de silence à 0%.

Le formalisme des réseaux de transitions récursifs (RTN) (*cf. §1.1.2*) semble être l'outil le mieux adapté dans cette situation ; en effet, il convient de créer plusieurs sous-graphes, selon les types de séquences, et de les appeler dans les graphes de découpage. Il existe un obstacle technique : les RTN ne fonctionnent pas en mode caractère dans INTEX ; on ne peut donc pas parcourir les séquences de caractères soudés pour insérer les séparateurs. Heureusement, grâce aux Corpus P0 et Dictionnaires P0, déjà segmentés caractère par caractère, nous pouvons faire le processus inverse, c'est-à-dire que nous pouvons utiliser les RTN (avec variables) pour parcourir les Corpus P0 et Dictionnaires P0 et regrouper les caractères inséparables en enlevant les séparateurs entre eux. Ces RTN seront appliqués en mode « remplacement » par le troisième module de normalisation (*cf. §1.2.2.3*).

Cette méthode est divisée en 3 étapes : la première rassemble des caractères inséparables en segments plus grands ; la deuxième traite le résultat de la première étape en fusionnant certains segments inséparables en syllabes ; et la dernière regroupe certaines syllabes inséparables afin d'obtenir des mots. L'idée de diviser le regroupement en plusieurs étapes et de les appliquer en cascade nous permet de diminuer la complexité et le nombre de transducteurs. Supposons, par exemple, que nous pouvons diviser le regroupement en 3 étapes et que chacune possède 10 transducteurs, alors 30 transducteurs (10+10+10) qui fonctionnent en cascade donnent le résultat équivalent à 1 000 transducteurs (10x10x10) d'un seul regroupement.

4.1.2.1 Regroupement des caractères inséparables

En analysant la forme syllabique segmentée par la méthode par caractères (*cf. Figure 4.1*) sans considérer le phonème muet, nous constatons que seules les consonnes peuvent être autonomes, ni les voyelles[9] ni les diacritiques ne le sont (*cf. §2.1.2, §2.1.3*). En conséquence, nous pouvons regrouper les V_a, V_s, V_p et D avec leur C_i, le noyau de la syllabe à l'écrit (*cf. §2.2.3*), comme le montre la figure ci-dessous. Quant au C_f, il est souvent ambigu (entre être C_f de cette syllabe ou C_i de la syllabe suivante (*cf. §2.3.3*)) et nous ne pouvons le regrouper que dans certains cas.

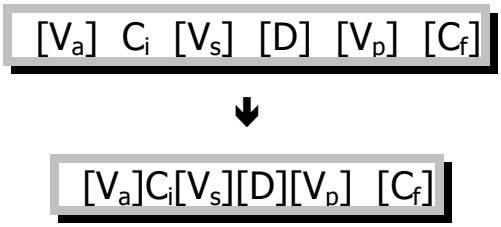

Figure 4.6 Regroupement des caractères inséparables

V_a	: Caractère vocalique antéposé	D	: Signe diacritique
C_i	: Caractère consonantique initial	V_p	: Caractère vocalique postposé
V_s	: Caractère vocalique suscrit ou souscrit	C_f	: Caractère consonantique final

Le phonème muet (*cf. Figure 4.2*) peut être regroupé de la même façon. Nous rappelons que les caractères vocaliques [i] et [u] sont aussi des V_s et que le signe de phonème muet est également un D.

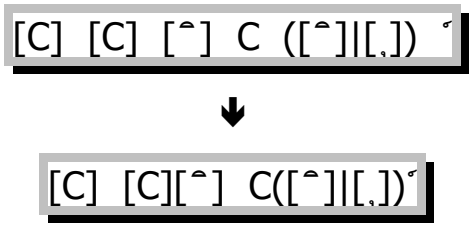

Figure 4.7 Regroupement des caractères inséparables dans un phonème muet

C	: Caractère consonantique	◌ͥ	: Caractère vocalique [u]
◌ͤ	: Caractère vocalique [i]	◌ͫ	: Signe de phonème muet

Après ce regroupement, les textes deviennent des séquences de segments (approximativement des syllabes sans phonèmes muets).

[9] À l'exception des voyelles isolantes (*cf. Tableau 2.3*).

4.1.2.1.1 Graphe de regroupement « Replace 1 »

Nous avons construit un graphe nommé « Replace 1 » : un RTN qui fait appel à plusieurs transducteurs avec variables (*cf. §1.1.3*). Dans chaque transducteur, différentes formes syllabiques, comprenant des caractères inséparables, sont décrites.

Les transducteurs parcourent les textes P0 dans lesquels les caractères sont séparés les uns des autres par des espaces blancs. Ils mémorisent chaque caractère d'un chemin dans une variable numérotée et produisent en sortie une séquence formée des mêmes caractères mais sans espaces blancs entre eux. Ainsi, les caractères reconnus par « Replace 1 » sont regroupés. Les transducteurs sont détaillés ci-dessous :

4.1.2.1.1.1 Sous-graphes

Nous commençons d'abord par créer des sous-graphes selon les types de caractères thaï :

- Consonnes initiales (Consonant) : ce sont les noyaux des syllabes à l'écrit. Tous les caractères consonantiques du thaï (*cf. §2.1.1*) peuvent être consonnes initiales, y compris les deux caractères obsolètes : « ฃ » [khɔ̌ː khùat] et « ฅ » [khɔːkhon] que nous pouvons trouver dans des documents anciens.

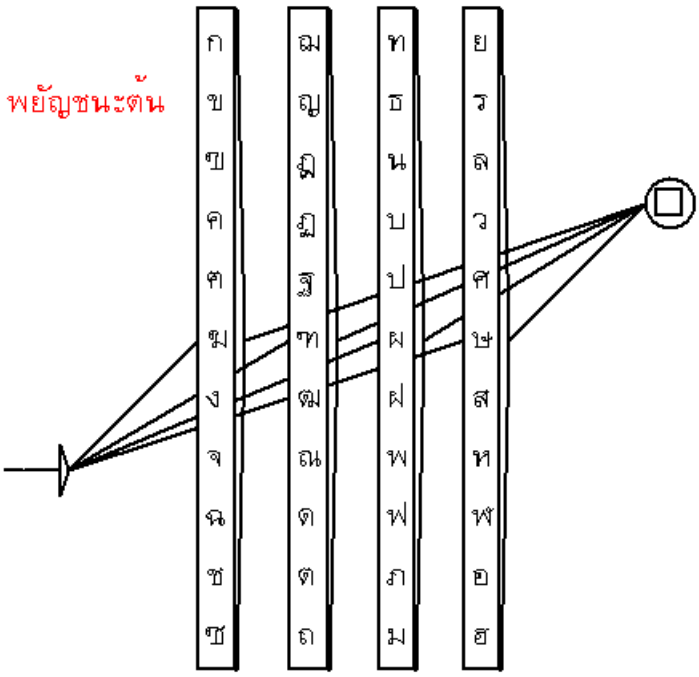

Figure 4.8 Graphe « Consonant »

- Consonnes initiales-1 (Consonant-1) : ce sont toutes les consonnes initiales sauf le « ก » [kɔ: kàj].

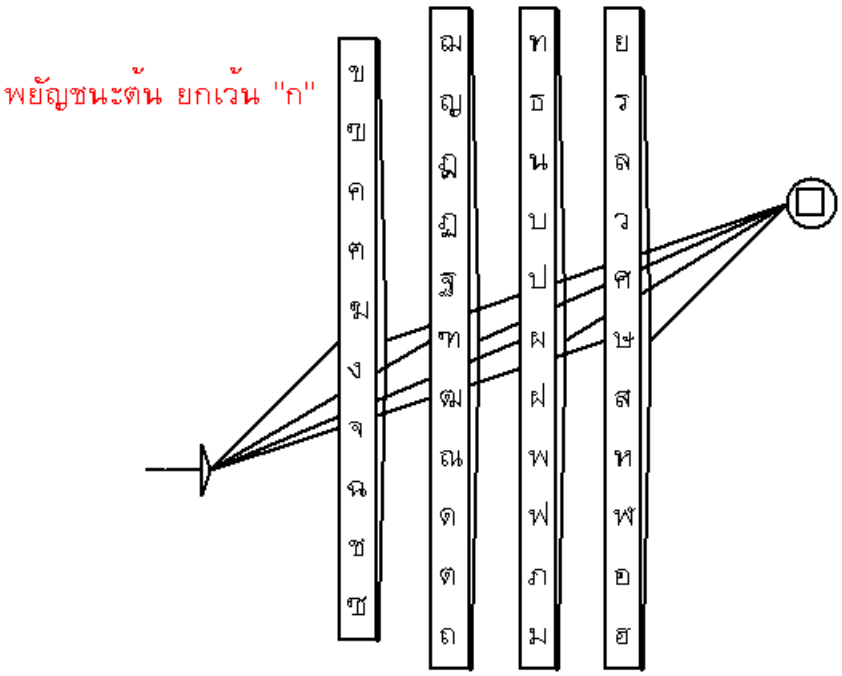

Figure 4.9 Graphe « Consonant-1 »

- Consonnes initiales-2 (Consonant-2) : ce sont toutes les consonnes initiales sauf les consonnes ambiguës (Consonant-3) et le « บ » [bɔ: bajmáj].

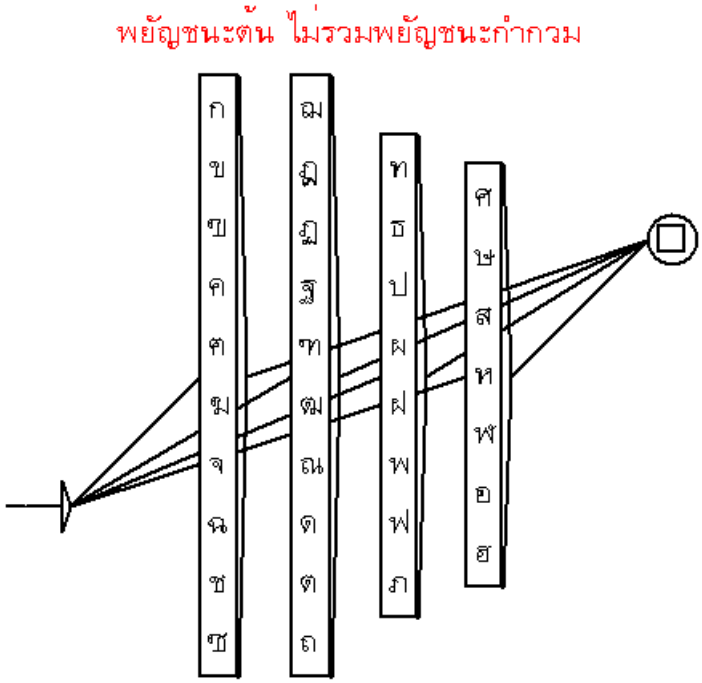

Figure 4.10 Graphe « Consonant-2 »

- Consonnes initiales-3 (Consonant-3) : ce sont les consonnes ambiguës, c'est-à-dire, celles qui peuvent être la deuxième consonne des agglomérats consonantiques. En effet, lorsque deux consonnes sont contiguës, la deuxième est souvent ambiguë parce qu'elle peut faire partie de l'agglomérat, être la consonne finale de cette syllabe ou bien être la consonne initiale de la syllabe suivante.

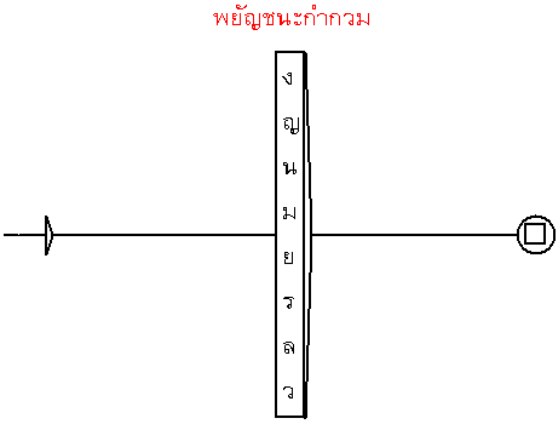

Figure 4.11 Graphe « Consonant-3 »

- Consonnes finales (Speller) : ce sont toutes les consonnes thaï à l'exception des 9 suivantes qui ne peuvent s'utiliser en position finale : « ฃ » [khɔ̌ː khùat], « ฅ » [khɔː khon], « ฉ » [chɔ̌ː chĭŋ], « ฌ » [chɔː chɣː], « ผ » [phɔ̌ː phŭŋ], « ฝ » [fɔ̌ː făː], « ห » [hɔ̌ː hĭːp], « อ » [ʔɔː ʔàːŋ] et « ฮ » [hɔː nókhûːk].

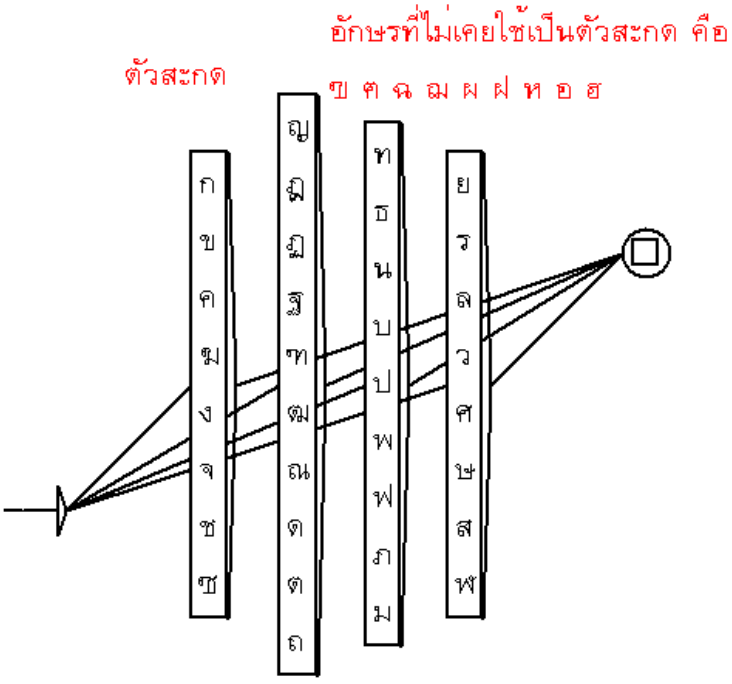

Figure 4.12 Graphe « Speller »

- Consonnes finales-1 (Speller-1) : ce sont toutes les consonnes finales sauf le « ว » [wɔː wɛ̌ːn].

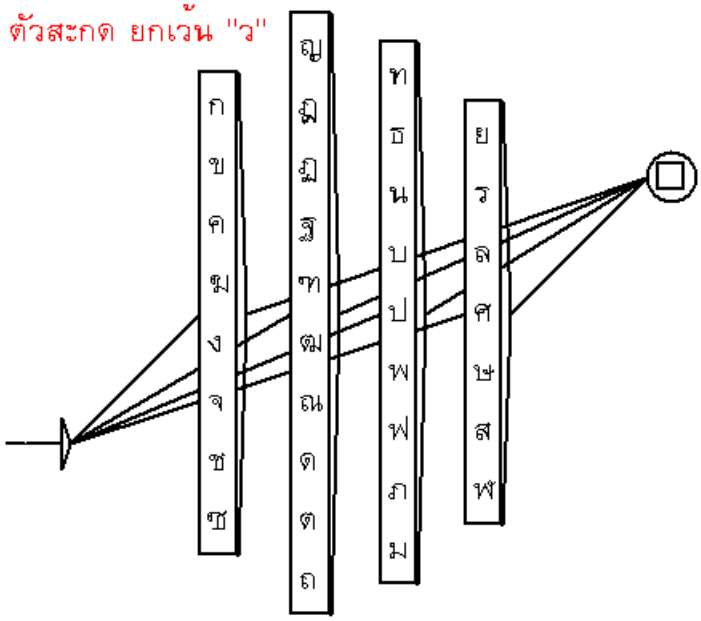

Figure 4.13 Graphe « Speller-1 »

- Marques de tons (Tone) : ce sont les 4 graphèmes tonals du thaï (*cf. §2.1.3.1*).

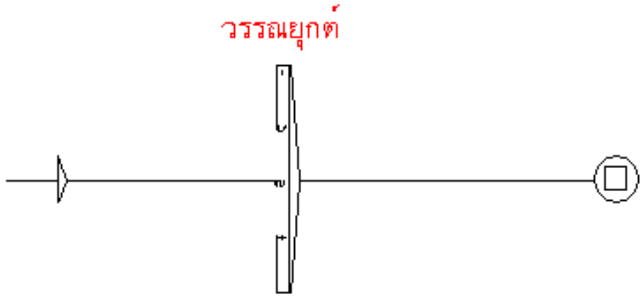

Figure 4.14 Graphe « Tone »

- Marques de tons-1 (Tone-1) : ce sont toutes les marques de tons sauf le ton bas « ◌̀ » [máj ʔèk].

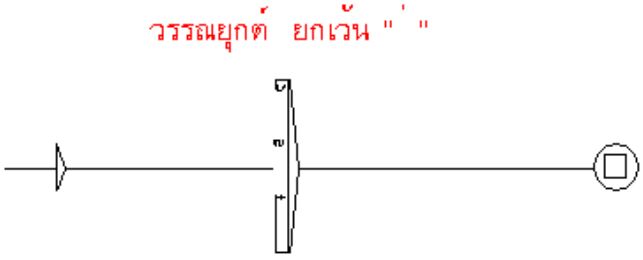

Figure 4.15 Graphe « Tone-1 »

4.1.2.1.1.2 Graphes principaux

Les graphes principaux de regroupement sont des RTN avec variables. Chaque graphe traite des formes syllabiques spécifiques comprenant des caractères inséparables et des éléments exigés. Étant donné que les agglomérats consonantiques posent des problèmes d'ambiguïté, nous ne les prenons en considération que dans les cas non-ambigus.

- Le graphe « Replace 1.1 » traite les voyelles « ไ- » [aj] et « ใ- » [aj] qui sont des voyelles antéposées et sont également des caractères inséparables. Elles s'accompagnent obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement les suivre.

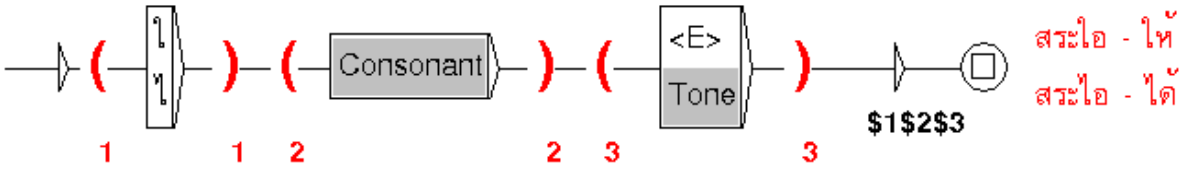

Figure 4.16 Graphe « Replace 1.1 »

- Le graphe « Replace 1.2 » traite les voyelles « -ะ » [a], « -า » [a:] et « -ำ » [am] qui sont des voyelles postposées et sont également des caractères inséparables. Elles s'accompagnent obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement suivre la consonne initiale.

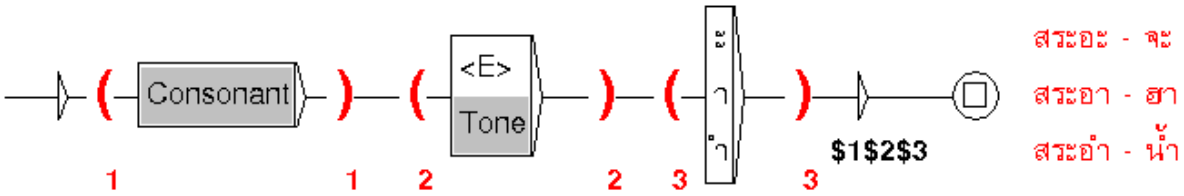

Figure 4.17 Graphe « Replace 1.2 »

- Le graphe « Replace 1.3 » traite la voyelle « ี- » [i:] (voyelle suscrite) et les voyelles « ุ- » [u] et « ู- » [u:] (voyelles souscrites). Toutes les trois sont des caractères inséparables qui s'accompagnent obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement les suivre.

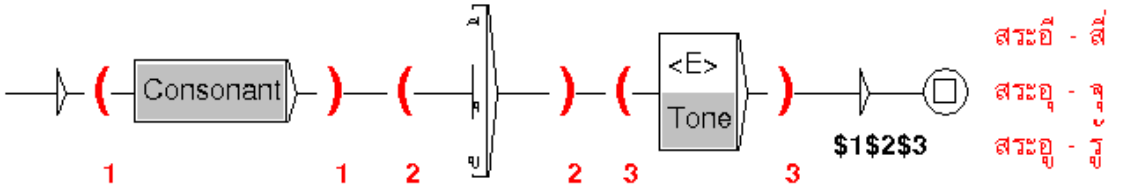

Figure 4.18 Graphe « Replace 1.3 »

- Le graphe « Replace 1.4 » traite les voyelles « ◌ิ » [i] et « ◌ึ » [ɯ] sans marque de ton. Ce sont des voyelles suscrites et des caractères inséparables qui s'accompagnent obligatoirement d'une consonne initiale.

Figure 4.19 Graphe « Replace 1.4 »

- Lorsque les voyelles « ◌ิ » [i] et « ◌ึ » [ɯ] s'écrivent avec une marque de ton, une consonne finale est exigée.[10] Cette forme est traitée par le graphe « Replace 1.5 ».

Figure 4.20 Graphe « Replace 1.5 »

Remarque : Selon la troisième règle de l'application des graphes (*cf. §1.1.2*), « Replace 1.5 », étant plus long, est prioritaire par rapport à « Replace 1.4 ».

- Le graphe « Replace 1.6 » traite la voyelle « ◌ื » [ɯː] qui est une voyelle suscrite et est également un caractère inséparable. Elle s'accompagne obligatoirement d'une consonne initiale et d'une consonne finale. Lorsque cette dernière est absente, le « อ » [ʔɔː ʔàːŋ] la remplace automatiquement (*cf. §2.2.2.4.1*). Une marque de ton peut éventuellement apparaître entre la voyelle en question et la consonne finale (ou le « อ » [ʔɔː ʔàːŋ]).

[10] CHARNYAPORNPONG 1983

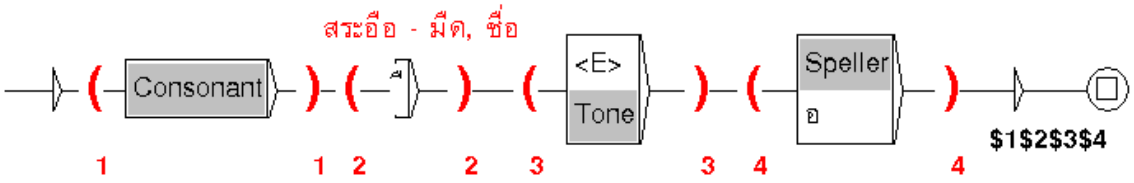

Figure 4.21 Graphe « Replace 1.6 »

- Le graphe « Replace 1.7 » traite les voyelles « เ- » [eː], « แ- » [ɛː] et « โ- » [oː] : des voyelles antéposées inséparables qui s'accompagnent obligatoirement d'une consonne initiale. Ce sont également des voyelles de phonème long qui peuvent accueillir la voyelle « -ะ » [a] à la fin des syllabes pour se transformer en voyelles de phonème court « เ-ะ » [e], « แ-ะ » [ɛ] et « โ-ะ » [o] (*cf. Tableau 2.3*). Une marque de ton peut éventuellement suivre la consonne initiale.

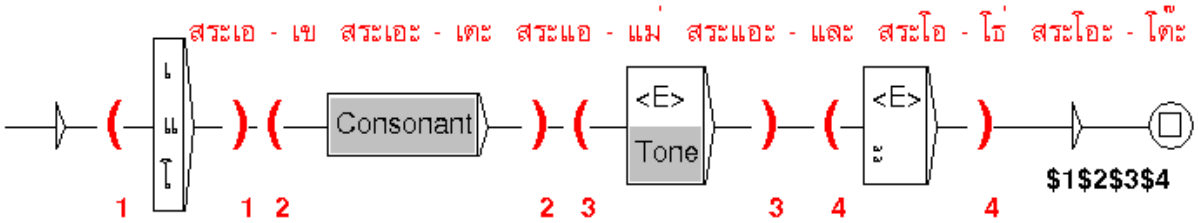

Figure 4.22 Graphe « Replace 1.7 »

<u>Remarque</u> : Selon la deuxième règle de l'application des graphes (*cf. §1.1.2*), « Replace 1.7 » qui commence plus à gauche, est prioritaire par rapport à « Replace 1.2 ». Ainsi la séquence « เ อ ะ » est reconnue par « Replace 1.7 » (non par « Replace 1.2 ») et sera transformée en « เอะ » (pas en « เ อะ »).

- Le graphe « Replace 1.8 » traite la voyelle « เ-า » [aw] : voyelle composée de deux caractères vocaliques inséparables « เ- » [eː] et « -า » [aː]. Elle s'accompagne obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement apparaître entre la consonne initiale et le dernier caractère vocalique.

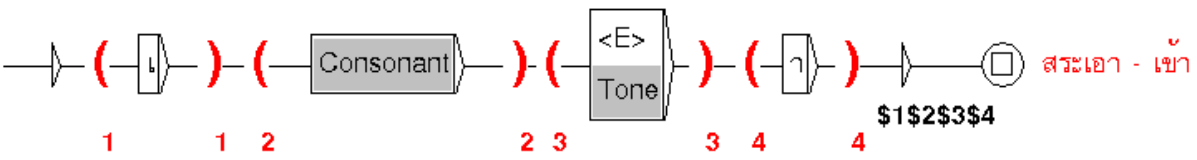

Figure 4.23 Graphe « Replace 1.8 »

Remarque : Par la priorité (*cf. §1.1.2*), la séquence « เ อ า » sera reconnue par « Replace 1.8 » (et sera transformée en « เอา »), non par « Replace 1.7 » (qui la transformerait en « เอ า »), ni par « Replace 1.2 » (qui la transformerait en « เ อา »).

- Le graphe « Replace 1.9 » traite la voyelle « เ-าะ » [ɔ] : voyelle composée de trois caractères vocaliques inséparables « เ- » [eː], « -า » [aː] et « -ะ » [a]. Elle s'accompagne sans ambiguïté d'une ou deux consonnes initiales. Une marque de ton peut éventuellement apparaître avant le « า » [aː].

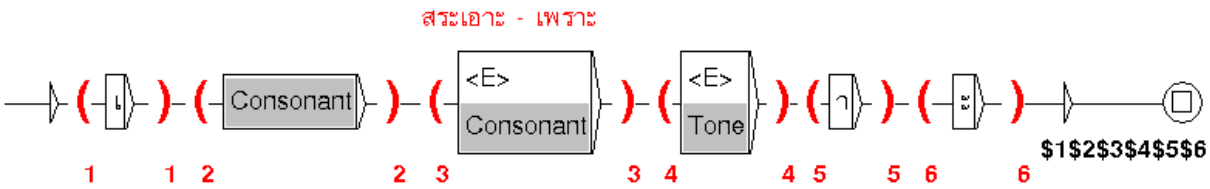

Figure 4.24 Graphe « Replace 1.9 »

Discussion : Dans le cas ci-dessus, la deuxième consonne (si elle existe) n'est pas du tout ambiguë parce que cette forme ne peut pas se diviser. En effet, la deuxième consonne ne peut pas être la consonne initiale de la deuxième syllabe : la forme « Cาะ » n'existe pas ; elle ne peut pas non plus être la consonne finale de la première syllabe : le « า » [aː] ne peut pas commencer une nouvelle syllabe. Elle fait donc partie de l'agglomérat consonantique initial, ce qui permet de regrouper les six éléments.

- Le graphe « Replace 1.10 » traite les voyelles « เ-ีย » [iaː] et « เ-ียะ » [ia], voyelles composées respectivement de trois et de quatre caractères. Ces voyelles sont inséparables et s'accompagnent obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement apparaître entre le « -ี » [iː] et le « ย » [jɔː jáːk].

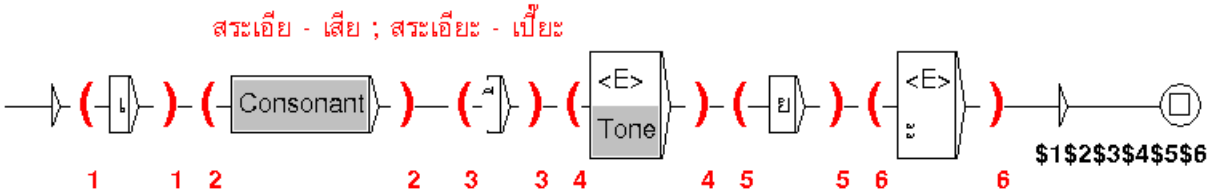

Figure 4.25 Graphe « Replace 1.10 »

- Le graphe « Replace 1.11 » traite la voyelle « เ-ือ » [ɯaː], voyelle composée de trois caractères. Elle s'accompagne obligatoirement d'une consonne initiale. Une marque de ton peut éventuellement apparaître entre le « -ื » [ɯː] et le « อ » [ʔɔː ʔàːŋ].

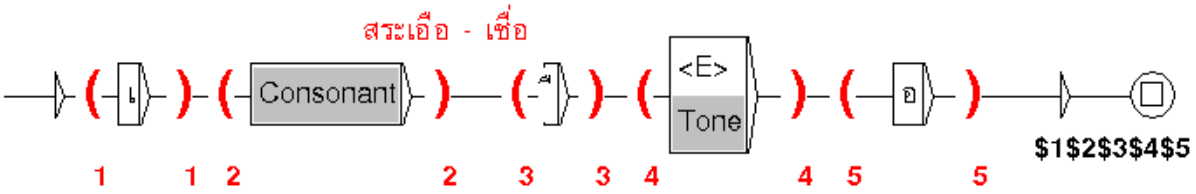

Figure 4.26 Graphe « Replace 1.11 »

- Le graphe « Replace 1.12 » traite les voyelles « เ-ือะ » [ɯa], voyelle composée de quatre caractères. Elle s'accompagne sans ambiguïté d'une ou deux consonnes initiales. Une marque de ton peut éventuellement apparaître entre le « -ื » [ɯː] et le « อ » [ʔɔː ʔàːŋ].

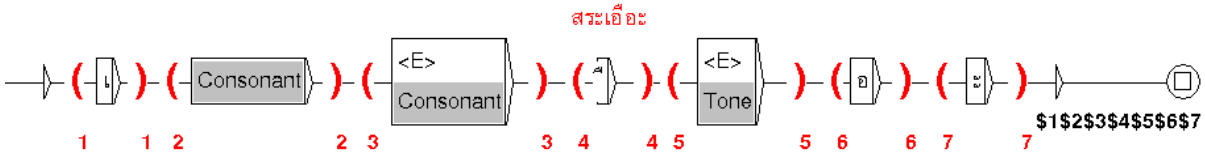

Figure 4.27 Graphe « Replace 1.12 »

Discussion : Dans le cas ci-dessus, la deuxième consonne (si elle existe) n'est pas du tout ambiguë parce que cette forme ne peut pas se diviser. En effet, la deuxième consonne ne peut pas être la consonne initiale de la deuxième syllabe : la forme « ื◌อะ » n'existe pas ; elle ne peut pas non plus être la consonne finale de la première syllabe : le « -ื » [ɯː] ne peut pas commencer une nouvelle syllabe. Elle fait donc partie de l'agglomérat consonantique initial, ce qui permet de regrouper les sept éléments.

