Elaboration d'un composant syntaxique à base de grammaires d'arbres adjoints pour le vietnamien

Phuong Le-Hong

Nancy-Université
Université Nancy 2
UFR mathématiques et informatique — École doctorale IAEM Lorraine
Département de formation doctorale en informatique

Elaboration d'un composant syntaxique à base de grammaires d'arbres adjoints pour le vietnamien

THÈSE

présentée et soutenue publiquement le 22 octobre 2010

pour l'obtention du

Doctorat de l'université Nancy 2

(spécialité informatique)

par

Lê Hồng Phương

Composition du jury

Rapporteurs :	Isabelle Tellier, Professeur à l'Université d'Orléans Jacques Chauché, Professeur à l'Université de Montpellier 2
Examinateurs :	Anne Boyer, Professeur à l'Université Nancy 2 Tri Doi Tran, Professeur à l'Université Nationale du Viêtnam à Hanoï
Directeurs de thèse :	Laurent Romary, Directeur de Recherche INRIA Azim Roussanaly, Maître de Conférence à l'Université Nancy 2

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

Loria

Mis en page avec la classe thloria.

Remerciements

Je tiens tout d'abord à remercier Azim Roussanaly, mon encadrant, qui m'a guidé tout au long de ces années de thèse par sa direction et ses explications claires et précieuses. Je remercie également Laurent Romary, mon directeur de thèse, pour son encadrement, sa direction, son support et sa confiance pendant mes années de thèse.

Je voudrais remercier Isabelle Tellier et Jacques Chauché d'avoir accepté la lourde tâche de rapporteur. Je remercie aussi Anne Boyer et Trần Trí Dõi de me faire l'honneur de participer au jury.

Ma reconnaissance s'adresse également aux membres de l'équipe KIWI et de l'équipe TALARIS qui ont toujours été prêts à m'aider de mon stage du DEA à aujourd'hui. Remerciements en particulier à Anne Boyer, Armelle Brun, Sahbi Sidhom, Claire Gardent, Matthieu Quignard, Alexandre Denis, Yannick Parmentier, Eric Kow pour leur amitié et leur support durant ces années. Un grand merci à l'ensemble du personnel administratif et technique du laboratoire, notamment les assistantes d'équipes Antoinette Courrier, Cécilia Claude et Isabelle Blanchard.

De nombreux échanges ont initié plusieurs de mes travaux, je souhaite ainsi remercier Nguyễn Thị Minh Huyền, Mathias Rossignol, Hồ Tường Vinh, Nguyễn Phương Thái, Vũ Xuân Lương pour leur contribution et collaboration.

Je remercie le Département de Mathématiques, de Mécanique et d'Informatique de la faculté des Sciences, Université Nationale du Vietnam à Hanoi, pour m'avoir permis de suspendre mon travail durant les périodes passées en France.

Je ne saurais oublier le soutien de ma famille pour son encouragement durant ces longues années.

Table des matières

Liste des figures

Liste des tableaux

Première partie

Introduction générale

1

Introduction

Sommaire

1.1 Le composant syntaxique et ses applications

1.1.1 La notion de composant syntaxique

Le traitement automatique des langues naturelles (TAL) est décomposé en sous-tâches, reposant sur les différents niveaux d'analyse linguistique. La première tâche est l'analyse morphologique qui a pour but d'identifier des mots ou des morphèmes. La deuxième tâche est l'analyse syntaxique qui constitue l'identification des syntagmes et de leurs fonctions. La troisième tâche est l'analyse sémantique.

Chacune de ces tâches peut être elle-même décomposée. Pour l'analyse morphologique, on distingue ainsi la segmentation – l'identification des frontières de phrases et de mots, la lemmatisation – l'identification sous sa forme canonique et l'étiquetage – l'identification de la bonne catégorie morpho-syntaxique pour une forme donnée en contexte.

L'analyse syntaxique peut être décomposée dans les sous-tâches distinctes suivantes :
– le parenthésage – l'identification des frontières syntagmatiques majeures ;
– l'assignation de fonctions aux syntagmes distingués ou à leur tête ;

– la désambiguïsation syntaxique des têtes prédicatives (cadre de sous-catégorisation, actif/passif...) ;
– l'assignation d'une structure syntaxique globale (un arbre) à chaque phrase.
Il est clair que les tâches ne sont pas indépendantes les unes des autres et qu'à chaque niveau, des connaisances de niveaux différents sont nécessaires.

Nous concevons le composant syntaxique comme un module commun pour des applications diverses, qui peut être constitué de façon indépendante du domaine d'application, et de la nature de l'application.

1.1.2 Les applications du composant syntaxique

L'analyse syntaxique constitue aujourd'hui une étape essentielle de TAL dès lors qu'elle recherche une connaissance relativement fine des relations grammaticales présentes dans une phrase. Le composant syntaxique permet de modéliser et d'analyser la structure syntaxique de la phrase et ainsi de préciser les relations existantes entre ses constituants, ce qui constitue un apport d'information très important pour l'accès au sens de la phrase.

On peut distinguer les applications demandant une analyse syntaxique complète (la traduction) et celles qui peuvent se contenter d'une analyse partielle (l'indexation, l'extraction d'informations...). On peut également distinguer celles qui demandent un analyseur robuste, tolérant aux fautes, et celles qui reposent sur un analyseur exigeant qui est capable de détecter toutes les agrammaticalités.

Parmi les nombreuses applications du composant syntaxique, nous pouvons citer des applications directes qui peuvent faire l'impasse sur les composants de plus haut niveau [6] : la correction orthographique, l'indexation automatique, le résumé de textes, l'alignement automatique de textes, l'extraction de connaissances à partir de textes, la génération de phrases et la traduction automatique.

1.2 Les connaissances syntaxiques

1.2.1 Les formalismes de grammaires

La représentation des connaissances syntaxiques est un problème de recherche en soi qui est toujours actif depuis les années 1960 jusqu'à aujourd'hui. Les années 1960 ont vu l'adoption enthousiaste des travaux en grammaire générative inventée par Chomsky [32]. Les difficultés rencontrées par ce type de grammaires transformationnelles (peu efficaces, difficiles à maintenir et à mettre à jour) ont directement inspiré la nouvelle génération de modèles regroupés sous le nom de grammaires d'unification dont certains sont liés à une véritable théorie linguistique. A la différence des grammaires d'inspiration chomskyenne, les grammaires d'unification sont véritablement formalisées, c'est-à-dire qu'elles ont recours à un métalangage mathématique ou logique bien défini. Si l'on se limite aux modèles qui incorporent une théorie linguistique, on en retient aujourd'hui trois principaux[1] :
– la *grammaire d'arbres adjoints* (Tree Adjoining Grammar ou TAG) [78, 79] ;

1. On peut mentionner la *grammaire syntagmatique généralisée* (Generalized Phrase Structure Grammar ou GPSG) [64], aujourd'hui abandonnée.

– la *grammaire lexicale fonctionnelle* (Lexical Functional Grammar ou LFG) [17] ;
– la *grammaire syntagmatique guidée par les têtes* (Head-driven Phrase Structure Grammar ou HPSG) [145].

Avec la stabilisation des formalismes de syntaxe, on assiste à un effort de construction de grammaires à larges couvertures des langues, qui vise à la réutilisabilité de ces grammaires pour de multiples tâches ultérieures.

1.2.2 Le développement de grammaires à large couverture

Les besoins spécifiques du TAL ont contribué au développement d'études linguistiques et à la description générale des langues. En effet, les grands projets de description générale d'une langue sont exceptionnels en linguistique, au moins pour ce qui concerne la syntaxe. Un des résultats du TAL a été de fixer comme objectif l'élaboration de grammaires à large couverture pour une langue. Ces grammaires sont les ressources linguistiques les plus importantes des composants syntaxiques.

Les ressources linguistiques

Aujourd'hui, les grammaires à large couverture en particulier et les ressources linguistiques des langues en général jouent un rôle très important dans les applications de la technologie des langues. D'une part, les ressources linguistiques alimentent les différents processus des systèmes de TAL, et d'autre part ces ressources sont de plus en plus utilisées pour accompagner le travail de modélisation linguistique par des méthodes statistiques, qui tiennent une position de plus en plus importante dans les applications de TAL [128, 151].

Les conférences LREC [2] sur les ressources linguistiques et leur évaluation ont montré l'intérêt réservé aux ressources linguistiques par les équipes de recherche dans tous les domaines : annotation morpho-syntaxique, grammaires, corpus arborés, sémantique, web sémantique et ontologie, pragmatique, dialogue et multimodal... Dans le monde de la recherche, les questions d'acquisition et d'utilisation de ressources, ainsi que d'évaluation et de normalisation des ressources, des outils et des systèmes ont ainsi acquis une importance centrale. [128]

Les ressources linguistiques à grande échelle connaissent une diffusion croissante, notamment grâce à des structures comme le LDC [3] aux Etats-Unis et l'ELRA [4] en Europe, et l'AFNLP [5] en Asie. En particuler en France, le CNRTL [6] a été créé qui a pour but de fédérer au sein d'un portail unique, un ensemble de ressources linguistiques informatisées et d'outils de traitement de la langue.

Dans la section suivante, nous présentons un aperçu des grammaires existantes au niveau international, celles qui constituent les ressources les plus importantes des composants syntaxiques des systèmes de TAL. Nous nous concentrons, d'une part sur les grammaires consacrées aux langues indo-européennes, qui sont naturellement plus étudiées, et d'autre

2. International Conference on Language Resources and Evaluation
3. Linguistic Data Consortium
4. European Language Ressource Association
5. Asian Foundation of Natural Language Processing
6. Centre National de Ressources Textuelles et Lexicales

part, sur les langues asiatiques dites « isolantes » [7], famille à laquelle appartient le vietnamien.

Les grammaires à large couverture

Les grammaires électroniques à large couverture permettent de modéliser sinon toute la langue, du moins une part importante de celle-ci. Plusieurs projets ont été entrepris et menés à bien pour la création de grammaires, principalement pour les langues indo-européennes, qui sont les plus étudiées. La plupart des grammaires ont été construites en se basant sur des formalismes grammaticaux de haut niveau présentés à la section précédente.

Pour l'anglais, on peut citer le projet Alvey, basé sur GPSG, en Grande-Bretagne [66], le projet XTAG, basé sur les TAG, à l'Université de Pennsylvanie [56], le projet Lingo, basé sur HPSG, au centre CSLI de Stanford [41] ou le projet ParGram, basé sur LFG, chez Xerox [19]. Pour le français, on peut citer la grammaire FTAG fondée sur l'architecture XTAG, développée à l'Université de Paris 7. Il existe également une grammaire TAG pour le français développée au laboratoire LORIA [42], qui est enrichie par des constructions sémantiques [63, 139] (*cf* section 7.3). Pour l'allemand, une grammaire utilisant une extension du formalisme TAG à composants multiples [8] a été créée à l'Université de Tübingen [83].

Pour le chinois, l'équipe CKIP de l'Académie Sinica (Taiwan) a développé un analyseur syntaxique basé sur le formalisme ICG (Information-based Case Grammar), un formalisme de grammaire d'unification. Pour le thaï, à notre connaissance, il n'existe aujourd'hui aucune grammaire publiquement disponible.

Pour le vietnamien, dans un travail précédent, Nguyen [128] a prévu une première liste des phénomènes syntaxiques du vietnamien et constaté la faisabilité d'une représentation de ces phénomènes employant le formalisme TAG. Dans cette étude, seulement la représentation des groupes nominaux en TAG a été concrétisée par des exemples. En effet, il n'existe non plus aucune grammaire électronique de taille importante pour le vietnamien.

1.3 Travail de thèse

1.3.1 Objectifs de la thèse

Le vietnamien, quoique pratiqué dans le monde par environ 90 millions de locuteurs, ce qui le place au 14ième rang mondial, fait partie des langues encore peu représentées au sein de la communauté scientifique internationale, tant du point de vue de sciences que de celui du TAL. Les travaux en traitement automatique du vietnamien sont encore rares et sont souvent entrepris sans la participation des linguistes, qui restent assez « traditionnels ». Pourtant, due à la présence de plus en plus massive de la langue vietnamienne sur l'Internet et dans le processus de la globalisation, le traitement automatique du vietnamien joue un rôle de plus en plus important, tant du point de vue de la recherche fondamentale que de

7. Nous définissons cette notion au chapitre suivant, lors de la présentation des principes de base du vietnamien.

8. En anglais : Multicomponent TAG.

la recherche appliquée. Une difficulté majeure pour les tâches de traitement automatique du vietnamien est due à un degré d'ambiguïté élevé dans tous les niveaux de traitements.

Cette thèse s'inscrit dans le domaine du traitement automatique du vietnamien. Le travail présenté dans la thèse porte sur la construction d'outils et de ressources linguistiques pour les tâches fondamentales de traitement du vietnamien, notamment la construction d'une grammaire à large couverture et d'un analyseur syntaxique, les deux parties princiaples d'un composant synataxique du vietnamien.

Nous présentons dans la section qui suit les contributions précises de notre projet de thèse. Nous espérons que notre travail puisse établir une base de réflexion utile pour l'élaboration de projets d'ingénierie des langues de grande ampleur concernant le traitement automatique du vietnamien.

1.3.2 Contributions

Tout d'abord, nous avons développé une chaîne modulaire de prétraitements pour le vietnamien dont le rôle est d'appliquer à des corpus bruts une cascade de traitements de surface. Il s'agit d'un segmenteur en phrases, d'un segmenteur en unités lexicales, d'un reconnaisseur de mots redoublés et d'un étiqueteur morpho-syntaxique. Préalables nécessaires à une possible analyse, ces traitements peuvent également servir à préparer d'autres tâches.

En ce qui concerne les tâches de segmentation en phrases et l'étiquetage morpho-syntaxique de textes, notre recherche porte à la fois sur l'étude des fondements théoriques des modèles d'apprentissage supervisé, plus précisément des modèles log-linéaires, et l'exploration des connaissances utiles représentant des particularités du vietnamien. Ces connaissances ont été intégrées aux modèles d'apprentissage, qui ont donné de bons résultats pour les tâches concernées.

En ce qui concerne la tâche de segmentation en unités lexicales, nous avons proposé une méthode hybride et un algorithme pour le découpage automatique de textes vietnamiens en unités lexicales. Cette approche combine à la fois la technique des automates à états finis, l'analyse des expressions régulières et la stratégie de matching maximal qui est renforcée par une méthode statistique pour résoudre des ambiguïtés de segmentation.

Puis, nous avons effectué la modélisation de la grammaire vietnamienne en utilisant le formalisme des grammaires d'arbres adjoints lexicalisées. Nous avons développé un système qui extrait automatiquement une grammaire à large couverture à partir d'un corpus arboré du vietnamien. Les arbres élémentaires de la grammaire forment les structures syntaxiques de la langue vietnamienne.

Ensuite, nous avons adapté et enrichi un analyseur syntaxique du français pour construire un analyseur syntaxique pour le vietnamien. Le résultat de ce travail est un analyseur syntaxique profond du vietnamien qui permet à la fois l'analyse en constituants et en dépendances. En particulier, nous avons conçu un schéma d'annotation en dépendances permettant d'encoder des relations grammaticales pour le vietnamien. Nous avons proposé un algorithme pour l'extraction automatique des analyses en dépendances à partir des arbres de dérivation donnés par l'analyseur. Afin d'effectuer des évaluations significatives

de l'analyseur, nous avons étudié systématiquement les différentes procédures d'évaluation d'analyse syntaxique. Ces travaux nous ont permis d'expérimenter et de donner des évaluations quantitatives du système d'analyse. Notre système d'analyse en particulier et ses modules couplés en général atteignent des performances prometteuses dans les tâches de traitement automatique du vietnamien à l'heure actuelle.

Enfin, une autre contribution significative de notre travail est un ensemble de logiciels que nous avons implémentés et mis à disposition pour le téléchargement libre, y compris les codes sources. Ces logiciels ont vu un nombre élevé de téléchargements, d'environ 9.300 fois pour l'instant, ce qui justifie une contribution fructueuse pour la communauté du traitement automatique du vietnamien.

1.3.3 Publications

Cette thèse a donné, pour l'instant, les publications suivantes :

Journaux nationaux

– "Automated extraction of tree adjoining grammars from a treebank for Vietnamese", Phuong Le-Hong, Thi Minh Huyen Nguyen, Phuong Thai Nguyen and Thi Ha Phan, *Journal of Computer Science and Cybernetics*, Vietnam Academy of Science and Technology, ISSN 1813-9663, 2010 (accepted).

Conférences internationales avec comité de lecture

– "Automated extraction of tree adjoining grammars from a treebank for Vietnamese", Phuong Le-Hong, Thi Minh Huyen Nguyen, Azim Roussanaly and Phuong Thai Nguyen, *Proceedings of The Tenth International Conference of Tree Adjoining Grammars and Related Formalisms – TAG+10*, Yale University, New Haven, CT, USA, 2010.
– "An emprical study of maximum entropy approach for part-of-speech tagging of Vietnamese texts", Phuong Le-Hong, Azim Roussanaly, Thi Minh Huyen Nguyen and Mathias Rossignol, *TALN 2010*, Montréal, Canada, 2010.
– "Finite-state description of Vietnamese reduplication", Phuong Le-Hong, Azim Roussanaly and Thi Minh Huyen Nguyen, *The 7th Workshop on Asian Language Resources, ACL-IJCNLP 2009*, Suntec City, Singapore, 2009.
– "Building a large syntactically-annotated corpus of Vietnamese", Phuong Thai Nguyen, Xuan Luong Vu, Thi Minh Huyen Nguyen and Phuong Le-Hong, *The Third Linguistic Annotation Workshop, ACL-IJCNLP 2009*, Suntec City, Singapore, 2009.
– "A metagrammar for Vietnamese LTAG", Phuong Le-Hong, Azim Roussanaly and Thi Minh Huyen Nguyen, *The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms*, Tubingen, Germany, 2008.
– "A hybrid approach to word segmentation of Vietnamese texts", Phuong Le-Hong, Thi Minh Huyen Nguyen, Azim Roussanaly and Tuong Vinh Ho, *Springer LNCS 5196 - Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, Tarragona, Spain, 2008.
– "A maximum entropy approach to sentence boundary detection of Vietnamese texts", Phuong Le-Hong and Tuong Vinh Ho, *IEEE International Conference on Research, Innovation and Vision for the Future – RIVF 2008*, Vietnam, 2008.

– "Word segmentation of Vietnamese texts : a comparison of approaches", Quang Thang Dinh, Phuong Le-Hong, Thi Minh Huyen Nguyen, Cam Tu Nguyen, Mathias Rossignol, Xuan Luong Vu, *Proceedings of the 6th Language Resources and Evaluation Conference LREC 2008*, Marrakech, Morocco, 2008.
– "A lexicalized tree adjoining grammar for Vietnamese", Phuong Le-Hong, Thi Minh Huyen Nguyen, Laurent Romary and Azim Roussanaly, *Proceedings of the 5th International Conference on Language Resources and Evaluation*, Italia, 2006.

Conférences nationales avec comité de lecture

– "Building a syntactic annotation framework for Vietnamese", Thi Ha Phan, Thi Minh Huyen Nguyen, Phuong Le-Hong and Van Tang Luu, *Proceedings of the Third National Symposium on Research, Development and Application of Information and Communication Technology*, Vietnam, 2008.

1.3.4 Présentation du plan

Afin de clarifier nos contributions et les travaux liés, notre manuscrit se divise en huit chapitres principaux qui sont regroupés en trois parties.

Le chapitre 2 est dédié à une introduction de la langue vietnamienne. Le but de ce chapitre est de présenter une connaissance de base de cette langue suffisante pour que les lecteurs puissent comprendre les difficultés particulières liées à l'exploitation informatique de cette langue.

La deuxième partie du manuscrit se compose de quatre chapitres présentant les prétraitements de textes vietnamiens.

Tout d'abord, nous présentons dans le chapitre 3 l'apprentissage par les modèles log-linéaires. Récemment, les modèles log-linéaires ont été appliqués avec succès dans plusieurs problèmes d'apprentissage automatique, surtout dans le TAL. Ce chapitre sert de base pour les travaux que nous présentons dans les chapitres 4 et 6.

Ensuite, nous présentons dans les trois chapitres suivants les trois tâches fondamentales qui constituent les prétraitements de textes en vietnamien. Le chapitre 4 abord le découpage de textes en phrases. Le chapitre 5 discute la segmentation de textes en unités lexicales. L'étiquetage morpho-syntaxique de textes est présenté dans le chapitre 6.

La troisième partie du manuscrit se compose de trois chapitres présentant l'élaboration d'un composant syntaxique du vietnamien, le travail principal de notre projet de thèse.

Le chapitre 7 présente le formalisme syntaxique LTAG, ses propriétés linguistiques et informatiques intéressantes pour développer des composants syntaxiques dans des systèmes de traitement automatique de nombreuses langues.

Le chapitre 8 présente les algorithmes et le système d'extraction automatique de grammaires LTAG que nous avons conçu afin d'élaborer une grammaire à large courverture pour le vietnamien.

Le chapitre 9 présente le développement d'un système d'analyse syntaxique à base de grammaires LTAG pour le vietnamien, les évaluations quantitatives ainsi que des discus-

sions autour des résultats obtenus.

Enfin, le mémoire se conclut par une synthèse des travaux que nous avons effectués dans le cadre de cette thèse et des perspectives qui feront l'objet de nos recherches futures. Les trois annexes incluses à la fin du manuscrit présentent respectivement les informations spécifiques du corpus arboré du vietnamien, l'ensemble des logiciels disponibles et la reconnaissance de mots redoublés que nous avons développés.

2

Notions élementaires du vietnamien

Sommaire

Nous présentons dans ce chapitre une connaissance de base de la langue vietnamienne suffisante pour que les lecteurs puissent comprendre les difficultés particulières liées à l'exploitation informatique de cette langue isolante. Tout d'abord, nous présentons l'origine et la typologie du vietnamien. Puis, nous discutons les caractéristiques principales du vietnamien du point de vue de l'écriture, de la phonétique et du lexique. Enfin, nous présentons des attributs grammaticaux importants du vietnamien.

2.1 Origine et typologie

2.1.1 Origine de la langue vietnamienne

Le vietnamien appartient au groupe Viet-Muong, branche Mon-Khmer de la famille Austro-Asiatique [68]. Le vietnamien est une langue isolante et tonique. Le vietnamien ressemblait originairement aux langues non toniques de la branche Mon-Khmer. Son caractère tonique fut ultérieurment ajouté grâce aux échanges culturels du voisinage avec le thaï et le chinois [69].

Le Vietnam a été dominé pendant une dizaine de siècles par la Chine, à partir du deuxième siècle avant J.C. Sous la domination chinoise, le chinois devint la langue administrative. Durant cette période, le vietnamien a été enrichi par un nombre important de mots chinois prononcés « à la vietnamienne » et appelés des mots sino-vietnamiens. Au douzième siècle, apparut l'écriture « nôm » (une langue démotique), une écriture de type idéographique, qui permet une transcription purement vietnamienne. Au dix-septième siècle, le missionnaire jésuite français Alexandre de Rhodes mit au point une romanisation de l'écriture vietnamienne, actuellement dite « quốc ngữ », toujours en usage, après être devenue officielle au dix-neuvième siècle en Indochine française. Sous la colonisation française pendant cent ans, le vietnamien a également évolué par emprunt de mots et de constructions grammaticales françaises.

2.1.2 Type de langue et classification du vietnamien

Présentation des types de langues

Dans la typologie linguistique, on distingue deux types principaux de langues : *les langues synthétiques* et *les langues isolantes*. En fait, il est plus précis de concevoir les langues en terme d'un continuum, des langues strictement isolantes (un morphème pour mot) aux langues polysynthétiques (dans lesquelles un seul mot peut contenir autant d'informations qu'une phrase anglaise entière). De ce point de vue, on distingue souvent quatre types de langues : isolantes (analytiques), flexionnelles, agglutinantes, et polysynthétiques.

1. *Les langues isolantes* sont des langues qui expriment les divers rapports grammaticaux par des mots et des signes isolés. Un morphème correspond à un mot. Les mots sont typiquement invariables et en général, il est difficile de distinguer le radical des éléments grammaticaux. Les fonctions syntaxiques sont identifiées par l'ordre des mots dans la phrase. Le vietnamien et le chinois appartiennent à ce type de langues.

2. *Les langues flexionnelles* sont des langues dans lesquelles les lemmes changent de forme selon leur rapport grammatical aux autres lemmes. Chaque forme d'un même paradigme peut transmettre un ou plusieurs types de traits grammaticaux (genre, nombre, fonction syntaxique, temps, mode...) pouvant s'opposer (singulier contre pluriel, masculin contre féminin...). La flexion du verbe est souvent nommée la conjugaison. Un exemple représentant ce type de langues est le latin.

3. *Les langues agglutinantes* sont des langues qui présentent la caractéristique structurelle de l'agglutination. Elles utilisent l'accumulation d'affixes autour de radicaux pour manifester les rapports grammaticaux. Les mots d'une langue agglutinante sont analysables en une suite de morphèmes nettement distincts. Un exemple représentatif est le turc.

4. *Les langues polysynthétiques* sont des langues dans lesquelles des suffixes et des morphèmes grammaticaux se synthétisent sur un radical donné aboutissant à la formation de longs mots et phrases. Un mot peut exprimer l'idée qui correspond à une phrase entière dans une langue non-synthétique. Un exemple est le mohawk.

Ton			
Initial (consonne)	Rime		
	prétonal	son noyau (voyelle)	final (consonne/semi-voyelle)

Table 2.1 – Composition phonétique d'une syllabe vietnamienne

Le vietnamien

Le vietnamien est une langue typiquement isolante. Ses propriétés isolantes principales se caractérisent par les phénomènes suivants :
- Les mots sont morphologiquement invariables. Il n'y a ni conjugaison des verbes, ni accord des noms ou des adjectifs.
- Le vietnamien a une unité qui s'appelle « tiếng ». Cette unité constitue à la fois un morphème, une syllable et un mot simple. Par exemple, la phrase « `Cô ấy là sinh viên.` »[9] est une phrase qui contient quatre mots, cinq syllabes et cinq morphèmes.
- Les sens grammaticaux sont déterminés par l'ordre des mots et les mots outils, par exemple :
  - Pour dire « Je le vois », on dit « `Tôi gặp nó` » ; alors que si l'on change l'ordre des mots pour « `Nó gặp tôi` », ça veut dire que « Il me voit ».
  - Pour dire « Je l'ai vu », on ajoute un mot outil `đã` avant le prédicat pour représenter le passé : « `Tôi đã gặp nó` ».

2.2 Ecriture et phonétique

Le vietnamien possède 41 sons, dont 23 consonnes, 13 voyelles simples, 3 diphtongues et 2 semi-voyelles (et une consonne non lettre), représentés par 29 lettres dans l'alphabet comme suit (en majuscule et miniscule) :

`Aa Ăă Ââ Bb Cc Dd Đđ Ee Êê Gg Hh Ii Kk Ll Mm`
`Nn Oo Ôô Ơơ Pp Qq Rr Ss Tt Uu Ưư Vv Xx Yy`

De plus, le vietnamien est une langue tonique. Il y a cinq accents qui sont utilisés pour représenter six tons[10] : `ngang` (`a`, sans accent), `huyền` (`à`, accent grave), `hỏi` (`ả`, accent retombant), `ngã` (`ã`, accent remontant), `sắc` (`á`, accent aigu) et `nặng` (`ạ`, accent intensif).

Le vietnamien est une langue monosyllabique. Chaque syllabe a une structure phonétique rigoureuse, représentée dans la Table 2.1. Dans l'écriture, les syllabes sont séparées par des espaces.

De plus, le vietnamien importe les quatre lettres `f`, `j`, `w`, `z` pour écrire les mots étrangers empruntés par transcription phonétique, surtout les termes scientifiques.

9. Elle est étudiante.
10. Ici, le caractère `a` représente une voyelle quelconque, par exemple `a`, `e`, `o`, `i`...

2.3 Lexique

Les unités de base du vietnamien sont des syllabes séparées. Les mots peuvent être constitués d'une unique syllabe, ou construits par une composition des syllabes.

2.3.1 Unité de base : la syllabe

Comme nous l'avons constaté à la section 2.1.2, la langue vietnamienne possède une unité spéciale appelée « tiếng ». Cette unité correspond à la fois à une syllabe du point de vue phonologique mais aussi à un morphème du point de vue syntaxique, à un sémantème du point de vue de la structure du mot, et à un mot du point de vue des constituants de la phrase. Il existe trois types de « tiếng » :

1. « tiếng » ayant un sens réel comme `viết` (écrire), `nói` (parler), `sông` (rivière), `núi` (montagne)... Il peut constituer à lui seul un mot complet du point de vue syntaxique et sémantique. Ce type de mot est appelé *mot lexical*.
2. « tiếng » comme `nhưng` (mais), `mà` (que), `tuy` (bien que)... qui ne peut pas être un constituant de phrase à lui seul, mais qui est utilisé pour composer un constituant lexical de phrase, est appelé *mot outil*[11].
3. « tiếng » qui est employé uniquement pour la création de mots composés. Il peut s'agir de mots d'origine chinoise comme `sơn` (montagne), `thuỷ` (eau), `bất` (sens négatif)... ou de mots dont le sens n'est pas défini indépendamment comme `nhố` dans `nhố nhăng` (extravagant), `đẽ` dans `đẹp đẽ` (beau)...

2.3.2 Unités lexicales

Parmi les diverses définitions du mot en vietnamien, les linguistes sont parvenus à un consensus et considèrent comme un mot la plus petite unité ayant un sens spécifié et une structure stable, et utilisée pour composer des constituants de phrase [55, 188].

Une grande partie (15, 69%, *cf.* Section 2.3.4) des mots vietnamiens sont des mots simples repérés exactement comme des morphèmes et des syllabes morphologiquement invariables. Il existe également un nombre très important de mots composés qui se composent de plusieurs syllabes. La structure d'un mot est décrite en se basant sur le nombre de syllabes qu'il contient (pour distinguer les mots mono- ou multi-syllabiques), et sur la manière dont sont composés les mots complexes (pour déterminer leurs éléments et la relation entre ces éléments). Le dictionnaire vietnamien contient donc trois types d'unités lexicales : des mots simples, des mots complexes, et des expressions figées.

1. Des mots simples ou mots monosyllabiques correspondant aux catégories 1 et 2 de « tiếng ». Par exemple `đi` (aller), `cười` (rire), `bàn` (table), `sách` (livre)...

2. Des mots complexes qui ont plus d'une syllabe. Ce sont des mots réduplicatifs et des mots composés sémantiquement.

– *Mots réduplicatifs* : Les mots réduplicatifs sont construits par combinaison phonique, généralement de deux syllabes. On entend par combinaison phonique un phénomène de répétition et de symétrie. La réduplication peut être totale pour composer des mots

11. Il est appelé ailleurs *mot vide* ou *mot grammatical*.

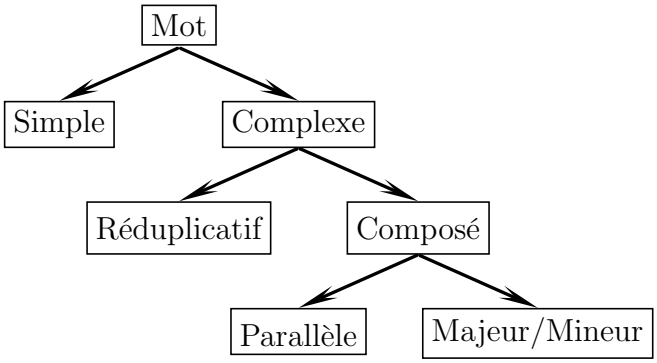


FIGURE 2.1 – Formes des mots vietnamiens

réduplicatifs complètement, par exemple `hao hao` (ressembler un peu), `đo đỏ` (un peu rouge), `chầm chậm` (un peu lent), `phăng phắc` (très silencieux). La réduplication peut être également partielle, par exemple `lí nhí` (très petit), `càu nhàu` (bougonner), `nhỏ nhắn` (mignon, menu).
Parmi les mots de cette catégorie, nous pouvons distinguer une classe de mots dont aucune des deux syllabes n'a de sens, par exemple `trục trặc` (détraqué, en panne), et une classe de mots dont une syllabe a un sens qui se trouve spécialisé par la répétition, par exemple `nhỏ` (petit) ⇒ `nhỏ nhắn` (mignon).
Les mots réduplicatifs sont généralement construits en suivant certaines règles de combinaison bien précises. Nous avons donc présenté un modèle computationnel qui se base sur des tranducteurs d'états finis pour la production et la reconnaissance automatique d'une large classe de mots réduplicatifs [102]. Nous présentons ce modèle computationnel dans l'Annexe C de manière plus détaillée.

– *Mots composés* : Ces mots sont composés à partir des syllabes en suivant une combinaison sémantique. On distingue deux classes de mots composés, en fonction de la composition sémantique des syllabes composantes.
La première classe contient des mots dits *composés parallèles* dont la composition représente une coordination sémantique des syllabes. Chaque syllabe a son propre sens et joue un rôle égal, et l'ordre des mots est donc généralement variable (à l'exception de quelques mots composés dont une ou toutes les syllabes sont d'origine chinoise, pour lesquels l'ordre des syllabes est fixe). Par exemple :

`quần` (pantalon)	`áo` (chemise)	`quần áo`/`áo quần` (vêtement)
`sông` (rivière)	`núi` (montagne)	`sông núi`/`núi sông` (pays, patrie)
`giang` (rivière)	`sơn` (montagne)	`giang sơn` (pays, patrie)

TABLE 2.2 – Exemples des mots composés parallèles du vietnamien.