- Le graphe « Replace 1.13 » traite le caractère « ◌ั » [máj hăn ʔakàːt] qui est soit une forme modifiée de la voyelle « -ะ » [a] (*cf. §2.2.2.3.1*), soit un élément des voyelles « ◌ัวะ » [ua] et « ◌ัว » [uaː] (*cf. Tableau 2.3*). C'est un caractère inséparable qui s'accompagne obligatoirement d'une consonne initiale et d'une consonne finale[11]. Une marque de ton peut éventuellement apparaître après le caractère en

[11] Pour les voyelles « ◌ัวะ » [ua] et « ◌ัว » [uaː], nous considérons, dans ces deux cas, le « ว » [wɔː wĕːn] comme consonne finale.

question. La consonne finale peut en plus accueillir une des voyelles « -ะ » [a], « -า » [aː], « -ิ » [i] et « -ุ » [u].[12]

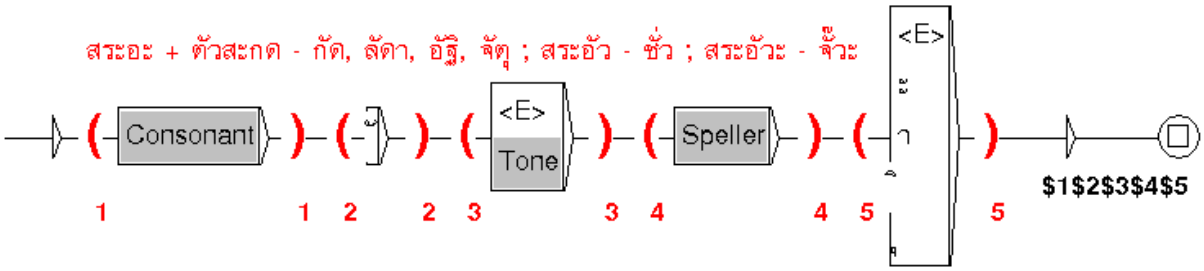

Figure 4.28 Graphe « Replace 1.13 »

- Le graphe « Replace 1.14 » traite les formes « เ-็- » et « แ-็- », formes modifiées respectivement des voyelles « เ-ะ » [e] et « แ-ะ » [ɛ] qui prennent une consonne finale (*cf. §2.2.2.3.2, §2.2.2.3.3*). Les deux formes s'accompagnent obligatoirement d'une consonne initiale et d'une consonne finale. Une deuxième consonne initiale (autre que le « ก » [kɔː kàj]) est autorisée mais aucune marque de ton ne l'est.

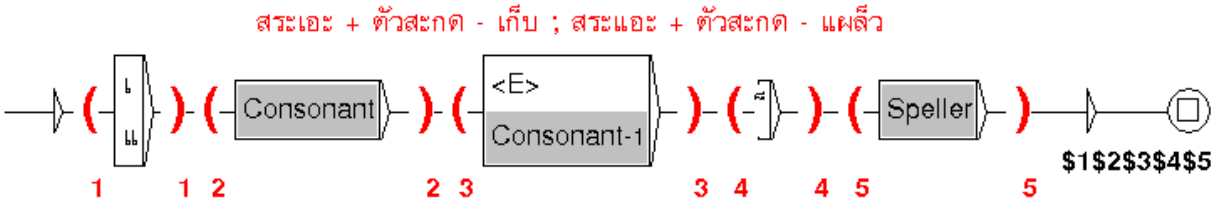

Figure 4.29 Graphe « Replace 1.14 »

Discussion : Si la deuxième consonne initiale était le « ก » [kɔː kàj], le graphe ci-dessus serait ambigu parce que le « ก » [kɔː kàj] et le signe de phonème court « ็ » [máj tàjkhúː] pourraient former la syllabe autonome « ก็ » [kɔ̂] (*cf. §2.2.2.3.4*). Ce qui veut dire que ce graphe pourrait regrouper à tort plusieurs syllabes autonomes.

Remarque : Aucune marque de ton n'est admise dans ces deux formes afin de ne pas encombrer le signe de phonème court qui occupe cette position.

- Le graphe « Replace 1.15 » traite les formes « ็-อ- » et « ็-ว- », formes modifiées respectivement des voyelles « เ-าะ » [ɔ] et « ั-วะ » [ua] qui prennent une consonne

[12] C'est le cas des mots empruntés du pāli-sanskrit.

finale (*cf. §2.2.2.3.4, §2.2.2.3.6*). Les deux formes s'accompagnent obligatoirement d'une consonne initiale (autre que le « ก » [kɔ: kàj]) et d'une consonne finale. Aucune marque de ton n'est autorisée.

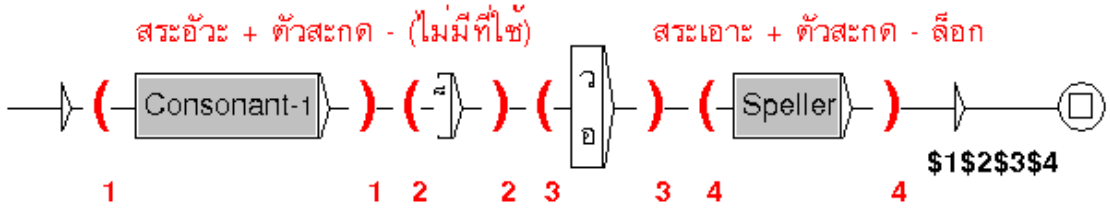

Figure 4.30 Graphe « Replace 1.15 »

Remarque : Même si le système d'écriture du thaï autorise la forme « ◌ั◌ว◌ », nous n'avons trouvé aucun mot des dictionnaires comportant cette syllabe.

Discussion : Si nous trouvons dans un texte une séquence « เ c_1 c_2◌ั ว c_3 », faut-il la regrouper en « เc_1 c_2◌ัc_3 » avec « Replace 1.7 » et « Replace 1.15 », ou en « เ$c_1$$c_2$◌ั ว c_3 » avec « Replace 1.14 » ? Grâce à la remarque ci-dessus, nous validons la deuxième solution qui est mise en œuvre à l'aide de « Replace 1.14 », déjà prioritaire sur « Replace 1.7 » par le critère de longueur de la séquence reconnue.

- Le graphe « Replace 1.16 » traite la forme « เ◌ิ◌ » : forme modifiée de la voyelle « เ-อ » [ɣ:] qui prend une consonne finale (*cf. §2.2.2.3.5*). Cette forme s'accompagne obligatoirement d'une consonne initiale et d'une consonne finale. Une marque de ton peut éventuellement apparaître entre le « ◌ิ » [i] et la consonne finale.

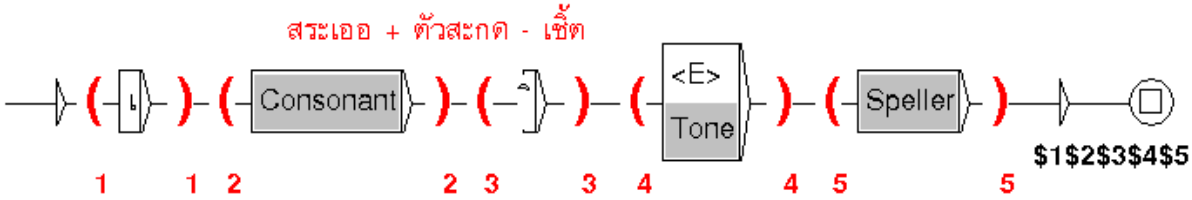

Figure 4.31 Graphe « Replace 1.16 »

- Le graphe « Replace 1.17 » traite la forme « เ◌ิ◌ » de « Replace 1.16 » dans laquelle un phonème muet, comportant une consonne avec le signe de phonème muet, est écrit devant la consonne finale. Cette forme est utilisée par certains mots empruntés d'origine européenne.

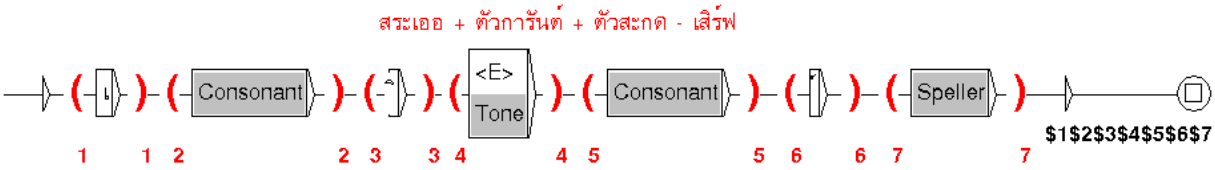

Figure 4.32 Graphe « Replace 1.17 »

- La voyelle « -อ » [ɔː] et la forme « $c_i c_f$ », forme supprimée de la voyelle « โ-ะ » [o] (*cf. §2.2.2.1.3*) sont souvent ambiguës et ne peuvent pas être regroupées à moins qu'elles soient écrites avec une marque de ton. Dans ce cas, une consonne initiale (autre que les consonnes ambiguës et le « บ » [bɔː bajmáj]) et le « อ » [ʔɔː ʔàːŋ] ou une consonne finale (autre que le « ว » [wɔː wĕːn]) sont exigés.

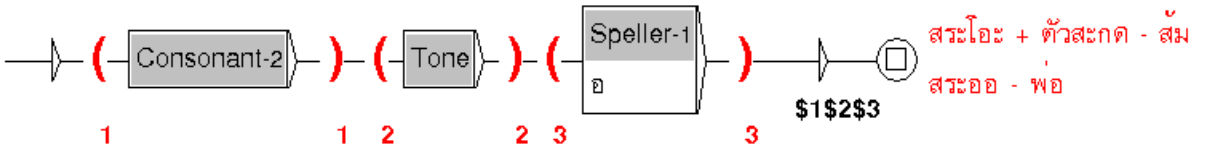

Figure 4.33 Graphe « Replace 1.18 »

Discussion : Si la consonne initiale était une consonne ambiguë (*cf. Figure 4.11*), elle serait équivoque parce qu'elle pourrait soit faire partie de l'agglomérat consonantique de la syllabe précédente à gauche, soit être la consonne initiale de cette syllabe.

- Si la consonne initiale est le « บ » [bɔː bajmáj] et que la marque de ton ne soit pas le ton bas, elles doivent s'accompagner du même type d'éléments que dans le graphe précédent.

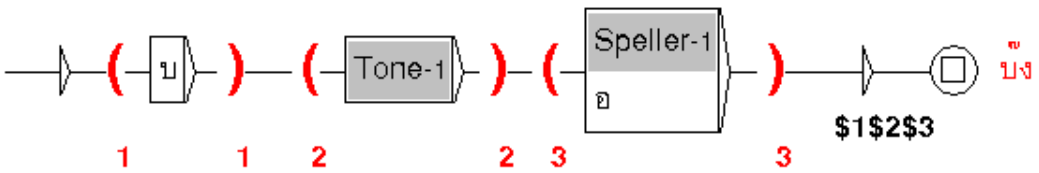

Figure 4.34 Graphe « Replace 1.19 »

Remarque : Le « บ » [bɔː bajmáj] et le ton bas peuvent former la syllabe autonome « บ่ » [bɔ̀ː] (*cf. Tableau 2.13*) : une forme supprimée de la voyelle « -อ » [ɔː] (*cf. §2.2.2.1.2*).

- D'après les deux derniers graphes, si l'élément tout de suite après la marque de ton est le « ว » [wɔː wɛ̌ːn], cela signifie qu'on a une forme réduite de la voyelle « ◌ัว » [uaː] qui prend obligatoirement une consonne finale (*cf. §2.2.2.2.2*).

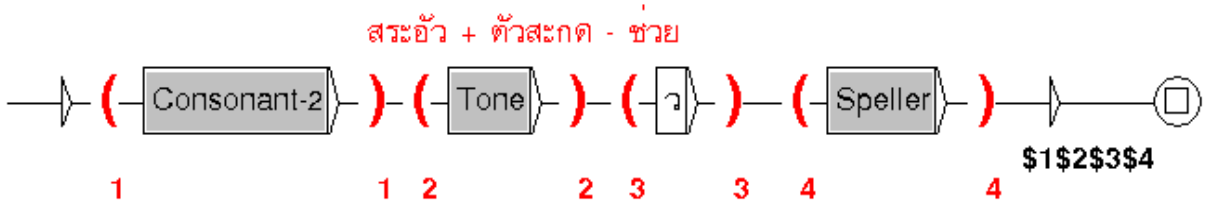

Figure 4.35 Graphe « Replace 1.20 »

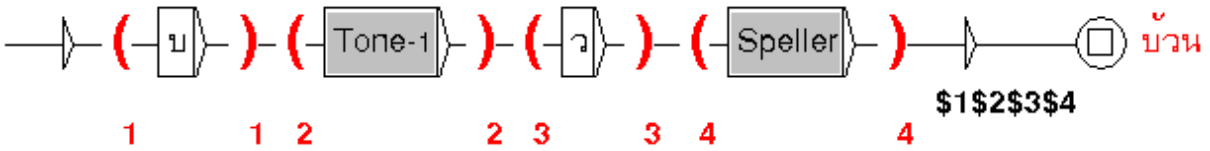

Figure 4.36 Graphe « Replace 1.21 »

Remarque : La consonne finale après le « ว » [wɔː wɛ̌ːn] peut être le « ว » [wɔː wɛ̌ːn]. Il s'agit de la forme archaïque de la voyelle « ◌ัว » [uaː] que nous pouvons trouver dans des documents anciens.

- Lorsqu'il reste, après tous les graphes précédents, une consonne ambiguë suivie d'une marque de ton, nous pouvons les regrouper par le graphe suivant.

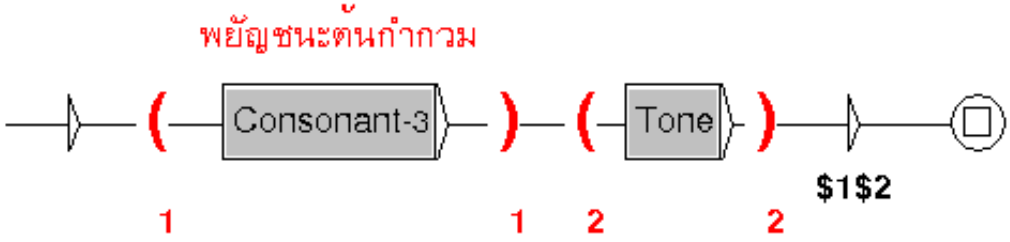

Figure 4.37 Graphe « Replace 1.22 »

Remarque : Le graphe ci-dessus n'ayant que 2 éléments est le moins prioritaire par rapport aux autres graphes qui sont plus longs.

4.1.2.1.1.3 Phonèmes muets

Comme le montre la Figure 4.7, nous regroupons les phonèmes muets en décrivant leurs formes syllabiques par les graphes suivants :

- La structure générale des phonèmes muets est composée d'une consonne, suivie éventuellement d'une voyelle « ◌ิ » [i] ou « ◌ุ » [u] et terminée par le signe de phonème muet « ◌์ » [thanthákhât].

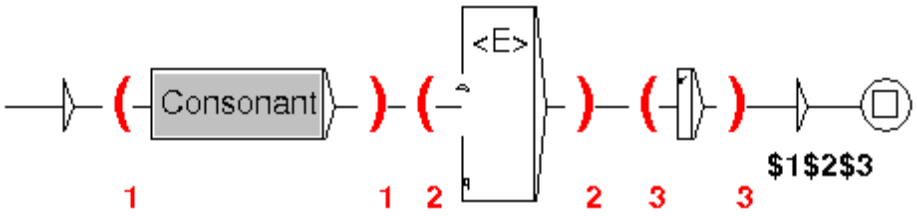

Figure 4.38 Graphe « Replace 1.23 »

- Un phonème muet peut être composé de plusieurs caractères (*cf. §2.1.3.2 et Figure 2.3*). Les cas non-ambigus sont décrits dans la figure suivante. Tous les phonèmes muets se terminent par le signe « ◌์ » [thanthákhât].

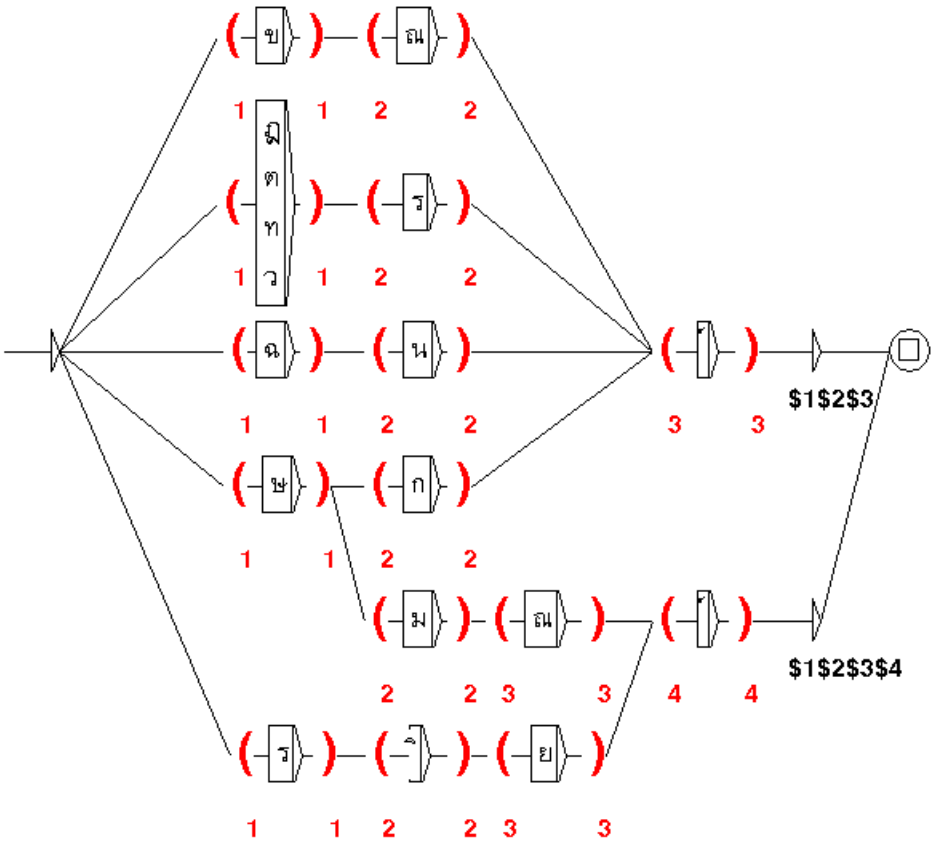

Figure 4.39 Graphe « Replace 1.24 »

4.1.2.1.1.4 Cas particuliers

Pour les cas particuliers, nous décrivons des syllabes autonomes courantes par les graphes de regroupement ci-dessous :

- La syllabe autonome « กี๋ » [kô] est la forme modifiée de « เก๊าะ » (*cf. §2.2.2.3.4*), composée du « ก » [kɔː kàj] et du signe de phonème court « ◌๋ » [máj tàjkhúː].

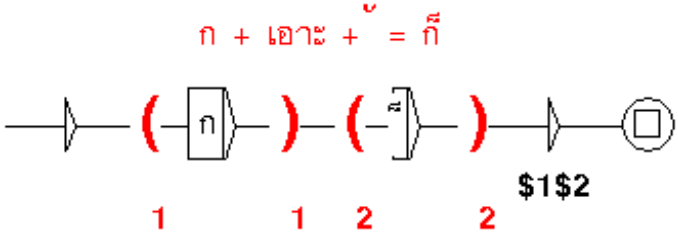

Figure 4.40 Graphe « Replace 1.25 »

- La syllabe autonome « บ่ » [bɔ̀ː] est une forme supprimée de la voyelle « -อ » [ɔː] (*cf. §2.2.2.1.2*), composée du « บ » [bɔː bajmáj] et du ton bas « ◌่ » [máj ʔèk].

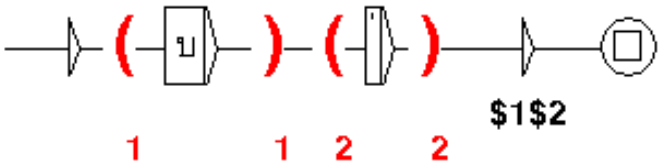

Figure 4.41 Graphe « Replace 1.26 »

- Les caractères vocaliques « ฤ » [rɔː rɯ́] et « ฦ » [lɔː lɯ́] sont des voyelles isolantes de phonème court qui acceptent le caractère vocalique « -ๅ » [lâk khâŋ jaːw] pour se transformer en voyelles isolantes de phonème long, respectivement « ฤๅ » [rɯːˑ] et « ฦๅ » [lɯːˑ].

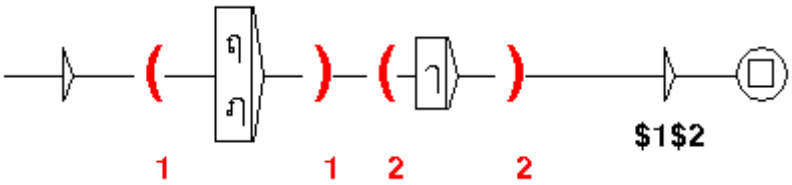

Figure 4.42 Graphe « Replace 1.27 »

- En thaï, il n'existe que 20 syllabes écrites avec le « ไ- » [sàrà ʔaj máj múan]. La plupart sont déjà prises en compte par « Replace 1.1 ». Les autres, écrites avec un agglomérat consonantique, sont traitées par le graphe ci-dessous.

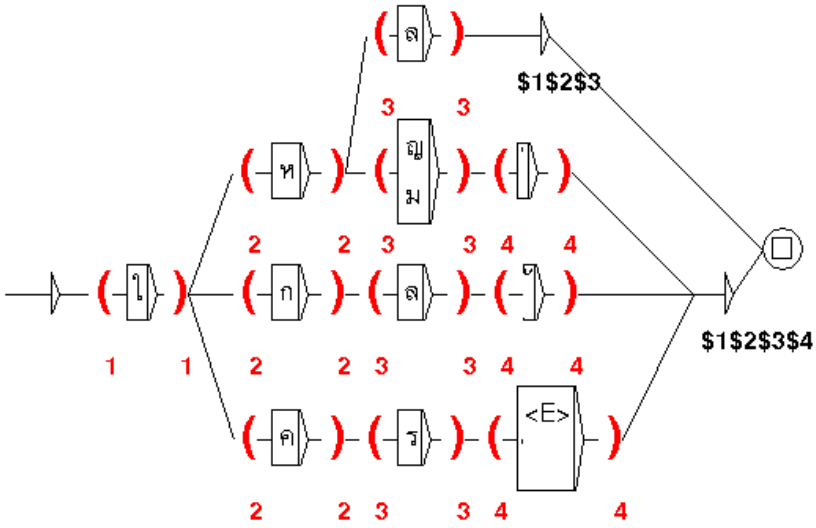

Figure 4.43 Graphe « Replace 1.28 »

- Les mots « เจริญ » [càryːn] et « เสริญ » [sɤ̌ːn] sont des mots courants qui théoriquement sont graphiquement divisibles en deux segments mais cela n'aurait aucun sens[13]. Nous les décrivons directement dans le graphe suivant.

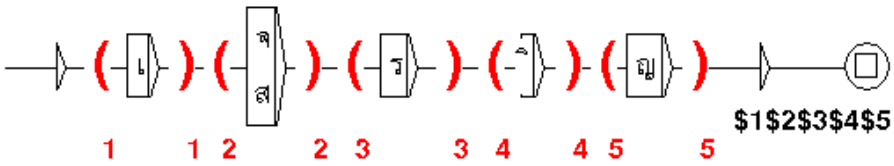

Figure 4.44 Graphe « Replace 1.29 »

Remarque : Les derniers graphes montrent que nous pourrions en fait travailler davantage sur les mots courants non-ambigus afin de les regrouper en une seule unité. Néanmoins, avant de réaliser de tels graphes, une étude approfondie est préférable.

Tous les graphes appelés par « Replace 1 » sont présentés dans l'Annexe IV.

4.1.2.1.2 Corpus P1

Après avoir appliqué le graphe « Replace 1 » au « Corpus P0 », le texte devient une suite de segments (approximativement des syllabes), nommé « Corpus P1 », et sa liste de lexèmes est plutôt une liste de syllabes (*voir la figure ci-dessous*). Nous remarquons que le nombre de formes simples se réduit de 608 dans le Corpus P0 (*cf. Figure 4.3*) à 323 dans ce corpus, soit 47%[14] de moins. Cela permet d'espérer une réduction du bruit.

[13] SAWAMIPAK 1990
[14] (323-608)x100/608 = -46,88%

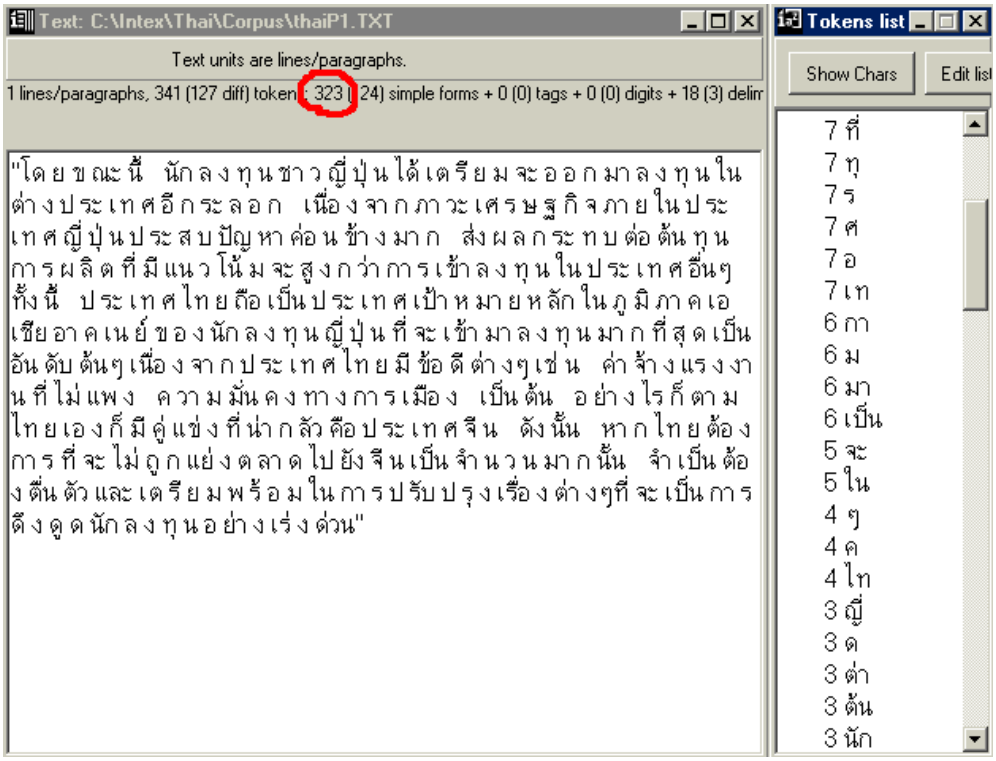

Figure 4.45 Corpus P1 et ses lexèmes

Nous pouvons toujours cacher les espaces blancs en désactivant l'option « Display tags » (comme dans le Corpus P0) pour que le texte soit affiché dans sa forme originale.

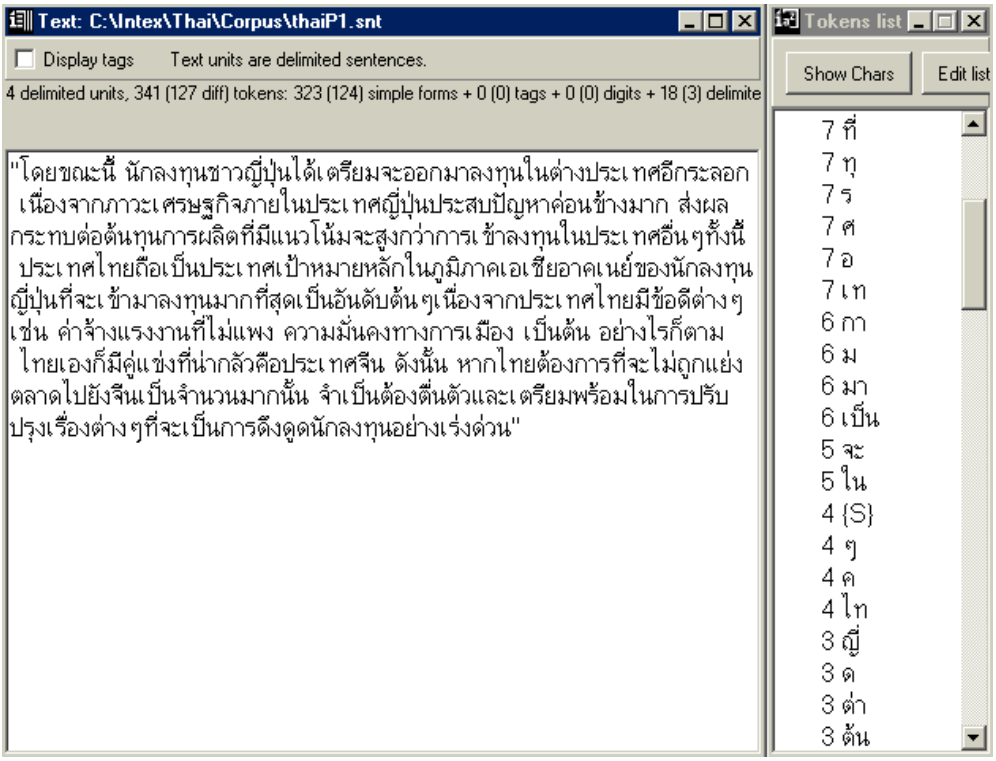

Figure 4.46 Corpus P1 avec les espaces blancs cachés

4.1.2.1.3 Dictionnaires P1

Nous appliquons également « Replace 1 » aux entrées des Dictionnaires P0 pour qu'elles aient les mêmes formes que les lexèmes du Corpus P1. Prenons l'exemple du Dictionnaire P0 de la section §4.1.1.4, nous obtenons ainsi le Dictionnaire P1 ci-dessous :

ขึ้น ชื่อ,ขึ้นชื่อ.V
ค อ ย ล์,คอยล์.N
จุ ฬา,จุฬา.N
ฉ กา จ ฉ ก ร ร จ์,ฉกาจฉกรรจ์.ADJV
ชา ติ พัน ธุ์,ชาติพันธุ์.N
ช่วง สิ ท ธิ์,ช่วงสิทธิ์.V+Law
อัฐิ,อัฐิ.N
เกราะ,เกราะ.N
เจริญ,เจริญ.V
เสิร์จ,เสิร์จ.N
ใคร,ใคร.PRO

4.1.2.1.3.1 Nombre de lexèmes

Nous constatons que le nombre de lexèmes par entrée lexicale dans le Dictionnaire P1 est beaucoup moins important que dans le Dictionnaire P0. Prenons l'exemple du §4.1.1.4 (un Dictionnaire P0), nous le comparons avec le même dictionnaire en version P1 :

Dictionnaire P0	Nombre de lexèmes	Dictionnaire P1	Nombre de lexèmes
ข ึ ้ น ช ื ่ อ	8	ขึ้น ชื่อ	2
ค อ ย ล์	5	ค อ ย ล์	4
จ ุ ฬ า	4	จ ุ ฬ า	2
ฉ ก า จ ฉ ก ร ร จ์	10	ฉ ก า จ ฉ ก ร ร จ์	8
ช า ต ิ พ ั น ธ ุ์	10	ช า ต ิ พ ั น ธ ุ์	4
ช ่ ว ง ส ิ ท ธ ิ์	10	ช่ ว ง ส ิ ท ธ ิ์	4
อ ั ฐ ิ	4	อัฐิ	1
เ ก ร า ะ	5	เกราะ	1
เ จ ร ิ ญ	5	เจริญ	1
เ ส ิ ร ์ จ	6	เสิร์จ	1
ใ ค ร	3	ใคร	1

Tableau 4.2 Nombre de lexèmes des Dictionnaires P0 et P1

Le nombre total de lexèmes des Dictionnaires P1 est diminué de 226 977 (dans la version P0) à 126 054, soit environ 44%[15] de réduction.