La deuxième classe constitue des mots dits *composés majeur/mineur*, dont la composition manifeste une subordination sémantique de normalement deux syllabes. Une syllabe jouant le rôle de majeur porte un sens général, et l'autre syllabe jouant le rôle de mineur limite le domaine du majeur. Dans ce cas, l'ordre des syllabes est important, habituellement majeur-mineur. Par exemple :
– La syllable majeure `xe` porte le sens général qui signifie « le véhicule ». Elle est utiliée

pour constituer des mots comme `xe đạp` (vélo), `xe hơi` (voiture), `xe hoả` (train).
- De façon similaire, la syllable `xanh` (bleu/vert) joue le rôle de majeur dans les mots composés `xanh lam` (bleu), `xanh lá` (vert), `xanh lè` (très vert) ; et la syllable `làm` (faire) est la syllable majeure dans les mots composés `làm việc` (travailler), `làm duyên` (minauder), `làm giàu` (s'enrichir).

3. Des expressions figées et des locutions qui sont généralement aussi considérées comme des unités lexicales ; par exemple, `há miệng chờ sung` (attendre que les alouettes tombent toutes rôties), `được ăn cả ngã về không` (jouer à quitte ou double).

La Figure 2.1 synthétise les formes différentes de mots du vietnamien. Cette complexité des formes des mots vietnamiens constitue une difficulté majeure pour la tâche de segmentation automatique des textes en unités lexicales et, par conséquent, pour les traitements suivants, surtout l'étiquetage automatique de textes et l'analyse syntaxique de textes en vietnamien.

2.3.3 Mots empruntés

Dans l'histoire de son développement, le vietnamien a été principalement en contact avec le chinois [12] puis avec le français [13]. Cela explique l'emprunt très important de mots d'origine chinoise et de plusieurs mots français.

Mots empruntés au chinois

Une grande partie du vocabulaire vietnamien est composée de mots empruntés au chinois, avec une prononciation vietnamisée. Il y a beaucoup de syllabes sino-vietnamiennes qui peuvent ou non être utilisées indépendamment. Quand elles participent à la formation de nouveaux mots ou termes, elles devraient suivre l'ordre de combinaison des syllabes. Par exemple : `cơ khí` (mécanique) ⇒ `cơ khí hoá` (mécanisation) ; `tin học` (informatique) ⇒ `tin học hoá` (informatisation).

Mots empruntés au français et aux langues occidentales

Les mots empruntés au français sont classés en deux catégories :
- Transcription phonique directe, par exemple `ghiđông` (guidon), `pêđan` (pédale), `pênixilin` (pénicilline), `pôpelin` (popeline).
- Transcription phonétique réduite, par exemple `phanh` (frein), `len` (laine), `tôn` (tôle).

Il y a un problème à noter qui concerne les transcriptions phonétique directes. Parmi les mots appartenant à cette catégorie, la plupart des mots scientifiques ainsi que les noms propres occidentaux dont l'écriture est encore variable : leurs syllabes peuvent être attachées (`pôpelin`), séparées par des espaces (`pô pe lin`), ou séparées par des tirets (`pô-pe-lin`). La dernière forme est encouragée mais cette recommandation n'est toujours pas respectée en réalité. De plus, il n'existe pas réellement de consensus ou de standard concernant ces transcriptions [14] et la tendance moderne tend à maintenir ces mots empruntés sous leur forme originale.

12. Le Viêtnam a été dominé par la Chine pendant mille ans et il y a eu plusieurs guerres ensuite.
13. Le Viêtnam a eu une centaine d'années de colonisation française.
14. On peut trouver aussi pour « popeline » une transcription `pô-pơ-lin`

Longueur	#	%
1	6303	15, 69
2	28416	70, 72
3	2259	5, 62
4	2784	6, 93
≥ 5	419	1, 04
Somme	40181	100

TABLE 2.3 – Longueurs des mots vietnamiens mesurées en syllabes

En bref, l'écriture non standardisée des mots phonétiquement transcrits reste actuellement un problème pour l'analyse automatique des textes vietnamiens.

2.3.4 Statistique des mots

Le lexique vietnamien édité récemmement par le Centre de Lexicographie du Viêtnam (Vietlex [15]) contient 40181 mots qui sont utilisés largement dans la langue parlée et la littérature contemporaine. Ces mots sont composés de 7729 syllabes. Nous rappelons que le vietnamien est une langue inflexionnelle, chaque mot a une forme invariable et unique.

La Table 2.3 présente une statistique intéressante sur les longueurs de mots mesurées en syllabes. Tout d'abord, il y a environ 81, 55% des syllabes qui sont également des mots ; 15, 69% de mots sont des mots simples. Puis, il y a 70, 72% de mots composés qui sont composés de deux syllabes. Enfin, il y a 13, 59% de mots qui sont composés d'au moins trois syllabes.

2.4 Grammaire

2.4.1 Catégories syntaxiques

Comme nous l'avons présenté dans la section 2.1.2, le vietnamien est une langue isolante dans laquelle chaque mot a une forme unique qui ne peut pas être modifiée par dérivation ou flexion. Les relations grammaticales ne se manifestent pas par la flexion mais par l'ordre des mots et des mots outils. La classification en catégories syntaxiques n'est donc pas morphologiquement évidente.

Récemment, dans le cadre d'un projet national [16], nous avons développé une classification des mots pour le vietnamien [127]. Cette classification contient 12 catégories syntaxiques qui sont généralement universelles et obtient un haut consensus des linguistes. La Table 2.4 présente les catégories syntaxiques, leurs étiquettes ainsi que quelques exemples pour chaque catégorie.

Nous présentons dans ce qui suit plus finement les cinq catégories les plus importantes.

15. *http ://www.vietlex.com/*
16. Projet VLSP : Vietnamese Language and Speech Processing

No.	Catégorie	Notation	Description	Exemples
1.	Nom	N	désigne des entités	nhà (maison), máy tính (ordinateur)
2.	Pronom	P	remplace un nom	tôi (je), vậy (ça, cela), nó (il/elle)
3.	Verbe	V	désigne un procès, un état	đi (aller), ăn (manger), trả lời (répondre)
4.	Adjectif	A	désigne une propriété	trắng (blanc), to (grand)
5.	Adverbe	R	modifie un verbe ou un adjectif	đã (temps passé), sẽ (temps futur), rất (très)
6.	Déterminant	L	détermine un nom	những, các
7.	Numéral	M	quantifie un nom	một (un), mười (dix)
8.	Préposition	E		xuống (à), vào (en, dans)
9.	Conjonction subordonnée	C	désigne une relation subordonnée entre des constituants de la phrase	nếu (si), tuy nhiên (pourtant), vì (puisque, car)
10.	Conjonction coordonnée	CC	désigne une relation coordonnée entre des constituants de la phrase	và (et), hoặc (ou)
11.	Interjection	I	ajouté pour exprimer l'opinion du locuteur, indépendant de la structure de la phrase	ôi, chao, á
12.	Mot modal	T	ajouté pour exprimer l'opinion du locuteur, attaché à la structure de la phrase	thì, mà, ơi, nhé

TABLE 2.4 – Les catégories syntaxiques du vietnamien

Noms

Les noms peuvent être répartis dans les sous-catégories suivantes :

1. *Noms concrets* : Ce sont des noms désignant des choses qui peuvent exister sous forme individuelle, par exemple sách (livre), người (personne). Au pluriel, ces noms doivent être précédés par un nom individuel (voir ci-dessous), par exemple hai cuốn sách (deux livres). Ici, l'usage du nom individuel cuốn est important car en vietnamien on ne pourrait pas dire hai sách (hai est « deux »).
2. *Noms abtraits* : Ce sont des noms qui désignent un concept abstrait, par exemple suy nghĩ (pensée), thái độ (comportement). Au pluriel, ces noms n'ont pas besoin d'être précédés par un nom individuel, par exemple, on peut dire những suy nghĩ

(des pensées), hai thái độ (deux comportements).

3. *Noms individuels* : Les noms individuels jouent le rôle d'affixe des noms, ils se trouvent avant des noms concrets (et après des déterminants). Par exemple, les noms individuels cuốn, cái, chiếc se trouvent toujours avant les noms concrets dans les syntagmes nominaux suivants : hai cuốn sách (deux livres), ba cái bút (trois stylos), bốn chiếc kẹo (quatre bonbons). Il faut noter que ce type de nom ne correspond pas exactement au déterminant ou au quantifieur dans des langues occidentales, en effet il est une particularité du vietnamien. Les linguistes vietnamiens utilisent souvent le terme « classificateur » pour indiquer ce type de nom puisqu'il permet de classifier/distinguer les noms concrets et les noms abstraits.
4. *Noms d'unité de mesure* : Ce sont des noms qui désignent une unité déterminant une quantité, par exemple lít (litre), nắm (poignée). Ces noms se trouvent également avant des noms concrets (et parfois des noms abstraits) dans les syntagmes, par exemple một lít nước (un litre d'eau), một nắm muối (une poignée de sel).

Par ailleurs, comme dans toutes les langues, les noms peuvent être également sous-catégorisés en noms communs et noms propres. Comme dans les langues occidentales, les noms propres se distinguent à l'écrit par une initiale majuscule.

Pronoms

Les pronoms sont classifiés comme suit :

1. *Pronoms personnels* : Les pronoms personnels se distinguent non seulement par la personne et le nombre, mais aussi par la relation sociale ou familiale entre le locuteur et la personne désignée, ainsi que le sentiment. Par exemple tous les mots tôi, tao, ta, cháu, con, em, anh, chị, cô, bác... peuvent être utilisés pour dire « je ». En outre, plusieurs de ces mots peuvent être également utilisés pour dire « tu/vous ».
2. *Pronoms déictiques et démonstratifs* : Ce sont les pronoms utilisés pour indiquer le temps ou l'espace, par exemple đây (ici), này (ce), bây giờ (maintenant).
3. *Pronoms de quantité* : Ces pronoms sont utilisés pour indiquer la quantité, par exemple bấy nhiêu (tant, autant).
4. *Pronoms de qualité* : Ce sont les pronoms utilisés pour référencer une action ou une qualité, par exemple thế, vậy (comme ceci, comme cela).
5. *Pronoms interrogatifs* : Ce sont les pronoms pour l'interrogation, par exemple ai (qui), gì (quoi, que), đâu (où), bao giờ (quand), bao nhiêu, mấy (combien), sao, thế nào (comment).

Verbes

Les verbes sont classifiés comme suit :

1. *Verbes intransitifs* : Ce sont des verbes qui n'ont pas de complément d'objet, par exemple ngủ (dormir), phát triển (développer), hát (chanter).
2. *Verbes transitifs* : Ce sont des verbes qui font passer directement l'action d'un sujet sur un objet, par exemple viết (écrire), tặng (offrir), cải tiến (améliorer)
3. *Verbes d'orientation* : Ce sont des verbes de mouvement qui incluent une certaine direction. Ces verbes peuvent suivre d'autres verbes pour indiquer la « direction »

de ces derniers. Par exemple `vào` (entrer) ; `nó vào` (il entre), `nó chạy vào` (il entre en courant), ici le verbe `chạy` (courir) précède le verbe d'orientation `vào`.

4. *Verbe d'impression* : Ce sont des verbes qui expriment un état ou un processus psychologique, par exemple `hiểu` (comprendre), `yêu` (aimer). Ces verbes ont aussi le caractère transitif comme les verbes transitifs ci-dessus ; la différence est qu'ils peuvent s'associer à un adverbe de degré, par exemple `rất yêu` (aimer beaucoup).
5. *Verbes d'état* : Ce sont des verbes qui indiquent l'existence des objets, et qui peuvent être utilisés sous forme impersonnelle. Par exemple `còn` (rester), `có` (avoir) ; `tôi còn gạo` (j'ai encore du riz), `còn gạo` (il reste du riz).
6. *Verbes volitifs* : Ce sont des verbes qui désignent une volonté, comme `muốn` (vouloir), `dám` (oser). Ces verbes sont suivis par d'autres verbes ou des phrases complètes.
7. *Verbes de réception (passifs)*[17] : Ce sont des verbes qui désignent un état de réception. Contrairement aux verbes volitifs, le sujet n'exerce pas d'action. Ces verbes doivent être suivis par un verbe ou un nom. Par exemple : `bị`, `phải`, `được` (état passif, dans un sens négatif ou positif) ; `bị phạt` (être puni), `được khen` (être félicité).
8. *Verbes comparatifs* : Ce sont des verbes qui expriment une comparaison entre le sujet et un objet, par exemple `bằng` (être égal, égaler), `hơn` (être supérieur, surpasser).
9. *Verbes de transformation* : Ce sont des verbes qui correspondent à une transformation comme `nên`, `trở nên` (devenir). Ces verbes sont suivis par un complément indiquant le résultat de la transformation. Le complément peut être un adjectif.
10. *Verbe « là »* : C'est la copule qui peut se traduire en français par « être ».

Adjectifs

Les adjectifs sont classifiés comme suit :

1. *Adjectifs qualitatifs* : Ce sont des adjectifs qui désignent une qualité, et qui peuvent être précédés par un adverbe de degré comme `rất` (comme les verbes d'impression). Ces adjectifs peuvent être suivis par un nom ou un groupe verbal qui limite le domaine de la qualité exprimée, par exemple `giỏi` (bon), `giỏi toán` (bon en mathématiques).
2. *Adjectifs quantitatifs* : Ce sont des adjectifs qui désignent une propriété quantifiable, par exemple `cao` (haut). Ces adjectifs peuvent être utilisés avec soit un adverbe de degré, soit un complément de quantité ou un repère qui détermine la quantité, par exemple `cao 2 mét` (de 2 mètres de hauteur).

Adverbes

Les adverbes sont catégorisés comme suit :

1. *Adverbe de temps* : Ce sont des adverbes qui expriment les sens grammaticaux de temps, par exemple `đã` (déjà, temps passé), `sẽ` (temps futur). Ces mots outils jouent le rôle temporel que joue dans les langues flexionnelles comme le français la conjugaison des verbes aux divers temps grammaticaux.

17. Noter que la passivation n'est pas habituelle en vietnamien.

2. *Adverbe de degré* : Ces mots se trouvent éventuellement avant ou après les adjectifs ou les verbes d'impression pour exprimer un sens grammatical de degré, par exemple rất/lắm (très, trop, beaucoup) rất đẹp (très belle), xấu lắm (très mauvais), cực (extrêmement).
3. *Adverbe de rapport* : Ces adverbes expriment un rapport de constance d'une action ou d'un état dans un contexte temporaire, par exemple cũng (aussi, également) vẫn (encore).
4. *Adverbe de négation, d'affirmation* : Ces adverbes sont utilisés pour indiquer la négation ou l'affirmation d'un prédicat, par exemple không (négation), có (affirmation intensive).
5. *Adverbe impératif* : Ce sont des adverbes utilisés pour exprimer l'impératif, par exemple hãy (suggestion positive), đừng (suggestion négative), phải (devoir).

2.4.2 Syntaxe

Dans cette section, nous présentons une brève introduction aux structures syntagmatiques et syntaxiques du vietnamien.

Procédés syntaxiques

Comme nous l'avons vu à la section 2.1.2, les sens grammaticaux se manifestent par l'ordre des mots, les mots outils, le redoublement des mots, ainsi que l'intonation du locuteur pour l'oral.

1. *L'ordre des mots* permet de distinguer les différents rapports entre les constituants des phrases. En vietnamien, les constituants se mettent toujours dans l'ordre *tête–complément*. Par exemple :
   - Nom–modifieur : Le nom máy (machine) et ses deux modifieurs tính (calculer), mạnh (puissant) dans les mots et groupes nominaux suivants :
     - máy tính (ordinateur)
     - máy mạnh (puissante machine)
     - máy tính mạnh (puissant ordinateur)
   - Verbe–complément d'objet : Le verbe đọc (lire) et son complément sách (livre) dans le groupe verbal đọc sách (lire un livre).
   - Verbe–adverbe de manière : Le verbe ăn (manger) et son adverbe de manière chậm (lentement) dans le groupe verbal ăn chậm (manger lentement).
2. *Les mots outils* sont utilisés pour exprimer le pluriel dans les groupes nominaux, le temps dans le groupes verbaux, la conjonction dans les structures de coordination ou de subordination. Voici quelques exemples :
   - Le pluriel : những sinh viên (les étudiants), các sinh viên (des étudiants) ; ici les mots outils những, các sont utilisés avant le nom concret sinh viên (étudiant) pour exprimer le pluriel.
   - Le temps : Nam đã đến (Nam est arrivé) ; ici, le mot outil đã est utilisé avant le prédicat đến (arriver) pour exprimer le temps passé.
   - La conjonction de coordination et de subordination : Les deux mots gà (poulet), mẹ (mère) peuvent être combinés directement pour gà mẹ (la mère poule) ou ils

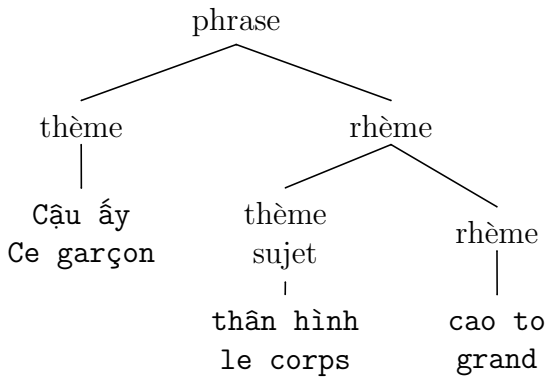


FIGURE 2.2 – Structure thème–rhème de la phrase `Ce garçon, son corps est grand.`

sont combinés avec le mot outil `và` (et) pour exprimer la coordination dans `gà và mẹ` (le poulet et la mère) ou avec le mot outil `của` (de) pour exprimer la possession dans `gà của mẹ` (le poulet de la mère).

3. *La forme redoublée des mots* permet notamment de modifier l'intensité des adjectifs. Le redoublement peut être combiné avec des mots outils pour souligner un constituant. Par exemple :
   - Redoublement des mots d'une syllabe : `xanh` (bleu ou vert) ⇒ `xanh xanh` (bleuâtre ou verdâtre, sens atténué).
   - Redoublement des mots de deux syllabes : `lúng túng` (perdre contenance), `lúng ta lúng túng` (perdre contenance, sens aggravé).

4. *L'intonation du locuteur* peut changer le sens de la phrase, par exemple transformer une affirmation en négation.

Structure thème–rhème

Du point de vue de la grammaire fonctionnelle, le vietnamien appartient aux langues avec préférence du *thème*[18]. Une langue avec préférence du thème organise sa syntaxe de façon à souligner la distinction entre le thème et le rhème (le commentaire, ce que le thème exprime) [107]. Cette propriété se manifeste en vietnamien par les phénomènes syntaxiques suivants :

- Le sujet d'une phrase ne peut pas être identifié par la morphologie (car il n'y a pas de variation morphologique en vietnamien) ni par sa position dans la phrase. De plus, il existe souvent des phrases ayant des « sujets doubles », en réalité un thème et un sujet. Par exemple, la Figure 2.2 illustre une phrase de structure thème–rhème en vietnamien.
- La passivation n'est pas une construction naturelle, car c'est le thème et non pas le sujet qui joue le rôle le plus important dans la construction de la phrase.
- Il n'existe pas de sujet impersonnel en vietnamien, car le sujet n'est pas obligatoirement présent. Par exemple, pour dire « `Il fait très chaud ici` », on dit tout simplement « `Ở đây rất nóng` » (Ici très chaud, `Ở đây` = Ici, `rất` = très, `nóng` = chaud).

18. En anglais : *topic prominent languages*

2.5 Bilan

Nous avons présenté dans ce chapitre les bases de la langue vietnamienne : origine, type de langue, composition des mots, catégories syntaxiques et structure syntaxique.

Nous discutons dans les chapitres qui suivent les travaux que nous avons effectués afin de construire des ressources et des outils de traitement du texte en vietnamien.

Deuxième partie

Prétraitements de textes vietnamiens

3

Apprentissage par les modèles log-linéaires

Sommaire

Dans ce chapitre, nous présentons en détail l'apprentissage par les modèles log-linéaires. Récemment, les modèles log-linéaires en général et ses cas particuliers comme les modèles d'entropie maximale ou les champs aléatoires conditionnels en particulier sont appliqués avec succès dans plusieurs problèmes d'apprentissage automatique, surtout dans le traitement automatique des langues naturelles.

Ce chapitre est organisé comme suit : dans un premier temps, nous présentons un survol de l'apprentissage, il s'agit de la formulation générale et de l'objectif de l'apprentissage. Dans un second temps, nous abordons trois tâches communes de l'apprentissage. Enfin, nous présentons comment ces tâches sont réalisées dans le cadre de l'apprentissage par les modèles log-linéaires.

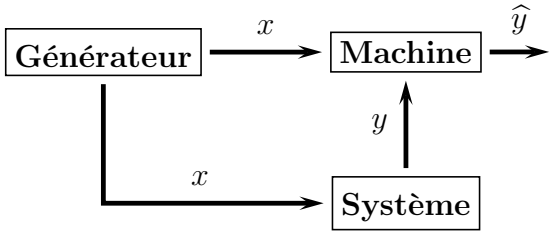


FIGURE 3.1 – La machine à apprentissage

3.1 Survol de l'apprentissage automatique

3.1.1 Formulation de l'apprentissage

L'apprentissage est le processus qui consiste à estimer une dépendance inconnue ou une structure inconnue (entrée, sortie) d'un système en utilisant un nombre limité d'observations.

La Figure 3.1 montre le scénario général de l'apprentissage [30]. Il y a trois composants : un **Generateur** qui engendre des vecteurs d'entrée aléatoires, un **Système** qui produit des sorties pour les vecteurs d'entrée, et une **Machine** à apprentissage qui estime un mapping inconnu (entrée, sortie).

Cette formulation est très générale et elle décrit plusieurs problèmes d'apprentissage qui se trouvent dans l'ingénierie et les statistiques, par exemple l'interpolation, la régression, la classification et l'estimation de densité. Les rôles des trois composants de cette formulation sont détaillés comme suit :

Générateur : Le générateur produit un vecteur aléatoire $x \in \mathbb{R}^d$ tiré indépendamment des autres à partir d'une probabilité fixe $p(x)$ qui est inconnue.

Système : Le système produit une valeur de sortie y pour chaque vecteur d'entrée x d'après une probabilité fixe $p(y|x)$ qui est aussi inconnue. Cette description inclut également le cas spécifique d'un système déterministe où $y = t(x)$, ou bien la formulation de régression $y = t(x) + \xi$, où ξ est un bruit aléatoire avec une moyenne de zéro.

Machine à Apprentissage : Dans le cas le plus général, la machine est capable d'implémenter un ensemble de fonctions $f(x, w), w \in \Omega$, où Ω est un ensemble de paramètres abstraits utilisés pour indexer l'ensemble des fonctions. Dans cette formulation, l'ensemble des fonctions implémentées par la machine pourrait être un ensemble de fonctions quelconques qui sont choisies à priori.

Par exemple, dans la régression paramétrique, l'ensemble des fonctions est spécifié comme un polynôme de degré fixe et la donnée d'apprentissage a une seule variable $x \in \mathbb{R}^1$. L'ensemble des fonctions implémentées par la machine est

$$f(x, w) = \sum_{i=0}^{M-1} w_i x^i,$$

où l'ensemble de paramètres Ω a la forme des vecteurs $w = (w_0, \ldots, w_{M-1})$, M est fixe.

3.1.2 Objectif de l'apprentissage

L'objectif de la **Machine** est de sélectionner une fonction qui donne la meilleure approximation de la réponse du **Système**. La **Machine** utilise un échantillon de données d'apprentissage produites par le **Générateur** et le **Système**. Les données sont indépendantes et distribuées identiquement d'après une fonction de probabilité jointe

$$p(x, y) = p(x)p(y|x).$$

Les données d'apprentissage de cette distribution sont notées

$$(x_i, y_i), \quad i = 1, \ldots, N.$$

La qualité d'une approximation produite par la **Machine** est mesurée par une fonction de perte $L(y, \widehat{y}) = L(y, f(x, w))$ qui mesure la différence entre la sortie y du **Système** et celle de la **Machine** pour une entrée x donnée. La perte est une valeur non-négative et son espérance est appelée *la fonction risque* :

$$R(w) = \int L(y, f(x, w))p(x, y)\, dxdy. \tag{3.1}$$

L'apprentissage est le processus d'estimer une fonction $f(x, w^*)$ qui minimise la fonction risque ($p(x, y)$ est inconnue). Nous notons $f(x, w^*)$ l'estimation optimale de $f(x, w)$ sur les données d'apprentissage. Si la probabilité $p(x, y)$ est connue, un problème d'apprentissage, par exemple la classification, pourrait être résolu par la minimisation de (3.1).

La tâche d'estimer la probabilité $p(x, y)$ en n'utilisant que des données d'apprentisage est très difficile. Afin d'obtenir une solution utile (et unique), en dehors de données, il est nécessaire d'ajouter une connaissance a priori. En pratique, cette connaissance est souvent reliée à l'ensemble des fonctions f de la **Machine**.

3.2 Tâches d'apprentissage

Le problème d'apprentissage générique présenté ci-dessus peut être divisé en trois classes de problèmes : l'estimation de la probabilité, la classification, et la régression.

L'approche classique de l'apprentissage divise le problème de l'apprentissage en deux étapes : *la spécification* et *l'estimation*. La spécification détermine la forme paramétrique des distributions inconnues en considération, tandis que l'estimation est l'évaluation des paramètres de ces distributions.

Les deux principes inductifs qui sont les plus utilisés dans l'apprentissage sont *le maximum de vraisemblance*[19] et *la minimisation du risque empirique*[20]. Le principe du maximum

19. En anglais : *maximum likelihood*
20. En anglais : *empirical risk minimisation*

de vraisemblance est de donner une forme spécifique au principe de la minimisation du risque empririque dans lequel des fonctions particulières de perte sont utilisées. Ces deux principes seront présentés par les tâches communes d'apprentissage dans les paragraphes qui suivent.

3.2.1 Estimation de probabilité

Dans cette tâche nous nous intéressons à estimer la probabilité de x représentée par la sortie de la **Machine** dans un ensemble des probabilités $\widehat{y} = f(x, w), w \in \Omega$; la sortie y du **Système** n'est pas utilisée.

Maximum de vraisemblance

Un critère naturel est *le maximum de vraisemblance* ou, de façon équivalente, la minimisation de la négation du log de la vraisemblance.

Si l'on utilise la fonction de perte

$$L(f(x, w)) = -\log f(x, w),$$

la fonction risque (3.1) devient

$$R(w) = \int -\log f(x, w)p(x)\, dx.$$

Cette fonction est souvent utilisée pour l'estimation de probabilité. La minimisation de cette fonction en utilisant les données d'apprentissage $X = (x_1, \ldots, x_N)$ nous donne $f(x, w^*)$.

L'approche classique pour l'estimation de probabilité restreint la classe des fonctions de probabilité à un ensemble paramétrique $f(x, w) \equiv p(x, w), w \in \Omega$, où w est un vecteur de dimension n. On suppose que la fonction optimale $p(x, w^*)$ appartient à cet ensemble. Etant donné un ensemble de données d'apprentissage X, la probabilité de cet ensemble est une fonction de w :

$$P(X|w) = \prod_{i=1}^{N} p(x_i; w), \tag{3.2}$$

et elle est appelée la *fonction de vraisemblance*[21]. Le principe du maximum de vraisemblance affirme qu'on devrait choisir le paramètre w qui maximise la fonction de vraisemblance. Ceci correspond à choisir un w^* et donc une distribution $p(x, w^*)$ qui est la plus apte à engendrer l'ensemble des données. Pour faciliter le calcul, on utilise souvent la log-vraisemblance. Ceci est équivalent à la minimisation de la fonction :

$$\widehat{R}_1(w) = -\sum_{i=1}^{N} \log p(x_i; w). \tag{3.3}$$

21. En anglais : *likelihood function*

Minimisation du risque empirique

Au contraire, si l'on utilise le principe de la minimisation du risque empirique, on estime empiriquement la fonction risque en utilisant les données d'apprentissage. Le risque empirique est le risque *moyen* des données d'apprentissage. L'espérance du risque est

$$R_2(w) = \int L(p(x;w))p(x)\ dx.$$

Cette espérance est approximée par le risque empirique

$$\widehat{R}_2(w) = \frac{1}{N}\sum_{i=1}^{N} L(p(x_i;w)). \tag{3.4}$$

La valeur optimale w^* est calculée par la minimisation du risque empirique (3.4) selon w.

On remarque que le principe de la minimisation du risque empirique est plus général que le principe du maximum de vraisemblance puisque la forme particulière de la fonction de perte n'est pas précisée. Si la fonction de perte est

$$L(p(x;w)) \equiv -\log p(x;w),$$

alors les deux principes sont équivalents.

3.2.2 Classification

Classification générale

Dans le problème de la classification binaire (deux classes), la sortie du système prend deux valeurs symboliques $y = \{0, 1\}$ correspondant à deux classes. La sortie de la machine a besoin donc de prendre seulement deux valeurs. L'ensemble des fonctions $f(x, w), w \in \Omega$ devient un ensemble de fonctions indicatrices. La fonction de perte souvent utilisée pour ce problème est la fonction qui mesure l'erreur de classification :

$$L(y, f(x, w)) = \begin{cases} 0, & \text{si } y = f(x), \\ 1, & \text{si } y \neq f(x). \end{cases} \tag{3.5}$$

Avec cette fonction de perte, la fonction risque (3.1) quantifie l'erreur de la classification. L'apprentissage devient alors le problème d'estimer la fonction indicatrice $f(x, w^*)$ qui minimise la probabilité de mauvaises classifications en utilisant les données d'apprentissage.

Classification classique

Le problème de la classification classique est un cas particulier du problème de la classification générale. Le modèle d'apprentissage est restreint comme suit : les probabilités conditionnelles pour chaque classe $p(x|y = 0)$ et $p(x|y = 1)$ sont estimées par l'estimation de probabilité classique et le principe du maximum de vraisemblance. Ces estimations

sont notées respectivement par $p_0(x, \alpha^*)$ et $p_1(x, \beta^*)$ pour indiquer qu'elles sont les fonctions paramétriques avec les paramètres choisis via le maximum de vraisemblance. Les probabilités de l'occurrence de chaque classe, $p(y = 0)$ et $p(y = 1)$ (les probabilités *a priori*) peuvent être estimées à partir des données d'apprentissage en comptant les fractions de l'occurrence de chaque classe. En utilisant le théorème de Bayes, on peut calculer les probabilités *a posteriori* qui déterminent la classe de chaque observation x :

$$p(y = 0|x) = \frac{p_0(x, \alpha^*)p(y = 0)}{p(x)} \tag{3.6}$$

$$p(y = 1|x) = \frac{p_1(x, \beta^*)p(y = 1)}{p(x)} \tag{3.7}$$

Le dénominateur de ces deux équations est une constante de normalisation qui est calculée par les probabilités conditionnelles et les probabilités a priori :

$$p(x) = p_0(x, \alpha^*)p(y = 0) + p_1(x, \beta^*)p(y = 1).$$

En effet, on n'a pas besoin de calculer cette constante puisque la règle de décision se base sur la comparaison de deux probabilités a posteriori, cela ramène à la comparaison des deux numérateurs. La règle de classification pour x est :

$$f(x) = \begin{cases} 0, & \text{si } p_0(x, \alpha^*)p(y = 0) > p_1(x, \beta^*)p(y = 1), \\ 1, & \text{sinon.} \end{cases} \tag{3.8}$$

3.2.3 Régression

Régression générale

La régression est le processus d'estimer une fonction réelle. La sortie du **Système** dans les problèmes de régression est une variable aléatoire qui prend des valeurs réelles et peut être interprétée par la somme d'une fonction déterministe et d'une erreur aléatoire de moyenne zéro :

$$y = t(x) + \xi,$$

où la fonction déterministe est la moyenne de la probabilité conditionnelle des sorties :

$$t(x) = \int yp(y|x)dy. \tag{3.9}$$

L'ensemble des fonctions $f(x, w), w \in \Omega$ supporté par la **Machine** n'est pas obligé d'inclure la fonction de régression (3.9). Une fonction de perte commune pour la régression est l'erreur carrée :

$$L(y, f(x, w)) = (y - f(x, w))^2. \tag{3.10}$$

L'apprentissage devient alors le problème de chercher une fonction $f(x, w^*)$ (un régresseur) qui minimise le risque

$$R(w) = \int (y - f(x, w))^2 p(x, y)dxdy \tag{3.11}$$

en utilisant les données d'apprentissage. Le risque mesure l'exactitude de la prédiction de la **Machine** sur les sorties du **Système**.

Régression classique

Dans la formulation classique du problème de régression, nous voulons estimer le vecteur des paramètres d'une fonction inconnue $f(x, w^*)$ en mesurant l'erreur de la fonction

$$y_k = f(x_k, w^*) + \xi_k,$$

où l'erreur ξ_k est indépendante de x et elle est distribuée selon une densité connue $p_\xi(\xi)$.