4.1.2.1.3.2 Retransfert du DELACF au DELAF

Les entrées du DELACF regroupées en une seule unité sans séparateur devront être retransférées dans le DELAF. Dans l'exemple ci-dessus, les cinq dernières entrées n'ont plus de séparateur, elles doivent être considérées comme des mots simples et replacées dans le DELAF. Les DELAF P1 comptent ainsi 1 918 entrées (1 912 entrées retransférées) et les DELACF P1, 31 749 entrées. L'idéal, ce serait de pouvoir faire revenir les 33 317 mots simples dans le DELAF en laissant les 350 mots composés dans le DELACF.

4.1.2.1.3.3 Suppression de lemme

Les mots replacés dans le DELAF qui ont les entrées identiques aux lemmes n'ont plus besoin de lemme. Dans ce cas, nous pouvons supprimer les lemmes comme ci-dessous. Cette solution est facultative et permet de réduire la taille du fichier dictionnaire.

อัฐิ,.N
เกราะ,.N
เจริญ,.V
เสิร์จ,.N
ใคร,.PRO

Des extraits des Dictionnaires P1 sont présentés dans l'Annexe V.

4.1.2.1.4 Application des Dictionnaires P1

L'application des Dictionnaires P1 sur notre corpus d'essai en version P1 reconnaît plus de mots comme mots simples (138) par rapport au Corpus P0 (3 seulement) mais beaucoup moins comme mots composés (179 contre 403). Au total, elle reconnaît 89 mots[16] de moins.

Notre corpus d'essai contient en fait 142 mots simples (*cf. §4.1.1.5*), c'est-à-dire qu'il y a 264 mots[17] reconnus à tort (bruit) dans la version P0. Lorsque nous reconnaissons 89 mots de moins dans la version P1, cela veut dire que nous pouvons réduire 34%[18] de bruit par

[15] (126054-226977)x100/226977 = -44,46%
[16] (138+179)-(3+403) = -89
[17] 406-142 = 264
[18] Le taux de réduction du bruit est calculé par : $\left(\frac{Nombre\ de\ bruit\ réduit}{Nombre\ de\ bruit\ d'origine}\right) \times 100\ \%$
Dans le cas ci-dessus, le taux de réduction du bruit = 89x100/264 = 33,71%

rapport au nombre de bruit dans la version P0. De plus, si nous n'y appliquons que les dictionnaires z1 (dictionnaires des mots courants), le chiffre est augmenté de 89 à 104 et le taux de réduction du bruit s'élève à 39%[19]. Dans tous les cas, le taux de silence reste toujours à 0%.

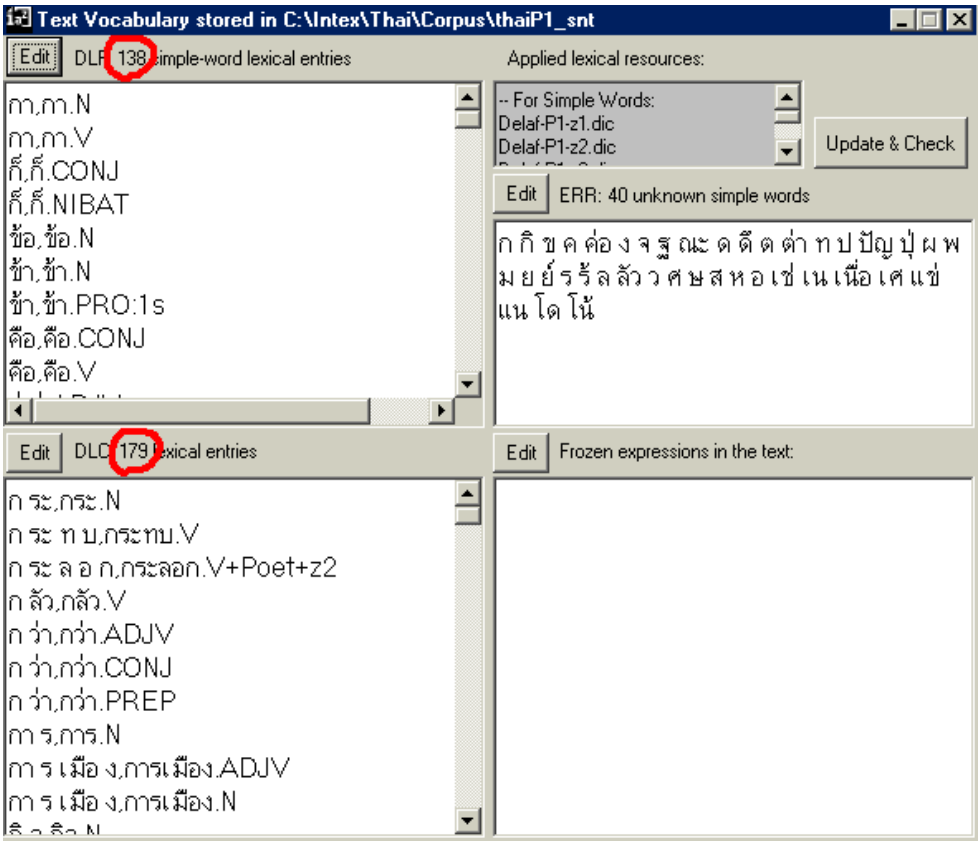

Figure 4.47 Vocabulaire du texte d'un Corpus P1

La réduction du bruit varie selon les textes. Prenons l'exemple du §4.1.1.6, la séquence « จะเข้ามา » est composée de 3 mots simples : « จะ », « เข้า » et « มา ». La méthode par caractères transforme cette séquence en « จ ะ เ ข ้ า ม า », ce qui correspond à 14 entrées des dictionnaires, soit 79%[20] de bruit. Cependant, grâce au graphe « Replace 1 », la séquence est transformée en « จะ เข้า มา » et il ne reste que 7 entrées correspondantes, soit 64%[21] de la réduction du bruit. En fait, les 36% de bruit qui restent sont du bruit au niveau lexical (mots à plusieurs parties de discours) et ne peuvent pas être éliminés dans cette phase.

D'après nos expériences sur différents documents, le bruit dans la version P1 est diminué de 15-40% environ par rapport à celui dans la version P0.

[19] 104x100/264 = 39,39%
[20] (14-3)x100/14 = 78,57%
[21] (14-7)x100/(14-3) = 63,63%

4.1.2.2 Regroupement des segments inséparables

Par le même principe que les caractères inséparables, certains segments, regroupés par « Replace 1 » sont encore considérés comme inséparables.

4.1.2.2.1 Graphe de regroupement « Replace 2 »

Le graphe « Replace 2 » sert à parcourir les Corpus P1 et Dictionnaires P1 afin de reconnaître des segments inséparables et de les regrouper. Les résultats obtenus après l'application du graphe « Replace 2 » sont les Corpus P2 et Dictionnaires P2.

4.1.2.2.1.1 Sous-graphe

- Nous commençons par décrire toutes les formes possibles des phonèmes muets (Karant), considérés comme segments inséparables. Ces phonèmes sont regroupés dans la phase précédente par les graphes « Replace 1.23 » et « Replace 1.24 ».

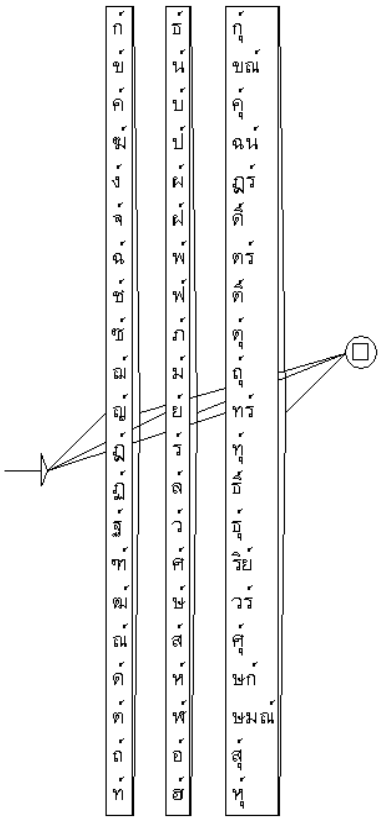

Figure 4.48 Graphe « Karant »

4.1.2.2.1.2 Graphes de regroupement des phonèmes muets

Le phonème muet est en fait un composant non-autonome dans une syllabe (*cf. §2.2.1*). Il s'écrit obligatoirement avec une syllabe principale, le noyau de phonème muet. En conséquence, nous pouvons les regrouper.

Remarque : Lorsqu'un phonème muet est présent, on connaît la limite à droite de la syllabe et on cherche à regrouper vers la gauche des composants minimaux qui peuvent former cette syllabe.

- La forme générale d'une syllabe comprenant un phonème muet est composée d'une syllabe principale, suivie éventuellement d'une consonne (ou le « น่ ») et terminée par un phonème muet.

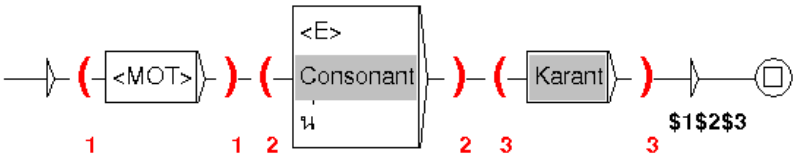

Figure 4.49 Graphe « Replace 2.1 »

Remarque : Le symbole formel <MOT> signifie une séquence de lettres entre deux séparateurs (*cf. Tableau 1.1*). Lorsque le graphe ci-dessus est appliqué aux textes P1, ce symbole est équivalent à un segment regroupé par « Replace 1 ».

Discussion : Une seule consonne ne peut pas être le noyau de la syllabe comportant un phonème muet. Lorsqu'on trouve une consonne isolée devant un phonème muet, c'est soit un autre élément du phonème muet qui ne pouvait pas être auparavant regroupé, soit la consonne finale de la syllabe à gauche, soit la deuxième consonne d'un agglomérat consonantique qui fait partie de la syllabe à gauche. Cette dernière est donc le noyau réel. Dans le troisième cas, cette consonne peut prendre une marque de ton (mais nous ne trouvons qu'un seul exemple : le « น่ » dans le mot « เสน่ห์ »). Nous pouvons enfin regrouper tous les segments considérés comme inséparables pour avoir une unité plus grande.

- Les mots comprenant la voyelle « -อ » [ɔ:], une consonne finale et un phonème muet sont des emprunts d'origine européenne. Puisqu'ils ne sont pas nombreux, nous pouvons les décrire dans le graphe suivant.

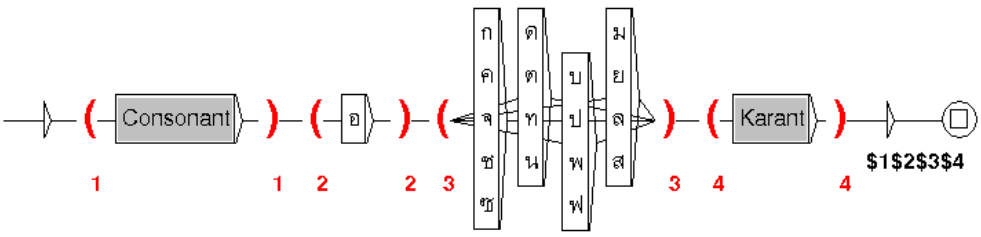

Figure 4.50 Graphe « Replace 2.2 »

- Le « รร » [rɔ́hăn], une forme modifiée de la voyelle « -ะ » [a] (*cf. §2.2.2.3.1*) est souvent ambigu à moins qu'il ne soit écrit avec un phonème muet. Dans ce cas, une des 9 consonnes finales ci-dessous peut s'y joindre.

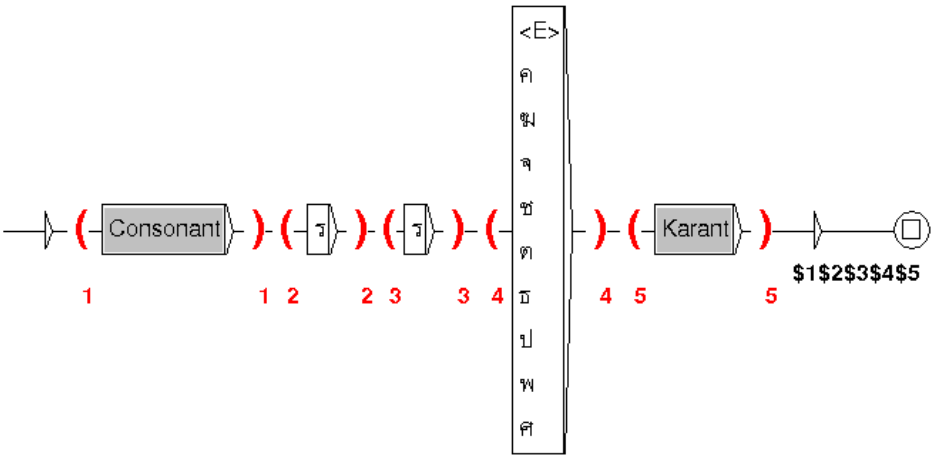

Figure 4.51 Graphe « Replace 2.3 »

4.1.2.2.1.3 Graphes de regroupement des signes pāli-sanskrits

Les 3 signes pāli-sanskrits (*cf. §2.1.4*) sont des caractères inséparables qui s'emploient avec des segments noyaux.

- Les signes « ฺ » [phinthú] et « ฺ » [níkkháhìt] sont écrits respectivement au-dessous et au-dessus d'une consonne ou d'une syllabe, représentée par <MOT>. Nous les regroupons ensemble.

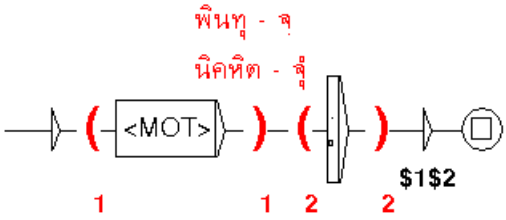

Figure 4.52 Graphe « Replace 2.4 »

Remarque : Le symbole <MOT> reconnaît dans le Corpus P1, une syllabe isolée aussi bien qu'une consonne isolée.

Discussion : Le signe « ฺ » [phinthú] a 2 usages : en pāli, il s'écrit sous une consonne pour indiquer que c'est la consonne finale ; en sanskrit, il s'écrit sous la première consonne d'un agglomérat consonantique (*cf. Tableau 2.6*). Cependant, les deux usages se mélangent souvent, y compris à l'intérieur d'un même mot. Par exemple, dans le mot « คนฺตฺวา »

[khantwaː], le premier signe indique la consonne finale « น » tandis que le deuxième indique l'agglomérat consonantique « ตว ». Puisqu'INTEX n'est pas en mesure de distinguer les deux usages, nous ne regroupons qu'une unité minimale comme dans le graphe ci-dessus.

- Le signe « ์ » [jaːmákkaːn] s'écrit sur la première consonne d'un agglomérat consonantique, nous pouvons donc regrouper les deux consonnes ensemble.

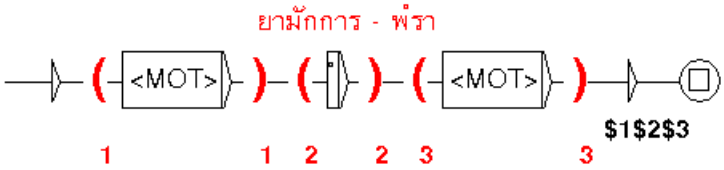

Figure 4.53 Graphe « Replace 2.5 »

Remarque : Chaque consonne peut être déjà regroupée avec d'autres caractères par « Replace 1 », c'est la raison pour laquelle nous employons le symbole <MOT> à la place de la consonne.

4.1.2.2.1.4 Exceptions avant « Replace 3 »

Pour éviter certaines erreurs dues à l'application de « Replace 3 » (expliqué plus loin dans ce chapitre), nous devons regrouper dans « Replace 2 » quelques segments particuliers.

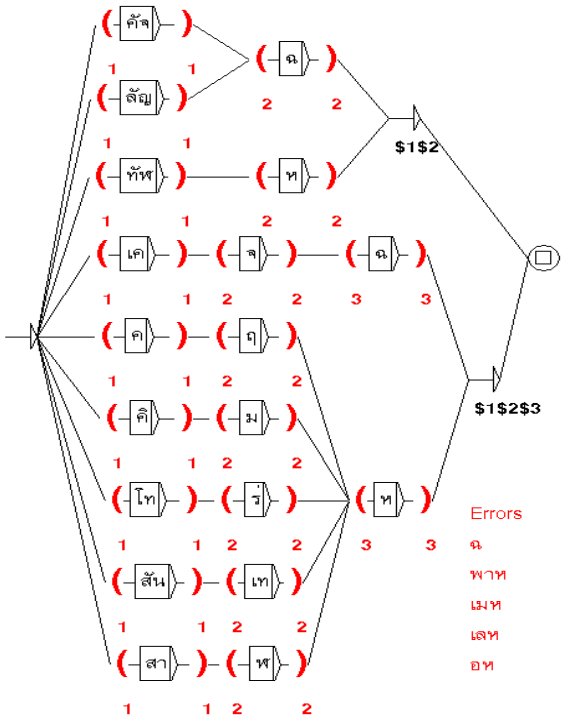

Figure 4.54 Graphe « Replace 2.6 »

Tous les graphes appelés par « Replace 2 » sont présentés dans l'Annexe VI.

4.1.2.2.2 Corpus P2

Nous appliquons le graphe « Replace 2 » aux « Corpus P1 » devenant ainsi « Corpus P2 » comme dans l'exemple ci-dessous. Nous remarquons que le nombre de formes simples est très légèrement réduit (d'un seulement dans notre corpus d'essai) mais cela varie d'un corpus à l'autre selon les composants utilisés dans les textes.

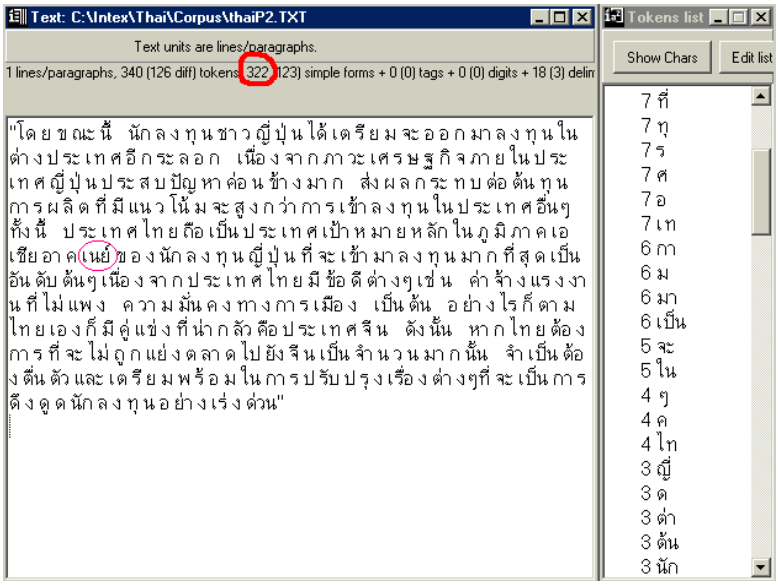

Figure 4.55 Corpus P2 et le lexème regroupé par « Replace 2 »

4.1.2.2.3 Dictionnaires P2

Après l'application du graphe « Replace 2 » aux « Dictionnaires P1 », ces derniers se transforment en « Dictionnaires P2 ». Néanmoins, la transformation ne modifie que des entrées comprenant des segments inséparables. Prenons l'exemple du §4.1.2.1.3 : parmi les 11 entrées proposées, seules les 4 ci-dessous sont concernées et modifiées par « Replace 2 ».

คอยล์,คอยล์.N

ฉ กา จ ฉ กรรจ์,ฉกาจฉกรรจ์.ADJV

ชา ติ พันธุ์,ชาติพันธุ์.N

ช่วง สิทธิ์,ช่วงสิทธิ์.V+Law

En comparant les entrées de chaque version, nous constatons une réduction importante du nombre de lexèmes :

Dictionnaire P0	Nombre de lexèmes	Dictionnaire P1	Nombre de lexèmes	Dictionnaire P2	Nombre de lexèmes
ขี้ น ชื่ อ	8	ขึ้น ชื่อ	2	ขึ้น ชื่อ	2
ค อ ย ล์	5	ค อ ย ล์	4	คอยล์	1
จุ พา	4	จุ พา	2	จุ พา	2

ฉ ก า จ ฉ ก ร ร จ์	10	ฉ ก า จ ฉ ก ร ร จ์	8	ฉ ก า จ ฉ ก รร จ์	5
ช า ติ พั น ธุ์	10	ช า ติ พัน ธุ์	4	ช า ติ พันธุ์	3
ช่ ว ง สิ ท ธิ์	10	ช่ว ง สิ ท ธิ์	4	ช่ว ง สิทธิ์	2
อั ฐิ	4	อัฐิ	1	อัฐิ	1
เ ก ร า ะ	5	เกราะ	1	เกราะ	1
เ จ ริ ญ	5	เจริญ	1	เจริญ	1
เ สิ ร์ จ	6	เสิร์จ	1	เสิร์จ	1
ใ ค ร	3	ใคร	1	ใคร	1

Tableau 4.3 Nombre de lexèmes des Dictionnaires P0, P1 et P2

Le nombre total de lexèmes des Dictionnaires P2 est diminué de 126 054 (dans la version P1) à 122 235, soit environ 3%[22] de réduction. Les DELAF P2 comptent 2 240 entrées (322 entrées retransférées) et les DELACF P2, 31 427 entrées.

Des extraits des Dictionnaires P2 sont présentés dans l'Annexe VII.

4.1.2.2.4 Application des Dictionnaires P2

L'application des Dictionnaires P2 sur notre corpus d'essai P2 donne le même résultat que dans la version P1. En effet, la réduction de lexèmes dans le corpus d'essai n'est pas assez importante pour avoir un effet sur le bruit.

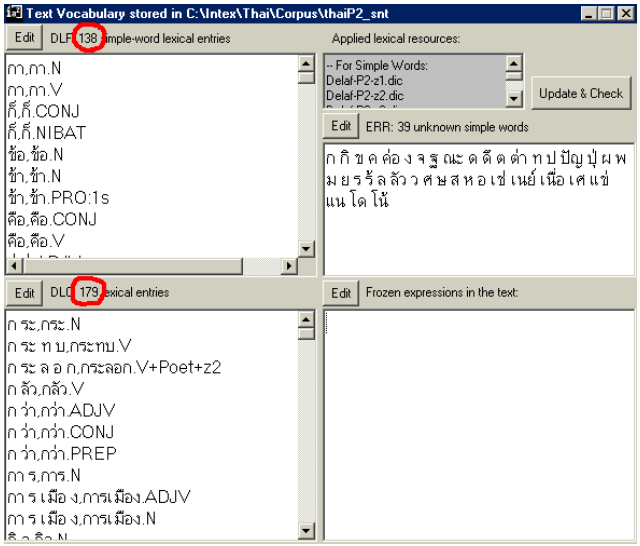

Figure 4.56 Vocabulaire du texte d'un Corpus P2

D'après nos expériences sur d'autres documents, le bruit dans la version P2 est diminué de 0,5-5% environ par rapport à celui dans la version P1.

[22] (122235-126054)x100/126054 = -3,03%

4.1.2.3 Regroupement des syllabes inséparables

Après les graphes « Replace 1 » et « Replace 2 », nous pouvons aller plus loin en travaillant avec certaines syllabes inséparables. En effet, on peut chercher à mieux exploiter l'idée des graphes de regroupement des phonèmes muets (*cf. §4.1.2.2.1.2*) qui trouvent d'abord une limite à droite d'une syllabe et essaient de regrouper vers la gauche des composants minimaux. Nous essayerons donc dans cette phase de trouver d'autres limites de syllabes et de les regrouper avec des composants minimaux afin d'avoir des unités plus grandes.

4.1.2.3.1 Graphe de regroupement « Replace 3 »

Le graphe « Replace 3 » est appliqué sur les Corpus P2 et Dictionnaires P2 pour qu'ils évoluent en Corpus P3 et Dictionnaires P3.

4.1.2.3.1.1 Sous-graphes

- En analysant toutes les syllabes dans nos dictionnaires, nous constatons que les 6 lettres suivantes ne peuvent être ni la consonne finale, ni la deuxième consonne des agglomérats consonantiques. Elles sont toujours utilisées pour commencer une nouvelle syllabe. Elles sont donc considérées comme des limites à gauche.

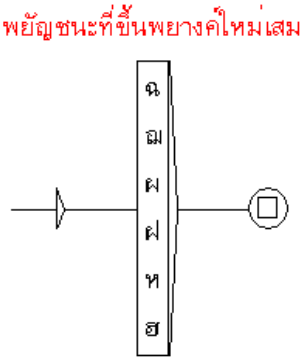

Figure 4.57 Graphe « Prefix-Cha »

- À part les phonèmes muets, nous ne trouvons pas d'autres limites à droite. Or, nous constatons que dans nos dictionnaires, il existe 3 lettres (« ฏ » [tɔː pàtàk], « ฒ » [thɔː phûːthâw] et « ฬ » [lɔː cùlaː]) qui ne servent jamais à commencer un mot. Les syllabes ci-dessous qui commencent toutes avec l'une de ces 3 lettres et qui sont regroupées par « Replace 1 » et « Replace 2 », ne peuvent pas non plus débuter un mot. En fait, ce sont juste des composants appartenant aux mêmes mots que des syllabes à gauche.

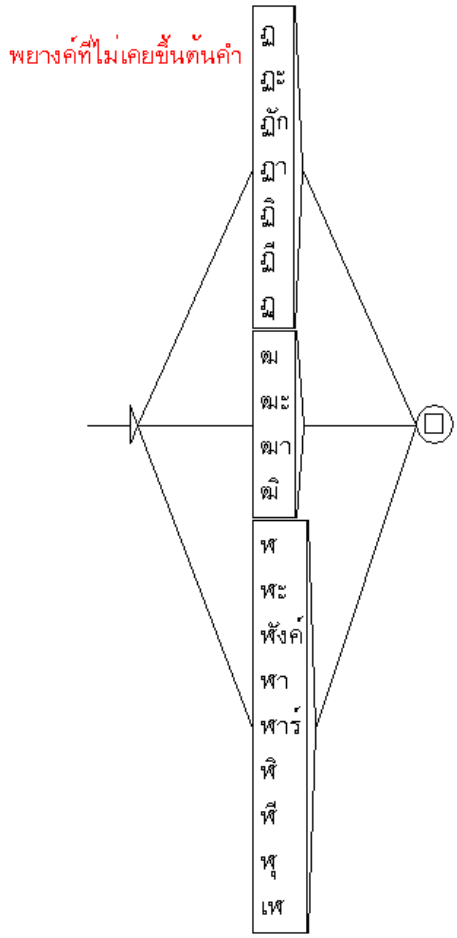

Figure 4.58 Graphe « Suffix-Syl »

- Le graphe ci-dessous décrit toutes les formes possibles des consonnes ambiguës (*cf. Figure 4.11*) qui sont regroupées avec des marques de tons par le graphe « Replace 1.22 » (*cf. Figure 4.37*).

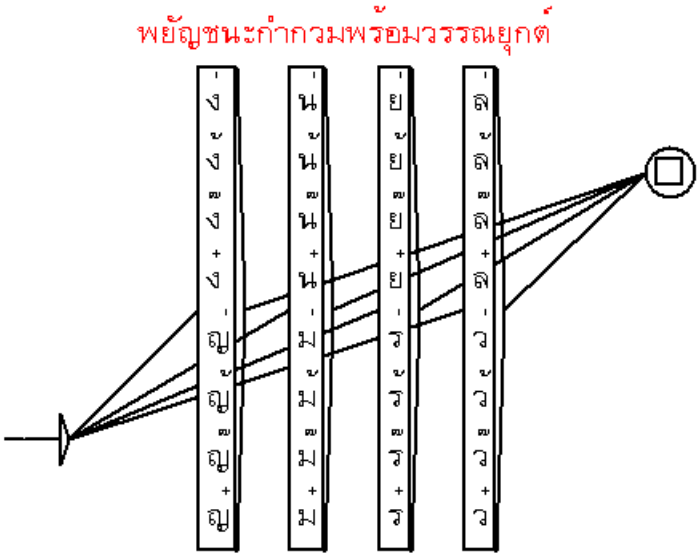

Figure 4.59 Graphe « Consonant-3-withTone »

4.1.2.3.1.2 Graphes principaux

- Les syllabes qui ne peuvent pas commencer un mot sont en fait des syllabes inséparables appartenant au même mot que la ou les syllabe(s) à gauche. Nous les regroupons par le graphe suivant.

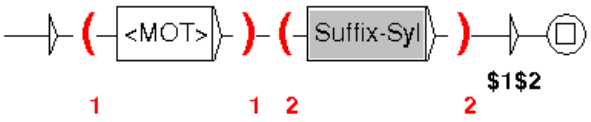

Figure 4.60 Graphe « Replace 3.1 »

- Lorsque nous trouvons une limite de syllabe à gauche (lettre qui commence toujours une nouvelle syllabe), nous essayerons de regrouper, à partir de cette limite vers la droite, des composants minimaux. Ces derniers peuvent être une syllabe regroupée par « Replace 1 » et « Replace 2 ». De plus, le graphe précédent peut éventuellement s'y joindre.

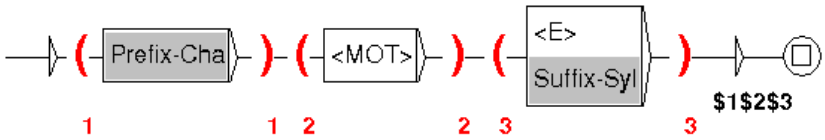

Figure 4.61 Graphe « Replace 3.2 »

- Si le deuxième élément du graphe ci-dessus est le « ว » [wɔː wɛ̌ːn], il peut correspondre soit à la forme réduite de la voyelle « ◌ัว » [uaː] qui prend obligatoirement une consonne finale (*cf. §2.2.2.2.2*), soit à la deuxième consonne d'un agglomérat consonantique. Dans le deuxième cas, le « อ » [ʔɔː ʔàːŋ] peut s'y joindre pour créer une syllabe de voyelle « -อ » [ɔː].