En se basant sur les données d'apprentissage $\mathcal{Z} = \{(x_i, y_i)\}_{i=1,\ldots,N}$, la vraisemblance de $\mathcal{Z}$ est

$$P(\mathcal{Z}\,|w) = \sum_{i=1}^{N} \log p_\xi(y - f(x, w)).$$

Si l'on suppose que l'erreur suit une loi normale avec la moyenne de zéro et une variance fixe σ, la vraisemblance devient

$$P(\mathcal{Z}\,|w) = -\frac{1}{2\sigma^2} \sum_{i=1}^{N} (y - f(x, w))^2 - N \log(\sigma\sqrt{2\pi}).$$

La maximisation de cette formule est équivalente à la maximisation de la fonction

$$\widehat{R}(w) = \frac{1}{n} \sum_{i=1}^{N} (y_i - f(x_i, w))^2,$$

qui est en effet le risque que l'on obtient en utilisant le principe de minimisation du risque empirique pour la fonction carrée de perte.

3.3 Apprentissage par les modèles log-linéaires

3.3.1 Motivations

Les modèles log-linéaires sont un des modèles probabilistes les plus utilisés dans le domaine du traitement des langues naturelles (TAL). Ils appartiennent à la famille des *modèles graphiques discriminants* qui permettent d'apprendre à annoter des données, en s'appuyant sur un ensemble d'exemples déjà annotés.

Deux cas particuliers de modèles log-linéaires introduits récemment et couramment utilisés sont *le modèle de Markov à entropie maximale* (Maximum Entropy Markov Model–MEMM) [116] et le modèle des *champs aléatoires conditionnels* (Conditional Random Fields–CRF) [94, 161].

Les modèles log-linaires ont le plus souvent été utilisés dans le TAL pour étiqueter des séquences d'unités linguistiques. Ils ont ainsi donné d'excellents résultats (souvent les meilleurs) pour la reconnaissance d'entités nommées [117], l'extraction d'informations [144], l'étiquetage en catégories syntaxiques [8, 166, 167] ou l'analyse syntaxique de surface [160]. Ils peuvent aussi être employés pour l'étiquetage de données structurées,

comme les corpus arborés [34, 36] et inspirer la conception de nouveaux analyseurs syntaxiques [26, 59, 169]. [22] Les modèles log-linéaires ont aussi été adaptés à l'annotation d'arbres et à la transformation de documents structurés [81, 80, 123].

Les modèles log-linéaires ont plusieurs avantages en comparaison avec d'autres modèles discriminatifs (comme les machines à vecteurs de support [23]), par exemple ses sorties probabilistes permettent à d'autres composants d'utiliser l'information de la confiance de décision dans une chaîne de traitements.

Nous présenterons de manière plus détaillée le succès des modèles log-linéaires dans les chapitres qui suivent quand nous aborderons l'utilisation de ces modèles dans le développement du segmenteur des textes vietnamiens en phrases (Chapitre 4) et de l'étiqueteur morphologique du vietnamien (Chapitre 6).

3.3.2 Modèles log-linéaires

Soit un exemple $(\mathbf{x}, \mathbf{y})$ où $\mathbf{x} = (x_1, \ldots, x_n)$ est un contexte (l'entrée ou l'observation) et $\mathbf{y} = (y_1, \ldots, y_n)$ est un label (la sortie ou l'annotation) du contexte $\mathbf{x}$. Les modèles log-linéaires sont un modèle conditionnel, définissant directement la probabilité conditionnelle $p(\mathbf{y} \mid \mathbf{x})$ de l'annotation sachant l'observation. Ils évitent ainsi de devoir modéliser l'observation et sa probabilité $p(\mathbf{x})$.

Nous nous intéressons aux cas des modèles pour l'annotation, et plus précisement pour l'annotation de séquences. Nous présentons ici ces modèles log-linéaires avec l'exemple des modèles de Markov à entropie maximale pour résoudre des tâches d'annotation de séquences. Pour cela, les vecteurs $\mathbf{x}$ et $\mathbf{y}$ correspondant à la séquence observée et son annotation sont représentés par des champs aléatoires $\mathbf{X} = \{X_1, X_2, \ldots, X_n\}$ et $\mathbf{Y} = \{Y_1, Y_2, \ldots, Y_n\}$ [24]. Les variables aléatoires de ces deux champs prennent respectivement leurs valeurs dans les ensembles $\mathcal{X}$, le domaine de définition de l'observation, et $\mathcal{Y}$, l'ensemble des labels. Les vecteurs $\mathbf{x}$ et $\mathbf{y}$ deviennent donc des réalisations des champs aléatoires $\mathbf{X}$ et $\mathbf{Y}$.

Les modèles de Markov à entropie maximale

Les modèles de Markov à entropie maximale [116] sont un modèle graphique dirigé et peuvent être considérés comme l'équivalent conditionnel des modèles de Markov cachés. Le graphe d'indépendances à la position i pour les MEMM est représenté sur la Figure 3.2.

Les MEMM respectent la propriété markovienne sur les variables aléatoires Y_i correspondant à l'annotation :

$$p(y_{i+1}|y_1, \ldots, y_i) = p(y_{i+1}|y_i).$$

Les MEMM ne genèrent pas l'observation comme dans les modèles de Markov cachés. A l'inverse, chaque variable aléatoire Y_{i+1} dépend non seulement de la variable aléatoire Y_i

22. D'après un texte présentant les généralités sur les modèles log-linéaires du projet CRoTAL, *http ://crotal.gforge.inria.fr/*.

23. Les machines à vecteurs de support (SVM) sont un ensemble de méthodes d'apprentissage supervisé utilisées pour la classification et la régression [174].

24. Un champ aléatoire est un ensemble de variables aléatoires. On représente ce champ par une majuscule en gras.

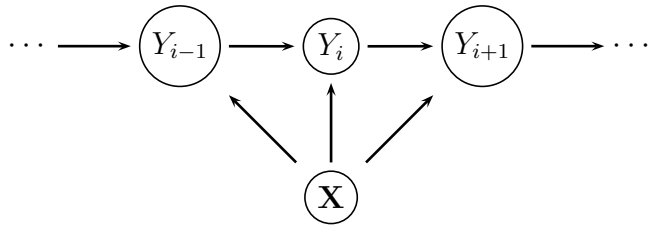


FIGURE 3.2 – Graphe d'indépendances d'un MEMM à la position i.

la précédant, mais aussi de l'ensemble du champ aléatoire $\mathbf{X}$. Ainsi, on a

$$p(y_{i+1}|y_1, \ldots, y_i, \mathbf{x}) = p(y_{i+1}|y_i, \mathbf{x}).$$

Avec ces dépendances, on obtient pour ce modèle la distribution de probabilité jointe suivante :

$$p(\mathbf{y}, \mathbf{x}) = p(\mathbf{x})p(y_1)\prod_{i=2}^{n} p(y_i|y_{i-1}, \mathbf{x}).$$

En divisant par $p(\mathbf{x})$ et en appliquant la loi de Bayes, on obtient la distribution de probabilité conditionnelle du modèle comme suit :

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{i=1}^{n} p(y_i|y_{i-1}, \mathbf{x}).$$

Ici, on suppose que $p(y_1) = p(y_1|y_0, \mathbf{x})$ pour simplifier les notations.

Chaque probabilité conditionnelle $p(y_i|y_{i-1}, \mathbf{x})$ est définie *pour chaque label* $y_i \in \mathcal{Y}$ et elle s'exprime de la façon suivante :

$$p(y_i|y_{i-1}, \mathbf{x}) = \frac{1}{Z(\mathbf{x}, y_{i-1})} \exp\left(\sum_{k=1}^{K} w_k \phi_k(y_{i-1}, y_i, \mathbf{x})\right). \tag{3.12}$$

Ces fonctions s'expriment donc sous la forme de l'exponentielle d'une combinaison linéaire de K fonctions ϕ_k et de coefficients w_k. Cette exponentielle est normalisée par la fonction de partition $Z(\mathbf{x}, y_{i-1})$ qui garantit que la fonction de transition représente bien une distribution de probabilité conditionnelle sachant $\mathbf{x}$ et y_{i-1} :

$$Z(\mathbf{x}, y_{i-1}) = \sum_{\mathbf{y} \in \mathcal{Y}^n} \exp\left(\sum_{k=1}^{K} w_k \phi_k(y_{i-1}, y_i, \mathbf{x})\right).$$

Les fonctions $\phi_k(y_{i-1}, y_i, \mathbf{x})$ sont appelées *les fonctions de traits* (feature functions) et sont des fonctions à valeurs réelles. Toutefois, elles sont souvent utilisées comme des fonctions binaires valant 1 si un test est vrai, 0 sinon. Les poids $w_1, \ldots, w_K$ associés à ces fonctions forment les paramètres du modèle MEMM.

En général, les fonctions de traits sont conçues pour extraire des attributs utiles de l'observation $\mathbf{x}$ qui aident à déterminer son label $\mathbf{y}$. Les poids w_k peuvent alors capturer

l'affinité ou l'importance des différents attributs de $\mathbf{x}$. Par exemple, si une partie x_i de la séquence $\mathbf{x}$ est un mot, des fonctions de traits peuvent utiliser les attributs de x_i comme « commence par une lettre majuscule », « commence par `B` », « est `Bernard` », « se compose de sept lettres », ou « est précédé par `Monsieur` ». Nous pouvons en général encoder des suffixes, des préfixes, des lexiques, des positions précédentes/suivantes de ponctuations... sous formes de fonctions de traits. Ces fonctions dépendent du problème considéré. Nous présenterons de manière plus détaillée la construction et des exemples de fonctions de traits pour le problème du découpage des textes en phrases et le problème de l'étiquetage syntaxique dans les chapitres qui suivent.

De plus, on voit que la définition des fonctions de traits implique que des dépendances longue distance sur l'observation peuvent être exprimées, ce qui n'est pas possible dans les modèles de Markov cachés où le label y_i choisi à la position i ne dépend que du label précédent y_{i-1} et du mot à la position i. Dans un modèle log-linéaire, les fonctions de traits ne portent pas uniquement sur x_i mais aussi sur $x_{i\pm1}$ et $x_{i\pm2}$ par exemple.

Annoter avec un MEMM

L'annotation avec un MEMM est similaire à l'annotation avec un modèle de Markov caché. Toutefois, contrairement aux modèles de Markov cachés où l'observation ne dépend que du label actuel, l'observation d'un MEMM peut dépendre du label précédent. Ainsi, nous pouvons penser que les observations sont associées aux transitions de labels, et les probabilités conditionnelles de la forme $p(y|y', \mathbf{x})$ peuvent être vues comme des fonctions de transition $p_{y'}(y, \mathbf{x})$, du label y' à son label suivant y. Pour chaque label y, nous avons donc $|\mathcal{Y}|$ fonctions de transition, chacune est représentée par une exponentielle décrite par la formule (3.12) ci-dessus.

Supposons que l'ensemble des fonctions de traits de chaque fonction de transition et leur poids sont connus, annoter une observation $\mathbf{x}$ revient à trouver l'annotation $\widehat{\mathbf{y}}$ qui maximise la probabilité conditionnelle de $\widehat{\mathbf{y}}$ sachant l'observation $\mathbf{x}$

$$\widehat{\mathbf{y}} = \arg\max_{\mathbf{y}\in\mathcal{Y}^n} p(\mathbf{y} \,|\, \mathbf{x}). \tag{3.13}$$

La connaisance des paramètres w_k et des fonctions $\phi_k, \forall k = 1, \ldots, K$ permet de calculer les fonctions de transitions $p_{y'}(y, \mathbf{x})$. Pour résoudre la tâche d'annotation, on a adapté l'algorithme de Viterbi au cas des MEMM.

On note $\delta_i(y_i)$ la probabilité de la meilleure annotation de la sous-séquence $x_1, x_2, \ldots, x_i$, où le label à la position i est fixé à $Y_i = y_i$. A l'étape initiale, on a $\delta_1(y_1) = p(y_1)$. L'étape récursive de l'algorithme de Viterbi est

$$\delta_{i+1}(y_{i+1}) = \max_{y_i\in\mathcal{Y}} \delta_i(y_i) \cdot p_{y_i}(y_{i+1}, \mathbf{x}).$$

Enfin, on a $p(\widehat{\mathbf{y}}|\,\mathbf{x}) = \max_{y_n\in\mathcal{Y}} \delta_n(y_n)$.

L'entropie maximale

Les modèles log-linéaires peuvent également être vus sous le paradigme de l'entropie maximale dans lequel nous choisissons une distribution de probabilité p qui maximise

l'entropie conditionnelle $H(p)$:

$$H(p) = -\sum_{\mathcal{X}\times\mathcal{Y}} p(\mathbf{x},\mathbf{y}) \log p(\mathbf{y} \mid \mathbf{x}) \tag{3.14}$$

sous K contraintes des estimations de $\phi_k, \forall k = 1, \ldots, K$:

$$\mathbb{E}_{p(\mathbf{x},\mathbf{y})}\phi_k(\mathbf{x},\mathbf{y}) = \mathbb{E}_{\widehat{p}(\mathbf{x},\mathbf{y})}\phi_k(\mathbf{x},\mathbf{y}), \tag{3.15}$$

où $\widehat{p}(\mathbf{x},\mathbf{y})$ est la distribution jointe observée des contextes et des labels dans l'ensemble d'apprentissage. Pour que le calcul soit possible et efficace, on approxime la distribution jointe $p(\mathbf{x},\mathbf{y})$ par le produit de la distribution empirique des contextes $\widehat{p}(\mathbf{x})$ et la distribution conditionnelle $p(\mathbf{y} \mid \mathbf{x})$ [13], on a

$$p(\mathbf{x},\mathbf{y}) = \widehat{p}(\mathbf{x}) \cdot p(\mathbf{y} \mid \mathbf{x}).$$

L'équation (3.15) devient

$$\sum_{(\mathbf{x},\mathbf{y})\in(\mathcal{X}\times\mathcal{Y})} \widehat{p}(\mathbf{x}) p(\mathbf{y} \mid \mathbf{x}) \phi_k(\mathbf{x},\mathbf{y}) = \sum_{(\mathbf{x},\mathbf{y})\in(\mathcal{X}\times\mathcal{Y})} \widehat{p}(\mathbf{x},\mathbf{y}) \phi_k(\mathbf{x},\mathbf{y}). \tag{3.16}$$

Parmi les modèles satisfaisant ces contraintes, le modèle qui a l'entropie maximale est le modèle le plus uniforme [149].

3.3.3 Apprentissage

La tâche d'apprentissage d'un modèle log-linéaire est l'estimation de la distribution de probabilité (3.12). En effet, c'est une instanciation concrète de la tâche d'estimation de probabilité présentée dans la sous-section 3.2.1. Pour estimer les paramètres du modèle, on utilise le principe de maximum de vraisemblance.

Le vecteur de paramètres du modèle $\mathbf{w} = (w_1, \ldots, w_K)$ est optimisé de manière à maximiser la log-vraisemblance conditionnelle régularisée des données d'apprentissage $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j=1,\ldots,N}$:

$$\mathcal{L}(\mathbf{w}) = \sum_{j=1}^{N} \log p(\mathbf{y}_j \mid \mathbf{x}_j; \mathbf{w}) - R(\mathbf{w}), \tag{3.17}$$

où N est la taille des données d'apprentissage, $\mathbf{y}_j$ est la sortie correcte pour l'entrée $\mathbf{x}_j$, et $R(\mathbf{w})$ est un terme de régularisation qui empêche le modèle de sur-spécifier [25] les données d'apprentissage.

Les deux méthodes de régularisation les plus utilisées sont les méthodes L_1 et L_2. La méthode de régularisation L_1 utilise la norme L_1, le terme de régularisation est donc défini par

$$R(\mathbf{w}) = c \sum_{k=1}^{K} |w_k|,$$

25. Ça veut dire que le modèle est optimisé pour l'échantillon d'apprentissage mais qu'il ne fonctionne pas aussi bien pour les autres échantillons ou une population plus grande.

où c est un paramètre qui contrôle le degré de la régularisation, il est souvent optimisé par validation croisée ou par l'utilisation d'une portion des données réservées. Cependant, la méthode de régularisation L_2 utilise la norme L_2, cela amène au terme de régularisation

$$R(\mathbf{w}) = c \sum_{k=1}^{K} w_k^2.$$

Il n'y a pas de différence considérable entre les deux méthodes en terme d'exactitude du modèle resultant [62]. En pratique, la régularisation L_1 est fréquemment utilisée puisqu'elle est plus rapide et requiert beaucoup moins d'espace mémoire que la régularisation L_2.

Il existe plusieurs méthodes et algorithmes pour estimer le vecteur de paramètres $\mathbf{w} = (w_1, \ldots, w_K)$ du modèle. Ils sont classifiés en deux approches principales : approche itérative et approche optimale.

- L'approche itérative inclut l'algorithme GIS–*Generalized Iterative Scaling* [45], l'algorithme IIS–*Improved Iterative Scaling* [143, 94] et l'algorithme SCGIS–*Sequential Conditional Generalized Iterative Scaling* [65].
- L'approche optimale utilise des algorithmes d'optimisation et d'analyse numérique. Les méthodes les plus utilisées sont l'algorithme quasi-Newton et ses variantes ; on peut citer par exemple la méthode BFGS [85, 136], la méthode L-BFGS [137] et la méthode OWL-QN [9]. Ces méthodes utilisent les algorithmes numériques de descente de gradient [62, 170].

Il existe également des méthodes de perceptron pour l'apprentissage [38]. Les travaux [40, 70] et [113] présentent en détail des comparaisons empiriques des méthodes et des algorithmes d'apprentissage pour les modèles log-linéaires.

3.4 Bilan

Nous avons présenté dans ce chapitre l'apprentissage en général et l'apprentissage par les modèles log-linéaires en particulier. La raison principale qui nous a orientés vers l'utilisation des modèles log-linéaires pour le dévelopment de systèmes de traitement pour les textes vietnamiens est la bonne performance de ces modèles appliqués à de nombreux problèmes du traitement automatique des langues. Nous présenterons dans les chapitres qui suivent l'application efficace de ces modèles dans le développement d'un système de découpage des textes vietnamiens en phrases et d'un étiqueteur syntaxique pour le vietnamien.

4

Découpage en phrases de textes vietnamiens

Sommaire

Dans ce chapitre, nous proposons une application des modèles log-linéaires présentés dans le chapitre précédent pour le découpage en phrases d'un texte en vietnamien. Au contraire de systèmes de découpage basés sur des règles prédéfinies, la procédure d'apprentissage de notre système ne requiert aucune règle prédéfinie manuellement, ni de lexique, ni d'informations spécifiques. Etant donné un corpus étiqueté avec les limites des phrases, le système apprend à classifier chaque occurrence de ponctuation comme étant une vraie ou fausse limite de phrase. La performance du système sur un corpus vietnamien est suffisante pour les niveaux suivants du traitement automatique d'un texte comme l'étiquetage et l'analyse syntaxique : $98,1\%$ de rappel. A notre connaissance, c'est le premier système pour le découpage automatique des textes en phrases pour le vietnamien. Le travail et le résultat concernant le découpage en phrases des textes vietnamiens que nous présentons ici est un enrichissement de notre résultat publié dans [98].

4.1 Introduction

De multiples phénomènes linguistiques sont liés au concept de « phrase », comme par exemple, les collocations, les expressions figées ou les coréférences. Il paraît donc naturel

que les tâches fondamentales du traitement automatique des langues comme la segmentation, l'étiquetage morphologique ou l'analyse syntaxique s'appuient sur ce concept et nécessitent un découpage préliminaire du texte en phrases.

A la lecture d'un texte, nous le découpons inconsciemment en phrases. Nous réalisons ce découpage par l'identification des limites de phrases. La détection des ponctuations n'est pas suffisante pour déterminer les limites ; nous utilisons également des informations liées au sens du texte. Le découpage n'est donc pas une tâche triviale, en fait, il est difficile à automatiser. A première vue, il semble que l'utilisation d'une liste des ponctuations qui marquent la fin des phrases comme « . », « ? », et « ! » est suffisante. Néanmoins, ces ponctuations ne sont pas utilisées seulement pour marquer la fin de phrases. Par exemple, une citation entre deux guillemets pourrait renfermer un nombre quelconque de ces ponctuations. Et le point est également utilisé pour marquer les abréviations, les nombres ordinaux, les points décimaux, les dates, les adresses électroniques, les adresses Internet *etc.* De plus, les ponctuations pourraient être utilisées pour marquer une abréviation et une limite de phrase en même temps. Ou encore, dans un but d'emphase, elles pourraient être utilisées plusieurs fois pour marquer une seule limite de phrase. Ainsi, nous pouvons considérer la détection des limites de phrase comme un problème de résolution d'ambiguïté.

Un récent article paru dans le journal *Computational Linguistics* [90] présente une étude approfondie des systèmes de détection des phrases et donne une description détaillée des méthodes et des systèmes pour l'identification des phrases pour les langues occidentales. En général, ces méthodes peuvent être classées en trois approches : les approches basées sur les règles prédéfinies, les approches d'apprentissage supervisé et les approches d'apprentissage non-supervisé. Les meilleurs résultats sont en général obtenus par les systèmes hybrides qui combinent ces approches.

Nous présentons dans ce chapitre un système de découpage en phrases de textes en vietnamien qui utilise une approche d'apprentissage statistique supervisé. C'est une étude de cas du modèle log-linéaire (ou bien le modèle de l'entropie maximale) présenté dans le chapitre précédent.

L'apprentisage du modèle se fait sur un corpus qui contient environ 20000 phrases manuellement découpées par des linguistes vietnamiens. En terme de performance, le modèle fournit un taux de précision de $98,1\%$ sur un corpus de test, et pour l'instant c'est le meilleur résultat obtenu pour cette tâche de prétraitement du vietnamien.

Les étapes de notre démarche sont les suivantes. Dans un premier temps, la section 4.2 présente la motivation de notre travail. Le modèle de l'entropie maximale est rappelé dans la section 4.3. Ensuite, la sélection des fonctions de traits pour le modèle est présentée dans la section 4.4. La méthode d'expérimentation et les résultats obtenus sont discutés dans la section 4.5. Enfin, nous présentons pour terminer la conclusion dans la section 4.6.

4.2 Motivation

Nous avons constaté que le découpage en phrases constitue un prétraitement qui a un effet important pour les niveaux suivants du traitement automatique d'un texte comme l'étiquetage et l'analyse syntaxique. L'unité traitée par un analyseur syntaxique est généralement une phrase. Lorsque l'entrée est un texte, il convient donc d'avoir recours au prétraitement. Les performances de l'analyseur sont ainsi directement liées à celles du prétraitement.

La raison qui nous a orientés vers l'utilisation des modèles log-linéaires pour le développement de notre système de découpage de textes en phrases est la bonne performance de ces modèles appliqués à de nombreux problèmes du traitement automatique des langues (TAL).

Comme nous l'avons présenté dans le chapitre précédent, les modèles log-linéraires sont également nommés *les modèles d'entropie maximale*. Ces modèles sont vus comme une approche d'apprentissage supervisé appliquée avec succès à un grand nombre de problèmes de classification linguistique dans lesquels les contextes sont utilisés pour prévoir les classes linguistiques. Ils offrent une manière claire de combiner diverses informations contextuelles afin d'estimer la probabilité d'une certaine classe linguistique apparaissant avec un certain contexte [13].

Les modèles log-linéaires ont démontré leur efficacité dans les tâches du TAL pour l'anglais, notamment la traduction automatique [31, 53], l'étiquetage de rôles sémantiques [165], l'analyse syntaxique [34, 134, 59, 169], l'étiquetage [148, 166, 167], la modélisation du langage [152], la classification des textes [133], la reconnaissance des entités nommées [15], et l'identification des phrases [150]. Récemment, les modèles log-linéraires ont été employés avec succès pour les traitements de plusieurs langues orientales telles que le traitement du chinois [108, 111, 164, 184], le traitement du japonais [171, 172]. Les modèles log-linéaires ont aussi reçu un intérêt considérable au sein de la communauté de TAL du français avec l'élaboration du projet CRoTAL [26] qui a pour objectif d'étudier et de développer de nouvelles techniques pour la manipulation de grandes masses de données textuelles [123, 95, 122]. Notre démarche s'incrit dans la poursuite de ces études en nous intéressant à l'apport de la méthode de l'entropie maximale pour le vietnamien et plus spécifiquement au problème du découpage en phrases de textes.

Nous présentons dans la section suivante le modèle probabiliste de la méthode de l'entropie maximale.

4.3 Le modèle probabiliste

Plusieurs problèmes du TAL peuvent être considérés comme des tâches de classification dans laquelles nous voulons estimer la probabilité d'une classe $\mathbf{y}$ étant donnée une observation ou une entrée $\mathbf{x}$, c'est-à-dire la probabilité $p(\mathbf{y} \mid \mathbf{x})$. Les observations dans les tâches de TAL contiennent ordinairement les mots et une observation particulière dépend de la

26. Projet « Conditional Random Fields for TAL ». Le projet se concentre plus particulièrement sur les champs aléatoires conditionnels, *http ://crotal.gforge.inria.fr/*.

nature de la tâche. Pour certaines tâches, une observation $\mathbf{x}$ peut inclure seulement un mot, cependant, pour d'autres tâches, $\mathbf{x}$ pourrait inclure quelques mots et leurs étiquettes syntaxiques.

Le modèle probabiliste que nous utilisons pour le découpage des textes en phrases est une instance des modèles log-linéaires généraux présentés dans le Chapitre 3. Nous adoptons une approche qui utilise un modèle log-linéaire puisqu'il permet l'inclusion des sources diverses d'informations sous forme de fonctions de traits.

Nous rappelons la distribution de probabilités des modèles log-linéaires présentée précédemment (*cf.* l'équation (3.12)) dans la formule suivante :

$$p(\mathbf{y} \mid \mathbf{x}; \mathbf{w}) = \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp \sum_{k=1}^{K} w_k \phi_k(\mathbf{x}, \mathbf{y}), \tag{4.1}$$

où $Z(\mathbf{x}, \mathbf{w})$ est le coefficient de normalisation

$$Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp \sum_{k=1}^{K} w_k \phi_k(\mathbf{x}, \mathbf{y}).$$

Ici, les données $(\mathbf{x}, \mathbf{y}) \in (\mathcal{X}, \mathcal{Y})$, où $\mathcal{X}$ est un ensemble fini d'observations et $\mathcal{Y}$ est un ensemble fini de labels. Les fonctions $\phi_k(\mathbf{x}, \mathbf{y})$ sont les fonctions de traits utilisées. Le vecteur des paramètres du modèle $\mathbf{w} = (w_1, w_2, \ldots, w_K) \in \mathbb{R}^K$ est à estimer.

Pour chaque ponctuation qui pourrait potentiellement marquer la fin de phrase (“.”, “ ?”, et “ !”), nous estimerons la probabilité conditionnnelle de la vraie ponctuation $\mathbf{y}$ étant donné son observation contextuelle $\mathbf{x}$. Cette probabilité caractérise la chance que la ponctuation est une vraie fin d'une phrase. Supposons que $\mathcal{Y} = \{\texttt{non}, \texttt{oui}\}$ est l'ensemble des labels possibles, et $\mathcal{X}$ est l'ensemble des observations possibles pour déterminer la fin de phrase. Les fonctions de traits sont des fonctions prenant des valeurs binaires sur les événements : $\phi_k(\mathbf{x}, \mathbf{y}) = \mathcal{X} \times \mathcal{Y} \to \{0, 1\}$ qui sont utilisées pour encoder l'information contextuelle à laquelle on s'intéresse.

La probabilité qu'une ponctuation soit une vraie fin de phrase dans une observation $\mathbf{x}$ est spécifiée par $p(\texttt{oui} \mid \mathbf{x})$. Nous utilisons une règle simple pour classifier les limites potentielles de phrases : une ponctuation potentielle est une vraie fin de phrase si et seulement si $p(\texttt{oui} \mid \mathbf{x}) > 0,5$.

Nous présentons dans la section suivante la sélection de fonctions de traits $\phi_k(\mathbf{x}, \mathbf{y})$ du modèle ci-dessus.

4.4 Les fonctions de traits

Les fonctions de traits utilisées dans le modèle devraient encoder des informations utiles pour déterminer les limites de phrases. Si une fonction de traits existe dans l'ensemble de fonctions, son coefficient correspondant irait contribuer à la probabilité conditionnelle $p(\mathbf{y} \mid \mathbf{x})$.

No.	Fonctions de traits
1.	Il existe ou non un blanc avant le caractère `EOS`
2.	Il existe ou non un blanc après le caractère `EOS`
3.	Le caractère `EOS`
4.	Le préfixe
5.	La longueur du préfixe
6.	Le premier caractère du préfixe est majuscule ou minuscule
7.	Le préfixe est dans une liste des abréviations prédéfinies ou non
8.	Le suffixe
9.	Le token précédent
10.	Le premier caractère du token précédent est majuscule ou minuscule
11.	Le token précédent est dans une liste des abréviations prédéfinies ou non
12.	Le token suivant
13.	Le candidat est en majuscule ou non

TABLE 4.1 – Les fonctions de traits utilisées pour le découpage en phrases.

Les limites potentielles de phrases sont identifiées par une lecture du texte pour trouver des séquences de caractères séparées par un blanc (les tokens) qui contiennent une des ponctuations « . », « ? », ou « ! » (sans les guillemets). Nous utilisons l'information du token contenant la ponctuation et ses informations contextuelles, c'est-à-dire les tokens à sa gauche et les tokens à sa droite.

Nous nommons `candidat` le token incluant un caractère marquant potentiellement la fin de phrase (le caractère `EOS`–end-of-sentence). La partie du `candidat` qui précède le caractère `EOS` est appelée le `préfixe`, et la partie qui suit le caractère `EOS` est appelée le `suffixe`. Nous examinons également le token précédent et le token suivant du `candidat`. Les positions des caractères `EOS` dans les phrases d'apprentissage sont également encodées. De plus, en vietnamien les caractères `EOS` « ? » et « ! » sont normalement précédés par un blanc. Donc pour améliorer la performance du modèle, nous avons ainsi besoin d'ajouter des traits encodant l'information de caractère blanc autour du `candidat`. Notre modèle utilise la liste de fonctions de traits indiquée dans la Table 4.1.

L'information contextuelle est automatiquement engendrée à partir des données d'apprentissage en utilisant les motifs de traits présentés ci-dessus. A titre d'illustration, quelques informations contextuelles utilisées pour déterminer le caractère potentiel `EOS` du « . » qui se trouve dans le token candidat *10.000USD* de la phrase suivante :

Những hacker máy tính sẽ có cơ hội chiếm giải thưởng trị giá 10.000USD và 10.000 đôla Singapore (5.882USD) trong một cuộc tranh tài quốc tế mang tên "Hackers Zone" được tổ chức vào ngày 13.5.1999 tại Singapore.

seraient `token-précédent`=*giá*, `token-suivant`=*và*, `préfixe`=*10*, `suffixe`=*000USD*.

Nous incluons dans la liste prédéfinie des abréviations les abréviations les plus utilisées dans les journaux vietnamiens, par exemple *GS* (professeur), *TT* (le premier ministre),

Rappel R	**Précision** P	F**-Mesure**
$98,10\%$	$97,20\%$	$97,64\%$

TABLE 4.2 – Performance du système de découpage des textes en phrases

UBND (comité populaire)...

4.5 Evaluation

4.5.1 Corpus

Malgré l'existence de débats en cours dans la communauté linguistique qui concernent plusieurs phénomènes linguistiques du vietnamien, presque tous les linguistes vietnamiens sont d'accord avec la définition d'une phrase vietnamienne qu'elle est l'unité linguistique minimale du discours. Une phrase est caractérisée non seulement par sa capacité à déclarer un fait, un avis ou un sentiment, mais aussi par son indépendence dans un texte [24, 131, 188].

Cette définition est précise et suffisamment générale. De plus, elle s'accorde sémantiquement avec celle des autres langues. C'est pourquoi, en dépit de la classification complexe des phrases vietnamiennes, en suivant cette définition, il n'est pas difficile pour les linguistes de déterminer et découper un texte vietnamien en phrases.

Pour faire l'apprentisage du modèle, nous utilisons un corpus qui contient environ 20.000 phrases manuellement découpées par des linguistes vietnamiens. La taille moyenne d'une phrase est de $23,54$ syllabes. Le corpus se compose principalement de textes du genre politique et social du journal *Tuổi trẻ* (La Jeunesse) ; une partie du corpus est du domaine de l'histoire du Viêtnam. Ce corpus est également utilisé comme la source de données d'un projet national dont l'objectif est de construire un corpus arboré pour le vietnamien [127].

4.5.2 Performance du système

Nous avons effectué 10 expérimentations sur le corpus de test. Dans chaque expérimentation, nous prenons 90% du corpus pour les données d'apprentissage (soit 18.000 phrases) et 10% du corpus pour l'ensemble de test (soit 2.000 phrases). La Table 4.2 montre les valeurs moyennes des précisions (P), rappels (R) et F-mesures du système ($F = \frac{2RP}{R+P}$).