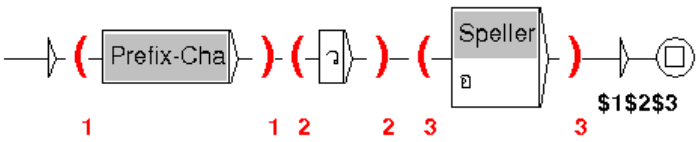

Figure 4.62 Graphe « Replace 3.3 »

- Si le deuxième élément de « Replace 3.2 » est une consonne ambiguë écrite avec une marque de ton, cet élément est certainement la deuxième consonne d'un agglomérat consonantique formé avec la consonne de gauche. Nous retombons sur le cas du graphe « Replace 1.18 » (*cf. Figure 4.33*) qui peut prendre soit le « อ »

[ʔɔː ʔàːŋ] pour former une syllabe de voyelle « -อ » [ɔː], soit une consonne finale (autre que le « ว » [wɔː wɛ̌ːn]) pour former une syllabe de voyelle « โ-ะ » [o] (*cf. §2.2.2.1.3*).

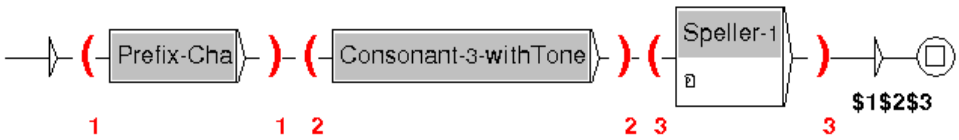

Figure 4.63 Graphe « Replace 3.4 »

- Si le troisième élément du graphe ci-dessus est le « ว » [wɔː wɛ̌ːn], il s'agit encore de la forme réduite de la voyelle « ◌ัว » [uaː] qui prend obligatoirement une consonne finale (*cf. §2.2.2.2.2*).

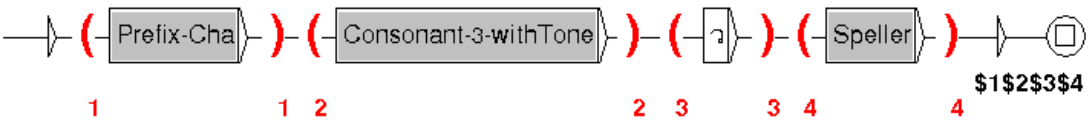

Figure 4.64 Graphe « Replace 3.5 »

Discussion : Le graphe « Replace 3 » peut causer des erreurs pour des mots dont la dernière syllabe s'écrit avec une des consonnes du graphe « Prefix-Cha » (*cf. Figure 4.57*) et une voyelle de forme supprimée ([a] ou [ɔː]) tels « คัจฉ » [khátchà] et « ทัพห » [thanhà]. En effet, le graphe « Replace 1 » va transformer ces deux mots en « คัจ ฉ » et « ทัพ ห » puis le graphe « Replace 3 » peut regrouper le « ฉ » [chɔ̌ː chǐŋ] et le « ห » [hɔ̌ː hǐːp] avec des syllabes à droite (comme « คัจ ฉxxx » et « ทัพ หxx »). Ces deux mots ne seront alors plus reconnus par nos dictionnaires.

Pour éviter cela, nous avons créé dans la phase « Replace 2 » un graphe d'exceptions (*cf. §4.1.2.2.1.4*) qui regroupe d'abord des cas pouvant poser le problème. Cependant, nous avons sacrifié les 5 mots suivants : « ฉ » [chɔ̌ː], « พาห » [phaːhà], « เมห » [meːhà], « เลห » [leːhà] et « อห » [ʔàhà] qui ne sont pas regroupés et peuvent ne pas être reconnus. En effet, ce ne sont pas des mots courants ; de plus, les syllabes initiales de ces mots sont fréquentes et elles peuvent très bien ne pas appartenir au même mot que la consonne du graphe « Prefix-Cha ». Par exemple, lorsqu'on trouve une séquence « พาหxx », il est plus probable qu'elle soit à analyser « พา »+« หxx » que « พาห »+« xx ». En conséquence, ces 5 mots-là ne seront pas reconnus à moins qu'ils ne soient directement suivis par un séparateur.

Tous les graphes appelés par « Replace 3 » sont présentés dans l'Annexe VIII.

4.1.2.3.2 Corpus P3

Le graphe « Replace 3 » transforme un « Corpus P2 » en « Corpus P3 » dans lequel des syllabes inséparables sont regroupées. Cela veut dire que le nombre de lexèmes du corpus diminue encore. Sur notre corpus d'essai, le nombre de formes simples réduit de 322 (dans la version P2) à 318 (dans cette version), soit environ 1%[23] de réduction.

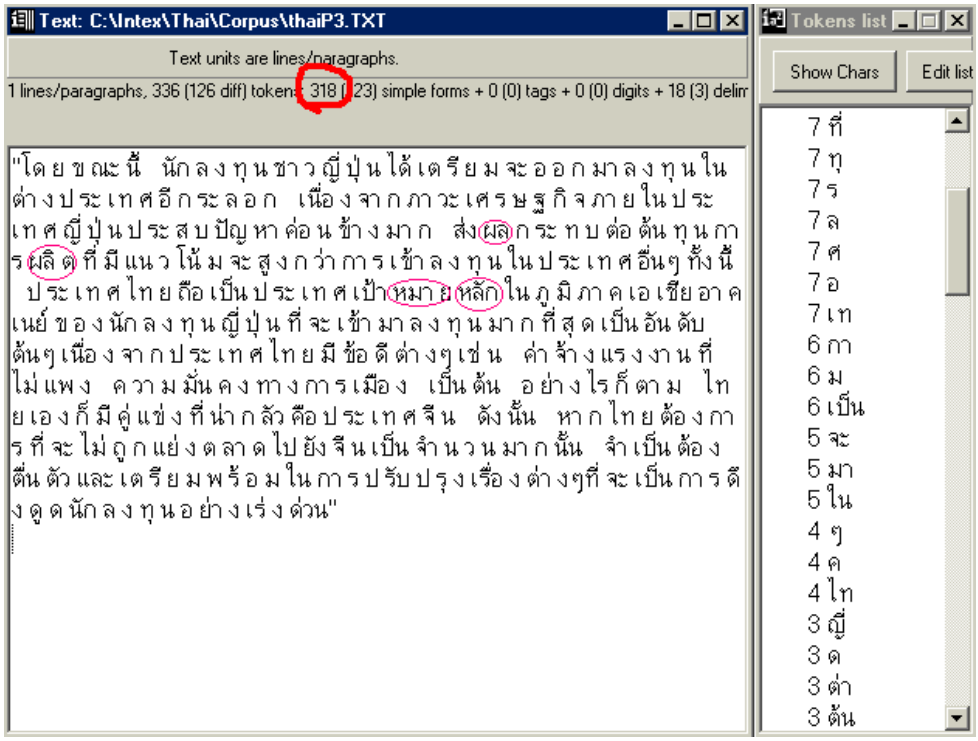

Figure 4.65 Corpus P3 et les lexèmes regroupés par « Replace 3 »

4.1.2.3.3 Dictionnaires P3

Seules les entrées des « Dictionnaires P2 » comprenant des syllabes inséparables sont concernées par l'application du graphe « Replace 3 ». Cette dernière fait évoluer les « Dictionnaires P2 » en « Dictionnaires P3 ». Dans l'exemple du §4.1.2.2.3, parmi les 11 entrées proposées, seules les 2 ci-dessous sont modifiées.

จุฬา,จุฬา.N

ฉกา จ ฉกรรจ์,ฉกาจฉกรรจ์.ADJV

Voici la table comparative des entrées dans chaque version du dictionnaire. Les éléments écrits en gras sont des constituants concernés par le regroupement dans la phase correspondante.

[23] (318-322)x100/322 = -1,24%

Dictionnaire P0	Nombre de lexèmes	Dictionnaire P1	Nombre de lexèmes	Dictionnaire P2	Nombre de lexèmes	Dictionnaire P3	Nombre de lexèmes
ขี้นชื่อ	8	**ขึ้น ชื่อ**	**2**	ขึ้น ชื่อ	2	ขึ้น ชื่อ	2
คอยล์	5	คอยล์	**4**	**คอยล์**	**1**	คอยล์	1
จุฬา	4	**จุฬา**	**2**	จุฬา	2	**จุฬา**	**1**
ฉกาจฉกรรจ์	10	ฉกาจ ฉกรรจ์	**8**	ฉกาจ ฉกรรจ์	**5**	**ฉกาจ ฉกรรจ์**	**3**
ชาติพันธุ์	10	**ชาติ พันธุ์**	**4**	ชาติ **พันธุ์**	**3**	ชาติ พันธุ์	3
ช่วงสิทธิ์	10	**ช่วง สิทธิ์**	**4**	ช่วง **สิทธิ์**	**2**	ช่วง สิทธิ์	2
อัฐิ	4	**อัฐิ**	**1**	อัฐิ	1	อัฐิ	1
เกราะ	5	**เกราะ**	**1**	เกราะ	1	เกราะ	1
เจริญ	5	**เจริญ**	**1**	เจริญ	1	เจริญ	1
เสิร์จ	6	**เสิร์จ**	**1**	เสิร์จ	1	เสิร์จ	1
ใคร	3	**ใคร**	**1**	ใคร	1	ใคร	1

Tableau 4.4 Nombre de lexèmes des Dictionnaires P0, P1, P2 et P3

Le nombre total de lexèmes des Dictionnaires P3 est diminué de 122 235 (dans la version P2) à 119 129, soit environ 3%[24] de réduction. Les DELAF P3 comptent 2 481 entrées (241 entrées retransférées) et les DELACF P3, 31 186 entrées.

Des extraits des Dictionnaires P3 sont présentés dans l'Annexe IX.

4.1.2.3.4 Application des Dictionnaires P3

L'application des Dictionnaires P3 sur le corpus d'essai en version P3 reconnaît plus de mots comme mots simples (140 au lieu de 138) mais moins comme mots composés (171 au lieu de 179). Au total, elle reconnaît 6 mots[25] de moins.

Nous savons qu'il reste 175 mots[26] considérés comme du bruit dans la version P1 et P2 (*cf. §4.1.2.1.4*). Ainsi, nous pouvons calculer le taux de réduction du bruit dans la version P3 qui est de 3%[27] environ par rapport à la quantité de bruit dans la version P2. Quant au taux de silence, il reste toujours à 0% parce que les 142 mots d'origine sont toujours reconnus.

D'après nos expériences sur différents documents, le bruit dans la version P3 est diminué de 1-5% environ par rapport à celui dans la version P2.

[24] (119129-122235)x100/122235 = -2,54%
[25] (140+171)-(138+179) = -6
[26] 264-89 = 175
[27] 6x100/175 = 3,42%

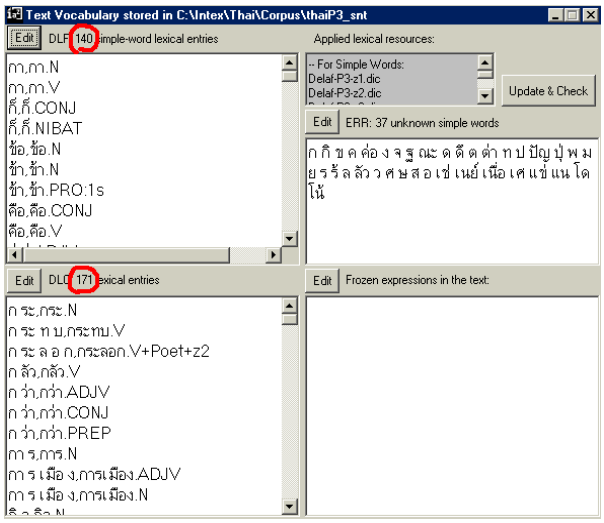

Figure 4.66 Vocabulaire du texte d'un Corpus P3

4.1.3 Conclusion

La méthode par caractères permet de reconnaître toutes les possibilités des mots codés dans les dictionnaires, c'est-à-dire que le taux de silence est très bas (près de 0%). Cependant, elle génère également une grande quantité de bruit (qui peut être réduit d'environ 2-15% par la réorganisation des dictionnaires). Pour mieux gérer le problème du bruit, nous avons proposé une méthode par syllabes qui comporte 3 phases appliquées en cascade, c'est-à-dire que chaque phase est appliquée au résultat de la phase précédente. Le dernier résultat obtenu permet de réduire jusqu'à 50% le nombre de lexèmes dans les textes et jusqu'à 40% la quantité de bruit par rapport à la méthode par caractères. De plus, en combinant avec la réorganisation des dictionnaires, le taux de réduction du bruit peut s'élever jusqu'à 50%, tout en gardant un taux de silence très bas. En fait, le silence dans cette dernière expérience n'est pas dû à la segmentation mais à l'absence des mots dans les DELAF et DELACF. Le Chapitre 5 détaillera davantage sur l'évaluation de nos méthodes.

On peut imaginer d'incorporer les 3 phases de la méthode par syllabes en un seul graphe de regroupement mais cela ne donne pas vraiment d'avantage ni en taille, ni en terme de rapidité. De plus, le graphe serait très grand et très complexe, ce qui entraînerait facilement des erreurs humaines lors de la construction et de la maintenance. Si l'on veut vraiment simplifier la tâche de regroupement, travailler en mode « Console », en appelant les 3 graphes de regroupement dans un même script (*cf. §4.3.2*), est un choix préférable.

Nous pouvons aussi, après « Replace 3 », créer « Replace 4 », « Replace 5 », etc. en travaillant davantage sur les syllabes non-autonomes : celles qui ne sont jamais employées seules. Cependant, une étude approfondie est nécessaire afin de ne pas introduire d'erreurs.

4.2 Segmentation en phrases

Faute de point indiquant la fin de phrase comme dans les langues européennes, la reconnaissance des phrases thaï n'est pas évidente. D'autant plus que l'emploi du point-virgule, du point d'exclamation et du point d'interrogation servant à déterminer la fin de phrase n'est pas non plus obligatoire. En effet, deux phrases différentes sont généralement séparées par un simple espace blanc. L'espace blanc est donc ambigu, puisqu'il peut également être utilisé au même titre que d'autres signes de ponctuation (*cf. §0.3.3*, *§1.5*).

En fait, la reconnaissance des phrases thaï se fait plutôt par la structure des phrases, donc après avoir reconnu bien plus que les mots et leur partie du discours. Cela exige une analyse complète de la structure des phrases thaï. Cependant, à cause de leur invariabilité, certains mots thaï ont une même forme pour le nom, le verbe et l'adjectif-verbe. En l'absence de particule ou de postposition servant à indiquer des rapports syntaxiques, comme cela existe dans les langues fléchies et agglutinantes, les parties de discours des mots thaï ne sont pas déterminées de manière univoque. Cela entraîne une difficulté de reconnaissance des phrases, d'autant plus que les structures des phrases thaï sont très variées. Il existe, par exemple, des cas où certains éléments peuvent être inversés et même des cas où certains éléments peuvent être facultatifs. Une étude approfondie sur les phrases thaï est nécessaire avant de parvenir à une segmentation des phrases par la syntaxe.

À cause de ces restrictions, il est préférable d'envisager d'autres procédures.

Nous proposons ci-dessous 2 méthodes formelles qui peuvent être employées ensemble : méthode par la ponctuation et méthode par mots-clés. Les transducteurs de segmentation des 2 méthodes seront appliqués en mode « insertion » par le premier module de normalisation (*cf. §1.2.2.1*).

4.2.1 Méthode par la ponctuation

Les signes de ponctuation d'origine européenne tels que le point d'exclamation, le point d'interrogation, les points de suspension et le point-virgule, même s'ils sont facultatifs en thaï, peuvent servir à déterminer les fins de phrases. Cependant, le signe qui sert le plus à segmenter les textes en phrases thaï est le nouveau paragraphe, symbolisé par <^>. En effet, un bon style de rédaction d'un texte en thaï favorise une phrase longue et complexe avec beaucoup de conjonctions. Il arrive souvent qu'un paragraphe contienne seulement une ou deux phrases. Si l'on arrive à détecter tous les débuts de paragraphe, on pourra reconnaître un certain nombre des phrases thaï.

- Nous utilisons le graphe ci-dessous pour détecter les signes de ponctuation et insérer le séparateur de phrases « {S} ». En effet, les signes de fin de phrase (autre que le point-virgule) peuvent être directement suivis par d'autres signes de ponctuation, tels que des parenthèses fermantes, des guillemets fermants, etc. Ils sont représentés par le symbole « #<PNC> »[28] en boucle. Les nouvelles phrases qui suivent peuvent également commencer par plusieurs signes de ponctuation tels que des parenthèses ouvrantes, des guillemets ouvrants, etc. Ils sont représentés par le symbole « <PNC># » en boucle.

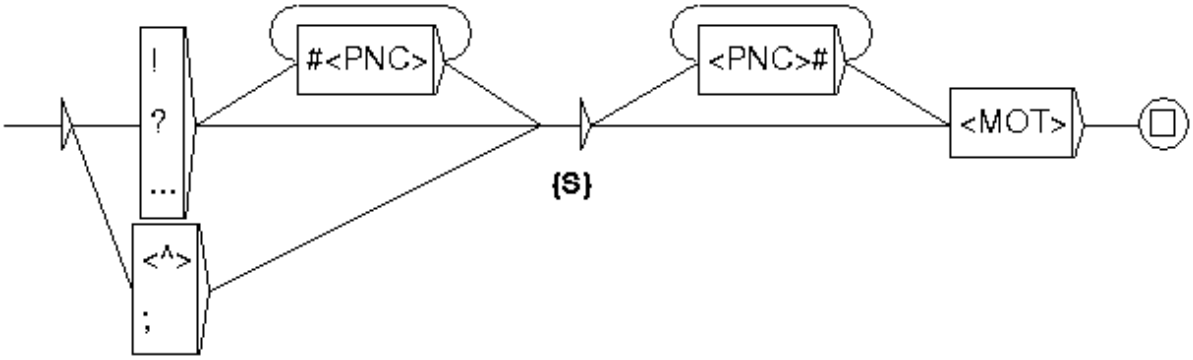

Figure 4.67 Graphe « Sentence-1 »

Rappel : Les espaces blancs devant les parenthèses fermantes et derrière les parenthèses ouvrantes (s'ils existent) sont préalablement supprimés par les Règles N° 2 et 3 de l'Algorithme 4.2.

4.2.2 Méthode par mots-clés

Compte tenu du fait que certains mots thaï sont toujours écrits au début ou à la fin des phrases, ils peuvent être employés comme mots-clés pour diviser une séquence de mots en deux phrases différentes, en ajoutant entre elles un séparateur de phrases. Nous rappelons que deux phrases thaï sont généralement séparées par un espace blanc qui se transforme en blanc insécable par la Règle N° 4 de l'Algorithme 4.2[29].

- Lorsque nous trouvons un mot suivi d'un blanc insécable et d'un mot-clé de début de phrase, nous insérons le séparateur de phrases devant ce mot-clé. En revanche, s'il existe des signes de ponctuation devant le mot-clé, le blanc insécable sera

[28] Le symbole <PNC> signifie un séparateur (*cf. Tableau 1.1*) tandis que le signe « # » interdit l'apparition du blanc (*cf. §1.1.1*).
[29] À l'exception des espaces blancs qui ne sont pas écrits entre deux caractères thaï, ils sont laissés tels quels.

absent et nous devons insérer le séparateur de phrases de la manière présentée dans le graphe ci-dessous :

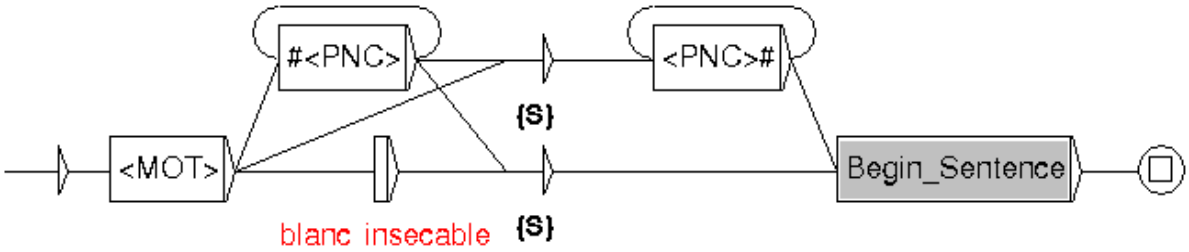

Figure 4.68 Graphe « Sentence-2 »

- Lorsqu'un mot-clé de fin de phrase est suivi d'un blanc insécable et d'un autre mot, nous insérons le séparateur de phrases après le blanc insécable. En revanche, s'il existe des signes de ponctuation derrière le mot-clé, le blanc insécable sera absent et nous devons insérer le séparateur de phrases de la manière présentée dans le graphe ci-dessous :

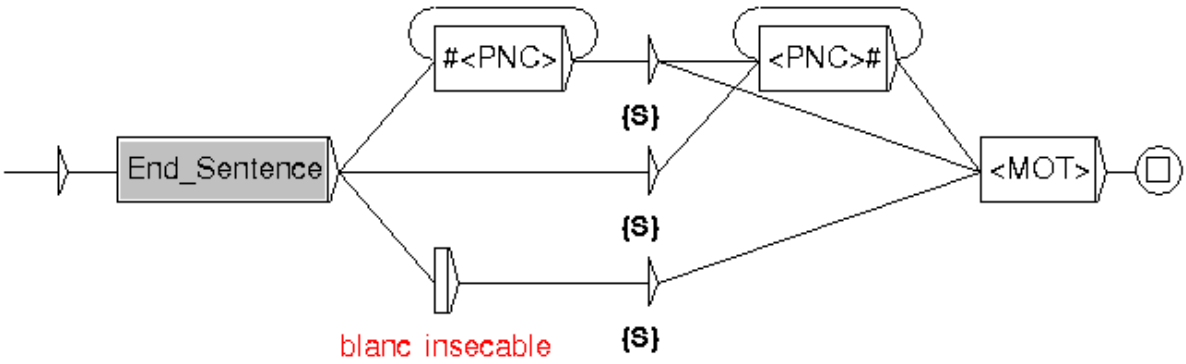

Figure 4.69 Graphe « Sentence-3 »

Remarque : Certains mots-clés ont les mêmes formes que d'autres mots dans les phrases. Seuls les séparateurs devant ou derrière ces mots-clés distinguent les deux usages car ils montrent l'intention des auteurs de commencer une nouvelle phrase. C'est pourquoi nous prenons en considération le blanc insécable et d'autres signes de ponctuation dans les graphes. Cependant, si les mots devant les mots-clés de début de phrase ou derrière ceux de fin de phrase ne sont pas écrits en caractères thaï, les blancs insécables seront absents et les deux graphes ci-dessus ne fonctionneront pas correctement. Mais ce cas est vraiment exceptionnel.

Tous les graphes de « Sentence » sont présentés dans l'Annexe X.

4.2.3 Conclusion

Les méthodes par la ponctuation et par mots-clés sont indépendantes. En combinant les deux, le taux de reconnaissance des phrases s'élève tandis que le taux de silence se réduit. Le Chapitre 5 détaillera davantage sur l'évaluation de nos méthodes.

Les mots-clés de début et de fin de phrase dépendent des types de documents à analyser. Ceux donnés dans l'Annexe X sont des mots quotidiens et sont sujets à discussion. Les utilisateurs peuvent facilement modifier ces listes. Néanmoins, il faut préparer des versions différentes de mots-clés (P0, P1, P2 et P3) selon les corpus utilisés (Corpus P0, Corpus P1, Corpus P2 ou Corpus P3). Pour cela, les mots-clés doivent subir les mêmes modifications que les Corpus.

Afin d'éviter cette complication, nous pouvons appliquer tous les graphes de segmentation en mode « Console » et dans ce cas, seule la version P0 des mots-clés suffit pour segmenter toutes les versions du corpus (*voir §4.3.2*).

4.3 Application des graphes de segmentation

Nous avons 2 possibilités pour appliquer les graphes de segmentation sur un corpus : en mode « Graphique » et en mode « Console » (ligne de commande).

4.3.1 Mode « Graphique »

La normalisation d'INTEX (*cf. §1.2.2*) en mode graphique dispose d'une restriction : ses 3 modules n'acceptent chacun qu'un seul graphe (ou fichier) à la fois. Lorsqu'on inclut la segmentation en phrases (le 1er module) dans la normalisation, on ne peut inclure qu'un seul graphe « Replace » dans le 3ème. Puis INTEX crée un fichier « snt » et le menu « Preprocessing » disparaît, on ne peut donc plus appliquer d'autres graphes de normalisation. Cette restriction nous interdit d'appliquer 3 graphes de remplacement après la segmentation en phrases.

Étant donné que chaque module est facultatif, une solution simple est de ne pas appliquer le 1er module lors des premières normalisations. Ainsi le menu « Preprocessing » ne disparaît pas et on peut l'appeler plusieurs fois en appliquant un graphe « Replace » à chaque appel. Répétons ainsi jusqu'à ce que l'on obtienne la version de corpus désirée et appliquons enfin le graphe « Sentence » pour obtenir un fichier « snt ». Ceci implique qu'on doit posséder les versions différentes de « Sentence » (P0, P1, P2 et P3 selon la version de mots-clés) pour les appliquer aux différentes versions du corpus.

4.3.2 Mode « Console »

Une autre solution plus flexible est de créer un fichier script (qui fonctionne en ligne de commande) dans lequel nous pouvons définir les programmes et leur ordre d'application. Nous incluons ainsi dans ce script le programme « PrepareText.pl » (*cf. Algorithme 4.1*) de la méthode par caractères (*cf. §4.1.1*), exécuté également en ligne de commande.

L'application de « Sentence » et celle de « Replace » sont en fait exécutées par le même programme « fst2txt.exe »[30] mais avec une option différente (« f » pour « insertion » et « s » pour « remplacement »). Après l'application de « Sentence », nous pouvons encore appliquer les 3 graphes « Replace » pour obtenir les différentes versions du corpus. Ainsi nous n'avons besoin que du graphe « Sentence P0 » car il est toujours appliqué sur un Corpus P0.

```
rem ************************************************
rem Partie 1 : Définition des variables d'environnement
rem ************************************************

set INTEX=C:\Program Files\Intex
set INTEXPRV=C:\MyIntex
set INTEXAPP=%INTEX%\App
set INTEXLNG=%INTEX%\Thai
set INTEXLNG0=%INTEXPRV%\Thai
set PATH=%PATH%;%INTEXAPP%

rem ****************************************
rem Partie 2 : Méthode par caractères (par PERL)
rem ****************************************

perl.exe "PrepareText.pl" "%1.txt" "GramText.txt" "%1-P0.txt"
rem %1 = Argument n°1 = Nom de fichier texte à segmenter

rem *****************************************
rem Partie 3 : Segmentation en phrases (par INTEX)
rem *****************************************

fst2txt.exe l g f "%INTEXLNG0%\SentenceP0.fst" "%1-P0.txt" - - - "%1-P0.snt"

rem ***************************************
rem Partie 4 : Méthode par syllabes (par INTEX)
rem ***************************************

fst2txt.exe d g s "%INTEXLNG0%\Replace1.fst" "%1-P0.snt" - - - "%1-P1.snt"
fst2txt.exe d g s "%INTEXLNG0%\Replace2.fst" "%1-P1.snt" - - - "%1-P2.snt"
fst2txt.exe d g s "%INTEXLNG0%\Replace3.fst" "%1-P2.snt" - - - "%1-P3.snt"
```

Algorithme 4.3 Script de segmentation « Console.bat »

Nous exécutons le script ci-dessus en lui donnant un argument : le nom de fichier [*texte*] à segmenter. Le script générera 5 fichiers : « [*texte*]-P0.txt », « [*texte*]-P0.snt », « [*texte*]-P1.snt », « [*texte*]-P2.snt » et « [*texte*]-P3.snt ». Les fichiers « snt » sont des textes

[30] SILBERZTEIN 2000, pp. 183-184

segmentés en phrases, prêts à être employés dans INTEX pour faire des analyses automatiques des textes thaï que nous détaillerons dans le Chapitre 6.

CHAPITRE 5 ÉVALUATION ET COMPARAISON

Ce chapitre est consacré à l'évaluation de nos deux méthodes de segmentation : segmentation en mots et segmentation en phrases. Nous comparons également nos résultats avec des méthodes concurrentes.

5.1 Méthode d'évaluation

5.1.1 Outil d'évaluation

Nous évaluons toutes les méthodes de segmentation avec « PARSEVAL » : le système d'évaluation le plus connu pour l'extraction et la recherche d'information. Proposé par « Grammar Evaluation Interest Group »[1], il est basé sur les notions de « Précision » et de « Rappel », définies ci-dessous :

- La précision permet de chiffrer la pertinence des résultats, c'est le rapport entre le nombre de réponses correctes fournies et le nombre de réponses fournies par le système :

$$Précision = \left(\frac{nombre\ de\ réponses\ correctes\ fournies}{nombre\ de\ réponses\ fournies} \right) \times 100\ \%$$

  Le contraire de la « Précision » est le « Bruit » qui est le rapport entre le nombre de réponses incorrectes et le nombre de réponses fournies par le système. Il peut être simplement calculé par :

$$Bruit = 100\% - Précision$$

[1] HARRISON *et al*. 1991

- Le rappel permet d'évaluer la quantité de réponses correctes fournies par rapport au nombre de réponses réellement attendues :

$$Rappel = \left(\frac{nombre\ de\ réponses\ correctes\ fournies}{nombre\ de\ réponses\ attendues} \right) \times 100\ \%$$

Le contraire du « Rappel » est le « Silence » qui est le rapport entre le nombre de réponses correctes mais non-fournies et le nombre de réponses attendues. Il peut être simplement calculé par :

$$Silence = 100\% - Rappel$$

5.1.2 Évaluation sur les mots

Étant donné que nos méthodes de segmentation en mots (*cf. §4.1*) n'ont pas pour l'objectif de trouver les frontières exactes de mots dans une séquence contiguë de caractères, mais de permettre la reconnaissance de mots par dictionnaires, nous évaluons directement le nombre de mots reconnus par les DELAF et DELACF du thaï en utilisant le système « PARSEVAL » décrit ci-dessus. Nous comparons les résultats de nos deux méthodes : méthode par caractères (*cf. §4.1.1*) et méthode par syllabes (*cf. §4.1.2*) en faisant des évaluations sur les corpus P0 et P3. Nous comparons également nos résultats avec ceux de la méthode « Maximal Matching »[2] (*cf. §0.3.2.1*) et de la méthode « Trigram »[3] (*cf. §0.3.2.2*). Cette dernière est la méthode utilisée dans le système « ORCHID »[4] (*cf. §0.3.4*).

5.1.3 Évaluation sur les phrases

Nos deux méthodes de segmentation en phrases (*cf. §4.2*) tentent d'ajouter le séparateur {S} au début de chaque phrase. Nous les évaluons sur le nombre de séparateurs qui sont insérés aux bons endroits par rapport au nombre d'insertions effectuées et au nombre d'insertions souhaitées. Ensuite, nous comparons notre résultat avec ceux de la méthode par règles statistiques[5] (*cf. §0.3.3.1*), de la méthode « POS Trigram »[6] et de la méthode « Winnow »[7] (*cf. §0.3.3.2*). Étant donné que les deux dernières méthodes mentionnées utilisent un autre outil d'évaluation, nous devons d'abord le convertir en « Précision » et « Rappel ».

[2] SORNLERTLAMVANICH 1993
[3] MEKNAVIN *et al*. 1997
[4] SORNLERTLAMVANICH *et al*.1997
[5] LONGCHUPOLE 1995
[6] MITTRAPIYANURUK et SORNLERTLAMVANICH 2000
[7] CHAROENPORNSAWAT et SORNLERTLAMVANICH 2001

5.2 Présentation des résultats

Nous utilisons 10 textes journalistiques venant de 10 domaines différents pour évaluer toutes les méthodes selon le système « PARSEVAL ». Tous les textes sont présentés dans l'Annexe XI.

5.2.1 Résultats sur les mots

5.2.1.1 Texte N°1

Domaine : Agriculture
Source : Journal « Matichon » du 17 mai 2003
Rubrique : Actualité
Taille : 777 mots dont 297 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	282	720	297	**39**	**95**
Par syllabes (P3)	282	557	297	**51**	**95**
Maximal Matching	211	330	297	**64**	**71**
Trigram	223	367	297	**61**	**75**

Tableau 5.1 Évaluation sur la reconnaissance des mots dans le Texte N°1

5.2.1.2 Texte N°2

Domaine : Art
Source : Magazine « Art & Culture » du 1er mai 2003
Rubrique : Art & Culture Club
Taille : 940 mots dont 375 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	355	833	375	**43**	**95**
Par syllabes (P3)	355	649	375	**55**	**95**
Maximal Matching	290	406	375	**71**	**77**
Trigram	294	417	375	**71**	**78**

Tableau 5.2 Évaluation sur la reconnaissance des mots dans le Texte N°2

5.2.1.3 Texte N°3

Domaine : Automobile

Source : Journal « Bangkok Business » du 11 mai 2003

Rubrique : Automobile

Taille : 1 045 mots dont 307 formes différentes

Méthode	**Nombre de réponses**			***Précision* (%)**	***Rappel* (%)**
	Correctes	**Fournies**	**Attendues**		
Par caractères (P0)	302	711	307	**42**	**98,4**
Par syllabes (P3)	302	555	307	**54**	**98,4**
Maximal Matching	224	348	307	**64**	**73**
Trigram	231	363	307	**64**	**75**

Tableau 5.3 Évaluation sur la reconnaissance des mots dans le Texte N°3

5.2.1.4 Texte N°4

Domaine : Cuisine

Source : Journal « Bangkok Business » du 16 mai 2003

Rubrique : @ Taste

Taille : 577 mots dont 227 formes différentes

Méthode	**Nombre de réponses**			***Précision* (%)**	***Rappel* (%)**
	Correctes	**Fournies**	**Attendues**		
Par caractères (P0)	227	484	227	**47**	**100**
Par syllabes (P3)	227	364	227	**62**	**100**
Maximal Matching	191	240	227	**80**	**84**
Trigram	195	257	227	**76**	**86**

Tableau 5.4 Évaluation sur la reconnaissance des mots dans le Texte N°4

5.2.1.5 Texte N°5

Domaine : Finance

Source : Journal « Bangkok Business » du 17 mai 2003

Rubrique : Économie

Taille : 818 mots dont 251 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	235	607	251	**39**	**94**
Par syllabes (P3)	235	459	251	**51**	**94**
Maximal Matching	177	289	251	**61**	**71**
Trigram	187	311	251	**60**	**75**

Tableau 5.5 Évaluation sur la reconnaissance des mots dans le Texte N°5

5.2.1.6 Texte N°6

Domaine : Immobilier

Source : Journal « Dailynews » du 17 mai 2003

Rubrique : Actualité

Taille : 1 245 mots dont 313 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	305	749	313	**41**	**97**
Par syllabes (P3)	305	572	313	**53**	**97**
Maximal Matching	244	374	313	**65**	**78**
Trigram	255	427	313	**60**	**81**

Tableau 5.6 Évaluation sur la reconnaissance des mots dans le Texte N°6

5.2.1.7 Texte N°7

Domaine : Justice

Source : Journal « Matichon » du 17 mai 2003

Rubrique : Justice

Taille : 629 mots dont 234 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	225	596	234	**38**	**96**
Par syllabes (P3)	225	456	234	**49**	**96**
Maximal Matching	169	260	234	**65**	**72**
Trigram	179	261	234	**69**	**77**

Tableau 5.7 Évaluation sur la reconnaissance des mots dans le Texte N°7

5.2.1.8 Texte N°8

Domaine : Littérature

Source : Journal « Bangkok Business » du 11 mai 2003

Rubrique : Judprakai Literature

Taille : 750 mots dont 316 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	312	661	316	**47**	**98,7**
Par syllabes (P3)	312	508	316	**61**	**98,7**
Maximal Matching	251	348	316	**72**	**79**
Trigram	252	349	316	**72**	**80**

Tableau 5.8 Évaluation sur la reconnaissance des mots dans le Texte N°8

5.2.1.9 Texte N°9

Domaine : Politique

Source : Journal « Thairath » du 17 mai 2003

Rubrique : Politique

Taille : 913 mots dont 322 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	299	790	322	**38**	**93**
Par syllabes (P3)	299	616	322	**49**	**93**
Maximal Matching	233	369	322	**63**	**72**
Trigram	243	381	322	**64**	**75**

Tableau 5.9 Évaluation sur la reconnaissance des mots dans le Texte N°9

5.2.1.10 Texte N°10

Domaine : Voyage

Source : Journal « Prachachart Turakij » du 15 mai 2003

Rubrique : Life Style

Taille : 459 mots dont 260 formes différentes

Méthode	Nombre de réponses			*Précision* (%)	*Rappel* (%)
	Correctes	Fournies	Attendues		
Par caractères (P0)	247	584	260	**42**	**95**
Par syllabes (P3)	247	441	260	**56**	**95**
Maximal Matching	200	279	260	**72**	**77**
Trigram	206	280	260	**74**	**79**

Tableau 5.10 Évaluation sur la reconnaissance des mots dans le Texte N°10

5.2.2 Bilan sur les mots

Les moyennes des 4 méthodes sur les 10 textes d'analyse ci-dessus sont présentées dans le tableau suivant :

Méthode	*Précision* (%)	*Bruit* (%)	*Rappel* (%)	*Silence* (%)
Par caractères (P0)	42	58	96	4
Par syllabes (P3)	54	46	96	4
Maximal Matching	68	32	75	25
Trigram	67	33	78	22

Tableau 5.11 Bilan sur les méthodes de segmentation en mots

D'après ce tableau, nous constatons que la méthode par syllabes n'apporte aucune amélioration du point de vue du rappel par rapport à la méthode par caractères mais qu'elle améliore la précision d'environ 13 points[8], c'est-à-dire que le bruit dans le corpus P3 est diminué d'environ 22%[9] par rapport au bruit dans la version P0. Nos deux méthodes sont moins bonnes en précision que les méthodes concurrentes. Par contre, elles sont meilleures en rappel, ce qui nous convient parce qu'un système d'analyse automatique de textes ne devrait laisser de côté aucun mot. D'après nos vérifications, nous confirmons que le silence dans nos deux méthodes n'a pas pour origine la segmentation, mais des noms propres, sigles, mots étrangers et mots familiers qui sont absents des dictionnaires. Quant au bruit assez élevé de nos méthodes, il pourrait être encore réduit par la réduction des transducteurs du texte (*cf. §6.4.2*) ou par les grammaires locales de levée d'ambiguïtés (*cf. §6.5*) que nous expliquerons dans le prochain chapitre.

[8] 54,19-41,57 = 12,62%
[9] (58,43-45,81)x100/58,43 = 21,60%

5.2.3 Résultats sur les phrases

Étant donné que la méthode par la ponctuation (*cf. §4.2.1*) et la méthode par mots-clés (*cf. §4.2.2*) sont indépendantes et peuvent être employées ensemble, nous avons combiné les deux pour insérer le séparateur de phrases dans les mêmes corpus que ceux utilisés pour l'évaluation sur les mots. Les résultats sont présentés dans le tableau ci-dessous :

Texte	Nombre d'insertions			*Précision* (%)	*Rappel* (%)
	Correctes effectuées	Effectuées	Souhaitées		
N°1 (Agriculture)	9	11	20	**82**	**45**
N°2 (Art)	18	19	33	**95**	**55**
N°3 (Automobile)	15	17	25	**88**	**60**
N°4 (Cuisine)	28	28	40	**100**	**70**
N°5 (Finance)	15	15	17	**100**	**88**
N°6 (Immobilier)	19	22	27	**86**	**70**
N°7 (Justice)	18	19	23	**95**	**78**
N°8 (Littérature)	24	28	28	**86**	**86**
N°9 (Politique)	18	22	30	**82**	**60**
N°10 (Voyage)	20	23	22	**87**	**91**

Tableau 5.12 Évaluation sur l'insertion de séparateurs de phrases dans 10 textes

5.2.4 Bilan sur les phrases

Faute de disposer des programmes, nous ne pouvons pas évaluer sur les mêmes corpus les méthodes concurrentes de segmentation en phrases et nous présentons simplement leurs résultats annoncés afin de pouvoir les comparer avec le nôtre qui est la moyenne des valeurs affichées dans le tableau ci-dessus. Pourtant, l'outil d'évaluation de 2 méthodes concurrentes est différent du nôtre car il marque toutes les occurrences d'espaces blancs dans un corpus avec deux étiquettes : « NSBS » (non-sentence-break space) ou « SBS » (sentence-break space). L'évaluation est donc faite par le système suivant défini par P. TAYLOR[10] :

Break-correct = (CB/RB) x 100%
Space-correct = (CS/RS) x 100%
False-break = (FB/RS) x 100%

[10] TAYLOR et BLACK 1998

avec

CB : Nombre de « SBS » correctement détectés

FB : Nombre de « SBS » erronés

CS : Nombre de « SBS » et « NSBS » correctement détectés

RB : Nombre de « SBS » de référence

RS : Nombre de « SBS » et « NSBS » de référence

Les résultats de l'évaluation sont présentés dans le tableau ci-dessous :

Méthode	**Break-correct (%)**	**Space-correct (%)**	**False-break (%)**
POS Trigram	80	85	9
Winnow	77	89	2

Tableau 5.13 Évaluation des méthodes concurrentes

Afin de pouvoir les comparer avec notre résultat, nous devons d'abord convertir le système ci-dessus en « PARSEVAL ». Nous remarquons que « Break-correct » correspond effectivement à notre « Rappel » mais la « Précision » doit être calculée par la formule suivante :

$$\text{Précision} = \frac{CB}{CB + FB}$$

$$= \frac{CB/RB}{CB/RB + FB/RB}$$

Les deux méthodes annoncent que la proportion entre « NSBS » et « SBS » dans leur corpus est environ 5 : 2, c'est-à-dire que :

$$\frac{RB}{RS} = \frac{2}{7}$$

ou

$$RB = \frac{RS}{3,5}$$

Donc

$$\text{Précision} = \frac{\text{CB/RB}}{\text{CB/RB} + \text{FB}\Big/\frac{\text{RS}}{3,5}}$$

$$= \frac{\text{Break correct}}{\text{Break correct} + (3,5 \times \text{False break})}$$

Nous pouvons donc présenter la précision et le bruit des deux méthodes concurrentes en bas du tableau ci-dessous :

Méthode	***Précision* (%)**	***Bruit* (%)**	***Rappel* (%)**	***Silence* (%)**
Nos méthodes	90	10	70	30
Par règles statistiques	81	19	*n.c.*[11]	*n.c.*
POS Trigram	72	28	80	20
Winnow	93	7	77	23

Tableau 5.14 Bilan sur les méthodes de segmentation en phrases

Puisque nous ne pouvons pas réaliser toutes les évaluations sur les mêmes corpus et que les résultats convertis sont simplement des valeurs approximatives, nous ne pouvons pas en conclure que notre comparaison est sûre et fiable. Cependant, les résultats obtenus permettent de montrer que nos méthodes sont très compétitives par rapport aux méthodes concurrentes.

En ce qui concerne la détection des limites de phrases, la précision est plus importante que le rappel. En effet, une limite de phrase erronée, donc le fait de couper en deux une phrase, gêne plus l'analyse syntaxique qu'une limite de phrase non-détectée, puisque cela revient à fusionner deux phrases.

5.3 Conclusion

La prise en compte du rappel et de la précision montre que notre méthode de segmentation en mots représente un progrès notable par rapport à l'état de l'art, avec une nette augmentation du rappel, qui est le critère de qualité le plus important pour le traitement des textes. Quant à notre méthode de segmentation en phrases, elle s'approche de la meilleure méthode connue en ce qui concerne ses performances en précision.

[11] *n.c.* : non communiqué

CHAPITRE 6 ANALYSE AUTOMATIQUE DE TEXTES THAÏ

Étant donné que les éléments essentiels d'INTEX (*cf. §1.2*) sont réalisés et adaptés pour le thaï, nous pouvons procéder à des analyses de textes. Nous utilisons le roman *SiPhanDin*[1] (Quatre Règnes) comme corpus d'analyse. Il a une taille de 1,5 Mo et est codé en ASCII étendu. Il comprend 1 452 116 caractères[2] (espaces non compris), ce qui est équivalent à un livre de poche de 1 000 pages environ.

Le système nous permet de faire 3 types d'analyses : une analyse morphologique, une analyse lexicale et une analyse syntaxique.

6.1 Analyse morphologique

6.1.1 Analyse des affixes

Cette analyse permet de reconnaître des parties de mots, c'est-à-dire des préfixes et des suffixes, en appliquant les dictionnaires d'affixes (*cf. §4.1.1.7.1*) sur un corpus P0. Ce dernier a été choisi pour ce type d'analyse parce qu'il avait été segmenté caractère par caractère, ce qui autorise l'identification partielle de mots par INTEX même dans les cas de mots composés avec adaptation morphologique (*cf. §2.3.2.2.2*). Par exemple, le préfixe « ราช » [raːchá] (roi) dans le mot « ราชาภิเษก » [raːchaːphîsèːk] (couronnement de roi) ne sera reconnu qu'avec le corpus P0 à cause du caractère vocalique inséparable « -า » [aː] qui est collé au dernier caractère du préfixe dans d'autres versions de corpus.

---

[1] PRAMOJ 2000

[2] Faute de séparateur de mots, nous ne sommes pas en mesure d'utiliser le nombre de mots dans le corpus comme référence.

Néanmoins, en utilisant le corpus P0, nous devons accepter un taux de bruit très élevé, plus particulièrement en raison des affixes de petite taille s'écrivant avec des caractères courants tel « พล » [phálá] (force) que nous trouvons 9 274 fois comme préfixe dans notre corpus d'analyse. Parmi ces 9 274 occurrences, seules 7 sont correctes. En effet, les petits affixes correspondent trop facilement à des parties de mots (*cf. §4.1.1.6*) et INTEX n'est pas en mesure de distinguer les bons des mauvais.

Un affixe plus grand pose moins de problèmes ; par exemple, le préfixe « เกียรติ » [kìa:ttì] (réputation) est correctement reconnu 10 fois sur 19 et le préfixe « วัฒน » [wátthánà] (prospérité), grâce au caractère peu courant « ฒ » [thɔ: phû:thâw], obtient la note de 16/17.

Après avoir fait une analyse des affixes sur le roman « SiPhanDin-P0 », nous avons identifié 468 préfixes et 15 suffixes. Parmi les 71 606 occurrences de préfixes repérées, 1 871 seulement étaient correctes (soit 3%[3] de précision) et parmi les 119 occurrences de suffixes repérées, 103 étaient correctes (soit 87%[4] de précision). Il n'y a aucun silence, c'est-à-dire que le rappel dans les deux cas est égal à 100%.

6.1.2 Analyse des phonèmes muets

Un phonème muet « Karant » est un élément dans une syllabe qui ne se prononce pas (*cf. §2.2.1*). Il est marqué par le signe de phonème muet « thanthákhât » (*cf. §2.1.3.2*). Pour examiner les phonèmes muets dans un corpus, nous appliquons le graphe « Karant » (*cf. Figure 4.48*) en mode « recherche de motifs par graphes » (*cf. §1.3.2.2*) sur un corpus P1 dans lequel les phonèmes muets sont préalablement regroupés par les graphes « Replace 1.23 » et « Replace 1.24 » (*cf. Figure 4.38 et Figure 4.39*).

Dans notre expérimentation sur le corpus « SiPhanDin-P1 » qui comporte exactement 2 119 phonèmes muets, notre méthode a identifié correctement 2 108 phonèmes, partiellement 9 phonèmes et 2 phonèmes n'ont pas été détectés du tout. Si nous considérons que les 9 phonèmes partiellement identifiés ont été à la fois détectés à tort et non détectés à tort, alors la précision est égale à 99,6%[5] et le rappel à 99,5%[6]. Les 2 phonèmes muets non-identifiés sont en fait le « ร์ » inclus dans la syllabe « เฟร์น » dont les caractères ont été regroupés par le graphe « Replace 1.17 » (*cf. Figure 4.32*). Ils ne sont donc pas détectés individuellement.

Le résultat complet de l'analyse morphologique est présenté dans l'Annexe XII.A.

[3] 1871x100/71606 = 2,61%
[4] 103x100/119 = 86,55%
[5] 2108x100/(2108+9) = 99,57%
[6] 2108x100/(2108+9+2) = 99,48%

6.2 Analyse lexicale

6.2.1 Analyse des mots simples et composés

Il s'agit de l'identification des mots simples et des mots composés dans un corpus respectivement au moyen des dictionnaires DELAF et DELACF et de l'association de ces mots aux informations lexicales correspondantes. Nous rappelons que la différence entre mots simples et mots composés dans INTEX est purement formelle (*cf. §3.2.1*).

Ici encore, le choix de la version (P0 - P3) du corpus à analyser est important. Plus la version est élevée, moins il existe de bruit produit, parce que le nombre de lexèmes dans une version supérieure est réduit par rapport à celui d'une version inférieure. Le tableau suivant montre le résultat des analyses de différentes versions de corpus.

Corpus	Nombre de lexèmes	Nombre de mots identifiés (Occurrences)		
		Simples	Composés	Total
SiPhanDin-P0	1 491 318	124 422	652 397	776 819
SiPhanDin-P1	777 831 (-48%)	398 254 (+220%)	296 166 (-54,6%)	694 420 (-10,6%)
SiPhanDin-P2	773 999 (-0,5%)	398 584 (+0,08%)	294 328 (-0,6%)	692 912 (-0,2%)
SiPhanDin-P3	759 587 (-2%)	402 569 (+1%)	281 904 (-4,2%)	684 473 (-1,2%)

Tableau 6.1 Analyse des mots simples et composés

Nous préférons donc analyser le corpus P3 dans lequel nous identifions les catégories grammaticales ainsi que leur effectif en nombre et en pourcentage. Le résultat est présenté dans le tableau suivant :

Catégorie	Occurrences	%
N	363 887	31,8
PRO	45 175	3,9
V	335 033	29,3
ADJV	296 675	25,9
PREP	47 475	4,2
CONJ	39 469	3,4
NIBAT	12 819	1,1
INTJ	3 098	0,3
EXP	3	0,0003

Tableau 6.2 Catégorie grammaticale du corpus « SiPhanDin-P3 »

D'après le tableau, nous constatons que l'ensemble formé par les noms, verbes et adjectifs-verbes représente déjà 87% des mots.

6.2.2 Analyse des expressions figées

Certains mots composés ne sont pas contigus et ne peuvent être identifiés que par des graphes décrits dans le répertoire DELAE (*cf. §1.2.3.5*).

- L'expression « ทั้ง...กับ... » [tháŋ…kàp…] est une conjonction qui relie 2 noms et qui signifie « et ».

  (ทั้งN$_0$กับN$_1$ = N$_0$ et N$_1$)

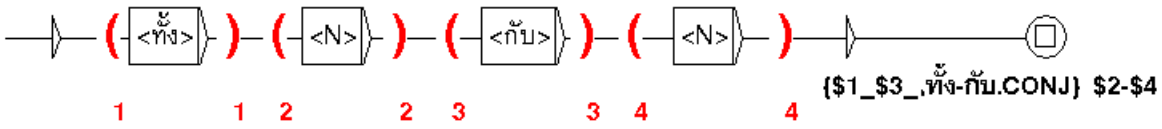

Figure 6.1 Expression figée « tháŋ…kàp… »

<u>Remarque</u> : En mettant les mots entre angles, nous travaillons avec leurs lemmes (*cf. §4.1.1.4.3*). Cela permet aux graphes de fonctionner avec toutes les versions de corpus.

- L'expression « ทั้ง...และ... » [tháŋ…lɛ́…] est un synonyme de l'expression ci-dessus.

  (ทั้งN$_0$และN$_1$ = N$_0$ et N$_1$)

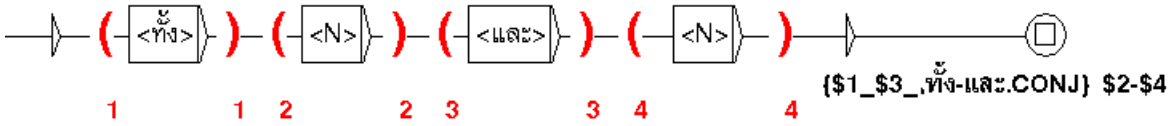

Figure 6.2 Expression figée « tháŋ…lɛ́… »

- L'expression « ทั้ง...ทั้ง... » [tháŋ…tháŋ…] est un adjectif-verbe qui relie 2 noms ou 2 verbes et qui signifie « simultané » ou « simultanément ».

  (ทั้งN$_0$ทั้งN$_1$ = N$_0$ et N$_1$ simultanés)

  (ทั้งV$_0$ทั้งV$_1$ = V$_0$ et V$_1$ simultanément)

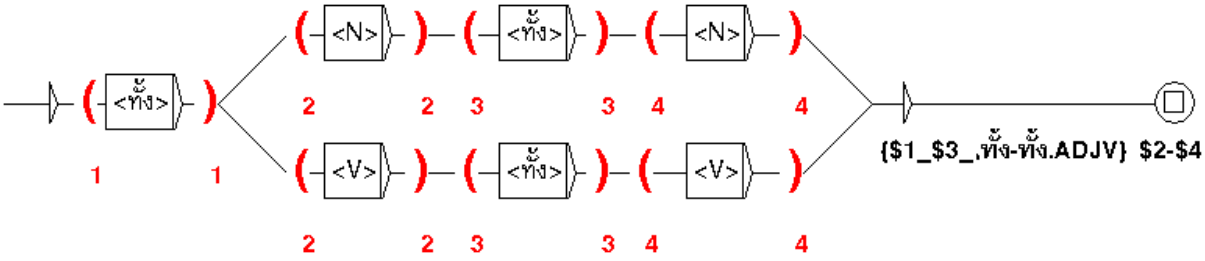

Figure 6.3 Expression figée « tháŋ…tháŋ… »

- L'expression « ...ใด...หนึ่ง » […daj…nɯ̀ŋ] est un adjectif-verbe utilisé avec un nom répété 2 fois et qui signifie « quelconque ».

  (N_0ใดN_0หนึ่ง = N_0 quelconque)

  L'idéal serait de pouvoir définir le <$1> dans la boîte comme la figure ci-dessous.

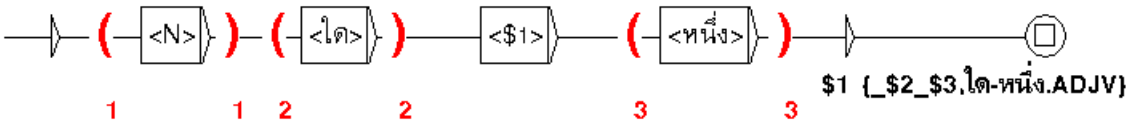

Figure 6.4 Expression figée « …daj…nɯ̀ŋ » idéale

INTEX n'accepte pas une telle définition et nous devons modifier le graphe ci-dessus qui devient :

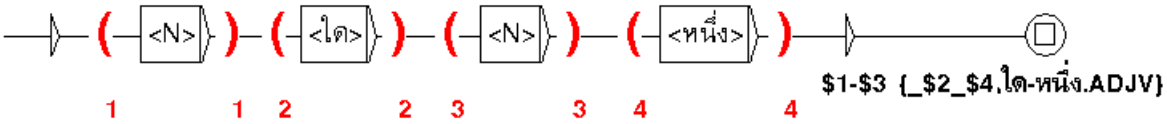

Figure 6.5 Expression figée « …daj…nɯ̀ŋ »

Discussion : le graphe modifié ci-dessus peut engendrer des erreurs en reconnaissant des noms différents dans les première et troisième position. C'est pourquoi nous indiquons simultanément le $1 et le $3 dans la sortie de graphe afin de pouvoir vérifier leur identité. Par exemple, l'expression suivante est correctement reconnue parce que $1 = $3 :

คนใดคนหนึ่ง → คน-คน {_ใด_หนึ่ง,ใด-หนึ่ง.ADJV}

Mais la séquence ci-dessous est reconnue à tort parce que $1 ≠ $3 :

คนใดที่หนึ่ง → คน-ที่ {_ใด_หนึ่ง,ใด-หนึ่ง.ADJV}

- L'expression « ...หนึ่ง...ใด » […nɯ̀ŋ…daj] est une variante de l'expression ci-dessus.

  (N_0หนึ่งN_0ใด = N_0 quelconque)

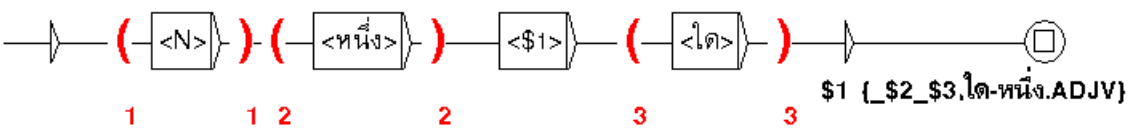

Figure 6.6 Expression figée « …nɯ̀ŋ…daj » idéale

Pour la même raison, nous devons le modifier. Ce dernier devient alors :

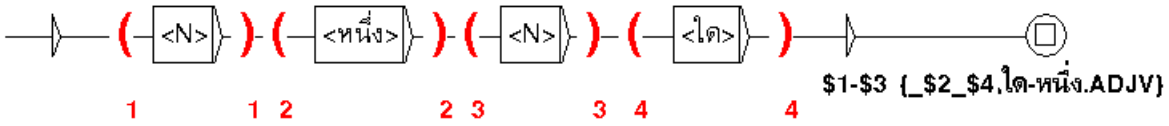

Figure 6.7 Expression figée « …nɯ̀ŋ…daj »

- L'expression « ...แล้ว...เล่า » […lɛ́ːw…lâw] est un adjectif-verbe qui accepte soit un nom, soit un verbe écrit 2 fois et qui signifie « répétitif » ou « répétitivement ».

  (N_0แล้วN_0เล่า = N_0 répétitif)

  (V_0แล้วV_0เล่า = V_0 répétitivement)

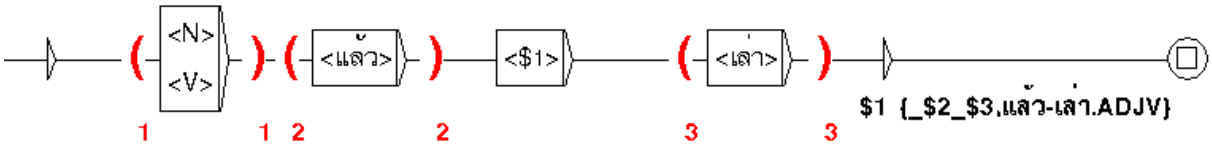

Figure 6.8 Expression figée « …lɛ́ːw…lâw » idéale

Pour la même raison, nous devons le modifier. Ce dernier devient alors :

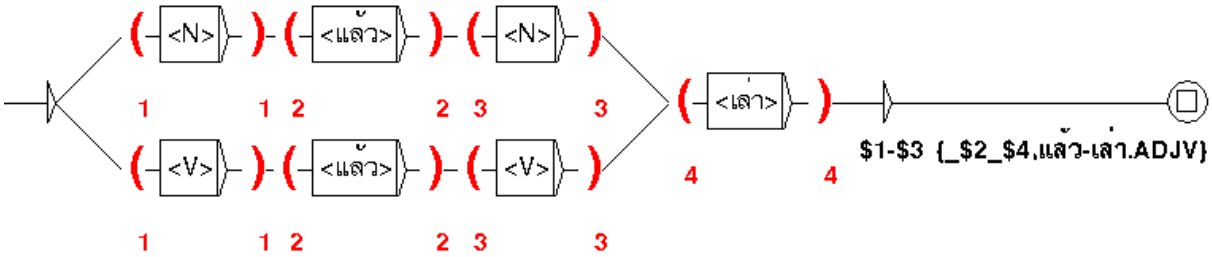

Figure 6.9 Expression figée « …lɛ́ːw…lâw »

Remarque : Afin de ne pas confondre le nom et le verbe, nous divisons le graphe en deux chemins comme dans la figure ci-dessus.

Après l'application de tous les graphes d'expressions figées ci-dessus sur le corpus « SiPhanDin-P3 » sur lequel nous avons appliqué les DELAF et DELACF, nous trouvons correctement 64 expressions figées, incorrectement 5 expressions, et 8 ne sont pas trouvées. Ainsi, la précision est égale à 93%[7] et le rappel à 89%[8].

Le résultat de l'analyse lexicale est présenté dans l'Annexe XII.B.

[7] 64x100/(64+5) = 92,75%
[8] 64x100/(64+8) = 88,89%

6.3 Identification d'expressions par des graphes

D'autres expressions qui ne sont décrites dans aucun dictionnaire peuvent être identifiées grâce aux graphes d'INTEX.

6.3.1 Nombres décimaux

Dans les textes thaï, les nombres sont souvent écrits par groupes de trois chiffres (soit en chiffres arabes <NB>, soit en thaï <TNB>) séparés les uns des autres par une virgule. Ils peuvent être précédés par un signe négatif. Les décimales sont toujours derrière un point.

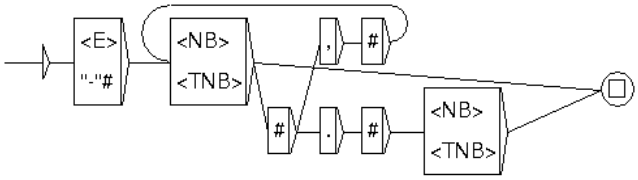

Figure 6.10 Expression des nombres décimaux

Nous appliquons ce graphe sur le corpus « SiPhanDin » quelle que soit la version, en mode « recherche de motifs par graphes » (*cf. §1.3.2.2*) et nous identifions les 219 nombres décimaux sans aucune erreur. C'est-à-dire que la précision et le rappel sont égaux à 100%.

6.3.2 Expressions numériques

Les nombres entiers écrits en toutes lettres sont identifiés par les graphes suivants :

- Le graphe ci-dessous reconnaît les nombres de 1 à 9 (*cf. Tableau 2.7*).