Afin d'évaluer l'importance de la taille du corpus d'apprentissage sur les résultats obtenus, nous avons également expérimenté le modèle sur des corpus d'apprentissage de taille variable. La Table 4.3 montre la dépendance de la performance du système à la taille du corpus d'apprentissage mesurée en nombre de phrases. On voit que la performance est monotone avec la taille du corpus. Même pour un corpus d'apprentissage de taille modeste de 2.000 phrases, on obtient un résultat bien meilleur que la base d'environ 60% si l'on considère simplement que toutes les ponctuations potentielles marquant la fin de phrase sont des vraies limites pour les phrases.

Nous avons effectué une analyse des erreurs du système pour déterminer les origines de

Taille	R	P
2000	0, 926	0, 919
4000	0, 949	0, 924
6000	0, 958	0, 950
8000	0, 970	0, 968
10000	0, 973	0, 969
12000	0, 974	0, 969
14000	0, 976	0, 970
16000	0, 979	0, 971
18000	0, 981	0, 972

TABLE 4.3 – Performance comme une fonction de la taille du corpus d'apprentissage

faux découpages. En effet, les erreurs observées du système proviennent souvent de deux sources :

– l'inconséquence de données de tests par rapport à la donnée d'apprentissage. Par exemple, il existe la « phrase » suivante dans l'ensemble de tests « *Chúng tôi tự hỏi sao lại đến nỗi thế ? Chẳng lẽ KBT ở đấy làm vì ? Nguồn tin của chúng tôi cho biết : "Những con thú ấy lái buôn sẽ đến mua hay thợ săn phải mang đi bán.* » Il s'agit d'une fausse annotation car ce texte devrait être découpé en trois phrases « *Chúng tôi tự hỏi sao lại đến nỗi thế ?* », « *Chẳng lẽ KBT ở đấy làm vì ?* », et « *Nguồn tin của chúng tôi cho biết : "Những con thú ấy lái buôn sẽ đến mua hay thợ săn phải mang đi bán.* »
– l'ambiguïté élevée du découpage au niveau des ponctuations « ! » et « ? ». En fait, en vietnamien, ces ponctuations peuvent être employées avec un blanc qui les précède ou non. Pour mieux prendre en compte cette ambiguïté, nous avons intégré une fonction de trait spéciale qui teste l'existence ou non d'un blanc avant le caractère `EOS`, pourtant il nous semble que ce trait n'est pas suffisamment robuste pour aider à résoudre ce type d'ambiguïté.

4.5.3 Logiciel

Nous avons dévelopé un logiciel nommé `vnSentDetector` qui implémente l'approche présentée pour le découpage automatique du texte vietnamien. Notre implémentation utilise la librarie `opennlp.maxent` [27] – un logiciel libre pour l'apprentissage et l'utilisation des modèles d'entropie maximale.

Le logiciel est implémenté en Java et distribué sous la licence GNU/GPL [28]. Il est également intégré à la plateforme `vnToolkit`, une application Eclipse Rich Client [29] qui est une plateforme générale pour l'intégration des outils et des ressources pour le traitement du vietnamien.

27. *http ://maxent.sourceforge.net/*
28. *http ://www.gnu.org/copyleft/gpl.html*
29. *http ://www.eclipse.org/rcp/*

Nous avons également adapté cette approche pour développer un logiciel qui découpe automatiquement des textes français en phrases. L'approche est robuste et le résultat est aussi comparable à ceux obtenus pour le vietnamien et d'autres langues, avec le taux de précision de $98,5\%$ sur un corpus du journal Le Monde [105].

Tous les logiciels sont en ligne et disponibles pour le téléchargement libre [30]. Le logiciel `vnToolkit` et ses modules sont décrits de manière plus détaillée en Annexe B.

4.6 Bilan

Dans ce chapitre, nous avons présenté un système de découpage en phrases de textes en vietnamien qui s'appuie sur un modèle log-linéaire. Cette étude a été menée jusqu'à la réalisation d'un outil disponible sous une licence de logiciel libre et les résultats expérimentaux démontrent l'efficacité de cette approche (taux de précision de $98,10\%$) malgré un corpus d'apprentissage de taille moyenne.

Pour notre étude, nous nous sommes limités à des textes de type journalistique. Ce système présente l'avantage d'être souple et facilement adaptable en utilisant, pour l'apprentissage, d'autres types de corpus. De plus, les tendances laissent espérer une amélioration des résultats en utilisant un corpus plus large.

Enfin, nous avons adapté le modèle ci-dessus pour développer un système automatique de découpage du français en phrases. Le système donne un bon résultat sur un corpus de test : 96% pour la précision [105] ; cela confirme la robustesse et l'efficacité des modèles log-linéaires dans ce problème de classification.

30. *http ://www.loria.fr/∼lehong/projects.php*

5

Découpage en unités lexicales de textes vietnamiens

Sommaire

Nous présentons dans ce chapitre une approche hybride pour le découpage automatique de textes vietnamiens en unités lexicales. L'approche combine à la fois la technique des automates à états finis, l'analyse des expressions régulières et la stratégie de matching maximal qui est renforcée par une méthode statistique pour résoudre des ambiguïtés de segmentation. L'approche est implémentée dans le logiciel `vnTokenizer` qui, en terme de performance pour le traitement du vietnamien, fournit un taux de précision de 98, 5%, le résultat de l'état de l'art pour cette langue. Les travaux et le résultat présentés dans ce chapitre sont l'amélioration de notre résultat qui a été publié précedemment dans [103].

5.1 Introduction

La tâche de découpage (ou de segmentation) d'un texte consiste à identifier dans un texte les différents segments (unités lexicales) comme les ponctuations, les symboles, les

nombres, les dates, les noms propres et les entités nommés, les mots...

Nous avons vu que le vietnamien est une langue alphabétique qui utilise des lettres latines comme des langues occidentales. En général, les écritures alphabétiques séparent les mots par des caractères blancs (des espaces) et un découpeur simple qui découpe des phrases en s'appuyant sur des espaces et des ponctuations est déjà assez précis [159].

Pourtant il y a des différences importantes entre le vietnamien et les langues occidentales qui engendrent des problèmes spécifiques au vietnamien. Premièrement, en vietnamien, les espaces sont utilisés pour séparer non seulement des mots mais aussi des syllabes qui constituent des mots composés (*cf.* section 5.2). Une séquence simple de trois syllabes `a b c` pourrait constituer trois mots `(a) (b) (c)`, deux mots `(a b) (c)`, deux mots `(a) (b c)` ou même un mot `(a b c)`. En conséquence, cela ne permet pas l'identification des mots composés en s'appuyant sur les blancs dans un texte.[31] Deuxièmement, les noms propres vietnamiens rendent plus ambiguë la segmentation à cause d'une transcription non standardisée. Par exemple, « Université Nationale » peut être écrit en vietnamien de plusieurs formes différentes : « `Đại học Quốc gia` », « `Đại học quốc gia` », ou « `Đại Học Quốc Gia` ».

De plus, du fait de la similarité du système de ponctuation avec le français ou l'anglais, on observe les mêmes problèmes que pour ces langues à propos des ponctuations, comme par exemple le problème du point après une abréviation.

Il y a plusieurs approches pour le découpage des textes en unités lexicales qui ont été proposées. Ces approches peuvent être classifiées généralement en deux méthodes : les approches qui sont basées sur un dictionnaire et les approches qui utilisent des méthodes statistiques. Cependant les meilleurs systèmes utilisent des approches hybrides [74].

Nous présentons dans ce chapitre une approche hybride pour le découpage des textes vietnamiens en unités lexicales. La méthode combine la technique des automates à états finis, l'analyse des expressions régulières, et la concordance maximale est renforcée par une méthode statistique pour résoudre des ambiguïtés de la segmentation.

Ce chapitre est organisé comme suit : la section suivante présente la construction d'un automate minimal à états finis qui code le lexique vietnamien. La section 5.3 présente la spécification du découpage des textes en unités lexicales. La section 5.4 discute l'application de l'automate lexical et la méthode hybride pour la segmentation des textes vietnamiens. Le découpeur pour le vietnamien et ses résultats expérimentaux sont présentés dans la section 5.5. Enfin, la section 5.6 conclut le chapitre en abordant quelques discussions et perspectives.

5.2 Représentation du lexique

Les automates déterministes à états finis ont été reconnus comme la meilleure représentation d'un lexique. Ils sont non seulement compacts mais aussi donnent un temps optimal pour l'accès aux données encodées [115]. Nous représentons le lexique vietnamien par un

31. On voit également des cas similaires en français, par exemple *pomme de terre* est un mot composé qui est constitué des mots *pomme*, *de*, et *terre*.

automate déterministe minimal.

Traditionnellement, la construction d'un automate minimal se déroule en deux étapes : la construction et la minimisation. Daciuk *et al.* ont récemment proposé un algorithme incrémental qui permet de construire un automate minimal en une seule étape. L'idée principale de cet algorithme intéressant est que la construction de l'automate se fait de façon incrémentale ; on ajoute les mots, un par un, à l'automate optimal en cours et on minimise l'automate résultat au fur et à mesure [44].

Nous avons implémenté cet algorithme pour encoder le lexique du vietnamien dans un automate à états finis minimal. A titre d'illustration, la Figure 5.1 montre l'automate minimal qui encode les noms des douze mois d'une année. Cet automate est capable de reconnaître le lexique qui se compose de douze mots donnés dans le tableau suivant :

một	(janvier)	bảy	(juillet)
hai	(février)	tám	(août)
ba	(mars)	chín	(septembre)
bốn	(avril)	mười	(octobre)
năm	(mai)	mười một	(novembre)
sáu	(juin)	mười hai	(décembre)

Cet automate possède 21 états dont deux sont des états finaux et 30 transitions.

L'automate minimal qui encode tout le lexique vietnamien (*cf.* section 2.3) contient 42.672 états dont 5.112 sont des états finaux. Il possède 76.249 transitions, le nombre maximal de transitions sortant d'un état est 85 et le nombre maximal de transitions entrant dans un état est 4.615. L'automate fonctionne dans le temps optimal dans le sens que le temps pour reconnaître un mot correspond au temps d'un seul parcours dans la machine déterministe, de l'état initial à un état final ; et la longueur du parcours est la longueur du mot mesuré en caractères.

Nous présentons dans la section 5.5 l'application de l'automate lexical pour le découpage des textes vietnamiens. Nous discutons tout d'abord la spécification de la tâche de découpage.

5.3 Spécification du découpage

En ce qui concerne la segmentation des textes en unités lexicales, le comité TC37/SC4 [32] de l'Organisation Internationale de Standardisation (ISO) a publié en 2006 une série de documents qui définissent un standard international pour le découpage des textes en mots. Ce standard vise principalement les langues dans lesquelles les limites de mots de leurs textes écrits ne peuvent pas être complètement identifiées par des propriétés typographiques (comme des espaces en anglais), par exemple le chinois, le japonais, le coréen, le thai et le vietnamien [72].

32. *http ://www.tc37sc4.org*

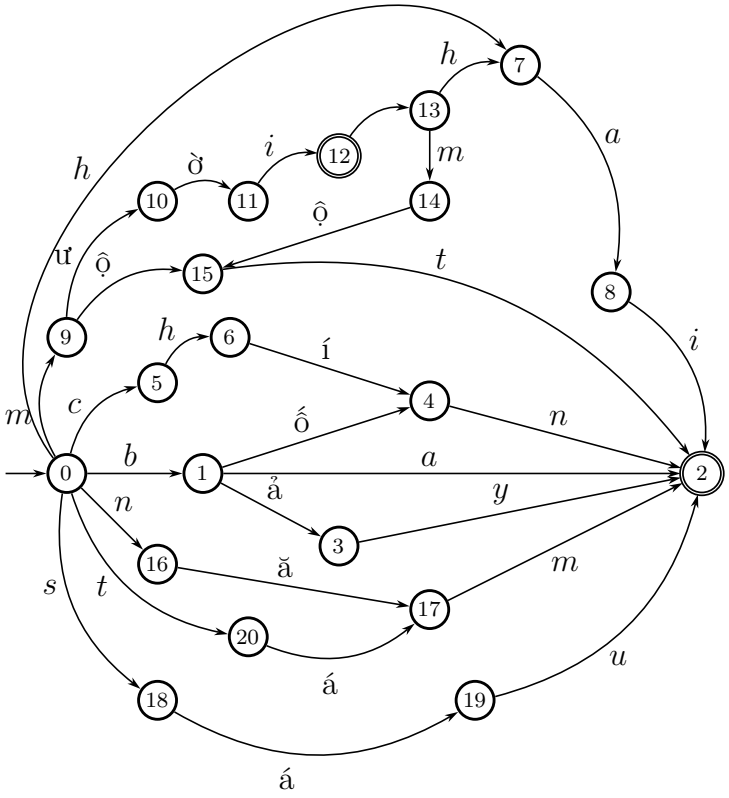


FIGURE 5.1 – Automate minimal qui encode les douze mois d'une année

5.3.1 Principes internationaux pour la validation d'une unité lexicale

Le comité TC37/SC4 de l'ISO a proposé des principes pour la formation et la validation d'une unité lexicale. En gros, ces principes sont basés à la fois sur le point de vue linguistique et le point de vue pragmatique (pratique). Nous citons ici les principes principaux sur lesquels nous nous appuyons pour développer une spécification pour le découpage du texte en vietnamien ; pour une liste complète des principes et méthodes proposés par l'ISO, veuillez référer à [72].

1. *Principe de morphème attaché* : Si un morphème est attaché à une unité linguistique, alors le résultat est un mot. Par exemple, le morphème `ăn` (manger) est un mot, le morphème `làm việc` (travailler) est un mot.
2. *Principe de l'intégrité lexicale* : Les règles syntaxiques ne peuvent pas référer à la structure interne de mots. Si une unité linguistique satisfait ce principe, elle tend à être un mot. Par exemple, l'unité linguistique `lấp lánh` (scintiller) est un mot car les règles syntaxiques ne peuvent pas référer aux syllabes qui la composent (ici `lấp` et `lánh`).
3. *Principe de l'imprévisibilité du sens* : Si le sens d'une unité linguistique ne peut pas être prédit en se basant sur ses composantes, elle est considérée comme un mot (ou, plus précisément, une unité lexicale). Par exemple, l'unité linguistique *nham nhở*

(parsemé de taches de couleurs différentes) est un mot. Ici, il n'y a pas de relation entre le sens de ce mot et ceux de ses syllabes.

4. *Principe de l'idiomatisme* : Si une unité linguistique a une propriété d'idiomatisme, elle est alors considérée comme une unité lexicale. Par exemple, l'unité `vậy mà` (pourtant, toutefois) est un mot.
5. *Principe de la colocalisation* : Si une unité linguistique a une propriété de colocalisation, c'est-à-dire ses composantes sont toujours colocalisées ensemble, elle est alors considérée comme une unité lexicale. Par exemple, l'unité `lượt thà lượt thượt` (trop long) est considérée comme un mot car la partie redoublée `lượt thà` est toujours colocalisée avec la partie originale `lượt thượt`.
6. *Principe de la productivité* : Si une unité linguistique montre une très faible productivité, c'est-à-dire elle est rarement utilisée pour produire des mots, elle tend alors à être une unité lexicale. Par exemple, les unités `nè`, `ủa` (voyons, euh) sont considérées comme des unités lexicales.
7. *Principe de la fréquence* : La fréquence est un indice de base pour le degré de lexicalisation d'une unité linguistique.

5.3.2 Principes pour la validation d'une unité lexicale en vietnamien

Nous avons developé, en collaboration avec un groupe de linguistes vietnamiens au Centre de la Lexicographie du Vietnam[33], un ensemble de règles pour le découpage des textes vietnamiens en unités lexicales. Ces règles sont conformes complètement aux sept principes recommandés par le standard de l'ISO abordé ci-dessus. Nous avons construit une liste des unités lexicales pour le vietnamien qui est composée de mots simples, de mots composés (y compris de mots réduplicatifs), d'expressions et de locutions. De plus, les noms propres, les entités nommées et les patrons réguliers comme les nombres, les dates sont également considérés comme des unités lexicales (*cf.* section 2.3.4).

5.4 Découpage des textes

5.4.1 Découpage des textes en unités lexicales

Un texte à découper est tout d'abord traité par un système de reconnaissance d'expressions régulières pour l'identification des patrons réguliers comme les noms propres, les abréviations communes, les ponctuations, les nombres, les dates, les temps, les adresses emails... Parmi les expressions régulières utilisées, il y a une expression particulière qui extrait les suites de syllabes du texte.

Le système de reconnaisance analyse le texte en utilisant une méthode gourmande dans laquelle toutes les expressions régulières sont examinées pour détecter des patrons et le patron le plus long est choisi. Par exemple, quand on analyse le patron `26-12-2008` avec deux expressions régulières de types `date` et `nombre`, celle de type `date` est choisie car elle donne le résultat `26-12-2008`, qui est un patron plus long qu'un nombre seul `26`.

33. *http ://www.vietlex.com*

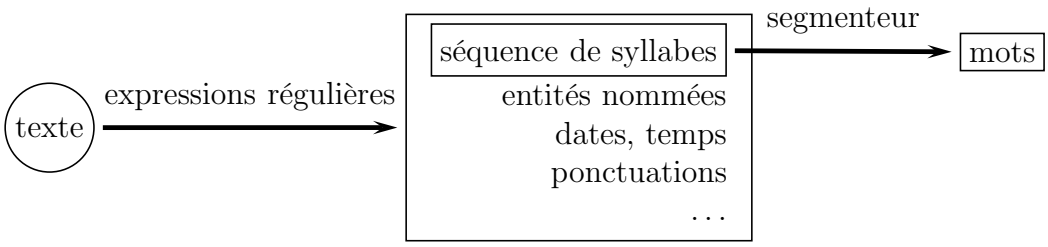


FIGURE 5.2 – Schéma général pour le découpage du texte en unités lexicales

Si un patron est une séquence de syllabes et d'espaces, il est ensuite passé à un segmenteur qui est en charge de le découper en mots. En général, une séquence de syllabes a plusieurs segmentations possibles, cependant, il y a typiquement une seule segmentation qui est correcte. Le schéma général pour le découpage du texte en unités lexicales est présenté dans la Figure 5.2.

Nous prenons un exemple pour illustrer le fonctionnement du système. Etant donné un texte extrait du corpus arboré vietnamien (le treebank vietnamien) comme suit :

`Chiều 23-3-2004, trong khi những lao động đang chờ đợi ngoài hành lang tại chi nhánh của công ty ở TP. HCM, chúng tôi làm việc trực tiếp với đại diện công ty Bitocimex.` [34]

Après l'analyse du texte par des expressions régulières, nous avons la liste des patrons de types corresspondants suivante :
– séquence de syllabes : `Chiều`
– date : `23-3-2004`
– ponctuation : `,`
– séquence de syllabes : `trong khi những lao động đang chờ đợi ngoài hành lang tại chi nhánh của công ty ở`
– entité nommée : `TP. HCM`
– ponctuation : `,`
– séquence de syllabes : `chúng tôi làm việc trực tiếp với đại diện công ty`
– entité nommée : `Bitocimex`
– ponctuation : `.`
Pour découper une phrase en mots, nous pouvons développer un segmenteur simple qui implémente une stratégie de « matching » maximal dans laquelle nous sélectionnons la segmentation qui contient le moins de mots [179]. Dans cette méthode, le segmenteur cherche la séquence la plus longue de syllabes qui commence à la position actuelle et qui est listée dans le lexique. Il prend la séquence comme un mot, déplace la position à la fin de la séquence et recommence à chercher la séquence suivante.

34. Au 23 mars 2004, nous avons discuté directement avec le représentant de la compagnie Bitocimex pendant que les travailleurs attendaient au couloir dans une branche de la compagnie à TP. HCM ville.

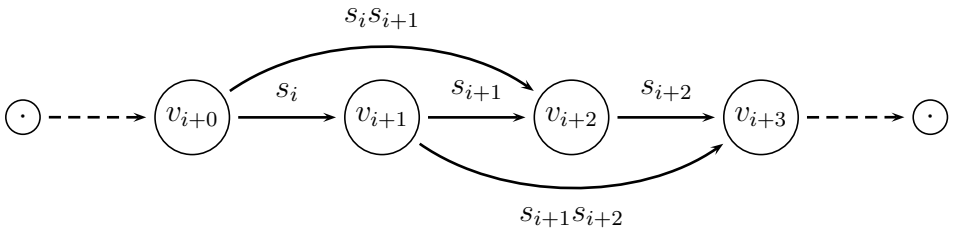


FIGURE 5.3 – Représentation graphique d'une phrase

Cette méthode marche assez bien car les mots longs sont plus susceptibles d'être plus corrects que les mots courts. Pourtant, c'est une méthode trop gourmande qui amène de temps en temps à des segmentations incorrectes à cause d'un nombre important de candidats de segmentation pour une suite de syllabes. Nous avons besoin alors de lister toutes les segmentations possibles et de concevoir une stratégie pour sélectionner la meilleure segmentation.

Nous présentons dans les paragraphes qui suivent notre approche pour proposer toutes les segmentations possibles d'une phrase en mots et notre méthode pour résoudre des ambiguïtés afin de sélectionner la meilleure segmentation pour la phrase.

Une phrase peut être formalisée comme une séquence de syllabes $s_1 s_2 \cdots s_n$ séparées par des espaces. Due au fait que la plupart des mots composés se composent de deux syllabes (*cf.* section 2.3.4), l'ambiguïté la plus fréquente concerne le cas où il y a trois syllabes consécutive et où toutes les deux segmentations possibles $(s_i s_{i+1})(s_{i+2})$ et $(s_i)(s_{i+1} s_{i+2})$ correctes, en dépendant du contexte (et parfois de la sémantique de la phrase). Ce type d'ambiguïté est appelé *l'ambiguïté de chevauchement*, et la séquence $s_i s_{i+1} s_{i+2}$ est appelée la séquence d'ambiguïté de chevauchement.

La phrase est représentée par un graphe orienté $G = (V, E)$, $V = \{v_0, v_1, \ldots, v_n, v_{n+1}\}$, Figure 5.3. Le graphe est linéaire dans le sens que ses sommets sont alignés avec les n syllabes de la phrase ; les sommets v_0 et v_{n+1} sont respectivement le sommet initial et le sommet final. Il y a un arc (v_i, v_j) si les syllabes adjacentes $s_{i+1}, s_{i+2}, \ldots, s_j$ composent un mot, pour tout $i < j$.[35] Si nous dénotons $\texttt{accept}(\mathcal{A}, s)$ le fait que l'automate lexical $\mathcal{A}$ accepte la chaîne de caractères s, la construction formelle d'un graphe pour une phrase est présentée dans Algorithme 1. Après avoir construit le graphe pour la phrase, nous pouvons proposer toutes les segmentations possibles pour la phrase en énumérant tous les chemins les plus courts sur le graphe, du sommet initial au sommet final.

Comme illustré dans la Figure 5.3, chaque séquence d'ambiguïté de chevauchement donne un groupe ambigu, donc, si un graphe a k groupes ambigus, il y a 2^k segmentations pour la phrase[36]. Par exemple, le groupe ambigu dans la Figure 5.3 entraîne deux segmentations $(s_i s_{i+1}) s_{i+2}$ et $s_i (s_{i+1} s_{i+2})$.

Nous discutons dans la sous-section suivante de la résolution de l'ambiguïté que nous avons proposée afin de choisir la meilleure segmentation s'il y a des ambiguïtés de che-

35. En particulier, si la syllabe s_i est elle-même un mot, on a un arc de v_i à v_{i+1}.
36. S'il y n'a pas de chevauchement entre les groupes.

Algorithm 1 Construction of a graph of a phrase $s_1 s_2 \dots s_n$

```
1: V ← ∅ ;
2: for i = 0 to n + 1 do
3:    V ← V ∪ {v_i} ;
4: end for
5: for i = 0 to n do
6:    for j = i to n do
7:       if (accept(A_W, s_i ··· s_j)) then
8:          E ← E ∪ {(v_i, v_{j+1})} ;
9:       end if
10:   end for
11: end for
12: return G = (V, E) ;
```

vauchement.

5.4.2 Résolution des ambiguïtés

Nous avons utilisé un modèle bigramme de langages qui est augmenté par la technique de lissage de l'interpolation linéaire pour la résolution des ambiguïtés.

Dans la modélisation de langage n-gramme, la probabilité $P(s)$ d'une chaîne s est exprimée comme un produit des probabilités des mots qui composent la chaîne. Supposons que la chaîne s se compose de m mots $s = w_1 \cdots w_m$ et pour simplifier la notation, on note la séquence de mots $w_i w_{i+1} \cdots w_j$ par w_i^j. La probabilité d'un mot est conditionnée à la présence des $n-1$ derniers mots, c'est-à-dire $P(w_i|w_1^{i-1}) \approx P(w_i|w_{i-n+1}^{i-1}), \forall i \geq n$. Nous avons donc :

$$P(s) = \prod_{i=1}^{m} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-n+1}^{i-1}). \tag{5.1}$$

Typiquement, la valeur de n est soit deux ou trois, cela ramène à un modèle bigramme ou trigramme respectivement. [37]

Afin d'estimer les probabilités $P(w_i|w_{i-1})$ de l'équation (5.1), nous pouvons utiliser les vraisemblances maximales de ces probabilités sur des données d'apprentissage. Dans le cas d'un modèle bigramme avec $n = 2$, l'estimation de $P(w_i|w_{i-1})$ est

$$P_{VM}(w_i|w_{i-1}) = \frac{P(w_{i-1}w_i)}{P(w_{i-1})} = \frac{c(w_{i-1}w_i)/N}{c(w_{i-1})/N} = \frac{c(w_{i-1}w_i)}{c(w_{i-1})},$$

où $c(\alpha)$ désigne le nombre de fois qu'on voit la chaîne α dans les données d'apprentissage et N est la taille mesurée en mots des données d'apprentissage.

En général, la vraisemblance maximale n'est pas une bonne estimation dans le cas où la taille des données d'apprentissage n'est pas assez importante en comparaison du nombre de

37. Pour que le terme $P(w_i|w_{i-n-1}^{i-1})$ soit significatif pour $i < n$, on peut ajouter au début de la chaîne des mots déguisés. Nous supposons alors qu'il y a $n-1$ tels mots déguisés au début de chaque phrase.

paramètres du modèle. Si c'est le cas, les données d'apprentissage ne pourraient pas fournir des estimations sûres pour les paramètres du modèle. Dans les modèles de langage, une probabilité bigramme zéro peut aboutir à des erreurs de modélisation. Pour cette raison, il y a plusieurs techniques de lissage qui ont été développées pour rectifier l'estimation de vraisemblance maximale afin de produire des probabilités plus précises. Non seulement les méthodes de lissage empêchent généralement des probabilités zéro, mais elles tentent aussi d'améliorer la précision du modèle dans son ensemble. Chaque fois qu'une probabilité est estimée à partir d'un nombre de données limité, le lissage a le potentiel d'améliorer de manière significative l'estimation [29].

Nous adoptons la technique de l'interpolation linéaire pour lisser le modèle. C'est une technique simple et efficace qui est largement utilisée dans le domaine de la modélisation de langage [61]. Dans cette méthode, le modèle bigramme est interpolé avec un modèle unigramme $P_{VM}(w_i) = c(w_i)/N$, un modèle qui reflète la fréquence des mots dans les données d'apprentissage. Nous prenons l'estimation

$$\widehat{P}(w_i|w_{i-1}) = \lambda_1 P_{VM}(w_i|w_{i-1}) + \lambda_2 P_{VM}(w_i), \tag{5.2}$$

où $\lambda_1 + \lambda_2 = 1$ et $\lambda_1, \lambda_2 \geq 0$.

Estimation des valeurs λ

L'objectif des techniques de lissage est d'améliorer la performance d'un modèle de langage. L'estimation des valeurs λ dans (5.2) concerne l'évaluation du modèle de langage.

La mesure la plus utilisée pour évaluer un modèle de langage est la probabilité que le modèle assigne à une donnée de test ou l'entropie de la donnée selon le modèle. Pour un modèle bigramme lissé qui a des probabilités $p(w_i|w_{i-1})$, nous pouvons calculer la probabilité $P(s)$ d'une phrase selon (5.1). La probabilité $P(T)$ d'un ensemble de tests T qui se compose de n phrases $s_1, s_2, \ldots, s_n$ est le produit des probabilités de toutes les phrases de l'ensemble

$$P(T) = \prod_{i=1}^{n} P(s_i).$$

L'entropie $H_p(T)$ de la donnée T est définie par

$$H_p(T) = \frac{-\log_2 P(T)}{N_T} = -\frac{1}{N_T} \sum_{i=1}^{n} \log_2 P(s_i), \tag{5.3}$$

où N_T est la taille du texte T mesurée en mots. L'entropie est liée à l'inverse de la probabilité moyenne que le modèle assigne pour les phrases de l'ensemble de tests ; plus l'entropie est faible, meilleure est la performance du modèle.

Nous réservons une portion de l'ensemble de tests pour estimer les valeurs de λ_1 et λ_2. Cette portion est appelée l'ensemble de validation. On définit $C(w_{i-1}, w_i)$ le nombre de fois où le bigramme (w_{i-1}, w_i) est observé dans l'ensemble de validation. Nous choisissons λ_1, λ_2 qui maximise

$$L(\lambda_1, \lambda_2) = \sum_{w_{i-1}, w_i} C(w_{i-1}, w_i) \log_2 \widehat{P}(w_i|w_{i-1}) \tag{5.4}$$

tel que $\lambda_1 + \lambda_2 = 1$ et $\lambda_1, \lambda_2 \geq 0$.

Les valeurs λ_1 et λ_2 peuvent être estimées par une procédure itérative présentée dans Algorithme 2.

Une fois tous les paramètres du modèle bigramme estimés, les probabilités lissées des bigrammes sont calculées selon (5.2). Ces probabilités sont utilisées pour calculer la probabilité de chaque segmentation d'une phrase s selon (5.1). La plus probable segmentation pour une phrase est celle qui a la plus grande probabilité et on la choisit comme la meilleure segmentation.

Algorithm 2 Estimation of values λ

$\lambda_1 \leftarrow 0.5, \lambda_2 \leftarrow 0.5$;
$\epsilon \leftarrow 0.01$;
repeat
  $\widehat{\lambda}_1 \leftarrow \lambda_1, \quad \widehat{\lambda}_2 \leftarrow \lambda_2$;
  $c_1 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_1 P_{VM}(w_i|w_{i-1})}{\lambda_1 P_{VM}(w_i|w_{i-1})+\lambda_2 P_{VM}(w_i)}$;
  $c_2 \leftarrow \sum_{w_{i-1},w_i} \frac{C(w_{i-1},w_i)\lambda_2 P_{VM}(w_i)}{\lambda_1 P_{VM}(w_i|w_{i-1})+\lambda_2 P_{VM}(w_i)}$;
  $\lambda_1 \leftarrow \frac{c_1}{c_1+c_2}, \quad \lambda_2 \leftarrow 1 - \widehat{\lambda}_1$;
  $\widehat{\epsilon} \leftarrow \sqrt{(\widehat{\lambda}_1 - \lambda_1)^2 + (\widehat{\lambda}_2 - \lambda_2)^2}$;
until $(\widehat{\epsilon} \leq \epsilon)$;
return λ_1, λ_2 ;

Nous présentons dans la section suivante les expérimentations et les résultats obtenus à l'aide de notre méthode.

5.5 Expérimentations

5.5.1 Préparation du corpus

Le corpus que nous utilisons pour évaluer la performance du découpeur est une collection de 1.264 articles appartenant de la section « Politique – Société » du journal *Tuổi trẻ* (La Jeunesse). Le corpus contient au total 507.358 unités lexicales qui sont manuellement découpées et vérifiées orthographiquement par des linguistes du Centre de Lexicographie du Viêtnam. Bien qu'il y ait plusieurs segmentations plausibles pour une phrase vietnamienne donnée, une seule segmentation est retenue. Nous supposons qu'il y a une seule segmentation correcte pour chaque phrase pour deux raisons. Premièrement, c'est la simplicité. Deuxièmement, nous ne connaissons actuellement pas de méthodes efficaces pour utiliser plusieurs segmentations dans des applications typiques concernant le traitement du vietnamien.

Nous effectuons une validation croisée sur le corpus. A chaque expérimentation, nous prenons 90% du corpus comme ensemble d'apprentisage ($\approx$ 456.600 unités lexicales) et 10% du corpus sont réservés comme ensemble de tests.

Step	λ_1	λ_2	ϵ
0	0,500	0,500	1,000
1	0,853	0,147	0,499
2	0,952	0,048	0,139
3	0,981	0,019	0,041
4	0,991	0,009	0,015

TABLE 5.1 – Estimation des valeurs λ

	Précision	**Rappel**	F**-Mesure**
Sans résolution d'ambiguïté	0,968	0,980	0,973
Avec résolution d'ambiguïté	0,970	0,985	0,977

TABLE 5.2 – Précision, rappel et F-mesure du système

5.5.2 Résultats

Dans chaque expérimentation, le modèle bigramme de langage est entraîné sur le corpus d'apprentissage. Une estimation des paramètres λ par l'Algorithme 2 est donnée dans la Table 5.1. Les paramètres estimés convergent après quatre itérations sachant que l'erreur donnée est $\epsilon = 0,03$.

Les résultats expérimentaux ci-dessus révèlent le fait que la technique de lissage qui se base sur l'interpolation linéaire rectifie bien des probabilités bigrammes et unigrammes. Elle améliore l'estimation et la précision entière du modèle.

La Table 5.2 présente les valeurs de précisions, rappels et mesures F du système en deux versions avec ou sans résolution d'ambiguïté. La précision est calculée comme le nombre d'unités lexicales communes sur le nombre d'unités lexicales automatiquement segmentées par le système ; le rappel est le nombre d'unités lexicales communes sur le nombre d'unités lexicales manuellement segmentées ; et la mesure F est la moyenne de ces deux valeurs.

Le système a une bonne performance avec le ratio rappel d'environ $98,5\%$. Pourtant, on voit que l'utilisation de la résolution d'ambiguïté améliore légèrement la précision du système. Cela peut s'expliquer par le fait que le modèle bigramme exploite une petite quantité de données d'apprentissage par rapport à la taille du modèle universel du vietnamien. Nous pensons que le modèle de résolution d'ambiguïté peut améliorer encore la précision du système s'il est entraîné sur de plus gros corpus.

5.5.3 vnTokenizer

Nous avons développé un logiciel appelé `vnTokenizer` qui implémente l'approche présentée pour le découpage automatique du texte vietnamien en unités lexicales. Le logiciel est implémenté en Java et il est indépendant des plateformes.

Le logiciel `vnTokenizer` a été utilisé dans plusieurs projets de recherche et d'application, par exemple le projet national du Viêtnam dont le but est de construire des ressources linguistiques et des outils essentiels pour le traitement des textes vietnamiens, y compris

un corpus arboré du vietnamien. Depuis sa première mise en ligne, les versions binaires et code source de `vnTokenizer` ont été téléchargées plus de 4.300 fois [38].

Il est intéressant de voir que `vnTokenizer` est le système le plus robuste pour le découpage automatique comme il a été montré dans les résultats d'une comparaison des systèmes de découpage du texte en vietnamien publié en 2008 [54].

Le logiciel `vnTokenizer` a été intégré à la plateforme générale `vnToolkit` qui est décrite de manière plus détaillée en Annexe B.

5.6 Bilan

Nous avons présenté dans ce chapitre une approche efficace pour le découpage de textes vietnamiens en unités lexicales qui donne une haute précision. L'approche a été implémentée dans `vnTokenizer`, un segmenteur automatique pour les textes vietnamiens.

Selon une analyse profonde des résultats des expérimentations, nous trouvons deux types de chaînes d'ambiguïté dans des textes vietnamiens :
– *des chaînes qui ont une ambiguïté de chevauchement*, et
– *des chaînes qui ont une ambiguïté de combinaison.*
Une séquence de syllabes $s_1 s_2 \dots s_n$ est appelée une chaîne d'ambiguïté de combinaison s'il s'agit d'un mot composé par lui-même et elle inclut des sous-séquences qui peuvent être aussi des mots par eux-mêmes dans un certain context. Par exemple, le mot `bà ba` (qui signifie « un type de pyjama ») peut être découpé en deux mots `bà` et `ba` (qui signifie « la troisième femme ») ; il existe des contextes dans lesquels les deux segmentations sont correctes.

L'approche que nous avons développée est capable de résoudre efficacement le premier type d'ambiguïté. Pourtant, le deuxième type d'ambiguïté n'est pas encore traité. Il y a trois raisons délicates pour ce type d'ambiguïtés. Premièrement, l'ambiguïté de combinaison requiert une considération syntaxique et sémantique de la segmentation, cependant il n'est pas évident de réaliser automatiquement cette tâche au niveau d'un prétraitement. Deuxièmement, nous observons que la fréquence de ce type d'ambiguïtés n'est pas élevée dans les textes vietnamiens. Troisièmement, l'utilisation d'un modèle trigramme de langage peut résoudre quelques cas concernant ce type d'ambiguïtés mais il y aurait une perte de vitesse de traitement du programme.

Finalement, nous trouvons que dans la majorité des cas, les erreurs de découpage sont dus à de nouveaux mots qui ne sont pas encodés dans le lexique. C'est-à-dire que les mots inconnus sont la raison principale des erreurs de découpage. Cela suggère une direction prioritaire pour améliorer encore le découpeur dans le futur : développer et intégrer au découpeur un outil qui est capable de reconnaître des nouveaux mots dans un texte. Pour développer un tel outil, nous pouvons utiliser des méthodes statistiques ou des connaisances linguistiques qui spécifient les règles de compositions des mots.

38. *http ://www.loria.fr/ lehong/tools/vnTokenizer.php*

6

Etiquetage de textes vietnamiens

Sommaire

Nous présentons dans ce chapitre une étude empirique de l'application de modèles log-linéaires pour l'étiquetage syntaxique de textes vietnamiens. Notre meilleur étiqueteur explore et inclut des connaissances utiles qui, en terme de performance pour l'étiquetage du texte vietnamien, fournit un taux de précision globale de $93,40\%$ et de $80,69\%$ pour les mots inconnus sur un ensemble de test du corpus arboré vietnamien. Notre étiqueteur est nettement supérieur à celui qui est en train d'être utilisé pour développer le corpus arboré vietnamien, et à l'heure actuelle c'est le meilleur résultat obtenu pour l'étiquetage du vietnamien. Les travaux et le résultat présentés dans ce chapitre ont été publiés dans [97].

6.1 Introduction

La tâche d'étiquetage consiste à associer à chaque mot une étiquette représentant une catégorie syntaxique (nom, verbe, adjectif, *etc*).

On peut considérer l'étiquetage comme une tâche de prédiction structurelle. Le but est de prédire un label complexe (une séquence de catégories syntaxiques) sur une entrée

complexe (une phrase entière). Cette tâche est difficile et très différente de la tâche de classification classique. Il y a au moins trois sources de difficulté. Premièrement, si l'apprentissage est effectué par un classifieur qui ne se base que sur le niveau de mots, il y aura une perte d'informations puisque les influences entre des étiquettes voisines ne sont pas prises en compte. Deuxièmement, les phrases ont des longueurs différentes, donc il n'est pas possible de représenter toutes les phrases par des vecteurs d'une même dimension fixe. Troisièmement, l'ensemble de toutes les séquences possibles d'étiquettes est trop grand–il constitue un ensemble de taille exponentielle par rapport au nombre de labels.

De nombreux systèmes pour l'étiquetage automatique en catégories syntaxiques ont été développés pour un large éventail de langues. Parmi les systèmes les plus performants, on trouve ceux qui s'appuient sur des techniques d'apprentissage automatique. Ces systèmes utilisent différentes méthodes pour l'apprentissage ; on peut citer les modèles de Markov cachés [16], l'apprentissage par transformation [18] et l'approche basée sur les modèles log-linéaires [148, 8, 166, 167]. Pour certaines langues comme l'anglais ou d'autres langues bien étudiées, ces systèmes ont atteint des niveaux de performance proches des niveaux humains.

Nous avons montré dans le chapitre 3 que les modèles log-linéaires ont démontré leur efficacité dans de nombreux problèmes de traitement automatique des langues. Nous avons vu également dans le chapitre 4 le succès de ces modèles pour le découpage du texte vietnamien en phrases. Nous nous inscrivons dans la poursuite de ces études en présentant dans ce chapitre une étude empirique de l'application de modèles log-linéaires pour l'étiquetage syntaxique du texte vietnamien.

Ce chapitre est organisé comme suit : dans un premier temps, la section 6.2 présente le modèle d'entropie maximale pour l'étiquetage du texte y compris la sélection des fonctions de traits et l'inférence pour le modèle. Ensuite, la section 6.3 aborde le corpus et les catégories syntaxiques du vietnamien qui sont utilisées dans le problème de l'étiquetage, les expérimentations et les résultats obtenus. Enfin, nous présentons pour terminer la conclusion dans la section 6.4.

6.2 Le modèle probabiliste

6.2.1 Le modèle et les fonctions de traits

Le modèle probabiliste que nous utilisons pour étiqueter du texte vietnamien est une instance des modèles log-linéaires généraux présentés dans le chapitre 3. Nous adoptons une approche qui utilise un modèle log-linéaire puisqu'il permet d'inclure une variété d'informations utiles en utilisant des fonctions de traits. De plus, le modèle log-linéaire ne suppose pas l'indépendance entre les variables.

Nous rappelons la distribution de probabilité des modèles log-linéaires présentée dans le chapitre 3 (*cf.* l'équation (3.12)) dans la formule suivante :

$$p(y|x;w) = \frac{1}{Z(x,w)} \exp \sum_{k=1}^{K} w_k \phi_k(x,y), \tag{6.1}$$

où $Z(x, w)$ est le coefficient de normalisation

$$Z(x, w) = \sum_{y \in \mathcal{Y}} \exp \sum_{k=1}^{K} w_k \phi_k(x, y).$$

Ici, les données $(x, y) \in (\mathcal{X}, \mathcal{Y})$, où $\mathcal{X}$ est un ensemble fini de séquences et $\mathcal{Y}$ est un ensemble fini d'étiquettes. Les fonctions $\phi_k(x, y)$ sont les fonctions de traits utilisés. Le vecteur des paramètres du modèle $w = (w_1, w_2, \ldots, w_K) \in \mathbb{R}^K$ est à estimer.

Dans le problème de l'étiquetage séquentiel, on a un texte qui contient des phrases. Soit une phrase du texte qui se compose de n mots $\mathbf{x} =< x_1, x_2, \ldots, x_n >$ [39] et soit $\mathcal{Y}$ l'ensemble des catégories syntaxiques, la tâche d'étiquetage est le processus d'assigner la séquence de catégories syntaxiques $\mathbf{y} =< y_1, \ldots, y_n >$ la plus probable pour la séquence de mots $< x_1, \ldots, x_n >$. Vu en termes de probabilités, le but revient à estimer la probabilité conditionnelle $p(y_1, \ldots, y_n | x_1, \ldots, x_n)$. Il est bien entendu impossible d'estimer correctement de telles probabilités pour des problèmes évidents de dispersion des données. En pratique, on peut néamoins simplifier le problème en faisant l'hypothèse d'indépendance suivante :

$$\widehat{\mathbf{y}} = \arg\max_{\mathbf{y} \in \mathcal{Y}^n} p(y_1, \ldots, y_n | x_1, \ldots, x_n) \approx \arg\max_{\mathbf{y} \in \mathcal{Y}^n} \prod_{i=1}^{n} p(y_i | h_i),$$

où y_i est l'étiquette du mot x_i et h_i est « l'historique » ou le contexte de (x_i, y_i).

Le modèle log-linéaire (6.1) pour l'étiquetage peut être réécrit comme suit :

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n; w) = \frac{1}{Z(x, w)} \exp \sum_{k=1}^{K} w_k \phi_k(x_1, \ldots, x_n, y_1, \ldots, y_n). \tag{6.2}$$

Pour prévoir l'étiquette y_i du mot x_i, nous utilisons des informations déjà connues, par exemple les deux étiquettes précédentes y_{i-2}, y_{i-1}, le mot actuel x_i et ses voisins, les positions des mots dans la phrase *etc.* Ces informations sont souvent appelées des contextes ou des historiques du modèle et dans les modèles log-linéaires elles sont représentées par des fonctions de traits ϕ_k. Ces fonctions permettent d'exprimer et d'incorporer des statistiques complexes, qui ne sont pas restreintes à des statistiques n-grammes comme dans un modèle de Markov caché. Par exemple, on peut utiliser des fonctions de traits pour examiner les préfixes et les suffixes des mots ou pour incorporer la position des mots dans une phrase, y compris des positions spéciales comme le début ou la fin de la phrase *etc.*

Pour que le modèle log-linéaire soit calculable, il est important que chaque fonction de traits dépende d'une seule étiquette, ou de deux étiquettes adjacentes. En pratique, on suppose souvent qu'une fonction de traits dépend seulement de deux étiquettes dont l'étiquette actuelle et la précédente.

39. Nous utilisons les crochets pointus $< \cdots >$ pour dénoter les séquences de longueur variable afin de les distinguer des vecteurs de taille fixe dénotés par des parenthèses $(\cdots)$.

Nous définissions un contexte h_i pour une position i comme suit :

$$h_i = \{y_{i-2}, y_{i-1}, \mathbf{x}, i\}.$$

Les fonctions de traits sont définies comme des fonctions binaires prenant des valeurs binaires $\{0, 1\}$ sur des contextes h_i et des étiquettes correspondantes. La forme générale de ces fonctions est

$$\phi_k(h_i, y_i) = \begin{cases} 1, & \text{si } h_i \text{ satisfait une condition et } y_i \text{ satisfait une condition,} \\ 0, & \text{sinon.} \end{cases}$$

A titre d'illustration, supposons que l'on veuille déterminer l'étiquette du sixième mot de la phrase anglaise suivante :

`Hispaniola/NNP quickly/RB became/VB an/DT important/JJ base/??` `from which Spain expanded its empire into the rest of the Western Hemisphere .`

Nous avons $i = 6$, $y_{i-2} =$ `DT`, $y_{i-1} =$ `JJ` et $\mathbf{x} =$<`Hispaniola`,`quickly`,...,`Hemisphere`,`.`>. L'ensemble des contextes contient tous les contextes possibles de la forme $(y_{i-2}, y_{i-1}, \mathbf{x}, i)$. L'ensemble des étiquettes possibles est $\mathcal{Y} = \{$`NN`, `NNS`, `Vt`, `Vi`, `IN`, `DT`, ...$\}$ [40]. Nous pouvons définir K fonctions de traits $\phi_k : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, $\forall k = 1, \ldots, K$. Par exemple :

$$\phi_1(h_i, y_i) = \begin{cases} 1, & \text{si le mot actuel } x_i \text{ est } \texttt{base} \text{ et } y_i = \texttt{NN}, \\ 0, & \text{sinon.} \end{cases}$$

$$\phi_2(h_i, y_i) = \begin{cases} 1, & \text{si le mot actuel } x_i \text{ se termine par } \texttt{ing} \text{ et } y_i = \texttt{VBG}, \\ 0, & \text{sinon.} \end{cases}$$

...

L'application de ces fonctions de traits au contexte actuel nous donne

$$\phi_1((\texttt{DT}, \texttt{JJ}, <\texttt{Hispaniola}, \cdots>, 6), \texttt{NN}) = 1$$
$$\phi_2((\texttt{DT}, \texttt{JJ}, <\texttt{Hispaniola}, \cdots>, 6), \texttt{Vt}) = 0$$
$$\ldots$$

Nous prenons comme autre exemple une phrase étiquetée dans les données d'apprentissage du corpus vietnamien : « `Những cây bằng lăng nở đầy hoa tím.` » [41] Les mots et leur étiquettte dans la phrase sont montrés dans le tableau suivant :

Position i	1	2	3	4	5	6	7	8
Mot x_i	Những	cây	bằng lăng	nở	đầy	hoa	tím	.
Etiquette y_i	L	Nc	N	V	A	N	A	.

40. Nous utilisons ici le jeu d'étiquettes du corpus arboré Penn Treebank [114].
41. Des fleurs violettes qui s'épanouissent.

$x_i = \texttt{bằng lăng}$	&	$y_i = N$
$x_{i-1} = \texttt{cây}$	&	$y_i = N$
$x_{i-2} = \texttt{Những}$	&	$y_i = N$
$x_{i+1} = \texttt{nở}$	&	$y_i = N$
$x_{i+2} = \texttt{đầy}$	&	$y_i = N$
$y_{i-1} = \texttt{Nc}$	&	$y_i = N$
$y_{i-2}y_{i-1} = \texttt{L Nc}$	&	$y_i = N$

TABLE 6.1 – Quelques traits définis pour l'historique h_3.

En ce qui concerne la position $i = 3$, on peut définir une fonction de traits comme suit :

$$\phi_{100}(h_i, y_i) = \begin{cases} 1, & \text{si } x_i = \texttt{bằng lăng} \text{ et } y_i = \texttt{N}, \\ 0, & \text{sinon.} \end{cases}$$

Si cette fonction existe dans l'ensemble des fonctions de traits du modèle, son paramètre correspondant va contribuer à la probabilité jointe $p(h_i, y_i)$ quand x_i est `bằng lăng` et quand y_i est `N`. Le paramètre w_{100} du modèle servira comme un poids pour une prédiction contextuelle qui associe au mot `bằng lăng` l'étiquette observée `N`.

De la même façon, on peut définir une autre fonction de traits comme suit :

$$\phi_{101}(h_i, y_i) = \begin{cases} 1, & \text{si } x_{i-1} = \texttt{cây} \text{ et } y_i = \texttt{N}, \\ 0, & \text{sinon.} \end{cases}$$

Cette fonction contribue à l'observation du fait que le mot `cây` précède une étiquette `N`. Ou encore des fonctions de traits :

$$\phi_{102}(h_i, y_i) = \begin{cases} 1, & \text{si } x_{i-2} = \texttt{Những} \text{ et } y_i = \texttt{N}, \\ 0, & \text{sinon.} \end{cases}$$

$$\phi_{103}(h_i, y_i) = \begin{cases} 1, & \text{si } y_{i-1} = \texttt{Nc} \text{ et } y_i = \texttt{N}, \\ 0, & \text{sinon.} \end{cases}$$

$$\phi_{104}(h_i, y_i) = \begin{cases} 1, & \text{si } y_{i-2} = \texttt{L}, \, y_{i-1} = \texttt{Nc} \text{ et } y_i = \texttt{N}, \\ 0, & \text{sinon.} \end{cases}$$

La Table 6.1 présente quelques traits extraits de l'historique h_3 qui contient le mot `bằng lăng`. Nous allons présenter tous les traits utilisés pour chaque contexte dans les expérimentations présentées ultérieurement, dans la section 6.3.

En pratique, nous pouvons définir un nombre quelconque de questions (traits) afin d'intégrer des connaissances utiles collectionnées à partir d'un corpus d'apprentissage. Pour un contexte $x \in \mathcal{X}$ donné, chaque étiquette de l'ensemble $\mathcal{Y}$ est mappée à un vecteur de

Algorithm 3 Search for a sequence of tags (T)

Require: Une phrase de test $< x_1, x_2, \ldots, x_n >$
Ensure: Une séquence d'étiquettes $< y_1, y_2, \ldots, y_n >$
1: Propose des étiquettes probables pour x_1 ;
2: Prend les k meilleures étiquettes et initialise $s_{1j}, \forall 1 \leq j \leq k$;
3: **for** $i = 2$ to n **do**
4: **for** $j = 1$ to k **do**
5: Propose des étiquettes probables pour x_i ;
6: Prend les k meilleures étiquettes pour x_i : $(y_{i1}, y_{i2}, \ldots, y_{ik})$;
7: **for** $u = 1$ to k **do**
8: $s_{ij} \leftarrow < s_{i-1,j}, y_{iu} >$;
9: **end for**
10: **end for**
11: **end for**
12: **return** s_{n1} ;

traits, par exemple

$$\begin{aligned}
\phi(\texttt{L}, \texttt{Nc}, < \texttt{Những}, \texttt{cây}, \cdots >, \texttt{L}) &= 10010101011010 \\
\phi(\texttt{L}, \texttt{Nc}, < \texttt{Những}, \texttt{cây}, \cdots >, \texttt{Nc}) &= 00110101010000 \\
\phi(\texttt{L}, \texttt{Nc}, < \texttt{Những}, \texttt{cây}, \cdots >, \texttt{N}) &= 00100101011001 \\
\phi(\texttt{L}, \texttt{Nc}, < \texttt{Những}, \texttt{cây}, \cdots >, \texttt{A}) &= 10110101111000 \\
&\cdots
\end{aligned}$$

Donc si l'on a K fonctions de traits $\phi_k, k = 1, \ldots, K$, on a un vecteur de traits $\phi(x, y) \in \mathbb{R}^K$ pour chaque $x \in \mathcal{X}$ et $y \in \mathcal{Y}$.

6.2.2 Inférence pour le modèle

Nous utilisons une procédure qui se base sur la recherche des *k-meilleurs* pour trouver la séquence d'étiquettes la plus probable pour une phrase donnée.

Etant donnée une phrase de test $< x_1, x_2, \ldots, x_n >$ et la séquence des étiquettes possibles $< y_1, y_2, \ldots, y_n >$, on a

$$p(y_1, \ldots, y_n | x_1, \ldots, x_n) = \prod_{i=1}^{n} p(y_i | h_i).$$

Soit s_{ij} la séquence des étiquettes qui a la probabilité la $j$ième plus importante jusqu'au le mot x_i inclus. La recherche de la séquence des étiquettes la plus probable est décrite par l'Algorithme 3.

Dans les expérimentations, nous utilisons $k = 10$. Les autres valeurs de k ont été essayées, mais elles n'ont pas donné de meilleurs résultats.

No.	Catégorie	Description	No.	Catégorie	Description
1.	Np	Nom propre	10.	M	Numéral
2.	Nc	Nom individuel	11.	E	Préposition
3.	Nu	Nom d'unité	12.	C	Conjonction subordonnée
4.	N	Nom commun	13.	CC	Conjonction coordonnée
5.	V	Verbe	14.	I	Interjection
6.	A	Adjectif	15.	T	Mot modal, auxiliaire
7.	P	Pronom	16.	Y	Abréviation
8.	R	Adverbe	17.	Z	Morphème lié
9.	L	Déterminant	18.	X	Inconnu

TABLE 6.2 – Les catégories syntaxiques du corpus arboré vietnamien

6.3 Expérimentations

Nous présentons dans cette section les expérimentations et les résultats obtenus à partir d'un certain nombre de modèles log-linéaires pour l'étiquetage présentés ci-dessus. La différence entre ces modèles est l'utilisation de différents ensembles de traits.

L'évaluation d'un modèle est effectuée de façon classique en utilisant une validation croisée. Le corpus est divisé en dix parties qui sont utilisées chaque fois pour tester un modèle entrainé à partir des neuf autres parties. On prend comme le résultat du modèle la moyenne des précisions des dix expérimentations.

6.3.1 Corpus

Le corpus que nous avons utilisé pour entraîner et tester les modèles est le corpus arboré du vietnamien [127], qui a été développé dans le cadre d'un projet national dont le but est de construire des ressources linguistiques et des outils nécessaires pour le traitement du vietnamien écrit.

Le corpus est actuellement composé de 10.163 phrases qui sont manuellement découpées en unité lexicales, étiquetées et analysées syntaxiquement. Les textes du corpus sont rassemblés à partir des sections sociales et politiques du journal Jeunesse du Vietnam. La longueur des phrases du corpus varie de 2 mots à 105 mots avec une moyenne de 22.

Nous avons vu dans le chapitre 2 que le vietnamien est une langue typiquement isolante, chaque mot a une forme unique et il n'y pas de flexion ni de formes morphologiques comme le genre, le nombre, le cas, *etc.* Les mots vietnamiens sont donc catégorisés en s'appuyant sur leur capacité de combinaison avec les autres, leur fonction syntaxique et leur sens général. La Table 6.2 liste les principales catégories syntaxiques utilisées dans le corpus arboré du vietnamien (*cf.* Table 2.4 du chapitre 2).

Outre les catégories syntaxiques classiques utilisées dans les langues occidentales comme le nom, le verbe... nous notons la présence des noms individuels et des mots modaux (voir section 2.4.1), qui expriment quelques nuances manifestées par la flexion dans les langues synthétiques. D'ailleurs, un morphème lié est un morphème qui ne se manifeste

pas comme lemme et qui n'existe jamais de manière isolée, mais qui est toujours rattaché à un autre morphème appelé base : comme `-ons` dans `ouvr-ons` ou `re-` dans `re-faire`.

Dans les sous-sections suivantes, nous présenterons les expérimentations et les résultats obtenus par les modèles log-linéaires que nous avons présentés.

6.3.2 Modèle de base

Tout d'abord, nous développons un modèle de base qui est similaire à un modèle trigramme dans lequel le contexte pour prédire la catégorie syntaxique d'un mot x_i est $\{y_{i-1}y_{i-2}x_ix_{i+1}\}$. Les fonctions de traits du modèle sont produites automatiquement par l'application des schémas de traits listés dans la Table 6.3. Ce modèle est appelé Modèle 0.

No.	Schéma de traits
1.	$x_i = X, y_i = T$
2.	$x_{i-1} = X, y_i = T$
3.	$x_{i+1} = X, y_i = T$
4.	$y_{i-1} = T_1, y_i = T$
5.	$y_{i-1} = T_1, y_{i-2} = T_2, y_i = T$

TABLE 6.3 – Les traits utilisés dans le modèle de base.

Ce modèle de base donne un taux de précision globale de $90,23\%$ qui est comparable à celui de l'étiqueteur utilisé dans [127] ($90,5\%$). En particulier, il présente un résultat médiocre pour les mots inconnus, c'est-à-dire les mots qui ne sont pas présents dans le corpus d'apprentissage, avec le taux de précision de $47,08\%$. Pour cette raison, nous nous intéressons dans un premier temps à étendre l'ensemble de traits afin d'améliorer la performance du modèle sur les mots inconnus.

6.3.3 Traits des mots inconnus

En partant de l'idée que les mots inconnus sont un cas particulier de mots rares dont nombre d'observations est inférieur à un seuil, nous pouvons considérer que les mots inconnus sont des mots rares.

Afin d'améliorer la capacité du modèle dans l'étiquetage des mots rares, nous avons ajouté des schémas de traits spéciaux au modèle de base. Le modèle obtenu est appelé Modèle 1a.

Nous définissons les mots rares commes des mots qui apparaissent moins que six fois dans le corpus d'apprentissage (ce seuil est sélectionné expérimentalement).

Les traits spéciaux sélectionnés pour traiter les mots rares sont listés à la Table 6.4. Les schémas utilisés pour les modèles composent un sous ensemble des schémas utilisés dans [148, 167] pour l'étiquetage du texte anglais.

Notons que nous n'incluons pas les traits de préfixe et de suffixe comme dans les étiqueteurs occidentaux puisque la notion d'affixe n'est pas définie pour les langues isolantes.

No.	Schéma de traits
1.	x_i contient un nombre, $y_i = T$
2.	x_i contient une lettre majuscule, $y_i = T$
3.	x_i ne contient que des lettres majuscules, $y_i = T$
4.	x_i contient un tiret, $y_i = T$

TABLE 6.4 – Traits pour les mots rares.

Pourtant, suite au mécanisme de la combinaison de syllabes qui permet la formation de nouveaux mots composés en vietnamien, quelques syllabes jouent un rôle similaire à celui des affixes dans les langues synthétiques. Par exemple, à partir du mot `biên tập` (éditer), on peut ajouter la syllabe `viên` à la fin du mot pour obtenir le nouveau mot `biên tập viên` (éditeur, une personne qui rédige). Cette syllabe peut être utilisée pour transformer la catégorie syntaxique des mots, d'un verbe à un nom, par exemple de `nghiên cứu` (rechercher) à `nghiên cứu viên` (chercheur) ou de `diễn` (jouer) à `diễn viên` (acteur) *etc.*

Un autre exemple concerne la syllabe `hoá` (lui-même est un mot signifiant « transformer »). Cette syllabe peut être ajoutée à la fin d'un nom ou d'un adjectif pour le transformer en un verbe. Par exemple, de `công nghiệp` (industrie) à `công nghiệp hoá` (industrialiser), ou de `hiện đại` (moderne) à `hiện đại hoá` (moderniser).

Les syllabes `viên` et `hóa` dans ces exemples peuvent être alors considérées comme des suffixes de mots composés reliés.

Cependant, il y a des suffixes qui n'ont pas d'effet sur la transformation de la catégorie syntaxique des mots. Par exemple, la syllabe `phó` dans le mot `phó giám đốc` (vice-directeur) ne change pas la catégorie syntaxique du mot `giám đốc` (directeur).

Afin de prendre en compte tous ces cas pour améliorer la capacité du modèle dans la prédiction des nouveaux mots composés, nous examinons comme des traits supplémentaires la première et la dernière syllabe (les affixes potentielles), et puis les groupes de deux premières et de deux dernières syllabes (le « noyau sémantique » potentiel) de ces mots composés. Le Modèle 1a augmenté par ces traits supplémentaires, qui sont listées dans la Table 6.5, nous donne le Modèle 1b.

No.	Schéma de traits
1.	$\sigma(1, x_i) = X$, $y_i = T$
2.	$\sigma(m, x_i) = X$, $y_i = T$
3.	$\sigma(1, x_i) = X_1, \sigma(2, x_i) = X_2$, $y_i = T$
4.	$\sigma(m, x_i) = X_1, \sigma(m-1, x_i) = X_2$, $y_i = T$
5.	Nombre de syllabes du mot composé x_i

TABLE 6.5 – Traits pour les syllabes d'un mot composé. $\sigma(j, x_i)$ est la fonction qui retourne la j-ième syllabe d'un mot qui se compose de m syllabes.

Il est intéressant que nous trouvions l'utilité de la longueur d'un mot mesurée en syllabes pour l'étiquetage de nouveaux mots. Quand nous ajoutons le schéma numéro 5 au Modèle

1b, le modèle résultant, Modèle 1c, est légèrement supérieur au Modèle 1b.

Les performances de ces quatre modèles sur le corpus de tests sont montrées dans la Table 6.6.

Modèle	**Précision globale**	**Précision de nouveaux mots**
Modèle 0	90, 23%	47, 08%
Modèle 1a	92, 64%	68, 92%
Modèle 1b	92, 85%	73, 23%
Modèle 1c	92, 92%	76, 92%

TABLE 6.6 – Précisions de quatre premiers modèles.

Ces résultats montrent l'utilité de l'information lexicale quand elle est utilisée dans les modèles 1*. Elles donnent une réduction importante de l'erreur de prédiction des nouveaux mots par rapport au Modèle 0. Ces résultats montrent également l'avantage d'inclure au modèle de base des traits particuliers du niveau syllabique.

6.3.4 Discussion des cas problématiques

Il y a beaucoup de mots qui peuvent avoir plusieurs catégories syntaxiques. De plus, nous avons vu que la mutation entre catégories syntaxiques est un phénomène fréquent des langues isolantes dans lesquelles les mots changent de catégorie syntaxique sans changer de forme morphologique. Ce phénomène introduit beaucoup d'ambiguïtés que l'étiqueteur devrait résoudre.

La matrice de confusions du Modèle 1c sur le corpus de tests est donnée dans la Table 6.7. Les étiquettes des lignes représentent les catégories correctes et celles des colonnes représentent les catégories présentées par le modèle. Par exemple, le nombre 41 à la position (N,V) est le nombre des noms communs (N) qui sont étiquettés incorrectement comme verbes (V). Nous voyons que les ambiguïtés entre des paires (N,A), (N,V) et (A,V) sont plus difficiles à résoudre pour le modèle. Cela montre et confirme la mutation fréquente entre ces catégories syntaxiques du vietnamien. Les confusions des trois principales catégories atteignent un taux important (65%) de l'erreur totale.

Il y a quelques types d'erreurs d'étiquetage. Premièrement, il y a des erreurs dues à des fautes inévitables de l'annotation du corpus. Par exemple, dans le corpus le mot `ông` est parfois annoté comme `Nc`, parfois comme `N` ou `P`, tandis qu'il devrait être systématiquement annoté `P`. Deuxièmement, il y a des erreurs qui proviennent des ambiguïtés syntaxiques systématiquement causées par la nature de la langue vietnamienne, par exemple les paires P/N, A/N, V/N et A/V sont très ambiguës, elles sont fortement dépendantes du contexte et ne sont pas faciles à annoter correctement. Troisièmement, on voit que l'étiqueteur a du mal à résoudre des ambiguïtés entre les noms propres, les noms individuels et les noms communs alors que ce n'est pas difficile pour les annotateurs humains de distinguer ces types de noms car ils possèdent des propriétés particulières.

Parmis ces trois types d'erreurs, il semble que la meilleure piste pour améliorer la précision

	N	Nc	Np	V	A	R	E	P
N	0	16	9	**41**	**20**	2	6	11
Nc	**24**	0	0	1	0	0	0	0
Np	12	0	0	4	1	0	0	0
V	**45**	0	2	0	**21**	14	12	0
A	**33**	0	0	**29**	0	6	2	1
R	5	0	0	16	3	0	4	1
E	2	0	0	10	0	0	0	0
P	5	0	0	0	0	1	0	0

TABLE 6.7 – Matrice de confusions pour Modèle 1c.

du modèle est de minimiser les erreurs du troisième type. Dans le paragraphe suivant, nous discutons comment nous incluons des sources additionnelles d'informations afin d'aider à résoudre des ambiguïtés liées à des noms propres et de ce fait améliorer la précision globale du modèle.

6.3.5 Traits pour la résolution des noms propres

Une raison principale qui mène aux erreurs de l'étiquetage du modèle est l'ambiguïté `Np/N`. En fait, dans la plupart des cas, il est assez évident pour l'humain de déterminer la catégorie syntaxique correcte, en utilisant des connaissances simples. En vietnamien, un nom propre composé de plusieurs syllabes est écrit sous une forme cohérente, dans laquelle la première lettre de chaque syllabe est toujours en majuscule, par exemple `Nguyễn Tấn Dũng`, `Hà Nội`. Pour aider le modèle à résoudre des ambiguïtés entre `Np` et `N`, nous incluons des traits qui examinent les syllabes d'un mot et les activent si leur première lettre sont en majuscule. La Table 6.8 montre le résultat du nouveau Modèle 1d qui inclut ces traits. Nous voyons une nette amélioration de la précision de nouveaux mots et de la précision globale du modèle.

Modèle	**Précision globale**	**Précision de nouveaux mots**
Model 1d	93, 13%	80, 62%

TABLE 6.8 – Précision du modèle qui distingue des noms propres.