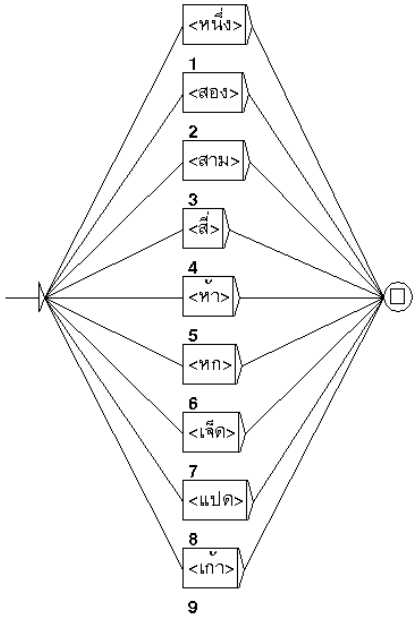

Figure 6.11 Graphe « Tnum1-9 »

- Le graphe ci-dessous reconnaît les nombres de 10 à 99.

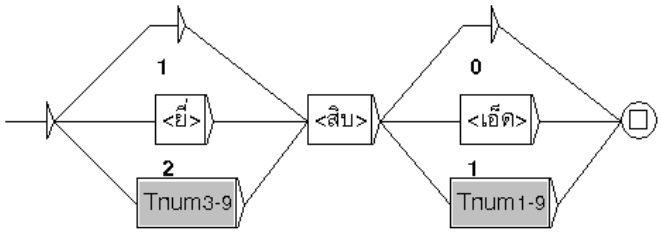

Figure 6.12 Graphe « Tnum10-99 »

C'est un RTN qui fait appel à deux sous-graphes : celui de la Figure 6.11 et celui ci-dessous :

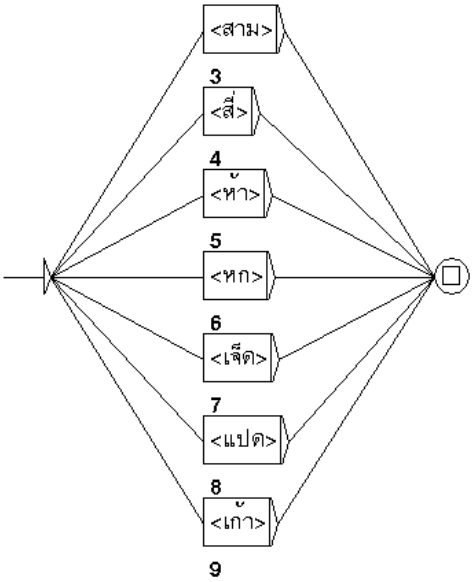

Figure 6.13 Graphe « Tnum3-9 »

- Les graphes ci-dessous sont également des RTN qui reconnaissent respectivement les nombres de 100 à 999, de 1 000 à 9 999, de 10 000 à 99 999 et de 100 000 à 999 999.

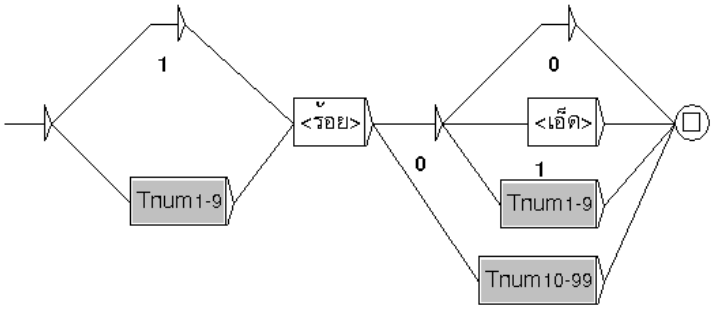

Figure 6.14 Graphe « Tnum100-999 »

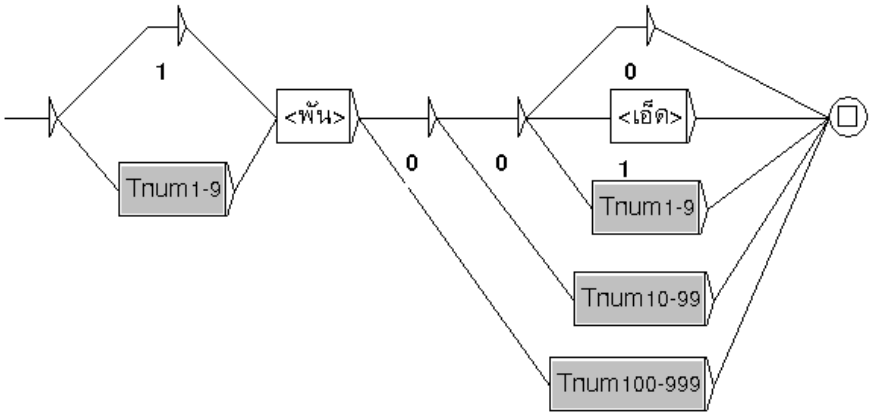

Figure 6.15 Graphe « Tnum1000-9999 »

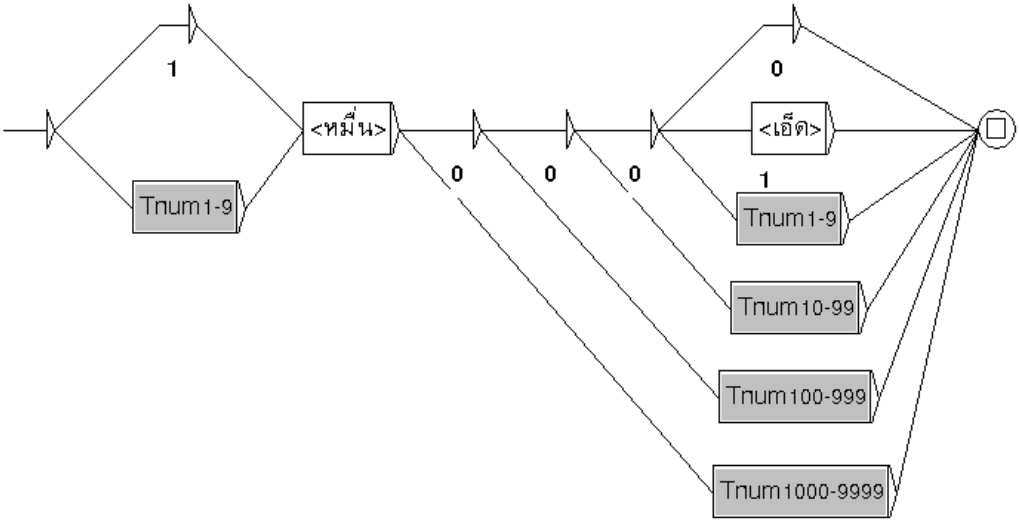

Figure 6.16 Graphe « Tnum10000-99999 »

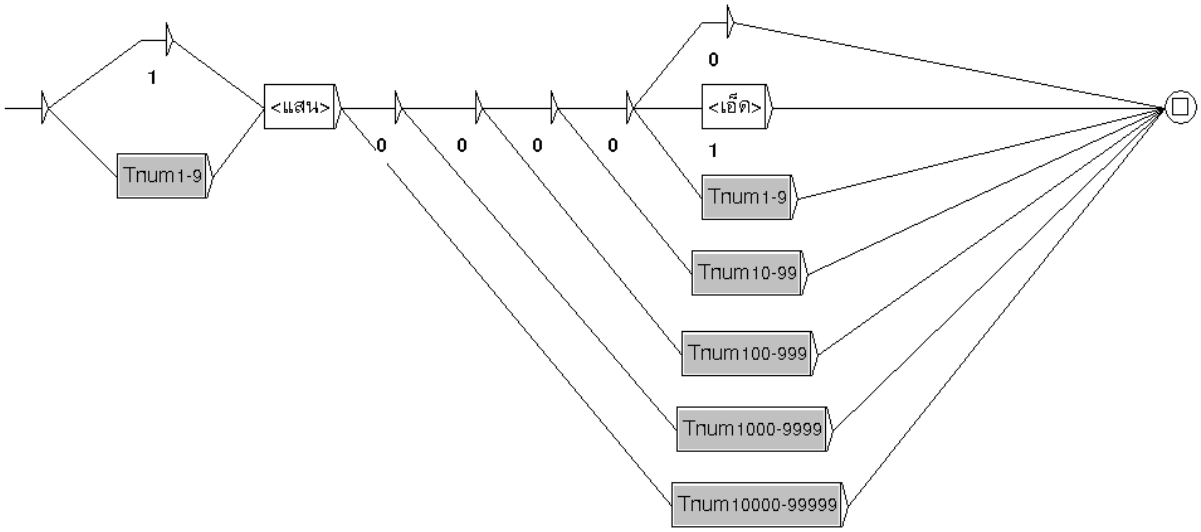

Figure 6.17 Graphe « Tnum100000-999999 »

- Le graphe ci-dessous reconnaît les nombres à partir de 1 000 000. C'est un RTN qui fait appel aux graphes ci-dessus et également à lui-même. En conséquence, il reconnaît théoriquement les nombres entiers d'un million jusqu'à l'infini[9].

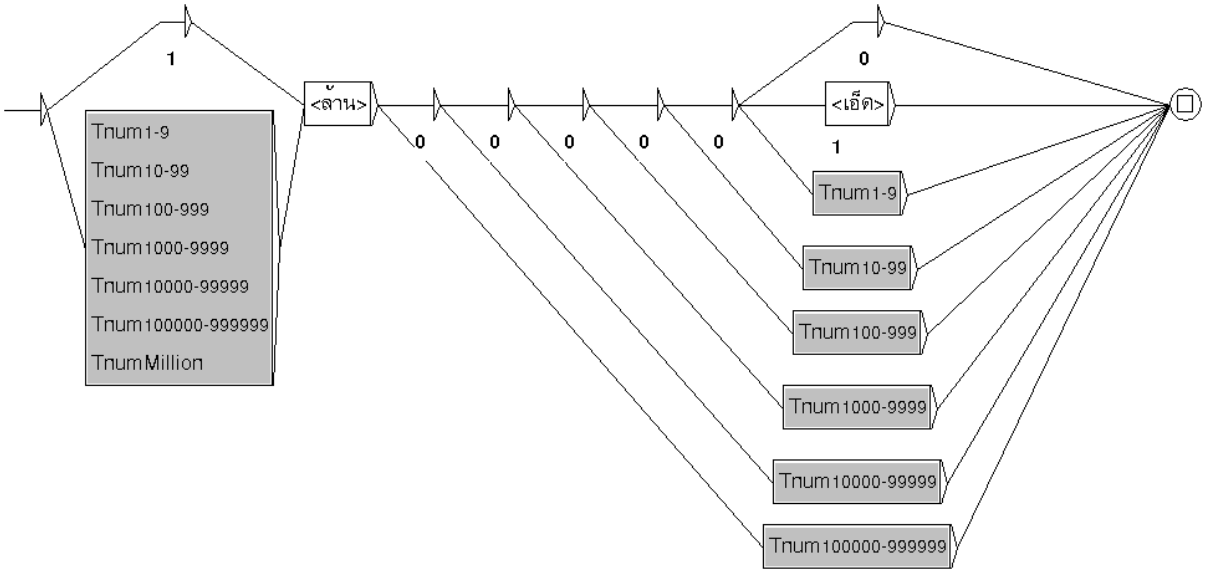

Figure 6.18 Graphe « TnumMillion »

- En combinant tous les graphes ci-dessus avec les mots « ศูนย์ » [sŭ:n] (zéro), « ลบ » [lóp] (moins) et « จุด » [cùt] (point), nous arrivons à reconnaître en principe toutes les expressions des nombres décimaux d'une valeur de moins l'infini jusqu'à l'infini[10].

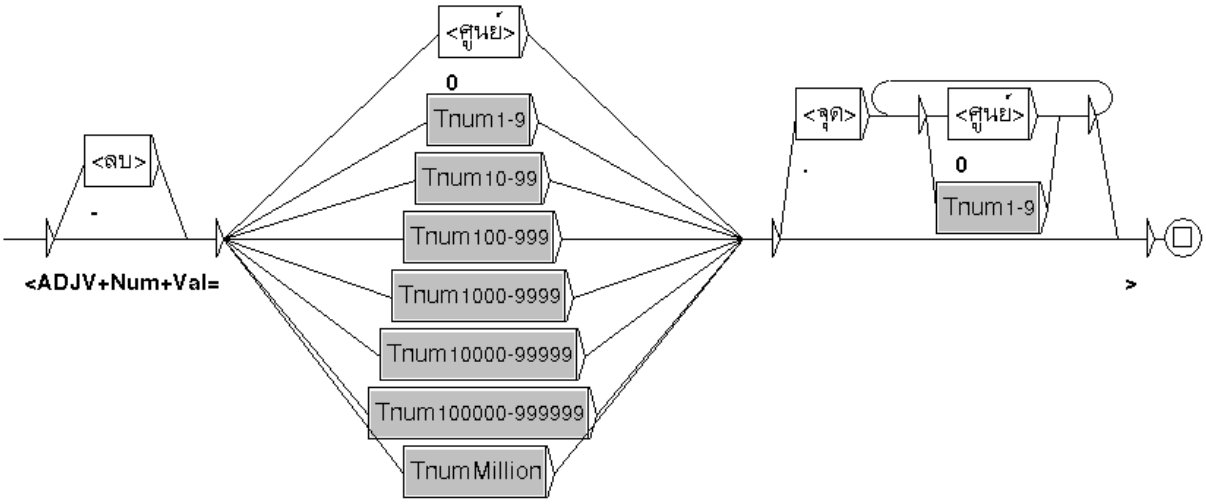

Figure 6.19 Expression numérique en thaï « RealNumberExpression »

[9] En pratique, INTEX limite le nombre d'imbrications possibles à 10 niveaux seulement [SILBERZTEIN 1998-1999a, p. 20].
[10] *Idem*

En appliquant le graphe ci-dessus sur le corpus d'analyse « SiPhanDin-P3 », nous arrivons à identifier correctement 2 835 expressions numériques écrites en toutes lettres avec 436 incorrectes et 4 manquantes. C'est-à-dire que la précision est égale à 87%[11] et le rappel à 99,9%[12].

6.3.3 Expressions de date

En thaï, il existe 3 façons d'exprimer la date : une forme moderne, une forme traditionnelle et une forme numérique. Toutes les trois sont reconnues grâce au graphe suivant :

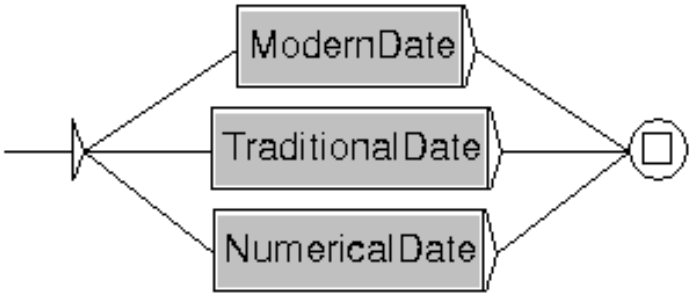

Figure 6.20 Expression de date en thaï

- Le graphe ci-dessous reconnaît des expressions de date modernes selon le calendrier solaire.

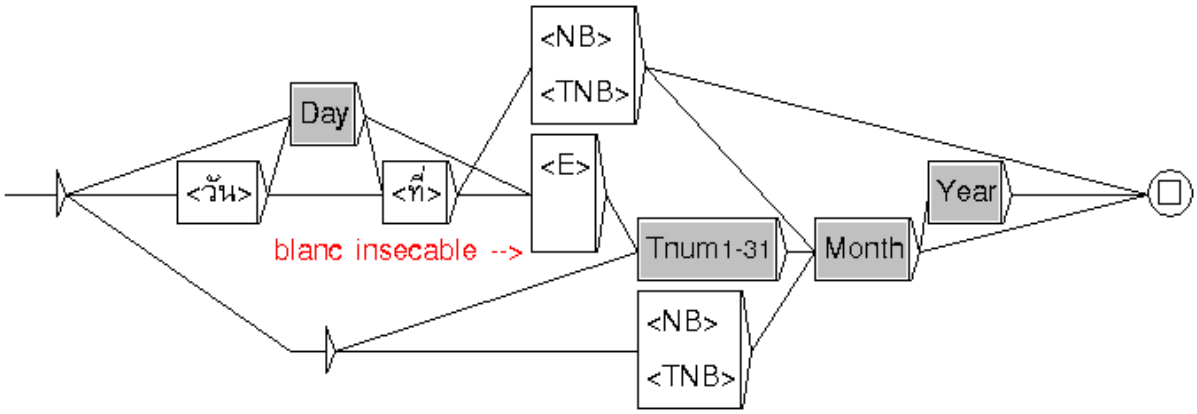

Figure 6.21 Expression de date « ModernDate »

- Le graphe ci-dessous décrit les 7 jours de la semaine du dimanche au samedi sous une forme complète ou abrégée.

[11] 2835x100/(2835+436) = 86,67%
[12] 2835x100/(2835+4) = 99,86%

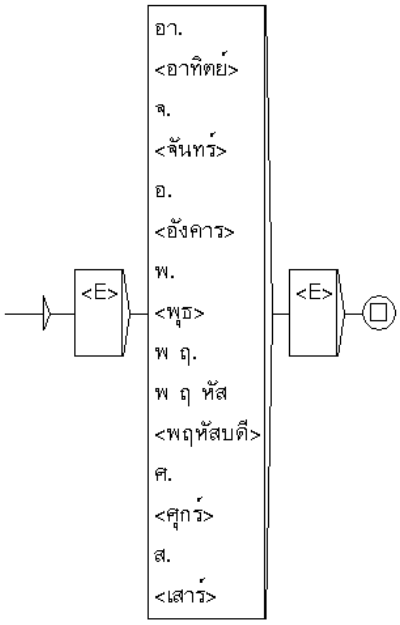

Figure 6.22 Graphe « Day »

- Le graphe ci-dessous est une version allégée du graphe «RealNumberExpression» (*cf. Figure 6.19*) qui ne reconnaît que les nombres entiers positifs de 1 à 31.

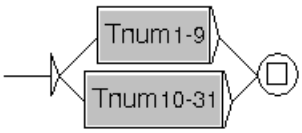

Figure 6.23 Graphe « Tnum1-31 »

- Le graphe ci-dessous décrit les 12 mois de l'année de janvier à décembre sous une forme complète ou abrégée.

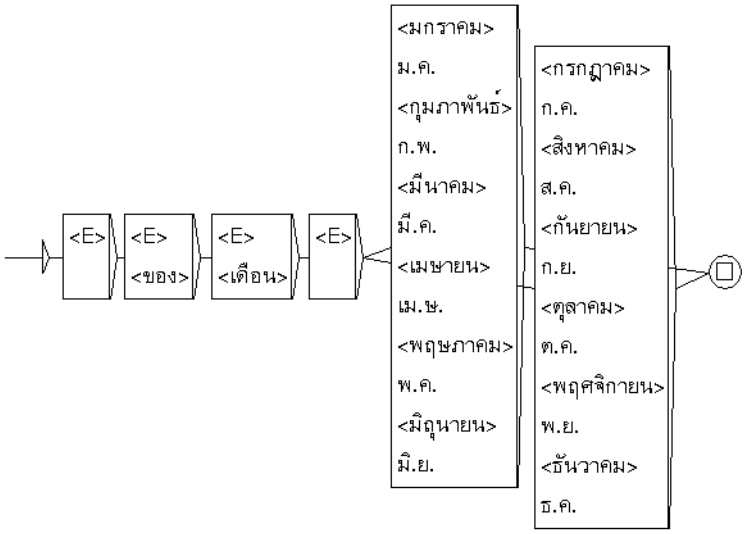

Figure 6.24 Graphe « Month »

- Le graphe ci-dessous décrit les façons d'exprimer l'année.

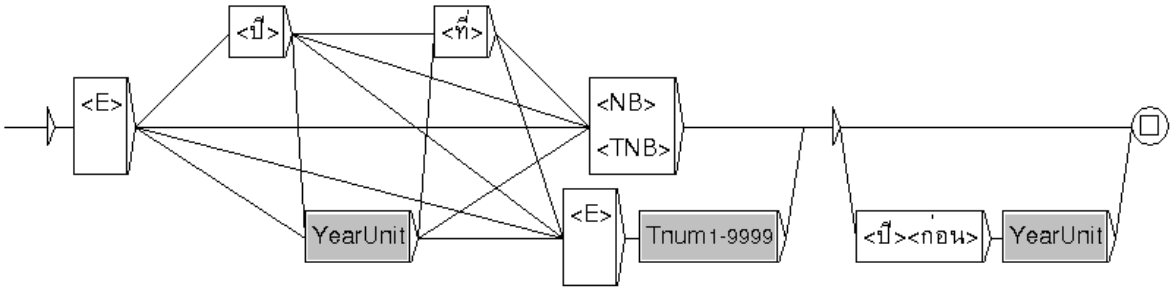

Figure 6.25 Graphe « Year »

- Les indications d'ère sont décrites dans le graphe ci-dessous. Il tient compte des calendriers chrétien, bouddhiste, islamique, etc. qui sont exprimés sous une forme complète ou abrégée.

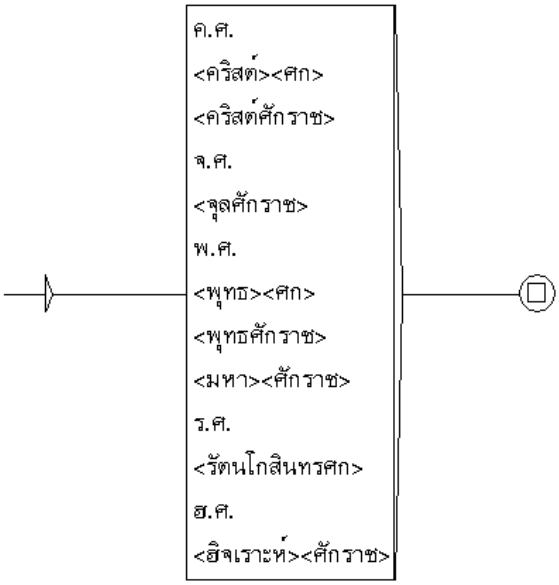

Figure 6.26 Graphe « YearUnit »

- Le graphe ci-dessous est une version allégée du graphe «RealNumberExpression» (*cf. Figure 6.19*) qui ne reconnaît que les nombres entiers positifs de 1 à 9999.

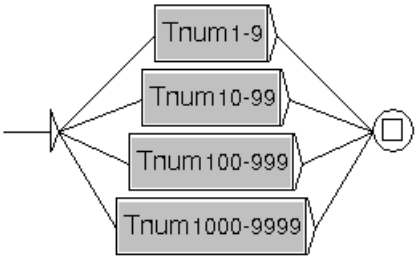

Figure 6.27 Graphe « Tnum1-9999 »

- Le graphe ci-dessous reconnaît des expressions de date traditionnelles selon le calendrier lunaire.

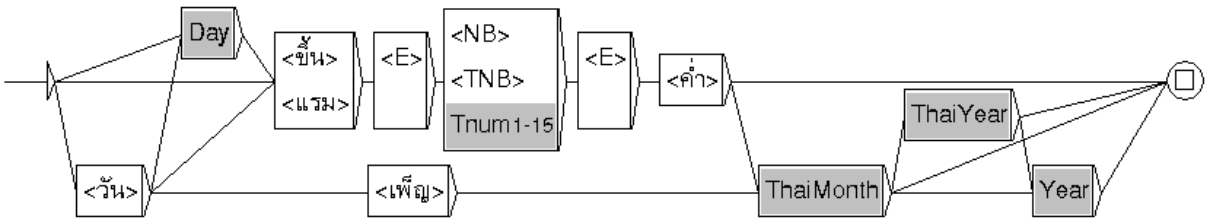

Figure 6.28 Expression de date « TraditionalDate »

- Les mois dans les expressions de date traditionnelles sont exprimés sous forme de nombres allant de un à douze.

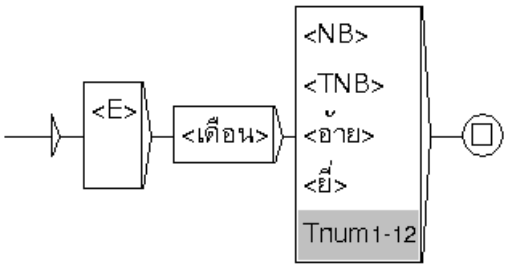

Figure 6.29 Graphe « ThaiMonth »

- Les années sont exprimées selon la tradition chinoise avec 12 signes d'animaux, chacun d'eux écrits sous différentes formes : une forme originale et ses variantes.

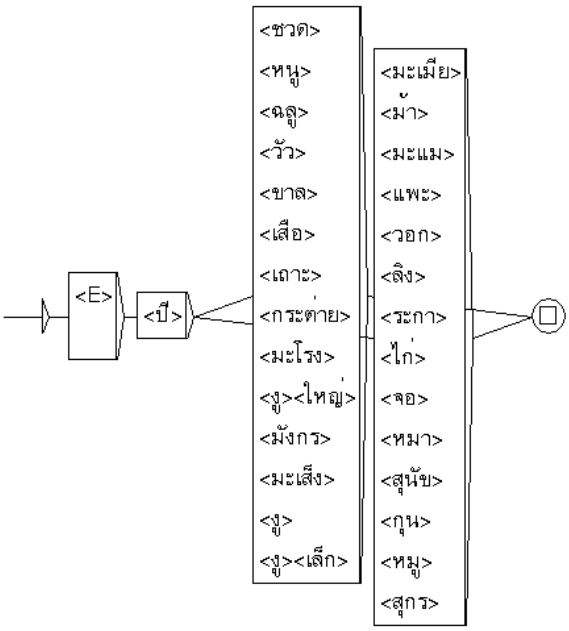

Figure 6.30 Graphe « ThaiYear »

- Le graphe ci-dessous reconnaît des expressions de date numériques tout en chiffres. La date, le mois et l'année sont séparés par le signe « / » ou « - ».

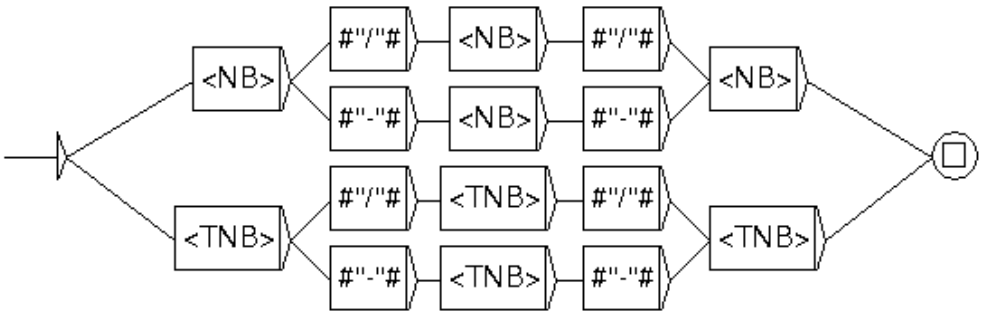

Figure 6.31 Expression de date « NumericalDate »

L'application des expressions de date sur le corpus « SiPhanDin-P3 » identifie correctement les 6 expressions sans aucune erreur. C'est-à-dire que la précision et le rappel sont égaux à 100%[13].

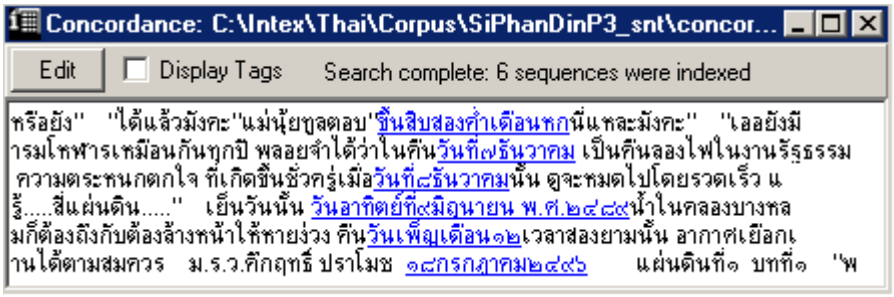

Figure 6.32 Résultat de la recherche des expressions de date

6.4 Analyse syntaxique

Après avoir appliqué les dictionnaires pour l'analyse lexicale, nous pouvons continuer avec des analyses syntaxiques en construisant des transducteurs du texte.