6.3.6 Le meilleur modèle

Nous avons vu que le vietnamien utilise souvent des mots outils entourant les principaux mots pour exprimer des nuances. De ce fait, l'utilisation de plusieurs mots voisins d'un mot peut renforcer la détermination de l'étiquette du mot. Nous considérons alors un contexte plus large en ajoutant deux schémas de traits qui examinent les mots voisins aux positions ± 2 du mot actuel. Le modèle obtenu est appelé Modèle 2, et c'est le meilleur modèle dans nos expérimentations. La liste complète des schémas de traits utilisés dans le meilleur modèle est montrée dans la Table 6.9.

A notre connaissance, ce modèle donne la meilleure performance pour l'étiquetage de

No.	Schéma	No.	Schéma
	Contextes de mots		*Contextes de syllabes*
1.	$x_i = X, y_i = T$	8.	$\sigma(1, x_i) = X,\ y_i = T$
2.	$x_{i-1} = X, y_i = T$	9.	$\sigma(m, x_i) = X,\ y_i = T$
3.	$x_{i+1} = X, y_i = T$	10.	$\sigma(1, x_i) = X_1, \sigma(2, x_i) = X_2,\ y_i = T$
4.	$y_{i-1} = T_1, y_i = T$	11.	$\sigma(m, x_i) = X_1, \sigma(m-1, x_i) = X_2,\ y_i = T$
5.	$y_{i-1} = T_1, y_{i-2} = T_2, y_i = T$	12.	Nombre de syllabes du x_i
6.	$x_{i-2} = X$	13.	Traits particuliers des noms
7.	$x_{i+2} = X$		propres

TABLE 6.9 – Traits utilisés dans le meilleur modèle.

textes vietnamiens. La Table 6.10 montre les précisions du modèle évalué aux niveaux des mots et des phrases.

Précision globale	Précision de mots inconnus	Précision de phrases
$93,40\%$	$80,69\%$	$31,40\%$

TABLE 6.10 – Précision du meilleur modèle.

Il est intéressant de voir que notre l'étiqueteur surpasse `vnQTAG`, l'étiqueteur qui utilise un modèle de Markov caché pour l'étiquetage du texte vietnamien [128, 130] dont la performance moyenne sur quatre corpus de tests est de $92,5\%$. Cependant, ces résultats ne sont pas comparables car les modèles sont entrainés et testés sur des corpus différents, et surtout sur des jeux d'étiquettes différents.

6.3.7 vnTagger

Nous avons développé un logiciel appelé `vnTagger` qui implémente l'approche présentée pour l'étiquetage du texte vietnamien. Le logiciel est écrit en Java et indépendant des plateformes. Le développement du logiciel a été facilité grâce à un logiciel libre du groupe de TAL à l'Université Stanford qui implémente un étiqueteur à base de l'entropie maximale pour l'anglais [42]. Ce logiciel implémente l'approche de classification en se basant sur la méthode d'entropie maximale. Il utilise une procédure de descente de gradient optimisée par le paramètre de lissage Gaussien pour maximiser la vraisemblance des données d'apprentissage [166].

Le logiciel `vnTagger` est aussi librement distribué sous la licences GNU/GPL. Depuis sa première mise en ligne, les versions binaires et code source de `vnTagger` ont été téléchargées plus de 450 fois [43]. Le logiciel a été utilisé dans de nombreux projets scientifiques et commerciaux pour préparer des corpus étiquetés de taille importante du texte vietnamien [86]. Le logiciel `vnTagger` a été intégré à la plateforme générale `vnToolkit` qui est décrite de manière plus détaillée en Annexe B.

42. *http ://nlp.stanford.edu/software/tagger.shtml*
43. *http ://www.loria.fr/ lehong/tools/vnTagger.php*

6.4 Bilan

Nous avons présenté dans ce chapitre l'utilisation d'un modèle d'entropie maximale et l'exploration des sources utiles d'informations pour développer un étiqueteur performant pour l'étiquetage du texte vietnamien.

Puisque le vietnamien est une langue isolante, le cas extrême d'une langue analytique, il est intéressant de constater que notre étude pourrait suggérer une approche efficace pour l'étiquetage des textes des autres langues analytiques de la région Sud-Est de l'Asie comme le thaï, le khmer, le laos, le burman...

Il est montré que l'utilisation d'un réseaux de dépendances bidirectionnelles avec une série de modèles d'entropie maximale fournit un meilleur résultat quand elle est appliquée à l'étiquetage du texte en anglais [166]. Nous espérons que l'application appropriée de cette approche peut aider à mieux résoudre des ambiguïtés dues à la mutation fréquente des catégories syntaxiques du vietnamien, et ainsi améliorer encore le résultat de l'étiquetage.

Il est également montré que l'intégration de ressources lexicales supplémentaires au corpus d'apprentissage aide à améliorer la performance de l'étiquetage syntaxique [52]. Il est évident que l'utilisation d'un dictionnaire extérieur a un avantage potentiel dans la prédiction des mots inconnus au cas où les mots ne sont pas présents dans le corpus d'apprentissage mais sonts présents dans le dictionnaire extérieur. Dans le travail futur, nous planifions l'intégration d'un lexique vietnamien [129] à notre étiqueteur afin d'améliorer encore sa performance.

Enfin, nous pensons que la précision de notre modèle peut être améliorée avec un corpus d'apprentissage de plus grande taille.

Notre étiqueteur partage un point faible avec les autres étiqueteurs du texte en vietnamien, c'est sa dépendance à la qualité du découpage du texte en unités lexicales. Cependant, le découpage ne peut pas atteindre le meilleur résultat quand il ne connaît pas les catégories syntaxiques potentielles des mots considérés. Dans le futur, nous voulons intégrer ces deux tâches du traitement pour améliorer encore notre résultat du découpage et de l'étiquetage du texte en vietnamien. Nous pensons que cette intégration elle-même est une tâche intéressante et particulière pour le traitement du vietnamien et elle fera l'objet de nos investigations futures.

Troisième partie

Elaboration d'une LTAG pour le vietnamien

7

Présentation du formalisme TAG

Sommaire

Nous présentons dans ce chapitre le formalisme syntaxique TAG, notamment ses propriétés linguistiques et informatiques. Le formalisme TAG a permis des analyses linguistiques originales et efficaces de plusieurs phénomènes pour un certain nombre de langues naturelles, comme l'anglais, le français, l'allemand. Les TAG présentent plusieurs propriétés intéressantes pour les traitements automatiques et elles sont largement employées pour développer des composants syntaxiques dans des systèmes de traitement automatique de nombreuses langues.

Ce chapitre est organisé comme suit : nous présentons d'abord en section 7.1 les unités de base ainsi que les opérations de combinaison du formalisme TAG. Ensuite, nous abordons en section 7.2 les propriétés formelles qui les rendent intéressantes pour les traitements automatiques. Enfin, nous présentons en section 7.3 les propriétés informatiques du formalisme qui justifient son utilisation en pratique.

7.1 Les grammaires d'arbres adjoints

Les Grammaires d'Arbres Adjoints (*Tree Adjoining Grammars*–TAG) ont été introduites par Joshi [78, 79]. A la différence des grammaires hors-contextes qui constituent un système de réécriture de chaînes, les grammaires TAG sont un système de réécriture d'arbres. Ainsi une grammaire TAG est composée d'arbres pouvant être combinés au moyen d'opérations de réécriture définies dans le formalisme TAG.

7.1.1 Arbres élémentaires

Les éléments primitifs d'une TAG sont des arbres élémentaires. Une TAG est *lexicalisée* si chaque arbre élémentaire contient au moins un nœud représentant un item lexical (qui est appelé *l'ancre* de l'arbre). Les arbres élémentaires sont minimaux en ce sens que tous et seulement les arguments de l'ancre sont encapsulés dans l'arbre. Les arbres d'une LTAG possèdent un domaine de localité étendu. Les contraintes grammaticales sont déclarées au-dessus des arbres élémentaires et elles sont indépendantes de tous les processus récursifs. Il y a deux types d'arbres élémentaires : arbres initiaux et arbres auxiliaires.

Les arbres initiaux se caractérisent par des nœuds intérieurs étiquetés par des symboles non-terminaux, et des nœuds feuilles qui sont soit étiquetés par des symboles terminaux, soit étiquetés par des symboles non-terminaux et marqués pour une substitution par le symbole $\downarrow$.

Les arbres auxiliaires disposent parmi les nœuds feuilles d'un nœud étiqueté par le même symbole que la racine de l'arbre ; ce nœud est appelé *nœud pied* et est marqué par le symbole $*$.

7.1.2 Deux opérations

Les arbres élémentaires se combinent par deux opérations de réécriture : la substitution et l'adjonction.

L'opération de *substitution*, représentée dans la Figure 7.1, consiste à substituer un nœud feuille étiqueté X d'un arbre α par un arbre β dont la racine est étiquetée X également.

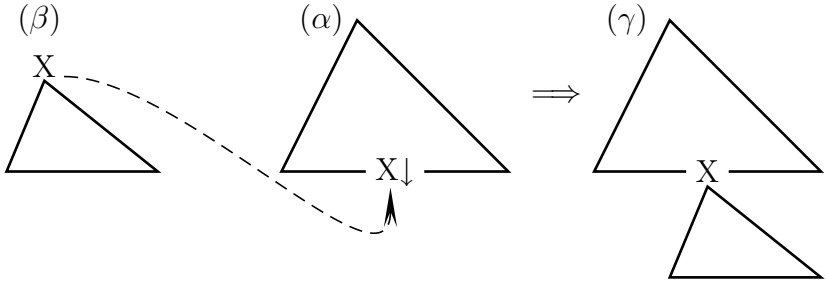


FIGURE 7.1 – Opération de substitution

L'opération d'*adjonction*, représentée dans la Figure 7.2, consiste à insérer un arbre à l'intérieur d'un arbre. Ainsi, dans la Figure 7.2, l'arbre auxiliaire β de racine X ayant un nœud feuille d'étiquette identique, est inséré à l'intérieur de l'arbre α sur un nœud interne étiqueté X, ce qui produit l'arbre γ. On note qu'il est interdit d'adjoindre un arbre sur un nœud marqué pour la substitution.

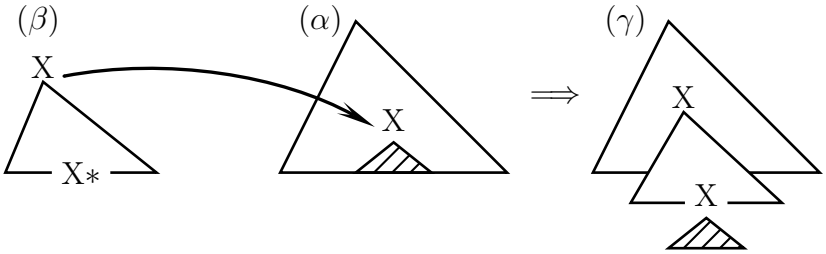


FIGURE 7.2 – Opération d'adjonction

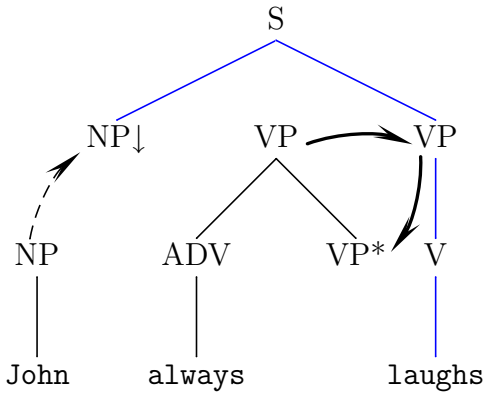


FIGURE 7.3 – Dérivation pour la phrase `John always laughs.`

Lorsque l'on applique une opération de réécriture sur deux arbres élémentaires, un nouvel arbre est produit, cet arbre est appelé *arbre dérivé*. Après réécritures successives, on obtient un arbre dérivé dont tous les nœuds feuilles sont des items lexicaux et il n'y a aucun nœud feuille marqué pour la substitution, on parle alors d'arbre dérivé *complet*. On appelle *dérivation* la réécriture d'un arbre dérivé complet à partir d'arbres élémentaires.

Un exemple de dérivation est donné dans la Figure 7.3. Si nous désignons par $\alpha_{\texttt{John}}$, $\alpha_{\texttt{always}}$ et $\alpha_{\texttt{laughs}}$ respectivement l'arbre associé au mot `John`, `always` et `laughs`, cette dérivation utilise les deux règles de réécriture du formalisme TAG comme suit :

– L'arbre $\alpha_{\texttt{John}}$ est substitué sur le nœud feuille d'étiquette NP de l'arbre $\alpha_{\texttt{laughs}}$, ce qui produit un arbre que nous appelons $\alpha'_{\texttt{laughs}}$;
– L'arbre (auxiliaire) $\alpha_{\texttt{always}}$ est adjoint sur le nœud interne d'étiquette VP de l'arbre $\alpha'_{\texttt{laughs}}$, ce qui produit finalement l'arbre dérivé qui est montré à gauche sur la Figure 7.4.

En plus de l'arbre dérivé complet, on définit une autre structure qui décrit les opérations de réécriture utilisées lors de la construction de l'arbre dérivé. Cette structure est appelée *l'arbre de dérivation*.

L'arbre de dérivation est une structure où les nœuds correspondent aux arbres élémentaires impliqués dans la dérivation, et chaque arc représente une opération de réécriture entre les deux arbres situés à ses extrémités. De plus, chaque nœud est étiqueté par l'adresse de Gorn [44] du nœud où a lieu l'opération de réécriture.

44. L'adresse de Gorn est définie comme suit : le nœud racine a pour adresse 0, le k-ième fils d'un nœud d'adresse j a pour adresse $j.(k-1)$.

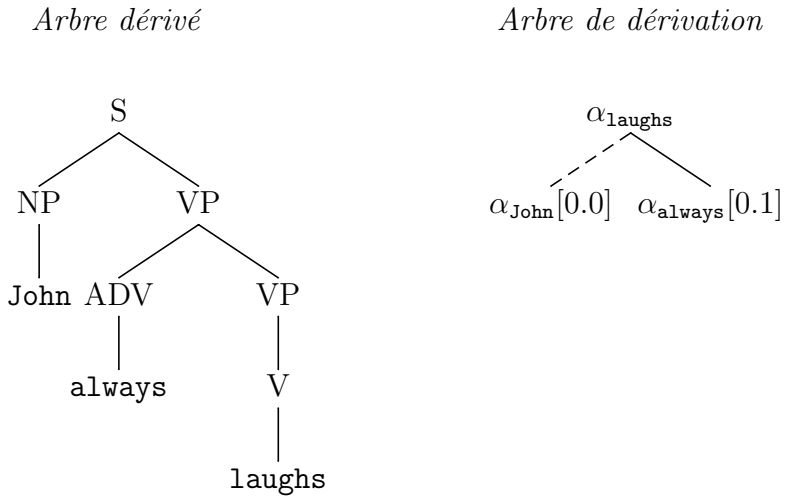


FIGURE 7.4 – L'arbre dérivé et l'arbre de dérivation pour la phrase `John always laughs`.

L'arbre de dérivation décrivant l'analyse de la phrase `John always laughs` est représenté à droite sur la Figure 7.4. Ici, les opérations d'adjonction sont représentées en trait plein, celles de substitution en pointillés, les adresses de Gorn des nœuds sur lesquels s'applique l'opération sont notées entre crochets.

On note quelques contraintes supplémentaires dans le formalisme TAG :

1. Il est interdit d'adjoindre deux arbres auxiliaires sur le même nœud d'un arbre donné. La conséquence en est que l'ordre entre deux adjonctions appliquées à un même arbre est indifférent. Quelle que soit l'adjonction réalisée en premier, l'arbre dérivé produit sera le même. On remarque donc que les nœuds frères d'un arbre de dérivation ne sont pas ordonnés.

2. L'opération d'adjonction peut être contrainte de manière à limiter les arbres auxiliaires pouvant s'adjoindre sur un nœud donné. Il existe trois types de contraintes d'adjonction :

   (a) *Adjonction obligatoire* : Le nœud en question doit obligatoirement être le site d'une adjonction au cours de la dérivation.

   (b) *Adjonction interdite* : Le nœud en question ne peut pas être le site d'une adjonction.

   (c) *Adjonction sélective* : Le nœud en question ne peut être le site d'une adjonction que pour certains arbres auxiliaires.

7.1.3 Les grammaires TAG à structures de traits

Les grammaires TAG à structures de traits (*Feature-Based TAG*–FBTAG) est une extension du formalisme TAG qui a été proposée par Vijay-Shanker et Joshi [177]. Dans cette version des grammaires TAG, chaque nœud d'un arbre élémentaire se voit associé

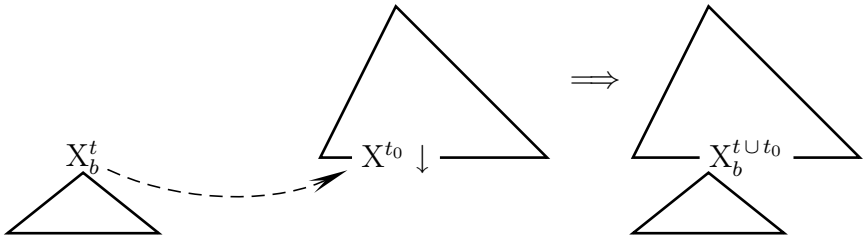


FIGURE 7.5 – Substitution pour FBTAG.

deux structures de traits [45] *top* et *bot* [46]. La structure *top* contient des traits référant aux relations d'un nœud avec ceux qui le dominent, et la structure *bot* aux relations de ce nœud avec ses nœuds fils. Ces structures de traits contraignent les dérivations de manières suivantes :

(substitution) La structure de traits *top* du nœud de substitution est unifiée avec la structure de trait *top* du nœud racine de l'arbre substitué. (Figure 7.5).

(adjonction) Si l'on appelle N le nœud site de l'adjonction, R et P le nœud racine et le pied respectivement de l'arbre auxiliaire adjoint, les structures de traits *top* des nœuds N et R sont unifiées, ainsi que les structures de traits *bot* des nœuds N et P. (Figure 7.6).

En fin de dérivation, lorsque l'on obtient un arbre dérivé complet, les structures de traits *top* et *bot* de chacun de nœuds de l'arbre sont unifiées.

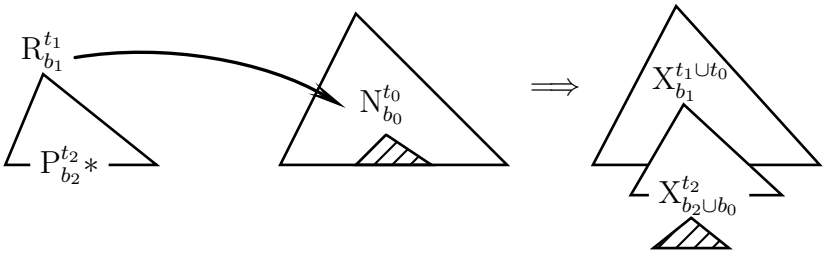


FIGURE 7.6 – Adjonction pour FBTAG.

Nous avons vu précédemment qu'il est possible de contraindre les adjonctions sur un nœud donné. Dans le formalisme FBTAG, les contraintes d'adjonction peuvent être représentées au moyen des structures de traits *top* et *bot* comme suit :
- dans le cas d'une adjonction obligatoire, il faut associer au nœud en question des structures de traits *top* et *bot* qui ne sont pas unifiables (compatibles) ;
- dans le cas d'une adjonction interdite, il faut associer à l'une des structures *top* ou *bot* un trait qui n'est unifiable avec aucun trait de la grammaire ;
- dans le cas d'une adjonction sélective, il faut associer aux structures *top* et *bot* des traits unifiables uniquement avec ceux des arbres autorisés à s'adjoindre.

45. Ces structures sont appelées également matrices attributs-valeurs.
46. Sauf les nœuds marqués pour la substitution qui ne contiennent qu'une structure *top*.

7.2 Propriétés formelles

Formellement, une grammaire d'arbres adjoints G est définie par le quintuplet $G = (\Sigma, \mathcal{N}, S, \mathcal{I}, \mathcal{A})$, où

- Σ est un ensemble fini de symboles terminaux ;
- $\mathcal{N}$ est un ensemble fini de symboles non-terminaux, $\mathcal{N} \cap \Sigma = \emptyset$. En pratique, ce sont les catégories syntaxiques ;
- $S \in \mathcal{N}$ est le symbole distingué ou l'axiome ;
- $\mathcal{I}$ est un ensemble fini d'arbres initiaux ;
- $\mathcal{A}$ est un ensemble fini d'arbres auxiliaires.

Une grammaire TAG lexicalisée (*Lexicalized TAG*–LTAG) est une grammaire TAG dont chaque arbre élémentaire de $\mathcal{I} \cup \mathcal{A}$ contient au moins un nœud feuille étiqueté par un symbole terminal.

Une grammaire lexicalisée peut être vue comme une fonction associant à chaque mot du lexique un ensemble de structures syntaxiques (arbres) représentant l'usage de ce mot dans les différentes phrases de la langue [139]. Notons que les grammaires lexicalisées présentent un avantage pratique : lors de l'analyse syntaxique, l'analyseur peut sélectionner une sous-grammaire suivant les mots de la chaîne à analyser, ce qui facilite la complexité en temps de l'analyse.

A partir de ces définitions d'une grammaire TAG et des opérations permises, il est possible de définir le langage généré par une grammaire TAG. Soit G une grammaire TAG. On note T_G l'ensemble des arbres dérivés complets engendrés par G. On appelle langage généré (ou encore langage couvert) par G, l'ensemble des chaînes de symboles terminaux situées sur les feuilles des éléments de T_G.

7.2.1 Classe de langage des grammaires TAG

En considérant la classification des langages de Chomsky [32], les grammaires TAG appartiennent à la famille des grammaires permettant d'engendrer tous les langages hors-contextes, ainsi que certains langages contextuels, comme par exemple le langage $a^n b^n e c^n d^n$ (Figure 7.7 [47]).

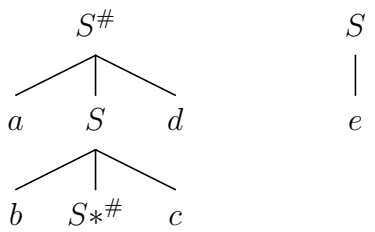


FIGURE 7.7 – Grammaire TAG engendrant le langage $a^n b^n e c^n d^n$.

Cependant, il existe des langages contextuels qui ne peuvent pas être engendrés par une

47. Le symbole # indique une adjonction interdite.

grammaire TAG, comme par exemple le langage contextuel $a^n b^n e^n c^n d^n$. De ce fait, les grammaires TAG sont considérées comme appartenant à la classe des grammaires légèrement sensibles au contexte [48].

7.2.2 Lexicalisation d'une grammaire hors-contexte

Une grammaire TAG permet de lexicaliser une grammaire hors-contexte finiment ambiguë [49] en conservant sa capacité générative forte [50]. La preuve de cette propriété est donnée par Joshi et Schabes dans [79].

On peut se demander si les grammaires TAG sont la plus petite classe de grammaires permettant de lexicaliser une grammaire hors-contexte. La réponse est négative. En effet, Schabes et Waters [158] définissent les grammaires d'arbres d'insertion (*Tree Insertion Grammars*–TIG) qui sont plus contraintes que les TAG et qui engendrent strictement la classe des langages hors-contexte.

7.2.3 Domaine de localité étendu

Dans une grammaire hors-contexte, chaque règle de la grammaire est représentable sous la forme d'un arbre de profondeur un (Figure 7.8). Cependant, les arbres des grammaires TAG peuvent avoir une profondeur quelconque.

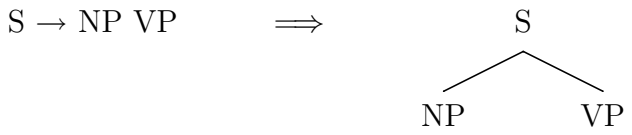


FIGURE 7.8 – Règle hors-contexte représentée sous forme d'arbre.

En conséquence, lors de la définition des arbres élémentaires de notre grammaire, il est possible d'associer à un prédicat une structure disposant d'un nœud représentant chacun des arguments que ce prédicat requiert. Ce qui est appelé généralement *Principe de Co-occurence Prédicat-Arguments* [2, 60]. La capacité des grammaires TAG à manipuler des structures dont le domaine de localité contient un nœud pour chacun des arguments du prédicat est appelée *domaine de localité étendu*.

Le domaine de localité étendu a un intérêt important en pratique. Il est possible de représenter en TAG certains phénomènes linguistiques tels que l'accord sur une seule et même structure, sans avoir à propager des traits supplémentaires. Par exemple, la représentation de l'accord sujet-verbe du français avec une grammaire hors-contexte est assez compliquée, comme montré dans la Figure 7.9.

48. En anglais : Mildly context sensitive grammars.

49. Une grammaire est dite finiment ambiguë si une phrase de longueur finie ne peut pas être engendrée par un nombre infini de dérivations. Elle ne comprend pas par exemple la règle $X \rightarrow X$, qui permet d'engendrer un nombre infini de dérivations.

50. Par capacité générative forte, on désigne l'ensemble des chaînes générées ainsi que leur structure dérivée.

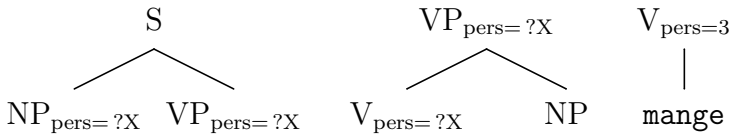


FIGURE 7.9 – Représentation de l'accord avec une grammaire hors-contexte.

Cependant, dans une grammaire TAG, l'accord peut se faire directement comme dans la Figure 7.10.

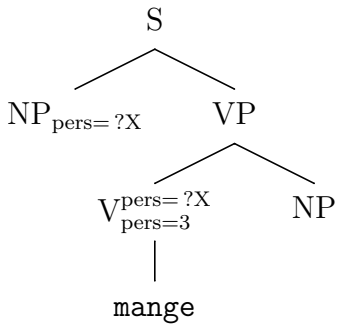


FIGURE 7.10 – Représentation de l'accord avec une grammaire TAG.

7.2.4 Dépendance à longue distance

La dépendance à longue distance réfère à la situation où un argument est réalisé à une distance éloignée du prédicat dont il dépend comme dans les exemples ci-dessous :

1. `Jean cueille une fleur.`
2. `La fleur [que Jean cueille].`
3. `La fleur [que Pierre pense [que Jean cueille]].`
4. `La fleur [que Pierre pense [que Marie lui a dit [que Jean cueille]]].`

Le pronom relatif `que` dans `que Jean cueille` est dépendant du verbe `cueillir`. On constate que le verbe dont le pronom relatif dépend est réalisé à un niveau d'enchâssement quelconque, et potentiellement non borné.

Le traitement des dépendances à longue distance est une question formelle qui a longtemps fait croire que l'usage de transformations était nécessaire en syntaxe pour rendre compte de ce phénomène. Une solution directe et élégante de ce phénomène a été donnée pour les TAG par Kroch et Joshi [93]. L'idée est de décrire la position des arguments du prédicat dans l'arbre élémentaire qui lui est associé et d'utiliser l'opération d'adjonction pour insérer les composantes récursives dans cet arbre élémentaire. Les exemples énumérés précédemment se traitent en TAG au moyen de la grammaire représentée dans Figure 7.11.

7.2.5 Expressions figées et mots composés

Comme l'a montré A. Abeillé *et al.* dans [1, 5], les grammaires TAG permettent une représentation originale des phénomènes non compositionnels, y compris les cas des expressions

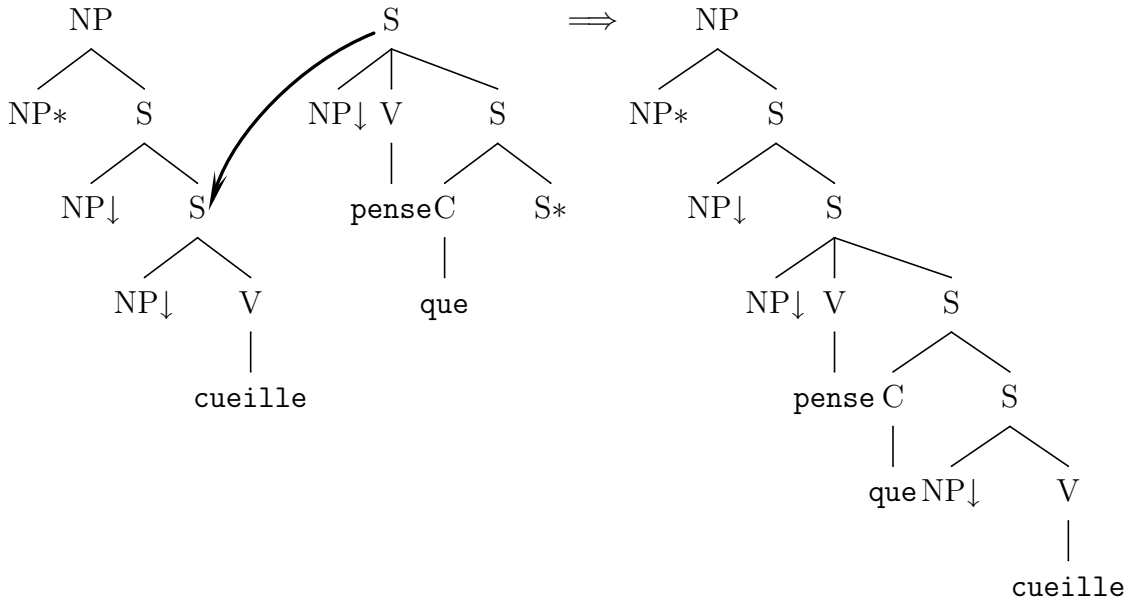


FIGURE 7.11 – Factorisation des composantes récursives hors du domaine de localité.

figées et des mots composés, des locutions conjonctives et prépositionnelles.

Nous appelons « figées » des phrases soit totalement idiomatiques, telle `casser sa pipe`, soit dont la combinatoire (par substitution synonymique) se bloque de façon imprévisible : l'expression `manger ses mots` est bonne, cependant, l'interprétation idiomatique `manger ses paroles` est impossible [2].

La plupart des catégories syntaxiques admettent des mots composés, généralement figés. En français, nous avons des noms composés comme `pomme de terre`, `cordon bleu`, des adjectifs composés comme `vert d'eau`, des adverbes composés comme `tout de suite`, des locutions prépositionnelles ou conjonctives comme `en face de`, `afin de`, des déterminants composés comme `la plupart`.

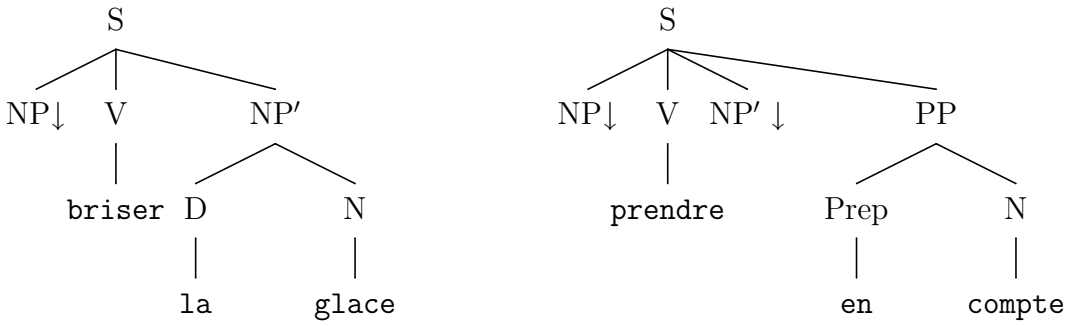


FIGURE 7.12 – Arbres élémentaires étendus pour expressions figées.

Les expressions figées et les mots composés peuvent être représentés en TAG par des arbres élémentaires étendus, c'est-à-dire en général plus profonds que les arbres élémen-

taires ordinaires dans lesquels la tête lexicale est constituée de tous les items figés de l'expression [2]. Par exemple, les arbres élémentaires associés aux expressions `briser la glace` et `prendre en compte` sont montrés dans la Figure 7.12.

7.3 Propriétés informatiques

Nous avons vu dans la section précédente les propriétés formelles du formalisme TAG qui permettent la description de la langue naturelle. Dans cette section, nous allons voir que ce formalisme présente également des propriétés informatiques qui justifient son utilisation dans le cadre du traitement automatique des langues.

7.3.1 Algorithmes et complexité d'analyse

Algorithmes

Une propriété des TAG très importante d'un point de vue du traitement automatique est que les grammaires TAG sont analysables en un temps polynomial. Les algorithmes existants pour les grammaires hors-contextes sont transposables aux grammaires légèrement contextuelles, et en particulier aux TAG.

Un algorithme de type CKY–Cocke-Kasami-Younger [7] a été défini pour la première fois pour les TAG en 1985 par Vijay-Shanker [175, 176]. Comme l'algorithme CKY pour la grammaire hors-contexte, il utilise la technique de programmation dynamique en supposant la mise en forme normale de la grammaire (ici en arbres binaires). L'algorithme a été implémenté mais le résultat est plutôt théorique que pratique [79].

Un algorithme de type Earley [57, 7] a été défini pour les TAG par Schabes et Joshi [155, 156]. Comme pour les grammaires hors-contextes, cet algorithme est efficace en pratique, sa complexité moyenne est inférieure à sa complexité dans le pire des cas pour une phrase de longueur donnée. Conformément à l'algorithme originel d'Earley, il est ascendant et procède de gauche à droite. La performance de l'algorithme est améliorée par une composante prédictive permettant la restriction du nombre d'hypothèses durant l'analyse. Au lieu de règles à points [51], il utilise des arbres à points qui séparent à chaque nœud l'arbre en une partie gauche (reconnue) et une partie droite (non reconnue). Cet algorithme a été implémenté dans les analyseurs du système XTAG [56] pour l'anglais et du système FTAG [2, 3] pour le français.

Un algorithme déterministe ascendant de type LR a initialement été défini par Schabes et Vijay-Shanker [157] mais s'est avéré incorrect. D'autres algorithmes de type LR ont été définis pour intégrer les problèmes d'ambiguïté inhérents aux langues naturelles par Nederhof [125] et Prolo [146].

Un algorithme bidirectionnel ascendant a été défini par Van Noord [173], implémenté dans l'analyseur de Sarkar [154]. Cet algorithme se focalise sur les analyses complètes qui doivent être obtenues avec une complexité la moins importante possible, sans donner une grande importance aux résultats partiels. Lopez [110] a proposé un algorithme bidirectionnel qui utilise une technique d'analyse par connexité. Au lieu d'arbres à points, cet

51. En anglais : *dotted rules*.

Complexité	**Temps**	**Espace**
Grammaires ambiguës		
pire des cas	$O(\|G\|^2 n^6)$	$O(\|G\|n^4)$
meilleur des cas	$O(\|G\|n)$	$O(\|G\|n)$
Grammaires non ambiguës		
pire des cas	$O(\|G\|^2 n^4)$	$O(\|G\|n^4)$
meilleur des cas	$O(\|G\|n)$	$O(\|G\|n)$

TABLE 7.1 – Complexités des algorithmes d'analyse pour grammaires TAG.

algorithme utilise des automates d'états finis pour représenter les arbres élémentaires. Les états de l'automate correspondent à un parcours d'un nœud vers un nœud voisin dans un arbre élémentaire. Tout parcours d'arbre (de gauche à droite, bidirectionnel à partir d'une ancre) peut alors se réaliser sur cet automate en traversant simplement ses états et ses transitions. Un avantage important de cet algorithme est qu'il permet d'obtenir des résultats partiels d'analyse plus étendus, donc d'améliorer la robustesse de l'analyse. C'est l'algorithme intégré à la plate-forme LLP2 que nous avons utilisée pour l'analyse de la grammaire du vietnamien (*cf.* Chapitre 9).

Les grammaires hors-contextes correspondent à des automates à pile non déterministes, les grammaires TAG à des automates à pile de piles (ou à piles emboîtées). D'autres algorithmes basés sur les automates à piles emboîtées ont également été proposés [12, 48].

Complexité de l'analyse

Pour les grammaires hors-contextes, le temps d'analyse d'une phrase au pire des cas avec un algorithme de type CKY ou Earley est $O(|G|^2 n^3)$ où $|G|$ est la taille de la grammaire (en nombre de règles) et n est la longueur de la phrase (en nombre de mots). Pour les TAG, le temps de traitement au pire des cas avec les mêmes algorithmes est $O(|G|^2 n^6)$, ici la taille de la grammaire est mesurée en nombre d'arbres élémentaires.

Avec un algorithme de type Earley, la complexité en temps au pire des cas pour des grammaires TAG non ambiguës est $O(|G|^2 n^4)$. Dans le meilleur des cas, la complexité de traitement est une fonction linéaire de la longueur de la phrase à analyser, pour les TAG comme pour les grammaires hors-contextes. Les complexités en temps et en espace avec ce type d'algorithme sont montrées dans la Table 7.1.

Les grammaires TAG sont donc plus coûteuses pour l'analyse automatique que les grammaires hors-contextes. Cependant, nous notons qu'au contraire des langages de programmation, les langues naturelles mettent en jeu en pratique des grammaires de grande taille et généralement ambiguës, mais pas forcement des phrases longues. De plus, à la différence des grammaires hors-contextes, les grammaires TAG peuvent toujours être focalisées à des unités concernées, ce qui permet de réduire en pratique la taille de la grammaire à considérer pour chaque phrase. C'est une propriété importante des grammaires TAG d'un point de vue traitement automatique qui justifie leur utilisation dans le cadre du traitement automatique des langues.

7.3.2 Analyseurs syntaxiques pour grammaires TAG

Plusieurs analyseurs syntaxiques ont été développés pour les TAG. Pour l'anglais, le système de référence est XTAG [56, 182], développé à l'Université de Pennsylvanie. Ce système comprend un module de représentation du lexique pour une TAG lexicalisée et un analyseur syntaxique de type bidirectionnel ascendant décrit par Sarkar dans [154]. Pour le français, on distingue principalement l'analyseur LLP2 [110, 43] et l'analyseur FRMG [50] qui est transformé par le générateur d'analyseurs DyALog [46].

7.3.3 Ressources linguistiques TAG

Des ressources linguistiques de taille importante ont été développées pour le formalisme TAG. Le système XTAG que nous avons cité ci-dessus inclut une grammaire TAG pour l'anglais. Pour le français, l'Université Paris 7 a développé une grammaire de couverture importante [2]. Il existe également une grammaire TAG pour le français développée au laboratoire LORIA [42]. Pour l'allemand, une grammaire utilisant une extension du formalisme TAG–les TAG à composantes multiples [52] a été créée à l'Université de Tübingen [83]. Une grammaire TAG pour le coréen a aussi été développée. [183].

7.4 Bilan

Nous avons présenté dans ce chapitre le formalisme TAG, ses propriétés formelles et informatiques intéressantes pour les traitements automatiques. Les différents intérêts du formalisme TAG ont conduit à de nombreux développements informatiques, notamment le développement de grammaires à large couverture, ainsi que des analyseurs syntaxiques de la langue.

C'est le formalisme TAG que nous avons choisi dans notre projet pour modéliser la grammaire vietnamienne. Nous allons maintenant nous intéresser au développement d'une grammaire TAG pour le vietnamien.

52. En anglais : Multicomponent TAG.

8

Extraction d'une grammaire LTAG pour le vietnamien

Sommaire

Nous présentons dans ce chapitre l'élaboration d'une grammaire LTAG pour le vietnamien. Il s'agit d'un système que nous avons développé qui extrait automatiquement des grammaires LTAG à partir d'un corpus arboré du vietnamien. Les travaux et le résultat présentés dans ce chapitre est l'amélioration de notre résultat qui a été publié dans [99, 100].

Ce chapitre est organisé comme suit. Tout d'abord, nous présentons en section 8.1 la motivation de l'induction automatique de grammaires et les systèmes existants pour l'apprentissage automatique de grammaires. Puis, la section 8.2 aborde les prototypes d'arbres de la grammaire cible. Ensuite, nous présentons en section 8.3 les algorithmes que nous

avons conçus pour l'extraction automatique de grammaires LTAG. Les expérimentations et la grammaire LTAG pour le vietnamien sont présentées en section 8.4. Enfin, nous concluons ce chapitre en discutant de quelques travaux futurs que nous envisageons pour étendre nos résultats.

8.1 Motivations

8.1.1 Problèmes liés au développement de grammaires

Le développement de grammaires est une tâche difficile. Les difficultés liées à l'écriture et à la maintenance de grammaires ont déjà suscité un bon nombre d'études [42, 58]. Parmi les limitations les plus importantes au développment de grammaires, nous pouvons citer :
- le temps de développement important ;
- la difficulté de conserver la cohérence entre les règles lorsque la grammaire atteint une certaine taille ;
- la difficulté de modulariser le développement ;
- la difficulté de représenter des concepts linguistiques généraux et de traiter des exceptions ;
- les problèmes liés à l'évaluation de la grammaire.

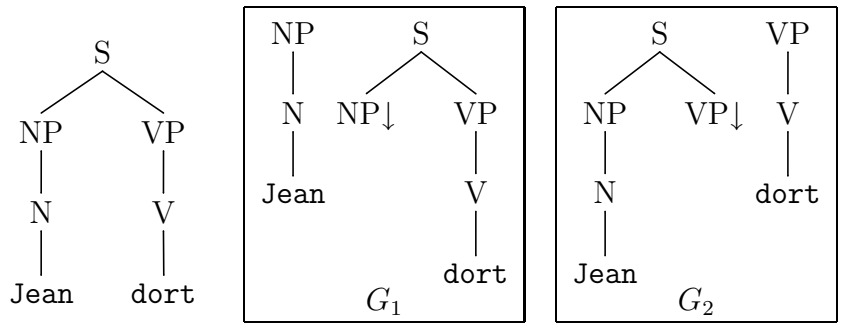
Ces difficultés mettent en évidence la nécessité de disposer d'outils adéquats, tels que des éditeurs de grammaires, des analyseurs syntaxiques et autres suites de tests.

8.1.2 Vers une production semi-automatique de grammaires

Il existe plusieurs possibilités pour faciliter le développpement de grammaires, parmi celles-ci nous pouvons citer l'utilisation d'un niveau d'abstraction [89] ou l'extraction automatique de grammaires à partir de corpus arborés.

Dans la première approche, l'abstraction en question correspond à une description des règles de la grammaire, appelée *méta-grammaire*, à partir de laquelle il doit être possible d'automatiser la génération de la grammaire. Cette description a pour but de factoriser l'information contenue dans la grammaire, en l'occurrence dans le cas des TAG, la factorisation concerne les structures arborescentes. Ces structures sont découpées en fragments élémentaires. Elles sont ensuite décrites en termes de combinaisons de ces fragments en utilisant un langage logique de description d'arbres. Les caractéristiques d'un système méta-grammatical sont décrites en détail dans de nombreux travaux concernant cette approche de production de grammaires, par exemple les publications [47, 139, 42] et [23]. Nous avons examiné cette approche pour construire une grammaire TAG de petite taille pour le vietnamien dans un travail publié précédemment [101].

Ici, nous nous intéressons tout particulièrement à la deuxième approche qui nous a permis de produire des grammaires de taille importante pour le vietnamien, le sujet original de notre travail de thèse. Dans le paragraphe qui suit, nous présentons les travaux antérieurs concernant l'extraction automatique de grammaires.



(a) arbre dérivé (b) grammaire G_1 (c) grammaire G_2

FIGURE 8.1 – An arbre dérivé et deux LTAGs qui peuvent engendrer cet arbre.

8.1.3 Les sytèmes d'extraction de grammaires existants

Les travaux existants sur l'extraction automatique de grammaires en général et de grammaires LTAG en particulier ne concernent que les langues flexionnelles. Xia a développé une méthode uniforme pour l'extraction de grammaires LTAG pour l'anglais, le chinois et le coréen [181, 180]. Chiang a développé un système pour extraire une grammaire LTAG à partir du corpus Penn Treebank pour l'anglais et cette grammaire a été utilisée dans un analyseur statistique de LTAG. Chen *et al.* a également développé une méthode pour extraire automatiquement des grammaires LTAG à partir du corpus Penn Treebank [28, 27] et il existe autres travaux qui sont basés sur son approche comme ceux de Johansen [75] et Nasr [124] pour le français, et Habash pour l'arabe [67]. Neumann a extrait des grammaires d'arbres lexicalisées pour l'anglais à partir du corpus Penn Treebank et pour l'allemand à partir du corpus NEGRA [126]. Park a extrait des grammaires LTAG pour le coréen à partir du corpus Sejong Treebank [138].

8.2 Les grammaires cibles

Etant donné un arbre dérivé, il pourrait y avoir plusieurs grammaires LTAG qui peuvent engendrer cet arbre. Par exemple, un arbre dérivé simple comme celui de la Figure 8.1a peut être produit soit par la grammaire G_1 dans la Figure 8.1b soit par la grammaire G_2 dans la Figure 8.1c.

Tandis que les deux grammaires sont TAGs, nous préférons toujours la grammaire G_1 à la grammaire G_2 puisqu'il est plus plausible linguistiquement qu'une phrase soit ancrée par un verbe plutôt que par un nom. Cette grammaire respecte également les principes de bonne formation des arbres élémentaires, notamment le principe de cooccurence prédicat-arguments qui précise que tout prédicat contient dans sa structure élémentaire au moins un nœud pour chacun des arguments qu'il sous-catégorise (sous forme d'un nœud à substitution ou de nœud pied) [2]. Ici, c'est le nœud NP qui est l'argument de l'ancre lexicale `dort`. Nous voulons donc utiliser des connaissances linguistiques pour développer un système qui produit G_1 plutôt que G_2.

Nous rappelons que le formalisme TAG est un framework général. Le formalisme peut être utilisé pour engendrer des langages formels comme le langage $\{a^n b^n c^n\}$. Son usage n'est pas limité aux langues naturelles, le formalisme lui-même n'impose pas de contraintes qui se basent seulement sur les propriétés des langues naturelles. Dans le but de l'extraction de grammaires de langues naturelles, nous voulons imposer de contraintes sur les grammaires cibles afin de modéliser les propriétés des langues naturelles. Ces contraintes sont basées sur des notions linguistiques bien définies comme la notion de *tête*. Les grammaires cibles ne forment donc qu'un sous-ensemble de toutes les grammaires LTAGs possibles. Dans l'exemple ci-dessus, le système ne produit que G_1.

Nous rappelons dans le paragraphe qui suit quelques notions syntaxiques importantes et présentons comment elles sont représentées dans des théories linguistiques et les grammaires LTAGs. Les notions abordées sont celle de tête et de ses projections, ainsi que celles d'arguments et de modifieurs.

8.2.1 Quelques notions syntaxiques importantes

Une notion importante dans un bon nombre des théories linguistiques comme la théorie X-bar [73] et la théorie GB (Government and Binding) [33] est la notion de *tête*. Une tête détermine les propriétés principales de la phrase à laquelle elle appartient et elle pourrait être projetée sur plusieurs niveaux différents. Nous appelons la chaîne formée par une tête et ses projections une *chaîne de projection*.

Dans la théorie X-bar, une tête X se projette sur $\overline{X}$, qui se projette sur XP comme dans la Figure 8.2a. Ici, X est une catégorie syntaxique quelconque, par exemple un verbe, et XP est un groupe syntaxique comme un groupe verbal.

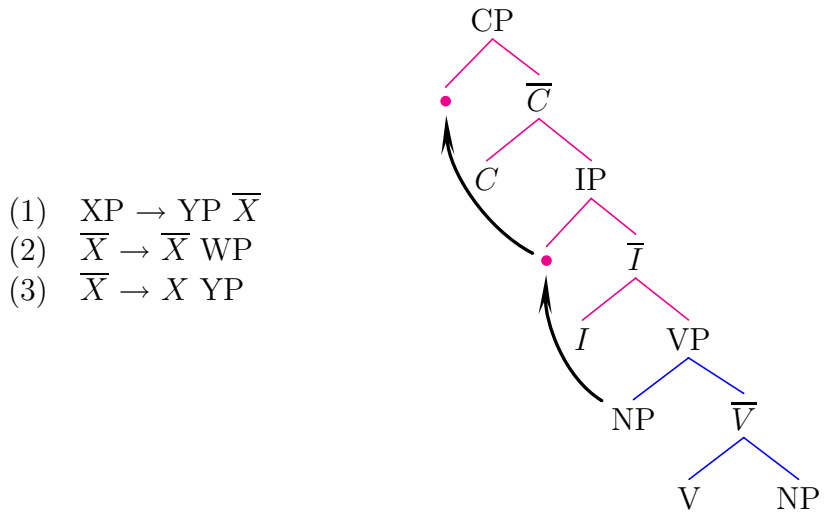
Il y a deux types de têtes dans la théorie GB : *les têtes lexicales* et *les têtes fonctionnelles*. Dans la Figure 8.2b, V est une tête lexicale (verbe), pendant que C (complément) et I (inflexion) sont des têtes fonctionnelles. Les projections de têtes lexicales et fonctionnelles sont appelées respectivement les projections lexicales et fonctionnelles. Les arcs dans la figure montrent le mouvement syntaxique d'une position plus basse à une position plus haute.

Une tête peut avoir des arguments, elle et ses projections peuvent être modifiées par d'autres catégories syntaxiques. Par exemple, un verbe peut se projeter sur un syntagme verbal et il peut avoir un ou plusieurs arguments, et un syntagme verbal peut être modifié par des syntagmes prépositionnels, des syntagmes adverbiaux, *etc.*

Dans un arbre élémentaire de grammaires TAG, les éléments têtes sont les ancres lexicales qui déterminent la sous-catégorisation et le contenu prédicatif de l'expression. Les têtes caractérisent la structure et la sémantique de leur arbre élémentaire.

8.2.2 Les prototypes des arbres élémentaires

Nous avons vu que TAG est un framework général, il n'est donc pas obligé de suivre une théorie linguistique particulière comme la théorie X-bar. Cependant, les notions de tête, de projection, d'argument et de modifieur sont largement acceptées au sein de la communauté des TAG, et en pratique on respecte souvent des conventions quand on

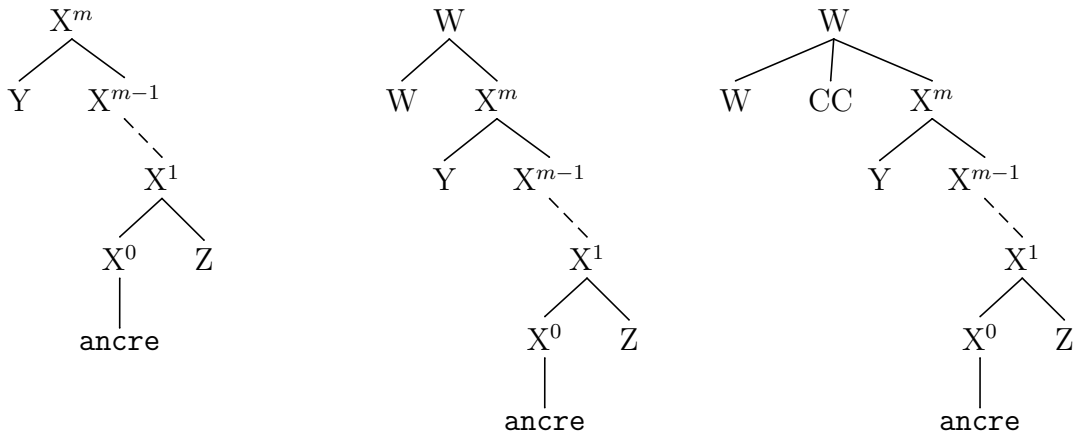


(a) Règles dans la théorie X-bar (b) Un syntagme dans la théorie GB

FIGURE 8.2 – Les notions de tête dans la théorie X-bar et la théorie GB

développe manuellement des grammaires TAG pour les langues naturelles. Par exemple, on utilise souvent des arbres initiaux pour exprimer des relations prédicat-arguments : l'ancre d'un arbre initial est la tête du nœud racine, et tous les arguments de la tête sont inclus dans le même arbre. Au contraire, les arbres auxiliaires sont utilisés pour exprimer des relations de modification où la catégorie du nœud racine et du nœud pied est celle de l'élément modifié, et le modifieur est un frère du nœud pied. Ces conventions sont formalisées par trois types d'arbres élémentaires qui sont définis selon les relations entre l'ancre de l'arbre élémentaire et les autres nœuds de l'arbre. Les trois prototypes d'arbres élémentaires sont montrés dans la Figure 8.3.

- Les arbres d'épines dorsales (spine-etree) pour les relations prédicat-arguments : un arbre d'épine contient une tête X^0, ses projections X^1, $X^2 \ldots$, X^m et ses arguments. Le chemin de X^0 à la racine X^m est la chaîne de projection. La tête X^0 est également l'ancre lexicale de l'arbre et ses arguments sont des nœuds feuilles attachés aux niveaux différents de l'arbre.
- Les arbres de modifieur (mod-etree) pour les relations de modification : la racine d'un arbre de modifieur a deux fils : un fils qui a la même étiquette (W) que la racine, tandis que l'autre fils X^m est un modifieur de son frère. Le nœud X^m est étendu en un arbre d'épine dont la tête X^0 est l'ancre de l'arbre entier.
- Les arbres de conjonction (conj-etree) pour les relations de coordination : dans un arbre de conjonction, les fils de la racine se composent de deux constituants conjoints et d'une conjonction (CC). Un constituant conjoint (ou un groupe conjoint) est étendu en un arbre d'épine dont la tête est l'ancre de l'arbre entier. Essentiellement, un arbre de conjonction est similaire à un arbre de modifieur sauf que sa racine a un fils supplémentaire qui est une conjonction.



(a) spine-etree (relation prédicat-argument) (b) mod-etree (relation de modification) (c) conj-etree (relation de coordination)

FIGURE 8.3 – Les trois prototypes d'arbres élémentaires

La similarité entre les prototypes de la Figure 8.3 et les règles dans la théorie X-bar est claire. Un arbre d'épine est un arbre qui combine les règles du premier et du troisième type de la théorie X-bar ; un arbre de modifieur combine les trois types de règles. Un arbre d'épine est également très similaire à la structure syntagmatique de base dans la théorie GB comme montrée dans la Figure 8.2b.

On note quelques explications des trois prototypes comme suit. Premièrement, chaque nœud de ces prototypes peut avoir zéro ou plusieurs fils ; s'il a plus d'un fils, l'ordre des fils n'est pas spécifié dans les prototypes. Par exemple, les prototypes permettent à des arguments Y et Z d'apparaître à gauche ou à droite de X^0. Deuxièmement, les arbres de modifieur et de conjonction sont des arbres auxiliaires du formalisme TAG, tandis que les arbres d'épine sont des arbres initiaux. Troisièmement, les notions de tête et d'ancre ne coïncident pas dans les prototypes : l'ancre d'un arbre d'épine est la tête du nœud racine, cependant l'ancre d'un arbre de modifieur ou de conjonction est la tête du syntagme du modifieur, elle n'est pas la tête du nœud racine.

Dans la section suivante, nous présentons les algorithmes d'extraction de grammaires LTAG à partir de corpus arborés qui permettent d'extraire automatiquement des arbres élémentaires conformes aux trois prototypes ci-dessus.

8.3 Algorithmes d'extraction

Les algorithmes d'extraction de grammaires prennent en entrée un corpus arboré, appelé corpus d'apprentissage et produisent une LTAG. Le corpus est traité arbre par arbre.

En général, l'extraction de grammaires LTAG à partir d'un corpus arboré comprend trois étapes. Tout d'abord, les structures arborées des phrases sont converties en arbres dérivés. Ensuite, les arbres dérivés sont décomposés en un ensemble d'arbres élémentaires conformes aux trois prototypes prédéfinis. Enfin, les arbres élémentaires invalides sont

No.	Catégorie	Description
1.	S	phrase déclarative simple
2.	VP	groupe verbal
3.	NP	groupe nominal
4.	PP	groupe prépositionnel
5.	N	nom commun
6.	V	verbe
7.	P	pronom
8.	R	adverbe
9.	E	préposition
10.	CC	conjonction coordonnée

TABLE 8.1 – Les catégories syntaxiques abordées dans les exemples.

éliminés en utilisant des connaissances linguistiques.

Nous présentons de manière plus détaillée dans les sous-sections suivantes les algorithmes réalisés dans ces trois étapes. A titre d'illustration pour les algorithmes présentés, nous donnons au fur et à mesure des exemples sur des arbres généraux ainsi que sur des arbres réels tirés d'un corpus arboré vietnamien. Les catégories syntaxiques du corpus vietnamien abordées dans ce chapitre sont listées dans la Table 8.1 (voir également la section 8.4).

8.3.1 Construction des arbres dérivés

Les structures des phrases dans le corpus arboré vietnamien suivent un format de parenthèses similaire au style du corpus anglais Penn Treebank [114]. Ces structures ne sont pas spécifiques au formalisme LTAG. En général, elles ne sont pas au format des arbres dérivés du formalisme LTAG dans lequel les têtes, les arguments et les modifieurs sont distingués explicitement. Nous devons donc convertir les structures arborées du corpus en arbres dérivés.

Tout d'abord, nous classifions chaque nœud d'une structure arborée en trois types : la tête, l'argument ou le modifieur. Ensuite, nous transformons la structure en un arbre dérivé en ajoutant des nœuds intermédiaires de sorte qu'à chaque niveau de l'arbre, les nœuds satisfassent exactement une des trois conditions suivantes :
- *relation prédicat-argument* : il y a un ou plusieurs nœuds, un nœud est la tête, les autres sont des arguments de la tête ;
- *relation de modification* : il y a exactement deux nœuds, un nœud est modifié par l'autre ;
- *relation de coordination* : il y a exactement trois nœuds dont deux nœuds sont coordonnés par une conjonction.

Pour trouver les têtes des syntagmes, nous avons construit une *table de percolation de tête*[53] [112, 37] pour le corpus arboré vietnamien. Cette table est utilisée pour sélectionner le fils tête d'un nœud. De plus, nous avons également construit une *table d'arguments* pour déterminer les types de compléments qu'une tête peut prendre. Cette table permet de

53. En anglais : *a head percolation table.*

Algorithm 4 PROCESS-CONJ(T)

Require: A tree T
Ensure: T with conjunctions processed
1: **for** $K \in T.\texttt{kids}$ **do**
2: **if** IS-PHRASAL(K) **then**
3: $K \leftarrow$ PROCESS-CONJ(K) ;
4: **end if**
5: **end for**
6: $(\mathcal{C}_1, \ldots, \mathcal{C}_k) \leftarrow$ CONJ-GROUPS($T.\texttt{kids}$) ;
7: **for** $i = 1$ to k **do**
8: **if** $\|\mathcal{C}_i\| > 1$ **then**
9: INSERT-NODE($T, \mathcal{C}_i$) ;
10: **end if**
11: **end for**
12: **if** $k > 2$ **then**
13: **for** $i = k$ **downto** 3 **do**
14: $\mathcal{L} \leftarrow \mathcal{C}_{i-1} \cup c_{i-1} \cup \mathcal{C}_i$;
15: $T^* \leftarrow$ INSERT-NODE($T, \mathcal{L}$) ;
16: $\mathcal{C}_{i-1} \leftarrow T^*$;
17: **end for**
18: **end if**
19: **return** T ;

marquer explicitement chaque frère d'une tête comme son argument ou son modifieur selon l'étiquette du frère, celle de la tête et la position du frère par rapport à la tête. D'ailleurs avec la *table des catégories syntaxiques*, ces trois tables constituent l'information spécifique du corpus arboré vietnamien qui sont nécessaires pour les algorithmes d'extraction. [54] Ces trois tables sont fournies en Annexe A.

Du fait de la spécificité des structures de conjonction comparées à celles d'argument et de modifieur, nous groupons dans un premier temps tous les groupes de conjonction d'une structure arborée en utilisant l'Algorithme 4 puis nous construisons l'arbre dérivé complet pour l'arbre résultant en utilisant l'Algorithme 5.

A titre d'illustration, la Figure 8.4 montre un arbre avec des groupes de conjonction avant et après l'application de l'Algorithme 4, où les c_i sont des conjonctions coordonnées et les $X_i, i = 1, 2, 3$ sont des groupes de conjonction. La Figure 8.5 montre une réalisation de l'Algorithm 5 où A_1, A_2 sont des arguments du fils tête H du nœud T et M_1, M_2 sont les modifieurs de H.

Ces deux algorithmes utilisent la fonction INSERT-NODE($T, \mathcal{L}$) montrée dans l'Algorithme 6 pour insérer un nœud intermédiaire entre un nœud T et une liste $\mathcal{L}$ de ses nœuds fils. Ce nouveau nœud est un fils de T, a le même label que T et prend $\mathcal{L}$ comme la liste de ses fils. La fonction CONJ-GROUPS($\mathcal{L}$) retourne k groupes de composants $\mathcal{C}_i, i = 1, \ldots, k$

54. A notre connaissance, c'est la première fois que ces tables sont abordées et publiés pour la langue vietnamienne.

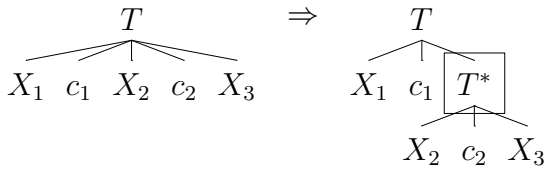


FIGURE 8.4 – La transformation de groupes de conjonction par l'Algorithme 4.

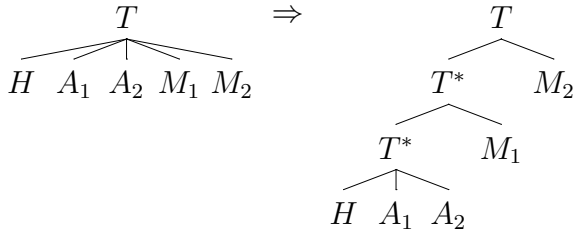


FIGURE 8.5 – Un exemple de la construction de l'arbre dérivé.

de $\mathcal{L}$ qui sont séparés par $k-1$ conjonctions $c_1, \ldots, c_{k-1}$. La fonction NEW-NODE(l) retourne un nouveau nœud de label l.

L'Algorithme 5 utilise quelques fonctions qui sont triviales. La fonction HEAD-CHILD(X) sélectionne le fils tête d'un nœud X selon une table de percolation de tête. La table de percolation de tête pour le corpus arboré vietnamien est montrée dans la Table A.1. La fonction IS-LEAF(X) vérifie si un nœud X est un nœud feuille ou non. La fonction IS-PHRASAL(X) vérifie si X est un nœud syntagmatique ou non[55]. La fonction ARG-NODES($H, \mathcal{L}$) (respectivement MOD-NODES($H, \mathcal{L}$)) retourne la liste des nœuds qui sont les arguments (respectivement les modifieurs) d'un nœud H. La liste $\mathcal{L}$ contient tous les frères de H.

Par exemple, la Figure 8.7 montre une structure arborée d'une phrase extraite du corpus arboré vietnamien « `Họ sẽ không chuyển hàng xuống thuyền vào ngày mai.` » (Ils ne livreront pas les marchandises au bateau demain.) dont la structure de parenthèses correspondante est montrée dans la Figure 8.6. Les têtes des phrases sont entourées.

L'arbre dérivé de la phrase donné par l'Algorithme 5 est montré dans la Figure 8.8. Les nœuds insérés sont carrés.

8.3.2 Extraction des arbres élémentaires

Dans cette étape, chaque arbre dérivé est décomposé en un ensemble d'arbres élémentaires. Les structures récursives de l'arbre dérivé sont découpées pour former des arbres auxiliaires ; les structures non-récursives qui restent seront extraites comme des arbres initiaux. Les arbres extraits appartiennent à un des trois prototypes des arbres élémentaires présentés dans la Figure 8.3.

55. Un nœud syntagmatique est défini comme un nœud qui n'est pas une feuille ou un préterminal. Cela signifie qu'il doit avoir un ou plusieurs fils, ou un fils qui n'est pas un terminal.

Algorithm 5 BUILD-DERIVED-TREE(T)

Require: A tree T whose conjunctions have been processed
Ensure: A derived tree whose root is T

if (not IS-PHRASAL(T)) **then**
  return T;
end if
$H \leftarrow$ HEAD-CHILD(T);
if not IS-LEAF(H) **then**
  for $K \in T.\texttt{kids}$ **do**
    $K \leftarrow$ BUILD-DERIVED-TREE(K);
  end for
  $\mathcal{A} \leftarrow$ ARG-NODES($H, \mathcal{L}$);
  $\mathcal{M} \leftarrow$ MOD-NODES($H, \mathcal{L}$);
  $m \leftarrow \|\mathcal{M}\|$;
  if $m > 0$ **then**
    $\mathcal{L} \leftarrow \{H\} \cup \mathcal{A}$;
    $T^* \leftarrow$ INSERT-NODE($T, \mathcal{L}$);
  end if
  $(M_1, M_2, \ldots, M_m) \leftarrow \mathcal{M}$;
  for $i = 1$ **to** $m - 1$ **do**
    $\mathcal{L} \leftarrow \{M_i, T^*\}$;
    $T' \leftarrow$ INSERT-NODE($T, \mathcal{L}$);
    $T^* \leftarrow T'$;
  end for
end if
return T;

Algorithm 6 INSERT-NODE($T, \mathcal{L}$)

Require: A tree T and its children list $\mathcal{L}$
Ensure: A new child node T^* of T whose kids are $\mathcal{L}$

$T^* \leftarrow$ NEW-NODE($T.\texttt{label}$);
$T^*.\texttt{kids} \leftarrow \mathcal{L}$;
$T.\texttt{kids} \leftarrow T.\texttt{kids} \setminus \mathcal{L}$;
$T.\texttt{kids} \leftarrow T.\texttt{kids} \cup \{T^*\}$;
return T^*;

Le processus de l'extraction implique la copie des nœuds des arbres dérivés pour construire des arbres élémentaires. Le résultat de l'extraction est trois ensembles d'arbres élémentaires : $\mathcal{S}$ contient des arbres d'épine, $\mathcal{M}$ contient des arbres de modifieur et $\mathcal{C}$ contient des arbres de conjonction.