6.4.1 Transducteurs du texte

En choisissant « Construct FST-Text » dans le menu « Text », nous pouvons voir toutes les structures possibles des phrases sous la forme des transducteurs du texte, dans lesquels chaque unité linguistique est représentée par une étiquette et chaque chemin du nœud initial vers le nœud terminal correspond à une analyse possible de chaque phrase. La version du corpus joue aussi un rôle important dans la complexité des transducteurs. Plus la version

[13] Cependant, le nombre des expressions trouvées dans ce corpus est trop peu élevé pour que les valeurs de la précision et du rappel soient fiables.

est élevée, moins les transducteurs sont complexes car il y a moins de bruit. Par exemple, la phrase « ฉันมารอกราบพระสงฆ์ » [chănmaːrɔːkràːpphrásŏŋ] (*cf. §2.3.3*) en version P0 a la structure suivante (48 étiquettes) :

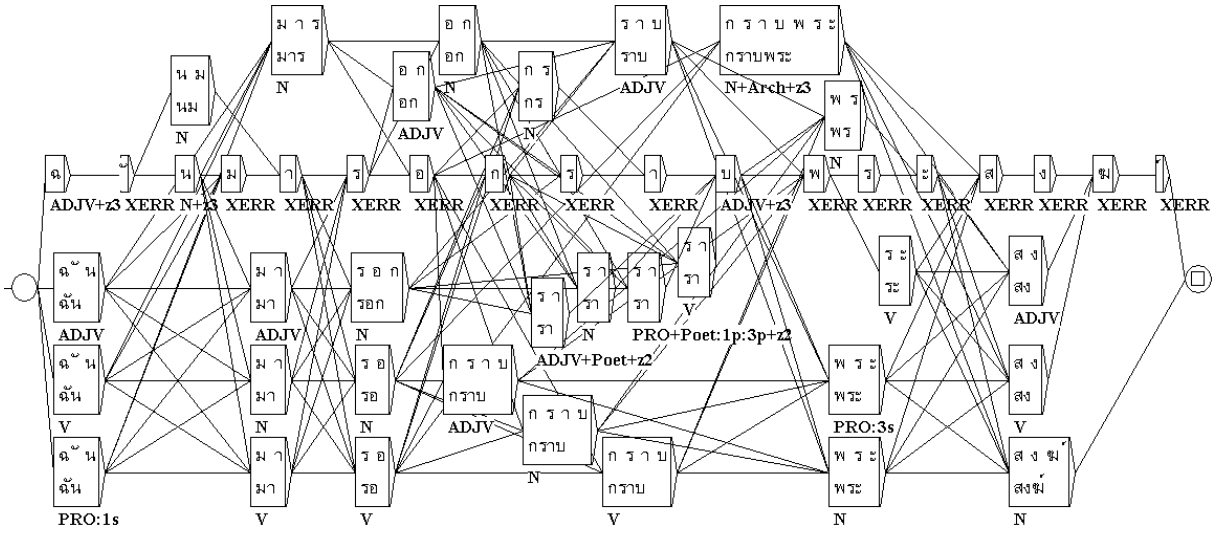

Figure 6.33 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » en version P0

En le comparant avec la même phrase en version P3, nous constatons que le transducteur ci-dessous est moins complexe (30 étiquettes) :

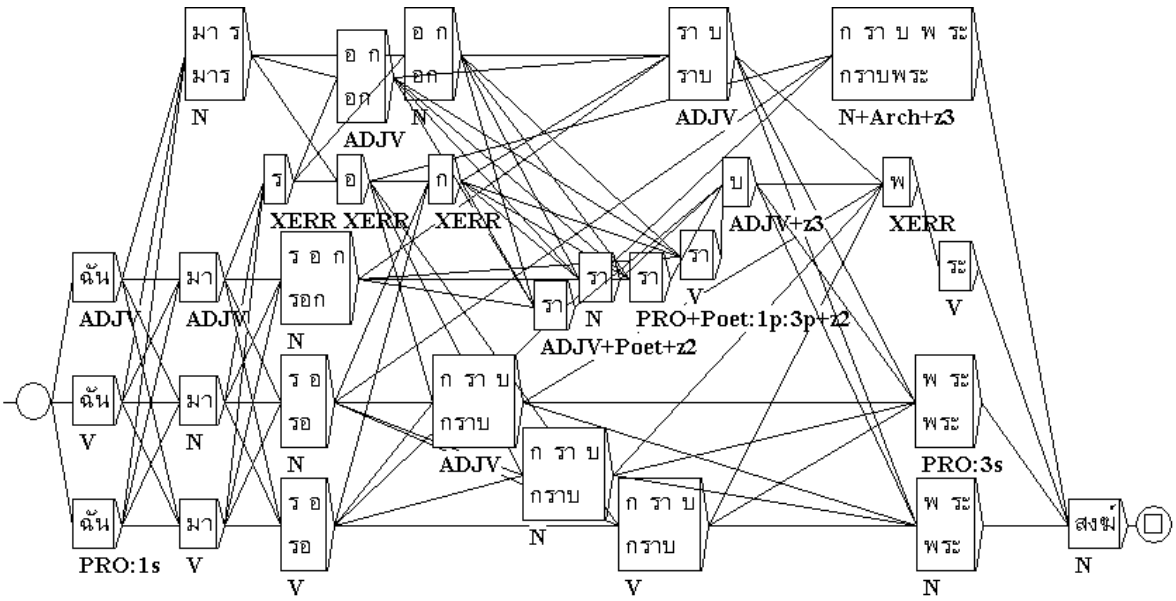

Figure 6.34 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » en version P3

Sachant que notre phrase ne comporte que des mots courants, nous pouvons encore réduire la complexité du transducteur en n'appliquant sur le texte que les dictionnaires z1 (27 étiquettes) :

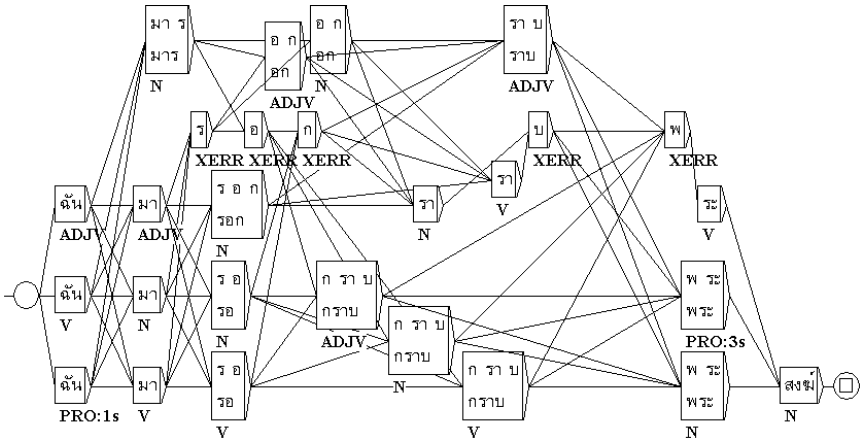

Figure 6.35 Transducteur du texte en version P3 appliqué uniquement avec les dictionnaires z1

6.4.2 Réduction des transducteurs du texte

Afin de réduire encore la complexité des transducteurs du texte, INTEX intègre l'option « Remove .Xxx lexical items » dans le menu de la construction du transducteur du texte (*cf. §1.3.4*). Cette option permet de ne pas présenter dans le résultat les mots inconnus (XERR) ainsi que les nœuds et les chemins associés, si et seulement si on peut passer par d'autres chemins concurrents et étiquetés. En conséquence, le transducteur du texte sera beaucoup plus simple comme dans les figures ci-dessous. Cependant, les noms propres absents des dictionnaires seraient également effacés avec cette option.

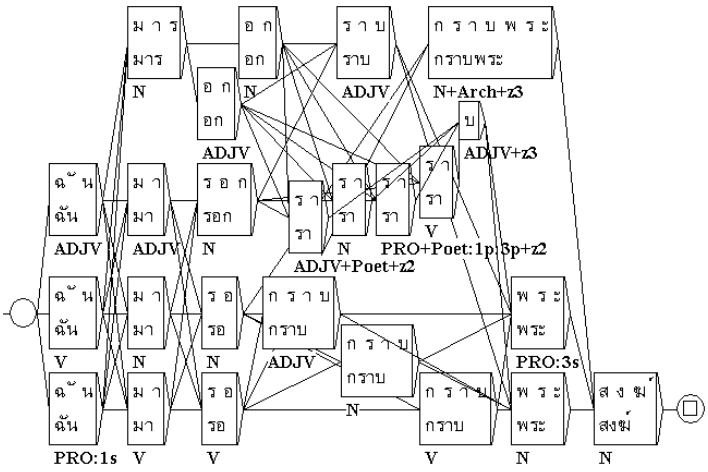

Figure 6.36 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » version P0 réduit

En comparant le transducteur ci-dessus (25 étiquettes) avec celui de la version P3 réduit ci-dessous (25 étiquettes également), nous constatons que les deux sont équivalents. En effet, la méthode de réduction donne le même résultat que la méthode de segmentation en mots par dictionnaires de S. RATTHAYANOND[14] (*cf. §0.3.2.1*).

[14] RATTHAYANOND 1992

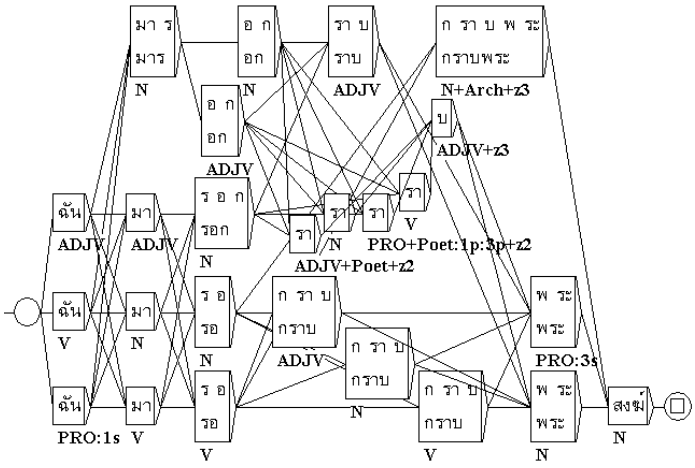

Figure 6.37 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » version P3 réduit

Remarque : Les mots supprimés du transducteur par la réduction restent dans le fichier de vocabulaire du texte. Il suffit de reconstruire le transducteur du texte en désactivant l'option de réduction pour retrouver le transducteur original.

Discussion : Même si, après la réduction, le transducteur du texte d'une version supérieure est équivalent à celui d'une version inférieure, la version supérieure est toujours plus avantageuse car il reste moins de bruit dans son vocabulaire du texte. La plupart des fonctionnalités d'INTEX comme par exemple, la recherche de motifs (*cf. §1.3.2*), l'étiquetage linéaire d'un texte (*cf. §1.3.3.3*) travaillent toujours sur le vocabulaire du texte et non sur le transducteur du texte. Ce qui veut dire qu'en utilisant ces fonctions avec un texte d'une version inférieure, on sera plus gêné par le bruit qu'avec celui d'une version supérieure.

En réduisant le transducteur du texte appliqué uniquement avec les dictionnaires z1 (*cf. Figure 6.35*), le résultat est encore simplifié (19 étiquettes) :

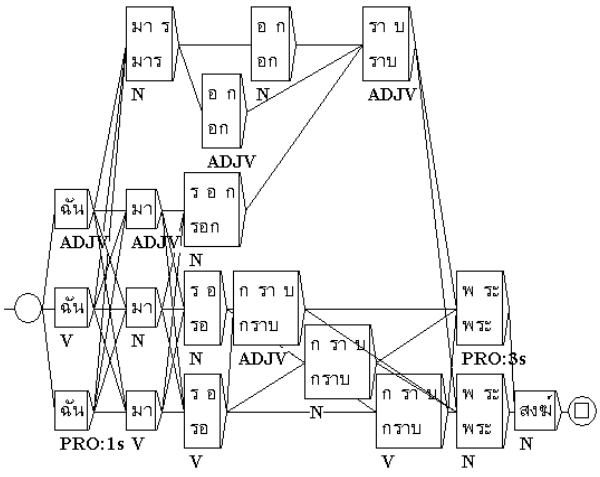

Figure 6.38 Transducteur du texte en version P3 appliqué uniquement avec les dictionnaires z1 et réduit

Grâce à la réduction, certains textes sont totalement désambiguïsés. Par exemple, le texte « อีกเพียงวันเดียว » [ʔiːkphiaːŋwandiaːw] en version P0 a la structure suivante (29 étiquettes) :

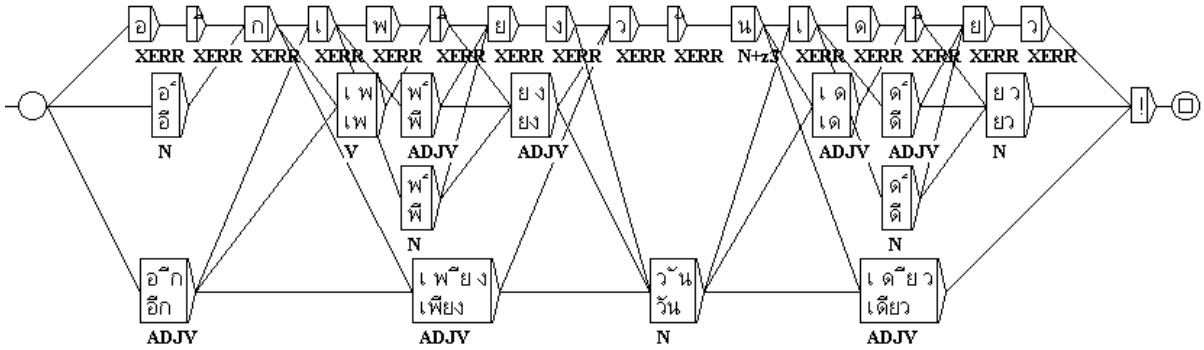

Figure 6.39 Transducteur du texte « ʔiːkphiaːŋwandiaːw » en version P0

Même si, avec la version P3, son transducteur du texte est fortement simplifié (10 étiquettes seulement), on ne peut toujours pas tout désambiguïser :

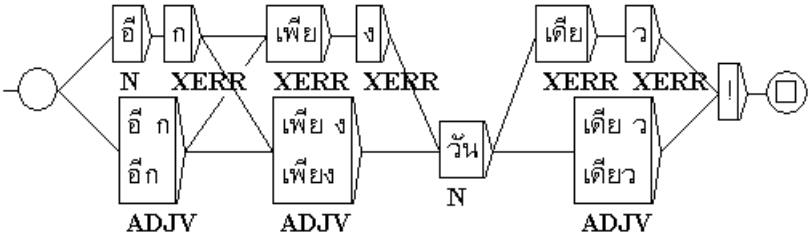

Figure 6.40 Transducteur du texte « ʔiːkphiaːŋwandiaːw » en version P3

Avec la réduction, le texte (quelle que soit la version) est totalement désambiguïsé comme dans la figure ci-dessous (4 étiquettes) :

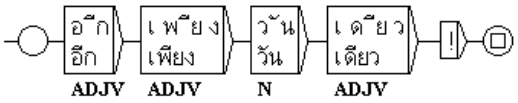

Figure 6.41 Transducteur du texte « ʔiːkphiaːŋwandiaːw » version P0 réduit

อีก [ʔiːk] (encore)
เพียง [phiaːŋ] (seul)
วัน [wan] (jour)
เดียว [diaːw] (un)

⇨ « Encore un seul jour ! »

Néanmoins, tous les textes ne peuvent pas être désambiguïsés de cette façon, il faut donc faire appel à une méthode plus élaborée.

6.5 Grammaires locales de levée d'ambiguïtés

Même si la structure de la phrase est vraiment complexe, elle a généralement une seule interprétation. Si nous connaissons la structure grammaticale de la phrase, nous pouvons créer une grammaire locale de levée d'ambiguïtés (*cf. §1.3.3.2*) sous forme d'un transducteur pour l'appliquer au transducteur du texte. Nous obtenons ainsi comme résultat un transducteur du texte désambiguïsé. Par exemple, la phrase ci-avant « ฉันมารอกราบพระสงฆ์ » [chănmaːrɔːkràːpphrásŏŋ] a la structure correspondant à la grammaire locale suivante :

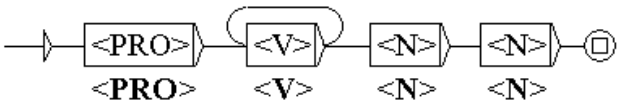

Figure 6.42 Grammaire locale de la phrase « chănmaːrɔːkràːpphrásŏŋ »

En appliquant cette grammaire sur le transducteur du texte, quelle que soit la version, nous obtenons le même résultat entièrement désambiguïsé ci-dessous :

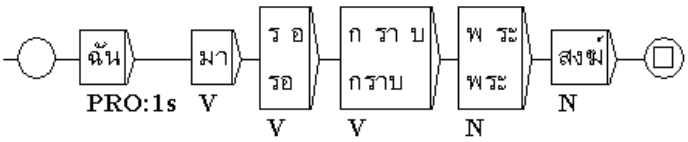

Figure 6.43 Transducteur du texte « chănmaːrɔːkràːpphrásŏŋ » désambiguïsé

ฉัน [chăn] (je)
มา [maː] (venir)
รอ [rɔː] (attendre)
กราบ [kràːp] (se prosterner)
พระ [phrá] (prêtre)
สงฆ์ [sŏŋ] (moine bouddhiste) } (bonze)

⇨ « Je viens, (j')attends, (je) me prosterne (devant le) bonze. »

En effet, une phrase thaï peut accepter plusieurs verbes successifs pour des actions consécutives (venir → attendre → se prosterner). Quant au complément d'objet dans cet exemple, c'est un groupe nominal de type NN.

Le travail de levée d'ambiguïtés consiste donc à analyser la syntaxe de la langue thaï afin de concevoir des graphes de grammaires locales, qui désambiguïsent chacun une partie des textes, dans le but que l'ensemble de ces grammaires désambiguïse tous les textes en entier. Cela demande une étude approfondie qui sort du cadre de ce travail. Cependant, nous présentons, à l'aide de quelques exemples ci-dessous, l'efficacité des grammaires locales :

- L'exemple suivant est une grammaire locale qui traite des interjections. En effet, certaines formes d'interjection du thaï peuvent correspondre à plusieurs catégories grammaticales. Mais si ces formes sont suivies d'un point d'exclamation, ce sont forcément des interjections.

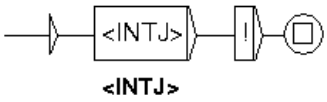

Figure 6.44 Grammaire locale de l'interjection

Par exemple, le mot « โถ ! » [thŏ:] a deux parties de discours : interjection (Mon pauvre !) ou nom (vase avec couvercle).

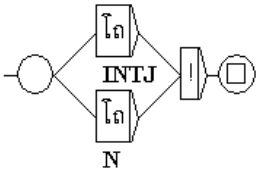

Figure 6.45 Transducteur du texte « thŏ: ! »

Après l'application de la grammaire locale de l'interjection, le transducteur du texte est désambiguïsé et réduit à :

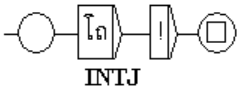

Figure 6.46 Transducteur du texte « thŏ: ! » désambiguïsé

Plus le mot est long, plus la grammaire est utile ; par exemple, l'interjection « แม่เจ้าโว้ย ! » [mê:câwwó:j] (Oh la la !) a une structure plus complexe à cause de la détection de parties des mots (*cf. §4.1.1.6.1*).

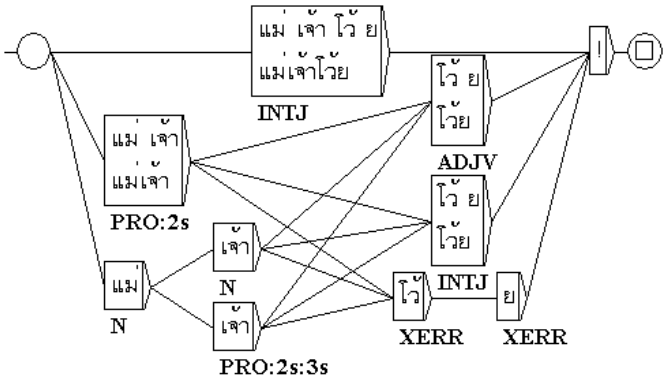

Figure 6.47 Transducteur du texte « mê:câwwó:j ! »

Grâce à la grammaire locale de l'interjection, le transducteur du texte est totalement désambiguïsé :

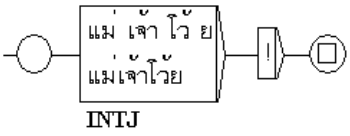

Figure 6.48 Transducteur du texte « mêːcâwwóːj ! » désambiguïsé

Discussion : Le point d'exclamation comme d'autres signes de ponctuation n'est jamais obligatoire en thaï. Si ce signe n'est pas présent dans le texte, la grammaire locale de l'interjection ci-dessus ne peut rien désambiguïser.

- L'exemple suivant est une grammaire qui traite les verbes auxiliaires. En thaï, les verbes auxiliaires apparaissent devant les verbes principaux. Lorsque l'on trouve de telles séquences, on élimine les autres possibilités par l'application du transducteur ci-dessous.

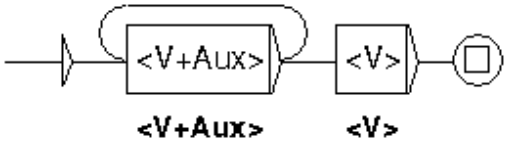

Figure 6.49 Grammaire locale du verbe auxiliaire

Par exemple, la phrase « เธอจะต้องมา » [thɣːcàtɔ̂ːŋmaː] (Elle devra venir.) est composée de 2 verbes auxiliaires : « จะ » [cà] indiquant le temps futur et « ต้อง » [tɔ̂ːŋ] indiquant une obligation, et d'un verbe principal : « มา » [maː] (venir). Elle a la structure suivante :

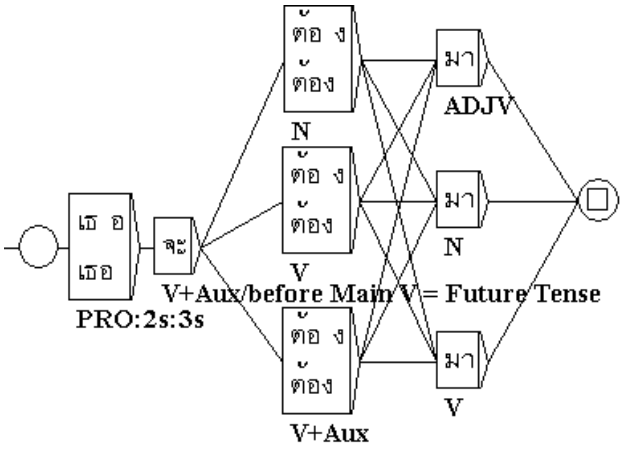

Figure 6.50 Transducteur du texte « thɣːcàtɔ̂ːŋmaː »

Grâce à la grammaire locale du verbe auxiliaire, le transducteur du texte est totalement désambiguïsé et se réduit à :

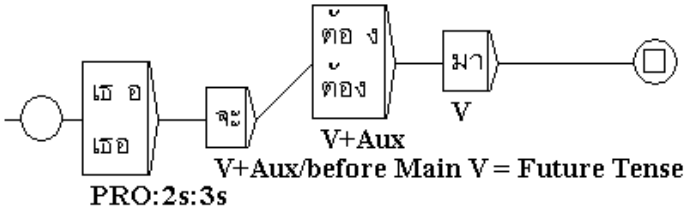

Figure 6.51 Transducteur du texte « thɣːcàtɔ̂ːŋmaː » désambiguïsé

Cette grammaire locale peut produire des erreurs en éliminant des chemins corrects. Considérons par exemple, la séquence « เธอจะต้องโทษ » [thɣːcàtɔ̂ːŋthôːt] qui a la structure suivante :

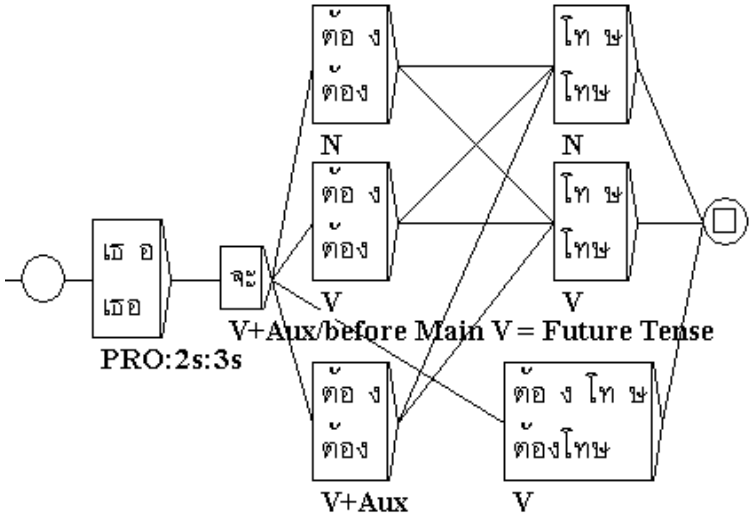

Figure 6.52 Transducteur du texte « thɣːcàtɔ̂ːŋthôːt »

Cette séquence est en fait ambiguë : la première interprétation est une partie de phrase, composée de 2 verbes auxiliaires : « จะ » [cà] indiquant le temps futur et « ต้อง » [tɔ̂ːŋ] indiquant une obligation, et d'un verbe transitif principal : « โทษ » [thôːt] (accuser) ; la deuxième interprétation est une phrase, composée d'un verbe auxiliaire : « จะ » [cà] indiquant le temps futur et d'un verbe intransitif principal qui est un mot composé : « ต้องโทษ » [tɔ̂ːŋthôːt] (être jugé coupable).

À cause du caractère glouton du transducteur avec boucle, la grammaire locale de la Figure 6.49 ne produit que l'unique résultat ci-dessous, qui est la première interprétation et qui signifie « Elle devra accuser… ».

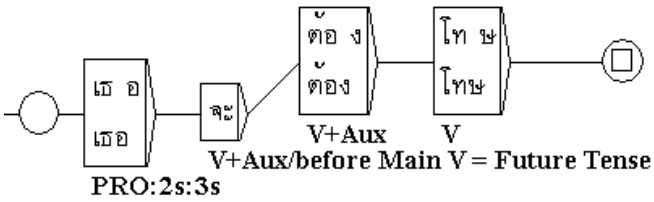

Figure 6.53 Transducteur du texte « thɣ:càtɔ̂:ŋthô:t » retenu par la grammaire locale

La deuxième interprétation (ci-dessous) qui signifie « Elle sera jugée coupable » est absolument ignorée.

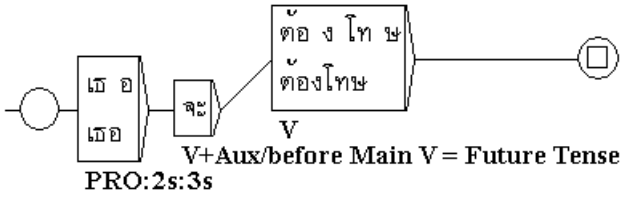

Figure 6.54 Transducteur du texte « thɣ:càtɔ̂:ŋthô:t » ignoré par la grammaire locale

Cet exemple montre la nécessité de faire plusieurs tests lorsqu'on écrit une grammaire locale avant de la mettre à la disposition des utilisateurs afin d'être certain de ne pas commettre d'erreur.

Une grammaire locale résout parfois l'ambiguïté de segmentation. Par exemple, la séquence « ตากลม » [ta:klom] (*cf. §2.3.3*) a 2 possibilités de découpage :

ตา/กลม [ta:/klom] (œil rond)

ตาก/ลม [ta:k/lom] (prendre l'air)

Mais son transducteur du texte a une structure plus complexe, car il contient plus de deux chemins :

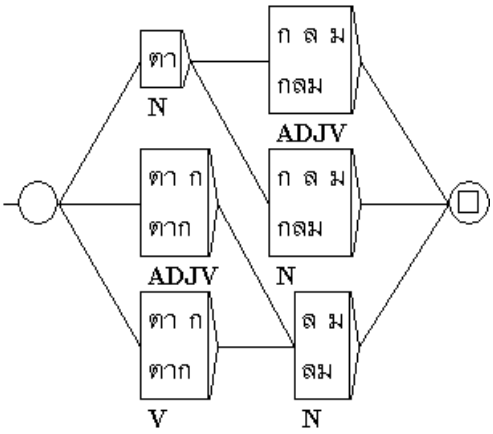

Figure 6.55 Transducteur du texte « ta:klom »

- Dans certains contextes, nous pouvons déterminer la segmentation correcte. Par exemple, si cette séquence est suivie du mot « สวย » [sŭa:j] (beau) ou « โต » [to:] (grand), nous validons la première segmentation (œil rond) ; si elle est suivie du mot « เย็น » [jen] (frais), nous validons la deuxième (prendre l'air). Même si le contexte ne permet pas de choisir le bon chemin, nous pouvons quand même en écarter quelques-uns. Ceci se fait par l'application de la grammaire locale suivante :

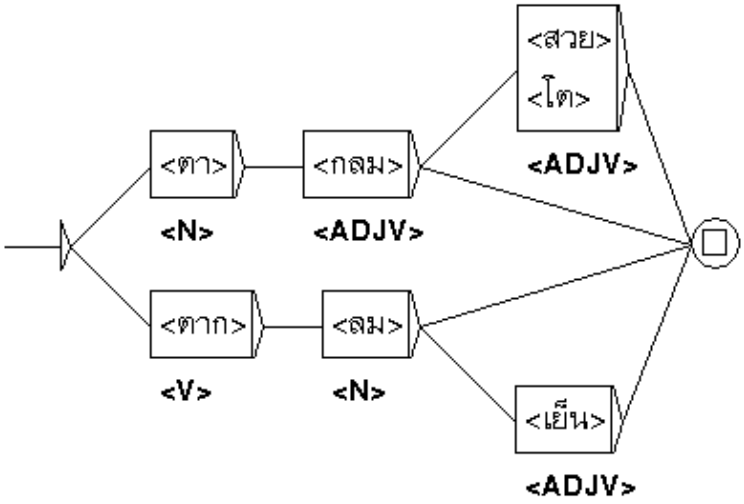

Figure 6.56 Grammaire locale de la séquence « ta:klom »

Par exemple, la figure ci-dessous est le transducteur du texte de la séquence « ตากลมสวย » [ta:klomsŭa:j] :

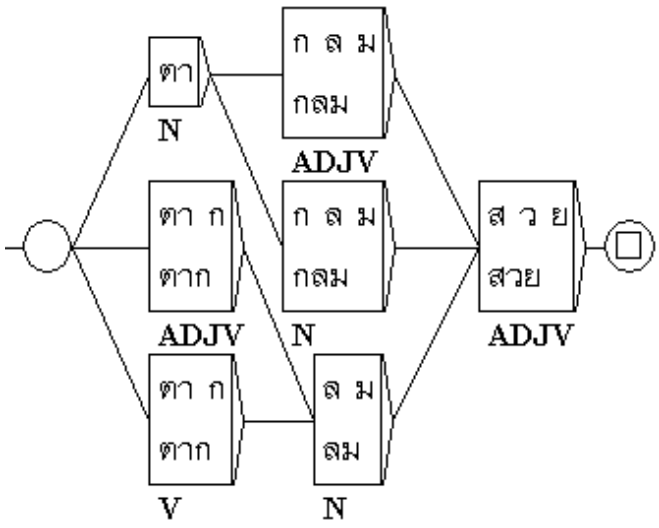

Figure 6.57 Transducteur du texte « ta:klomsŭa:j »

Après l'application de la grammaire locale de la Figure 6.56, le transducteur est entièrement désambiguïsé et se réduit à la figure ci-dessous qui signifie « bel(beaux) œil(yeux) rond(s) ».

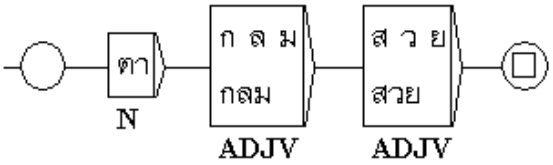

Figure 6.58 Transducteur du texte « ta:klomsŭa:j » désambiguïsé

La figure ci-dessous est le transducteur du texte de la séquence « ตากลมโต » [ta:klomto:] :

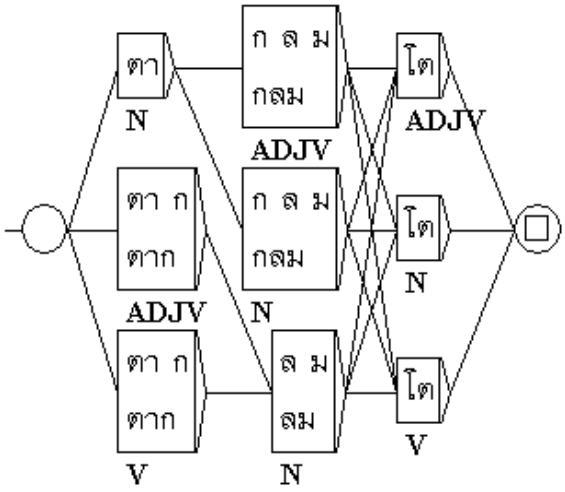

Figure 6.59 Transducteur du texte « ta:klomto: »

Après l'application de la grammaire locale de la Figure 6.56, le transducteur est entièrement désambiguïsé et se réduit à la figure ci-dessous qui signifie « grand(s) œil(yeux) rond(s) ».

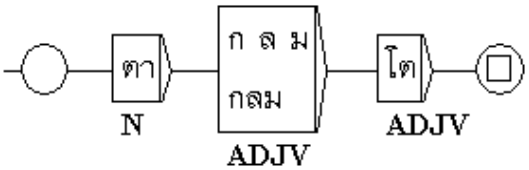

Figure 6.60 Transducteur du texte « ta:klomto: » désambiguïsé

La figure ci-dessous est le transducteur du texte de la séquence « ตากลมเย็น » [ta:klomjen] :

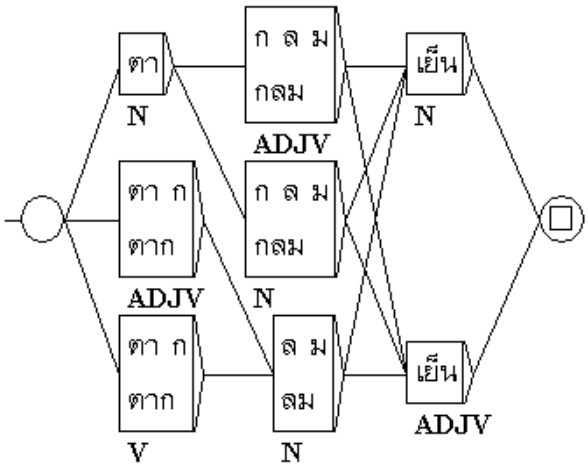

Figure 6.61 Transducteur du texte « ta:klomjen »

Après l'application de la grammaire locale de la Figure 6.56, le transducteur est entièrement désambiguïsé et se réduit à la figure ci-dessous qui signifie « prendre l'air frais ».

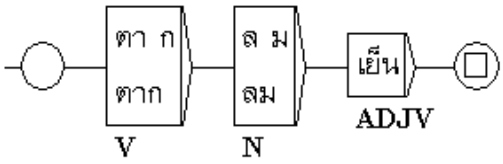

Figure 6.62 Transducteur du texte « ta:klomjen » désambiguïsé

Sans les contextes adéquats, la grammaire locale de la Figure 6.56 ne peut pas lever toutes les ambiguïtés. Malgré cela, elle peut quand même réduire le nombre de chemins de la Figure 6.55 qui devient :

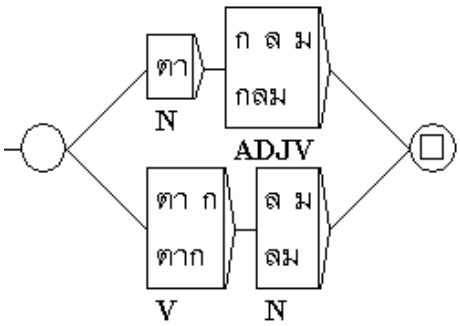

Figure 6.63 Transducteur du texte « ta:klom » partiellement désambiguïsé

Lorsque le transducteur du texte n'est pas entièrement désambiguïsé (*cf. Figure 6.63*), il faut essayer d'étendre la grammaire locale en prenant en compte d'autres contextes, d'autres contraintes ou d'autres informations qui pourraient éliminer les ambiguïtés restantes.

On pourrait imaginer que, lorsqu'on aura accumulé toutes les grammaires locales couvrant toute la syntaxe de la langue thaï, on ne soit plus gêné par les différentes versions du corpus (P0 - P3) car toutes les versions (même contenant énormément de bruit) seraient alors désambiguïsées et les transducteurs du texte se ramèneraient à une seule forme linéaire comme montré dans la plupart des exemples ci-dessus. Ce résultat constitue l'objectif idéal à atteindre pour tout analyseur lexical. Un futur projet de recherche pourrait consister à construire l'ensemble des grammaires locales du thaï dans ce but.

CONCLUSION ET PERSPECTIVES

Malgré les différences structurales entre la langue thaï et les langues occidentales, nous avons réussi, pour la première fois, à traiter le thaï dans le système INTEX©. Il s'agit d'une étape importante vers l'intégration dans la communauté RELEX. Des descriptions de la langue, sous forme de dictionnaires électroniques, selon le format DELA, et de graphes d'expressions, ont été créées. De nouvelles méthodes de segmentation en mots et en phrases, basées sur des expressions rationnelles et des transducteurs à nombre fini d'états, compatibles avec INTEX©, ont été proposées et évaluées. Nous avons constaté que notre méthode par syllabes (*cf. §4.1.2*) gagne en *précision* par rapport à notre méthode par caractères (*cf. §4.1.1*), tout en ne perdant rien sur le *rappel*. Par rapport aux autres méthodes de l'état de l'art, nos méthodes sont supérieures en termes de *rappel* pour la segmentation en mots (*cf. §5.2.2*) et s'approchent de la meilleure méthode connue en termes de *précision* pour la segmentation en phrases (*cf. §5.2.4*). Nous avons également montré que, finalement, notre système est capable de faire différents types d'analyse automatique de textes thaï, même plus que pour les langues européennes car nous fournissons plusieurs versions de corpus et de dictionnaires, selon la préférence des utilisateurs : P0 et P1 sont conçus pour l'analyse morphologique (*cf. §6.1*), P3 pour l'analyse lexicale (*cf. §6.2*), P1 et P2, en attendant un dictionnaire des syllabes (qui n'est pas encore réalisé), pour l'analyse syllabique. Quant à l'analyse syntaxique (*cf. §6.4*), elle peut, en cas d'ambiguïté, présenter ses résultats sous forme d'automates avec plusieurs chemins en parallèle, sur lesquels nous pouvons appliquer des grammaires locales de levée d'ambiguïtés (*cf. §6.5*) afin d'éliminer ou de réduire le bruit. Nous pouvons donc conclure que notre objectif de départ, qui est de concevoir et de réaliser un outil informatico-linguistique apte à effectuer des analyses automatiques de textes thaï, a été atteint.

Les méthodes de segmentation présentées dans notre travail pourraient s'adapter à d'autres langues sans séparateur telles que le chinois ou le japonais, et même au traitement de la parole, par reconnaissance des mots dans des séquences contiguës de caractères phonétiques.

Pour améliorer notre système, il sera nécessaire d'incorporer à nos dictionnaires davantage de mots, notamment des mots composés (au sens linguistique), et surtout d'avoir plus d'informations (lexicales, syntaxiques, sémantiques ou même pragmatiques) pour chaque entrée, afin d'affiner les grammaires locales de levée d'ambiguïtés et de construire des grammaires de phrases pour une analyse syntaxique syntagmatique qui transformera une phrase thaï en arbres.

Néanmoins, il reste encore quelques problèmes à résoudre, notamment en ce qui concerne la notion de lexème, qui n'est pas adaptée à la langue thaï, ainsi que le tri alphabétique « à la thaïlandaise », qui n'a pas pu être intégré, à cause de problèmes de compatibilité avec d'autres langues dans le système.

Étant donné que chaque langue a des caractéristiques spécifiques, une bonne solution serait de regrouper les langues partageant certaines particularités. Par exemple, diviser, dans un premier temps, les langues du système en deux groupes : langues avec séparateurs et langues sans séparateur[1]. La notion de lexème du premier groupe sera la même que celle utilisée traditionnellement, c'est-à-dire une séquence contiguë de lettres entre deux séparateurs, tandis que celle du deuxième groupe, qui inclut le chinois, le japonais et le thaï, sera définie par caractère. Cette solution donnera à notre système la capacité de traiter ces langues sans devoir ajouter de séparateurs de mots et produira un résultat tout à fait équivalent à notre méthode de segmentation par caractères (*cf. §4.1.1*) qui nous permet de faire plusieurs types d'analyse, y compris l'analyse morphologique. En revanche, le bruit très élevé (*cf. §4.1.1.6*) sera le problème principal dans ce deuxième groupe. Ce phénomène pourra être réduit, pour le thaï, en prenant en compte, dans la définition des lexèmes, des règles de composition syllabique, décrites dans notre méthode par syllabes (*cf. §4.1.2*). Ce qui veut dire que, dans l'avenir, le thaï et probablement chacune des langues particulières auraient une notion de lexème différente et feraient l'objet d'un traitement distinct.

[1] C'est le cas du système UNITEX [PAUMIER 2002] : un autre système d'analyse automatique de textes, créé en 2002 au sein de notre Laboratoire d'Informatique, Institut Gaspard Monge, Université de Marne-la-Vallée.

BIBLIOGRAPHIE

ASHER R.E., SIMPSON J.M.Y. (eds.) 1994. *The Encyclopedia of Language and Linguistics*, Vol. 9. Pergamon Press, Oxford.

CHARNYAPORNPONG Surin 1983. *A Thai Syllable Separation Algorithm*. Master Thesis of Engineering. Asian Institute of Technology, Pathumthani.

CHAROENPORNSAWAT Paisarn 1998. *การตัดคำภาษาไทยโดยใช้คุณลักษณะ* (*Feature-based Thai Word Segmentation*). [in Thai] Master Thesis of Engineering. Department of Computer Engineering, Chulalongkorn University, Bangkok.

CHAROENPORNSAWAT Paisarn, SORNLERTLAMVANICH Virach 2001. Automatic Sentence Break Disambiguation for Thai. *Proceeding of the 19th International Conference on Computer Processing of Oriental Languages (ICCPOL2001), May 2001*. Seoul, pp 231-235.

CHOI Sung-Woo 1996. *Hangul Code Manager (for KSC5601 – 1987 WANSUNG Code)*. Rapport interne: IGM 96 – 31. Université de Marne-la-Vallée.

CHOI Sung-Woo 1999. *Implantation de dictionnaires électroniques du coréen par automates finis*. Thèse de doctorat en Informatique. IGM, Université de Marne-la-Vallée.

CHROBOT Agata 1998. Flexion automatique des mots composés. In: KLEIN Jean René, LAMIROY Béatrice, PIERRET Jean-Marie (eds.) 1998, pp. 145-159.

COMRIE Bernard (ed.) 1987. *The World's Major Languages.* Croom Helm, London.

CONSTANT Matthieu 2002. On the Analysis of Locative Phrases with Graphs and Lexicon-Grammar: the Classifier/Proper Noun Pairing. In: RANCHHOD Elisabete, MAMEDE Nuno J. (eds.) 2002, pp. 33-42.

COURTOIS Blandine 1990. Un système de dictionnaires électroniques pour les mots simples du français. In: COURTOIS Blandine, SILBERZTEIN Max (eds.) 1990, pp. 11-22.

COURTOIS Blandine 1992. *Dictionnaire électronique des mots simples du français: DELAS V07-E1*. LADL, Université Paris VII.

COURTOIS Blandine, SILBERZTEIN Max (eds.) 1990. *Dictionnaires électroniques du français*. Langue Française, N° 87 (septembre 1990). Larousse, Paris.

COYAUD Maurice 1995. *Graphies et phonies (Introduction aux systèmes d'écriture)*. P.A.F., Paris.

COYAUD Maurice 1997. *Les langues dans le monde: Graphies et phonies (Introduction aux systèmes d'écriture)*, Tome 2. P.A.F., Paris.

DELOUCHE Gilles 1994. *Méthode de thaï*, Vol. 1. L'Asiathèque, Paris.

DISTER Anne 1998-1999. Développer des grammaires locales de levée d'ambiguïtés pour INTEX. In: FAIRON Cédrick (ed.) 1998-1999, pp. 233-247.

DISTER Anne (ed.) 2000. *Actes des Troisièmes Journées INTEX, 13-14 juin 2000*. Revue Informatique et Statistique dans les Sciences humaines, Vol. 36. C.I.P.L., Liège.

DISTER Anne 2001. Levée d'ambiguïtés sur les mots lexicaux et grammaticaux. In: LAPORTE Éric (ed.) 2001, pp. 105-126.

DUBOIS Jean, GIACOMO Mathée, GUESPIN Louis, MARCELLESI Christiane, MARCELLESI Jean-Baptiste, MÉVEL Jean-Pierre 1994. *Dictionnaire de linguistique et des sciences du langage*. Larousse, Paris.

DUGAS André 1997. *Le guide de la ponctuation*. Les Éditions LOGIQUES, Montréal.

FAIRON Cédrick (ed.) 1998-1999. *Analyse lexicale et syntaxique: Le système INTEX*. Lingvisticæ Investigationes, Tome XXII, Volume Spécial. John Benjamins, Amsterdam / Philadelphia.

FAIRON Cédrick 2000. *Structures non-connexes, Grammaire des incises en français : description linguistique et outils informatiques*. Thèse de doctorat en Informatique Fondamentale et Applications. LADL, Université Paris VII.

FRIBURGER Nathalie, DISTER Anne, MAUREL Denis 2000. Améliorer le découpage des phrases sous INTEX. In: DISTER Anne (ed.) 2000, pp. 181-199.

GARRIGUES Mylène 1993. *Méthode de paramétrage des dictionnaires et grammaires électroniques : Application à des systèmes interactifs en langue naturelle*. Thèse de doctorat en Sciences du langage. LADL, Université Paris VII.

GREFENSTETTE Gregory, TAPANAINEN Pasi 1994. What is a word, What is a sentence? Problems of Tokenization. In: KIEFER Ferenc, KISS Gábor, PAJZS Júlia (eds.) 1994, pp. 79-87.

GROSS Maurice 1975. *Méthodes en syntaxe*. Hermann, Paris.

GROSS Maurice 1981. Les bases empiriques de la notion de prédicat sémantique. In: GUILLET Alain, LECLÈRE Christian (eds.) 1981, pp. 7-52.

GROSS Maurice 1989a. La construction de dictionnaires électroniques. *Annales des Télécommunications*, Tome 44, N° 1-2, pp. 4-19. CNET, Paris.

GROSS Maurice 1989b. The use of finite automata in the lexical representation of natural language. In: GROSS Maurice, PERRIN Dominique (eds.) 1989, pp. 34-50.

GROSS Maurice 1997. The Construction of Local Grammars. In: ROCHE Emmanuel, SCHABES Yves (eds.) 1997, pp. 329-354.

GROSS Maurice, PERRIN Dominique (eds.) 1989. *Electronic Dictionaries and Automata in Computational Linguistics*. Proceedings of the LITP Spring School on Theoretical Computer Science, Saint-Pierre d'Oléron, May 1987. LNCS 377. Springer-Verlag, Berlin Heidelberg.

GUENTHNER Franz 1998. Constructions, classes et domaines : concepts de base pour un dictionnaire électronique de l'allemand. In: LE PESANT Denis, MATHIEU-COLAS Michel (eds.) 1998, pp. 45-55.

GUILLET Alain, LECLÈRE Christian (eds.) 1981. *Formes syntaxiques et prédicats sémantiques*. Langages, N° 63 (septembre 1981). Larousse, Paris.

HARRIS Zellig 1970. *Papers in Structural and Transformational Linguistics*. D. Reidel, Dordrecht.

HARRISON P., ABNEY S., BLACK E., FLICKINGER D., GDANIEC C., GRISHMAN R., HINDLE D., INGRIA B., MARCUS M., SANTORINI B., STRZALKOWSKI T. 1991. Evaluating syntax performance of parser/grammars of English. *Proceedings of the Workshop on Evaluating Natural Language Processing Systems*. 29th Annual Meeting of the Association for Computational Linguistics. Berkeley, pp. 71-77.

HIRANYAKARN Suwit 1999. *พจนานุกรม ฉบับนักเรียน* (*Dictionary : Students Edition*). [in Thai] 1st edn. IQ Book Center, Bangkok.

HUDAK Thomas John 1987. Thai. In: COMRIE Bernard (ed.) 1987, pp. 757-775.

HUDAK Thomas John 1994. Thai. In: ASHER R.E., SIMPSON J.M.Y. (eds.) 1994, pp. 4600-4601.

KANCHANAWAN Nitaya 1999. *ปัญหาการใช้ภาษาไทย* (*Problems in Thai Usage*). [in Thai] 2nd edn. Ramkhamhaeng University Press, Bangkok.

KAROONBOONYANAN Theppitak 1997. การเรียงลำดับคำไทยที่มีเครื่องหมายวรรคตอน (Sorting Thai Words with Punctuation Marks). [in Thai] *NECTEC Journal*, Vol. 14 (Jan.-Feb. 1997). NECTEC, Bangkok.

KAROONBOONYANAN Theppitak 1999. Standardization and Implementation of Thai Language. *Seminar on Enhancement of the International Standardization Activities in Asia Pacific Region (AHTS-1), March 1999*. CICC, Japan.

KAROONBOONYANAN Theppitak. *Thai Sorting Algorithms*. http://www.links.nectec.or.th/~thep/tsort.html

KAWTRAKUL Asanee, THUMKANON Chalathip, SERIBURI Sapon 1995. A Statistical Approach to Thai Word Filtering. *Proceeding of the 2nd Symposium on Natural Language Processing (SNLP'95), 2-4 August 1995*. Bangkok, pp 398-406.

KHAMTHAIDEE Samphan 1992. อัลกอริธึมไทยๆ ตอน ไล่เรียงลำดับไทย (Thai Style Algorithm: Sorting Thai). [in Thai] *Computer Review*, N° 91 (March 1992). Man Group, Bangkok.

KIEFER Ferenc, KISS Gábor, PAJZS Júlia (eds.) 1994. *Papers in Computational Lexicography: COMPLEX '94*. Proceedings of the 3rd International Conference on Computational Lexicography and Text Research, 7-9 July 1994. Budapest.

KLEIN Jean René, LAMIROY Béatrice, PIERRET Jean-Marie (eds.) 1998. *Théorie linguistique et applications informatique*. Actes du 16e Colloque européen sur la grammaire et le lexique comparés, 24-27 septembre 1997. Cahiers de l'Institut de Linguistique de Louvain, CILL 24. 3-4, Vol. 1. Louvain-la-Neuve.

KOOPTIWOOT Chompunuch 1999. *การตัดคำกำกวมในข้อความภาษาไทยด้วยการโปรแกรมตรรกะเชิงอุปนัย* (*Segmentation of Ambiguous Thai Words by Inductive Logic Programming*). [in Thai] Master Thesis of Science. Department of Computer Engineering, Chulalongkorn University, Bangkok.

KORNAI András (ed.) 1999. *Extended finite state models of language*. Studies in natural language processing. Cambridge University Press, Cambridge.

KOSAWAT Krit 2000. Procédure de reconnaissance des mots et des phrases thaï. In: DISTER Anne (ed.) 2000, pp. 241-255.

KRABUANRAT Wanphen 1996. *หลักภาษาไทย ฉบับนักเรียน-นักศึกษา* (*Thai Grammar for students*). [in Thai] Pattana Suksa Press, Bangkok.

LABELLE Jacques, LECLÈRE Christian (eds.) 1995. *Lexiques grammaires comparés en français*. Lingvisticæ Investigationes Supplementa 17. John Benjamins, Amsterdam / Philadelphia.

LAMIROY Béatrice (ed.) 1998. *Le Lexique-grammaire*. Travaux de Linguistique, N° 37 (Décembre 1998). Duculot, Bruxelles.

LAPORTE Éric 1988. *Méthodes algorithmiques et lexicales de phonétisation de textes : Applications au français*. Thèse de doctorat en Informatique. Université Paris VII.

LAPORTE Éric 1995. Levée d'ambiguïtés par grammaires locales. In: LABELLE Jacques, LECLÈRE Christian (eds.) 1995, pp. 97-114.

LAPORTE Éric 2001. Reduction of lexical ambiguity. In: LAPORTE Éric (ed.) 2001, pp. 67-103.

LAPORTE Éric (ed.) 2001. *Description et levée des ambiguïtés*. Lingvisticæ Investigationes, Tome XXIV, Fascicule 1. John Benjamins, Amsterdam / Philadelphia.

LAPORTE Éric, MONCEAUX Anne 1998-1999. Elimination of lexical ambiguities by grammars: The ELAG system. In: FAIRON Cédrick (ed.) 1998-1999, pp. 341-367.

LENGCHAROEN Eun 1992. *ภาษาไทยปริทัศน์* (*Thai language review*). [in Thai] Bamrungsarn, Bangkok.

LE PESANT Denis, MATHIEU-COLAS Michel (eds.) 1998. *Les classes d'objets*. Langages, N° 131 (septembre 1998). Larousse, Paris.

LONGCHUPOLE Sungkornsarun 1995. *Thai Syntactical Analysis system by method of splitting sentences from paragraph for machine translation*. [in Thai] Master Thesis of Engineering. King Mongkut's Institute of Technology Ladkrabang, Bangkok.

MAUREL Denis 1989. *Reconnaissance de séquences de mots par automates, adverbes de date du français*. Thèse de doctorat en Informatique. Université Paris VII.

MEKNAVIN Surapant, CHAROENPORNSAWAT Paisarn, KIJSIRIKUL Boonserm 1997. Feature-based Thai Word Segmentation. *Proceedings of the Natural Language Processing Pacific Rim Symposium 1997 (NLPRS'97), 2-4 December 1997*. Phuket, pp. 41-46.

MICROSOFT CORPORATION. *Microsoft Windows Codepage : 874 (Thai)*. http://www.microsoft.com/globaldev/reference/sbcs/874.htm

MICROSOFT CORPORATION. *Microsoft Windows Codepage : 949 (Korean)*. http://www.microsoft.com/globaldev/reference/dbcs/949.htm

MICROSOFT CORPORATION. *Microsoft Windows Codepage : 1252 (Latin I)*. http://www.microsoft.com/globaldev/reference/sbcs/1252.htm

MITTRAPIYANURUK Pradit, SORNLERTLAMVANICH Virach 2000. The Automatic Thai Sentence Extraction. *Proceedings of the 4th Symposium on Natural Language Processing 2000 (SNLP2000), May 2000*. Chiang Mai, pp 23-28.

NAGATA Masaaki 1999. A Part of Speech Estimation Method for Japanese Unknown Words using a Statistical Model of Morphology and Context. *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics, 20-26 June 1999*. Maryland, pp. 277-284.

PALMER David D., HEARST Marti A. 1997. Adaptive Multilingual Sentence Boundary Disambiguation. *Computational Linguistics*, Vol. 23, N° 2, pp. 241-267.

PANUPONG Vichin 1982. *โครงสร้างของภาษาไทย* (*The Structure of Thai*). [in Thai] Ramkhamhaeng University Press, Bangkok.

PAUMIER Sébastien 2001. Some remarks on the application of a lexicon-grammar. *Lingvisticæ Investigationes*, Tome XXIV, Fascicule 2, pp. 245-256. John Benjamins, Amsterdam / Philadelphia.

PAUMIER Sébastien 2002. *Unitex : Manuel d'utilisation*. IGM, Université de Marne-la-Vallée. http://www-igm.univ-mlv.fr/~unitex/manuel.html

POOVARAWAN Yuen, IMARROM Wiwat 1986. การแบ่งแยกพยางค์ไทยด้วยดิกชันนารี (Thai Syllable Separator by Dictionary). [in Thai] *Proceedings of the 9th Annual Meeting on Electrical Engineering of the Thai Universities, 3-4 December 1986*. Khonkaen.

PROMCHAN Pisit, TENG-AMNUAY Yunyong 1998. Performance Comparison of Thai Word Separation Algorithms. *Proceedings of the National Computer Science and Engineering Conference 1998 (NCSEC'98), 19-21 October 1998*. Bangkok.

RANCHHOD Elisabete, MAMEDE Nuno J. (eds.) 2002. *Advances in Natural Language Processing*. Proceedings of the Third International Conference, PorTAL 2002, June 2002. Faro, Portugal. LNAI 2389. Springer, Berlin / Heidelberg / New York.

RARUENROM Samphan 1991. *การแบ่งคำไทยด้วยพจนานุกรม* (*Dictionary-based Thai Word Separation*). [in Thai] Senior Project Report. Department of Computer Engineering, Chulalongkorn University, Bangkok.

RATTHAYANOND Somprathana 1992. *โครงสร้างข้อมูลสำหรับพจนานุกรมอิเล็กทรอนิกส์ภาษาไทย* (*Data Structure for Thai Electronic Dictionary*). [in Thai] Master Thesis of Science. Department of Computer Engineering, Chulalongkorn University, Bangkok.

ROCHE Emmanuel 1993. Une représentation par automate fini des textes et des propriétés transformationnelles des verbes. *Lingvisticæ Investigationes*, Tome XVII, Fascicule 1, pp. 189-222. John Benjamins, Amsterdam / Philadelphia.

ROCHE Emmanuel 1999. Finite state transducers: parsing free and frozen sentences. In: KORNAI András (ed.) 1999, pp. 108-120.

ROCHE Emmanuel, SCHABES Yves (eds.) 1997. *Finite-State Language Processing*. MIT Press, Cambridge, Massachusetts.

RONNAKIAT Nantana 1994. Thai Writing System: History. In: ASHER R.E., SIMPSON J.M.Y. (eds.) 1994, pp. 4601-4602.

ROYAL INSTITUTE 1996a. *พจนานุกรม ฉบับราชบัณฑิตยสถาน พ.ศ. ๒๕๒๕* (*The Royal Institute Dictionary 2525 B.E.*). [in Thai] 6th edn. Aksorn Charoen That, Bangkok.

ROYAL INSTITUTE 1996b. *อ่านอย่างไรและเขียนอย่างไร* (*How to read and how to write*). [in Thai] 13th edn. Comform, Bangkok.

SAVARY Agata 2000. *Recensement et description des mots composés – méthodes et applications*. Thèse de doctorat en Informatique Fondamentale. LADL, Université de Marne-la-Vallée.

SAWAMIPAK Duangkaew 1990. *การสร้างซอฟต์แวร์วิเคราะห์ไวยากรณ์ไทยภายใต้ระบบยูนิกซ์* (*Construction of Thai Syntax Analysing Software under UNIX*). [in Thai] Thammasart University Press, Bangkok.

SENELLART Jean 1998. Reconnaissance automatique des entrées du lexique-grammaire des phrases figées. In: LAMIROY Béatrice (ed.) 1998, pp. 109-125.

SENELLART Jean 1999. *Outils de reconnaissance d'expressions linguistiques complexes dans des grands corpus*. Thèse de doctorat en Informatique Fondamentale et Applications. LADL, Université Paris VII.

SILBERZTEIN Max 1989. *Dictionnaires électroniques et reconnaissance lexicale automatique*. Thèse de doctorat en Informatique. Université Paris VII.

SILBERZTEIN Max 1990. Le dictionnaire électronique des mots composés. In: COURTOIS Blandine, SILBERZTEIN Max (eds.) 1990, pp. 71-83.

SILBERZTEIN Max 1993. *Dictionnaires électroniques et analyse automatique de textes. Le système INTEX*. Masson, Paris.

SILBERZTEIN Max 1996. *INTEX 3.3 reference manual.* LADL, Université Paris VII.

SILBERZTEIN Max 1998-1999a. Les graphes INTEX. In: FAIRON Cédrick (ed.) 1998-1999, pp. 3-29.

SILBERZTEIN Max 1998-1999b. Traitement des expressions figées avec INTEX. In: FAIRON Cédrick (ed.) 1998-1999, pp. 425-449.

SILBERZTEIN Max 2000. *INTEX*. LADL, Université Paris VII.

SILBERZTEIN Max, BALVET Antonio, POIBEAU Thierry 2002. Tutorial : Intex et ses applications informatique. *Cinquièmes Journées Intex, 2-3 mai 2002*. Marseille. http://www.lim.univ-mrs.fr/INTEX-2002-FR.html

SIRIBOONMA Aurapin 1999. *Erreurs et Difficultés en français rencontrées par des étudiants thaïlandais au niveau universitaire, à partir de l'analyse des productions écrites*. Thèse de doctorat en Linguistique. Université Paris V.

SMYTH D. 1994. Tai Languages. In: ASHER R.E., SIMPSON J.M.Y. (eds.) 1994, pp. 4520-4521.

SORNLERTLAMVANICH Virach 1993. การตัดคำไทยในระบบแปลภาษา (Word Segmentation for Thai in Machine Translation System). [in Thai] *Machine Translation*, pp. 50-56. NECTEC, Bangkok.

SORNLERTLAMVANICH Virach (ed.) 1995. *Papers on Natural Language Processing : Multi-lingual Machine Translation and Related Topics (1987-1994)*. NECTEC, Bangkok.

SORNLERTLAMVANICH Virach, CHAROENPORN Thatsanee, ISAHARA Hitoshi 1997. *ORCHID: Thai Part-Of-Speech Tagged Corpus*. Technical Report: TR-NECTEC-1997-001. NECTEC, Bangkok.

SPROAT Richard, SHIH Chilin, GALE William, CHANG Nancy 1996. A Stochastic Finite-State Word-Segmentation Algorithm for Chinese. *Computational Linguistics*, Vol. 22, N° 3, pp. 377-404.

TAYLOR P., BLACK A. 1998. Assigning Phrase Breaks from part-of-speech Sequences. *Computer Speech and Language*, N° 12, pp. 99-117.

THAIRATANANOND Yupin 1981. *Towards the design of a Thai text syllable analyzer*. Master Thesis of Science. Asian Institute of Technology, Pathumthani.

THAWARANON Kobkool 1978. *การเว้นวรรคในการเขียนหนังสือไทย* (*Spacing in the Thai Writing*). [in Thai] Master Thesis of Arts. Department of Thai Language, Chulalongkorn University, Bangkok.

THEERAMUNKONG Thanaruk, SORNLERTLAMVANICH Virach, TANHERMHONG Thanasan, CHINNAN Wirat 2000. Character Cluster Based Thai Information Retrieval. *Proceedings of the 5th International Workshop on Information Retrieval with Asian Languages (IRAL2000), September-October 2000*. Hong Kong, pp 75-80.

THONGLOO Kamchai 1997. *หลักภาษาไทย* (*Thai Grammar*). [in Thai] 10th edn. Ruamsarn, Bangkok.

UPPAKITSILAPASARN 2000. *หลักภาษาไทย* (*Thai Grammar*). [in Thai] 9th edn. Thai Wattana Panich, Bangkok.

VARAKULSIRIPUNTH Ruttikorn, NGAMWIWIT Jongkol, JUNWUN Somsak, CHIWATTAYAKUL Suthatip, THIPCHAKSURAT Sakchai 1989. การตัดคำจากประโยคในภาษาไทยด้วยวิธีการเทียบคำที่ยาวที่สุด (Word Segmentation in Thai Sentence by Longest Word Mapping). [in Thai] In: SORNLERTLAMVANICH Virach (ed.) 1995, pp. 279-290.

YANAPRATHIP Thaweesak 1991. *พจนานุกรมนักเรียน ฉบับเฉลิมพระเกียรติ พ.ศ. ๒๕๓๐* (*Students Dictionary : Royal Celebrating Edition 2530 B.E.*). [in Thai] 7th edn. Wattana Panich, Bangkok.

Corpus

BOURGEOIS Paulette, CLARK Brenda 2002. *Franklin : Le Dessin de Franklin*. Deux Coqs d'Or, Paris.

DAILYNEWS THAILAND. *หนังสือพิมพ์เดลินิวส์* (*Dailynews Newspaper*). [in Thai] http://www.dailynews.co.th

MATICHON GROUP. *นิตยสารศิลปวัฒนธรรม* (*Art & Culture Magazine*). [in Thai] http://www.matichon.co.th/art

MATICHON GROUP. *หนังสือพิมพ์ประชาชาติธุรกิจ* (*Prachachart Turakij Newspaper*). [in Thai] http://www.matichon.co.th/prachachart

MATICHON GROUP. *หนังสือพิมพ์มติชน* (*Matichon Newspaper*). [in Thai] http://www.matichon.co.th/matichon

NATION GROUP. *หนังสือพิมพ์กรุงเทพธุรกิจ* (*Bangkok Business Newspaper*). [in Thai] http://www.bangkokbiznews.com

PRAMOJ Kukrit 2000. *สี่แผ่นดิน* (*Four Reigns*). [in Thai] 11th edn. Dokya 2000 Press, Bangkok.

VACHARAPHOL CO.,LTD. *หนังสือพิมพ์ไทยรัฐ* (*Thairath Newspaper*). [in Thai] http://www.thairath.co.th