Pour extraire des arbres élémentaires à partir d'un arbre dérivé T, nous trouvons tout d'abord un chemin de tête $\{H_0, H_1, \ldots, H_n\}$ de T[56]. Un nœud sur le chemin de tête est

56. Un *chemin de tête* d'un nœud T est le chemin unique de T à un nœud feuille où chaque nœud sauf

```
(S
  (NP (P Họ))
  (VP (R sẽ) (R không) (V chuyển)
    (NP (N hàng))
    (PP (E xuống)
      (NP (N thuyền)))
    (PP-TMP (E vào)
      (NP (N ngày mai)))))
```

FIGURE 8.6 – Une structure arborée du corpus

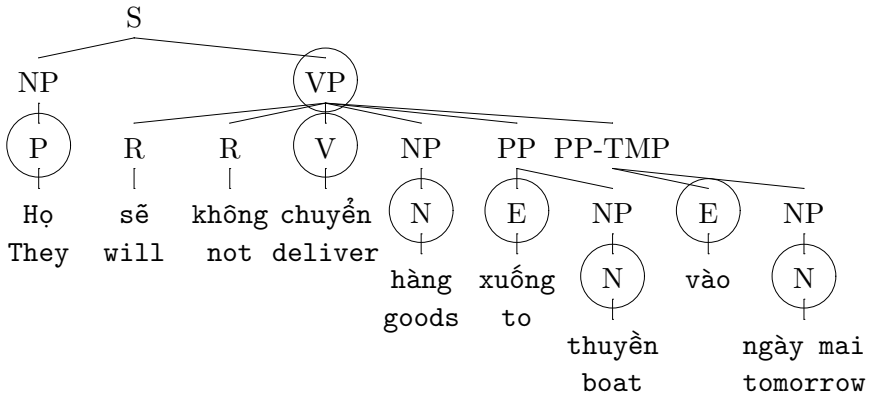


FIGURE 8.7 – L'arbre de la structure dans Figure 8.6.

appelé un *nœud de lien* (link node) si son label est similaire à celui de son père. Pour chaque père P et son fils tête H, nous prenons la liste $\mathcal{L}$ des frères de H et déterminons la relation entre H et $\mathcal{L}$. Si la relation est une coordination, un arbre de conjonction est extrait ; si c'est une relation de modification, un arbre de modifieur est extrait ; sinon, c'est une relation prédicat-argument et un arbre d'épine est extrait. L'Algorithme 7 montre l'algorithme d'extraction.

L'Algorithme 8 montre la fonction qui extrait un arbre d'épine. Il utilise la fonction MERGE-LINK-NODES(T) qui fusionne tous les nœuds de lien d'un arbre d'épine en un seul nœud. Les algorithmes 9 et 10 sont des fonctions qui respectivement extraient des arbres de modifieur et des arbres de conjonction.

Par exemple, à partir de l'arbre dérivé montré dans la Figure 8.8, 9 arbres élémentaires sont extraits par les algorithmes comme dans la Figure 8.10 et la Figure 8.9.

8.3.3 Filtrage des arbres élémentaires invalides

Pour tous les corpus arborés des langues, les erreurs d'annotation ne sont pas évitables. Ces erreurs mènent à des arbres élémentaires invalides. Un arbre élémentaire est considéré invalide s'il ne satisfait pas une certaine condition linguistique. Nous avons construit des

T est la tête de ses pères. Ici, $H_0 \equiv T$ et H_j est le père de sa tête H_{j+1}.

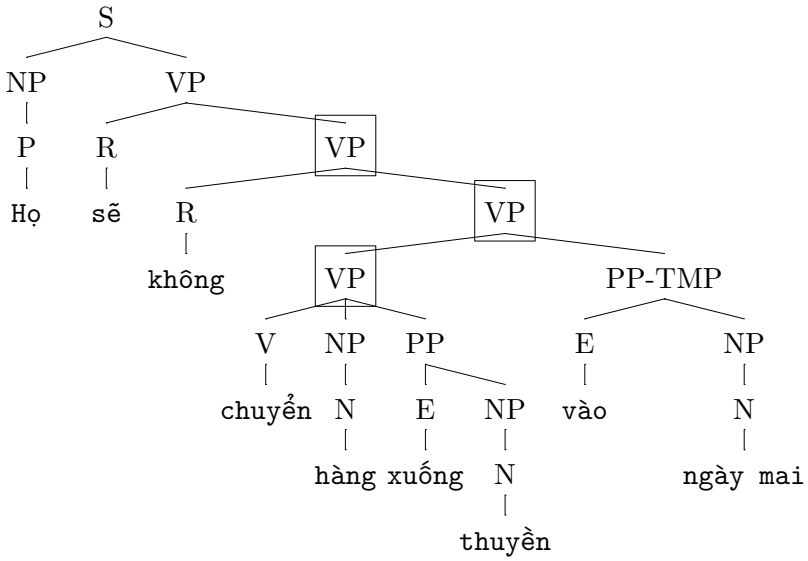


FIGURE 8.8 – L'arbre dérivé de l'arbre dans la Figure 8.7.

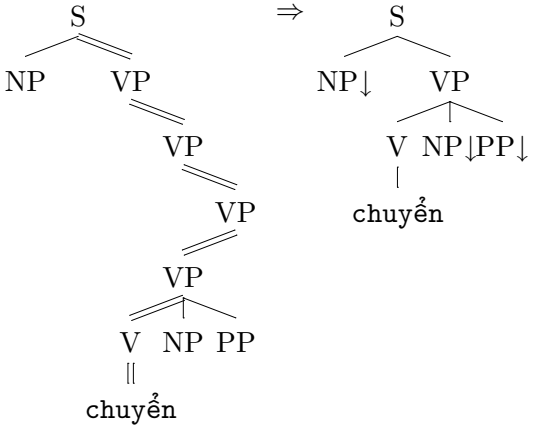


FIGURE 8.9 – Fusion des nœuds de lien pour construire un arbre d'épine.

règles linguistiques pour filtrer les arbres élémentaires invalides. En général, les règles sont classifiées en se basant sur deux types de conditions principaux.

Le premier type de condition aborde la bonne sous-catégorisation des arbres élémentaires. Il s'agit de la position d'une catégorie syntaxique par rapport à une autre. Par exemple, en vietnamien, un adjectif (ou un syntagme adjectival) est toujours placé après un nom (ou un syntagme nominal). Par conséquent, si un arbre élémentaire contient une structure d'ordre incorrect comme `(adjectif nom)`, cet arbre est considéré comme invalide et il est éliminé de l'ensemble des arbres élémentaires.

Le deuxième type de condition porte sur le nombre de compléments qu'une tête lexicale peut prendre. En général, le nombre d'arguments qu'une tête peut prendre ne dépasse pas 4 ; cependant, le système d'extraction donne quelques arbres initiaux dans lesquels il y a

Algorithm 7 BUILD-ELEMENTARY-TREES(T)

Require: T is a derived tree
Ensure: Sets $\mathcal{S}, \mathcal{M}, \mathcal{C}$ of elementary trees.

```
 1: if (not IS-PHRASAL(T)) then
 2:   return ;
 3: end if
 4: {H_0, H_1, ..., H_n} ← HEAD-PATH(T) ;
 5: ok ← false ;
 6: P ← H_0 ;
 7: for j ← 1 to n do
 8:   L ← SISTERS(H_j) ;
 9:   if |L| > 0 then
10:     Rel ← GET-RELATION(H_j, L) ;
11:     if Rel = Coordination then
12:       C ← C ∪ BUILD-CONJ-TREE(P) ;
13:     end if
14:     if Rel = Modification then
15:       M ← M ∪ BUILD-MOD-TREE(P) ;
16:       if j = 1 then
17:         S ← S ∪ BUILD-SPINE-TREE(P) ;
18:         ok ← true ;
19:       end if
20:     end if
21:     if Rel = Argument then
22:       if not ok and not IS-LINK-NODE(P) then
23:         S ← S ∪ BUILD-SPINE-TREE(P) ;
24:         ok ← true ;
25:       end if
26:     end if
27:   else
28:     if not IS-LINK-NODE(P) and IS-PHRASAL(P) then
29:       S ← S ∪ BUILD-SPINE-TREE(P) ;
30:     end if
31:   end if
32:   P ← H_j ;
33: end for
```

plus de cinq nœuds de substitution comme l'arbre de la Figure 8.11. Il est évident que ce n'est pas un bon arbre élémentaire et il est éliminé.

8.3.4 Comparaison avec les travaux existants

Notre approche de l'extraction de grammaires LTAG suit la méthode proposée par Xia [180]. Néanmoins, il y a quelques différences de conception et d'implantation entre nos algorithmes et ceux de Xia.

Algorithm 8 BUILD-SPINE-TREE(T)

Require: T is a derived tree
Ensure: a spine tree
1: $T_c \leftarrow \text{COPY}(T)$;
2: $P \leftarrow T_c$;
3: $H \leftarrow$ `NULL` ;
4: **repeat**
5: $H \leftarrow$ HEAD-CHILD(P) ;
6: $\mathcal{L} \leftarrow$ SISTERS(H) ;
7: **if** $|\mathcal{L}| > 0$ **then**
8: `Rel` $\leftarrow$ GET-RELATION($H, \mathcal{L}$) ;
9: **if** `Rel` = `Argument` **then**
10: **for** $A \in \mathcal{L}$ **do**
11: BUILD-ELEMENTARY-TREES(A) ;
12: A.`kids` $\leftarrow \emptyset$;
13: A.`type` $\leftarrow$ `Substitution` ;
14: **end for**
15: **else**
16: **for** $A \in \mathcal{L}$ **do**
17: P.`kids` $\leftarrow P$.`kids` $\setminus A$;
18: **end for**
19: **end if**
20: **end if**
21: $P \leftarrow H$;
22: **until** (H = `NULL`)
23: **return** MERGE-LINK-NODES(T_c) ;

Algorithm 9 BUILD-MOD-TREE(T)

Require: T is a derived tree
Ensure: a modifier tree
1: $T_c \leftarrow \text{COPY}(T)$;
2: $H \leftarrow$ HEAD-CHILD(T_c) ;
3: H.`kids` $\leftarrow \emptyset$;
4: H.`type` $\leftarrow$ `Foot` ;
5: $M \leftarrow$ MODIFIER(H) ;
6: $T' \leftarrow$ BUILD-SPINE-TREE(M) ;
7: **if** $|M$.`kids`$| > 1$ **then**
8: BUILD-ELEMENTARY-TREES(M) ;
9: **end if**
10: $M \leftarrow T'$;
11: **return** T_c ;

Premièrement, à l'étape de construction des arbres dérivés, nous groupons tout d'abord des groupes de conjonction avant de classifier complètement les arguments et les modifeurs

Algorithm 10 BUILD-CONJ-TREE(T)

Require: T is a derived tree
Ensure: a conjunction tree
1: $T_c \leftarrow \text{COPY}(T)$;
2: $H \leftarrow$ HEAD-CHILD(T_c) ;
3: BUILD-ELEMENTARY-TREES(H) ;
4: $K \leftarrow$ COORDINATOR(H) ;
5: BUILD-ELEMENTARY-TREES(K) ;
6: H.kids $\leftarrow \emptyset$;
7: H.type $\leftarrow$ Foot ;
8: K.kids $\leftarrow \emptyset$;
9: K.type $\leftarrow$ Substitution ;
10: **return** T_c ;

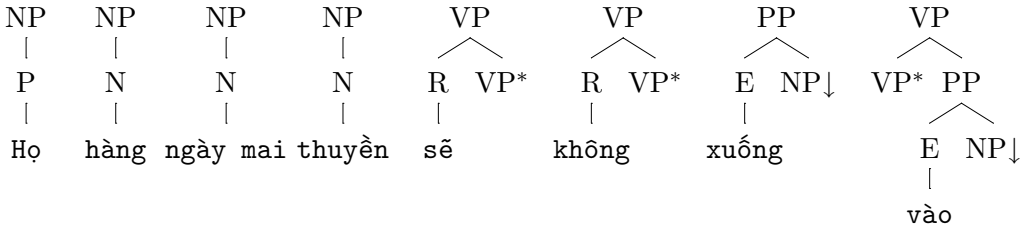


FIGURE 8.10 – Les arbres élémentaires extraits.

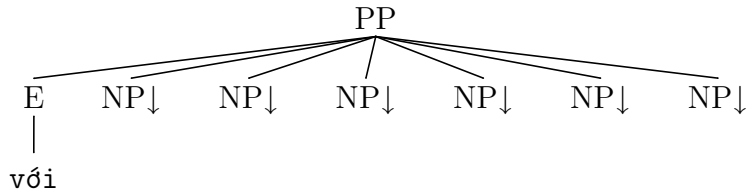


FIGURE 8.11 – Un arbre élémentaire invalide.

de l'arbre résultant. Nous pensons que cette approche est plus facile à compendre et à réaliser car les structures de conjonction sont assez différentes de celles d'argument ou de modifieur.

Deuxièmement, à l'étape de décomposition des arbres élémentaires, nous ne décomposons pas chaque nœud d'un arbre dérivé en deux parties *top* et *bottom* comme cela est effectué dans la méthode de Xia. Dans notre implémentation, les nœuds sont directement copiés pour construire des arbres extraits. En ne découpant pas les nœuds concernés en deux parties, nous améliorons l'efficacité du système (en terme de temps et de mémoire nécessaire) sur de gros corpus.

Troisièmement, le processus d'extraction est découpé en fonctions, chacune construit un type d'arbres élémentaires et elles s'invoquent mutuellement afin de répéter le processus d'extraction pour les sous-arbres dont les racines ne sont pas encore traitées. En dépit de l'utilisation de fonctions récursives, nos algorithmes d'extraction sont conçus de telle sorte

qu'il n'y ait pas de redondance ou de répétition des invocations de fonction : chaque nœud est assuré d'être visité seulement une fois. Une telle application de l'approche « diviser pour régner » de la conception d'algorithmes montre encore une fois son efficacité ; il est également plus facile à optimiser.

8.4 Expérimentations

8.4.1 Corpus arboré vietnamien

Récemment, un groupe de linguistes informaticiens a développé un corpus arboré pour le vietnamien [127], et c'est le premier corpus arboré sur lequel notre système d'extraction de grammaires a été utilisé.

Le développement d'un corpus arboré vietnamien est un sous-projet d'un projet national qui a pour but de construire des resources et des outils de base pour le traitement du vietnamien écrit et parlé [57]. Les textes du corpus sont collectés de la section « Politique – Société » du journal *Tuổi trẻ* (La Jeunesse). Le corpus est divisé en trois parties correspondant à trois niveaux d'annotation : découpage en phrases et en mots, annotation morpho-syntaxique et annotation syntaxique.

Le corpus arboré contient actuellement 10.163 phrases (225.085 mots). La longueur de phrases varie de 2 à 105 mots avec une longueur moyenne de 22. Il y a 9.314 phrases de longueur 40 mots ou moins. Le jeu d'étiquettes du corpus contient 38 catégories syntaxiques (35 catégories syntaxiques et 3 labels pour les catégories vides) et 17 labels fonctionels. Pour plus d'information, veuillez se référer au papier [127].

8.4.2 Résultats

Nous avons appliqué les algorithmes d'extraction sur le corpus arboré vietnamien et extrait deux grammaires LTAG. La première grammaire, G_1, utilise le jeu d'étiquettes originel du corpus. La deuxième grammaire, G_2 utilise un jeu réduit où quelques étiquettes sont fusionnées en une étiquette commune comme présenté dans la Table 8.2. La grammaire G_2 est plus petite que la grammaire G_1, le problème des données rares est donc moins sévère quand la grammaire G_2 est utilisée. De plus, il a été montré que la taille de grammaire est importante pour l'analyse légère de dépendance [58] et pour le super-étiquetage [11, 10].

Nous comptons le nombre d'arbres élémentaires et de schèmes élémentaires [59] de grammaires. Les tailles des deux grammaires sont fournies dans la Table 8.3.

Il y a 15.035 mots uniques dans le corpus et le nombre moyen d'arbres élémentaires qu'un mot ancre est autour de $3,07$. Nous comptons également le nombre de règles hors-contextes des grammaires [60]. Les grammaires G_1 et G_2 ont respectivement 851 et 727 règles hors-

57. Projet « Vietnamese Language and Speech Processing »

58. En anglais : Lightweight Dependency Analysis (LDA).

59. Un schème élémentaire est un arbre élémentaire sans item lexical.

60. Les règles sont simplement lues à partir des schèmes dans les grammaires LTAG. Par exemple, à partir du schème encodant une structure transitive montré dans la Figure 8.12, nous pouvons acquérir deux règles hors-contextes : `S` → `NP VP` et `VP` → `V NP`.

Catégorie	Etiquettes originelles	Etiquettes dans G_2
syntagmes nominaux	NP/WHNP	NP
syntagmes adjectivaux	AP/WHAP	AP
syntagmes adverbiaux	RP/WHRP	RP
syntagmes préposionnels	PP/WHPP	PP
phrases	S/SQ	S

TABLE 8.2 – Quelques étiquettes du corpus vietnamien sont fusionnées en une étiquette commune.

Type	Nombre d'arbres	Nombre de schèmes
G_1	**46.382**	**2.317**
Arbres d'épine	24.973	1.022
Arbres de modifieur	21.309	1.223
Arbres de conjonction	100	72
G_2	**46.102**	**2.113**
Arbres d'épine	24.884	952
Arbres de modifieur	21.121	1.093
Arbre de conjonction	97	68

TABLE 8.3 – Deux grammaires LTAG extraites du corpus arboré vietnamien.

contextes. Ces chiffres nous donnent une seconde vue de la taille des grammaires extraites.

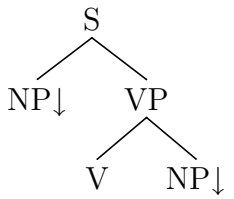


FIGURE 8.12 – Un schème encodant une structure transitive.

Afin d'évaluer la couverture du corpus, nous comptons le nombre de schèmes extraits par rapport à la taille du corpus. La Figure 8.13 ne montre pas de convergence du nombre de schèmes, ceci implique donc qu'il y a plusieurs schèmes inconnus. Ces résultats impliquent également que le corpus arboré actuel n'est pas assez important pour couvrir tous les schèmes grammaticaux de la langue vietnamienne.

8.4.3 Logiciel

Nous avons développé un logiciel appelé `LExtractor` qui implémente les algorithmes présentés pour l'extraction de grammaires LTAG pour le vietnamien. Le logiciel est écrit en Java et indépendant des plateformes. Il est librement distribué sous la licence GNU/GPL[61]. Le logiciel est très efficace en terme de vitesse. A titre d'illustration, l'extraction complète de la grammaire G_1 ne prend que 165 secondes sur un ordinateur de

61. *http ://www.loria.fr/∼lehong/tools/vnLExtractor.php*

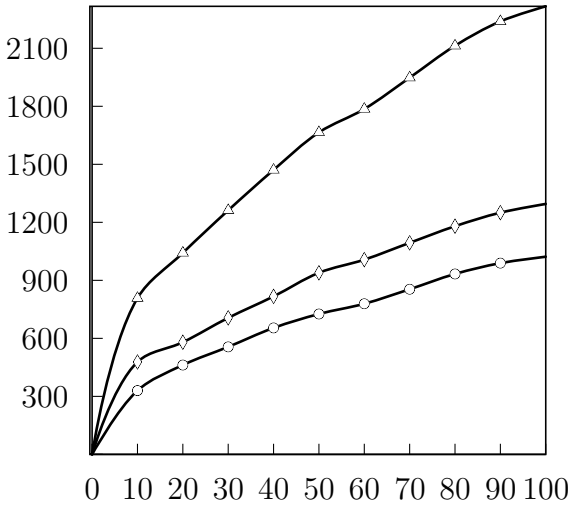


FIGURE 8.13 – L'accroissement des schèmes. L'axe x dénote le pourcentage du corpus utilisé pour l'extraction, l'axe y dénote le nombre de schèmes ($\triangle$), de schèmes initiaux (o) et de schèmes auxiliaires ($\diamond$).

bureau.

L'adaptation du logiciel pour extraire des grammaires LTAG de corpus arborés d'autres langues serait très facile puisque toutes les informations spécifiques des langues sont intentionnellement déplacées en dehors du système d'extraction. Afin d'utiliser le système sur un corpus arboré d'une langue, on a besoin de fournir des informations sur la langue, à savoir un jeu d'étiquettes, une table de percolation de tête et une table d'arguments.

8.5 Bilan

Nous avons présenté dans ce chapitre un système permettant d'extraire automatiquement des grammaires LTAGs à partir de corpus arborés. Le système a été utilisé pour extraire deux grammaires LTAG pour le vietnamien à partir du corpus arboré vietnamien.

Le nombre de schèmes extraits converge lentement par rapport à la taille du corpus utilisé. Ceci signifie qu'il y a plusieurs schèmes qui ne sont pas représentés dans le corpus arboré vietnamien.

Une limite de notre système d'extraction des grammaires TAG actuel concerne la construction des structures de traits pour les arbres élémentaires. En effet, le système ne peut pas produire des grammaires LTAG à structures de traits – les nœuds des arbres extraits ne sont pas décorés par des paires de structures de traits morpho-syntaxiques. Toutefois, le système actuel est suffisant pour construire une grammaire LTAG pour le vietnamien, sachant que le vietnamien est une langue isolante où il n'y pas de flexion, le phénomène original qui mène à une large utilisation des structures de traits dans les grammaires des langues occidentales. Nous envisageons de développer encore notre système pour qu'il puisse apprendre des grammaires LTAG à structures de traits afin de le rendre plus convenable pour les langues flexionnelles.

Dans nos travaux futurs, nous voulons expérimenter l'extraction de grammaires LTAG pour le français à partir d'un corpus arboré du français développé à l'Université Paris 7 [4]. Nous projetons également de réaliser une comparaison quantitative des structures syntaxiques du français et de celles du vietnamien. Nous pensons que la comparaison de deux grammaires pourrait révéler des relations intéressantes entre les deux langues car, pour des raisons historiques, la langue vietnamienne a été enrichie non seulement dans son vocabulaire mais également dans sa syntaxe par l'influence de la grammaire française.

9

Analyse syntaxique du vietnamien

Sommaire

Nous avons développé dans les chapitres précédents une chaîne modulaire de prétraitements pour le vietnamien dont le rôle est d'appliquer à des corpus bruts une cascade de traitements de surface qui est préalablement nécessaire à une possible analyse syntaxique. Nous présentons dans ce chapitre la construction et l'évaluation d'un analyseur syntaxique pour le vietnamien.

Le chapitre est organisé comme suit. Dans un premier temps, nous présentons le développement d'un analyseur syntaxique profond à base de grammaire d'arbres adjoints pour le vietnamien. Dans un deuxième temps, la section 9.2 aborde deux modes différents de représentation syntaxique d'une phrase : l'analyse en constituants et l'analyse en dépendances. Puis, nous présentons dans la section 9.3 la construction d'un schéma d'annotation en dépendances ainsi qu'un algorithme d'extraction des dépendances syntaxiques à partir des arbres de dérivations fournis en sortie de l'analyseur. Ensuite, nous discutons des différentes procédures d'évaluation de l'analyse syntaxique dans section 9.4. Enfin, la section 9.5 présente les évaluations de la performance de notre système d'analyse sur un

corpus de test ainsi que des discussions autour des résultats obtenus, afin de conclure le chapitre en section 9.6.

9.1 Introduction

Nous présentons dans ce chapitre la construction d'un analyseur syntaxique profond à base de grammaire LTAG pour le vietnamien. Avec la chaîne de prétraitements de textes et la grammaire LTAG à large couverture présentées dans les chapitres précédents, ce travail complète la construction d'un composant syntaxique pour le vietnamien – le sujet et l'objectif principal de notre travail de thèse. Nous résumons dans la Figure 9.1 la chaîne de traitements du composant syntaxique.

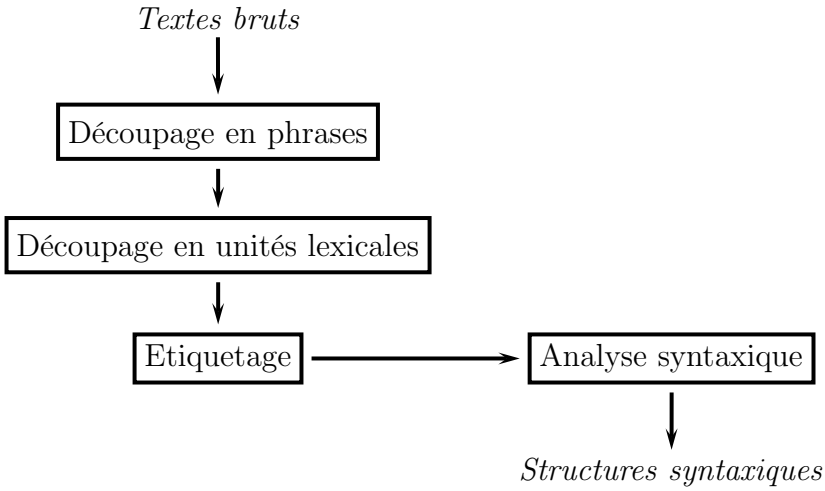


FIGURE 9.1 – La chaîne de traitements de textes vietnamiens

Nous avons vu dans le chapitre 7 que le formalisme TAG présente des propriétés intéressantes pour les traitements informatiques, notamment celle de la séparation entre données et programmes, ce qui permet d'utiliser le même analyseur pour des langues différentes. Cette séparation nous permet durant le travail de la thèse, de nous concentrer sur la construction de ressources linguistiques et le développement de prétraitements, sans avoir besoin de développer de nouveaux algorithmes d'analyse syntaxique.

Nous avons adapté et enrichi l'analyseur syntaxique LLP2[62] [153] développé au sein du laboratoire LORIA pour construire un analyseur syntaxique profond pour le vietnamien. L'analyseur LLP2 est originellement développé pour l'analyse syntaxique profonde du français. L'algorithme implémenté est celui de l'analyse par connexité décrit dans [110]. L'intégration d'un module de traitement de structures de traits et d'unification permet de prendre en compte les traits *top* et *bottom* aux nœuds des TAG. En d'autres termes, LLP2 a la capacité de traiter des TAG à structures de traits. Cependant, la version actuelle de LLP2 ne permet pas de prendre en compte les arbres auxiliaires décrivant des adjonctions englobantes (wrapping adjunction). Par conséquent, formellement, l'analyse est restreinte aux grammaires d'arbres insérés (TIG [158]). La grammaire LTAG du vietnamien que

62. LORIA LTAG Parser, version 2

nous avons élaborée ne contient pas d'adjonctions englobantes[63], l'analyseur LLP2 est donc suffisant pour effectuer des expérimentations de l'analyse syntaxique du vietnamien.

Nous avons adapté l'analyseur LLP2 en introduisant des interfaces générales à plusieurs modules de prétraitements de l'analyseur afin d'intégrer notre chaîne de prétraitements préalables d'une analyse, notamment les modules d'étiquetage et de découpage des phrases en unités lexicales. Nous pouvons constater que LLP2 est à l'heure actuelle un analyseur syntaxique TAG général, indépendant des langues.

Nous avons également enrichi LLP2 en ajoutant un module supplémentaire qui extrait une analyse en dépendances à partir de l'analyse donnée comme le résultat de LLP2. Il s'agit donc d'un analyseur syntaxique pour le vietnamien à la fois en constituants et en dépendances. L'intérêt de ce travail est l'avantage de la représentation en dépendances pour l'évaluation de la performance d'un analyseur syntaxique. Si les constituants permettent d'exprimer des générations structurales évidentes, les dépendances ont l'avantage de permettre une extraction plus directe des structures argumentales. Les structures de dépendances constituent également un format linguistique plus avantageux pour l'évaluation de la tâche d'analyse syntaxique.

Nous présentons dans la section suivante les deux modes de représentation de l'analyse syntaxique.

9.2 Analyse en constituants et en dépendances

La structure syntagmatique et la structure de dépendances sont deux modes différents de la représentation syntaxique d'une phrase. Tandis qu'une analyse en constituants représente l'emboîtement de constituants multi-mots, une analyse en dépendances représente les dépendances entre des mots individuels.

La dépendance syntaxique représente le fait que la présence d'un mot est légitimée par un autre mot qui est son gouverneur[64]. De plus, il s'agit de dépendances de surface, c'est-à-dire de relations entre formes fléchies, où toute forme fléchie, même sémantiquement vide, est représentée, et a un et un seul gouverneur, sauf pour la forme tête de la phrase[65] [22].

Une analyse en dépendances *typées* ajoute les étiquettes de relations grammaticales aux dépendances, comme *sujet* ou *objet indirect*.

Récemment, il y a eu de nombreux travaux sur l'analyse syntaxique en dépendances pour des langues bien-étudiées comme l'anglais [76, 135, 118, 91] et le français [22, 142, 124]. Les analyseurs de dépendances développés pour ces langues sont souvent des analyseurs statistiques qui sont entraînés sur des corpus disponibles pour la langue concernée. Nous pouvons classer l'architecture de ces analyseurs en deux types principaux :
– Les analyseurs qui emploient une méthode d'apprentissage statistique sur des corpus

63. Les arbres auxiliaires de la grammaire ne comprennent que des nœuds pieds qui se trouvent à gauche ou à droite des arbres.

64. Voir [82] pour un historique de la représentation en dépendances.

65. Pour obtenir que tout mot a exactement un gouverneur, on ajoute un nœud `root` comme gouverneur de la tête.

de dépendances extraits automatiquement à partir de corpus arborés de la langue et produisent directement des analyses en dépendances [76, 135, 21] ;
– Les analyseurs qui reposent sur un processus séquentiel dans lequel on produit l'analyse en constituants d'abord et l'extraction de dépendances ensuite [22, 51].

Il est évident que dans la dernière architecture, nous avons besoin d'un module qui prend en entrée les sorties en constituants de l'analyseur et les convertit en dépendances typées comme illustré en Figure 9.2 (l'exemple est extrait du papier [21]).

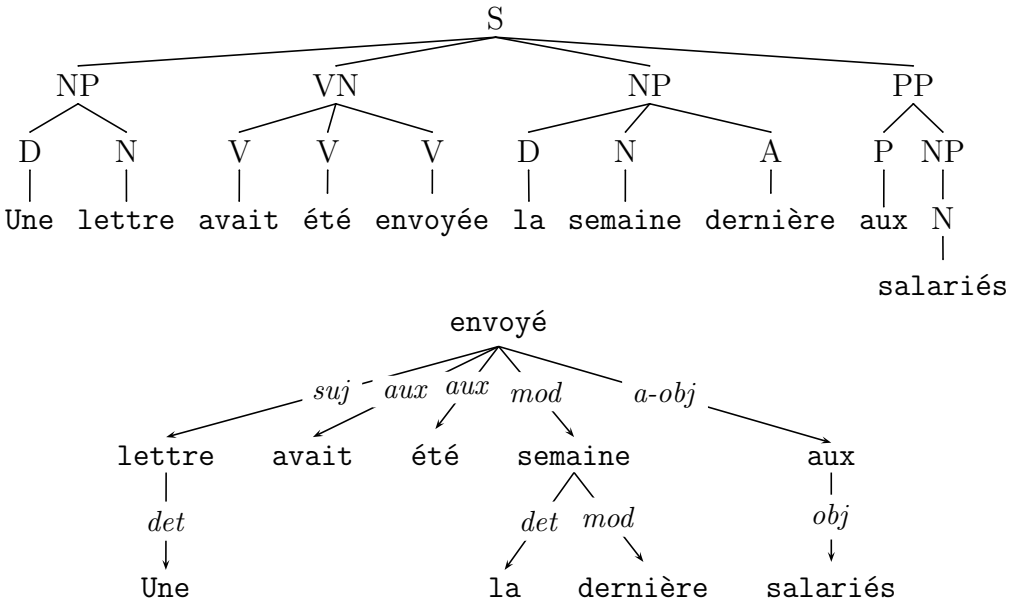


FIGURE 9.2 – L'analyse en constituants et en dépendances

9.3 Extraction d'analyses en dépendances

Avant de présenter l'extraction d'une analyse en dépendances à partir d'une analyse en constituants, nous abordons dans la section qui suit les schémas d'annotation en dépendances existants pour l'anglais et le français et le schéma que nous avons conçu pour encoder des relations grammaticales pour le vietnamien.

9.3.1 Schémas d'annotation en dépendances

Il existe de multiples schémas d'annotation en dépendances, comme le schéma EASy [141], ou des standards internationaux tels le schéma GR [25], défini comme un format plus adapté à l'évaluation d'analyseurs, le schéma Stanford Dependency (SD) [51] issu d'une conversion automatique du corpus arboré Penn Treebank ou le schéma PARC 700 [88] inspiré des structures fonctionnelles de la grammaire lexicale fonctionnelle. La multiplicité de ces schémas d'annotation tient pour partie à des choix linguistiques et pratiques différents, notamment sur le caractère surfaciel ou pas des dépendances. Par exemple, les relations de contrôle sont censées être encodées pour les quatre formats cités, mais les dépendances sur des mots sémantiquement vides n'apparaissent que pour GR et SD [22].

Nous préférons définir un schéma d'annotation uniquement en dépendances de surface

dont le format soit un pivot convertible vers les différents standards cités, et un pivot enrichissable ou convertible en dépendances plus profondes. La représentation actuelle contient 13 relations grammaticales représentant les dépendances correspondantes. Toutes les dépendances sont des relations binaires. Dans les paragraphes qui suivent, nous définissons ces dépendances. Les définitions utilisent les catégories syntaxiques du corpus arboré vietnamien [127], qui ont été également présentées dans le chapitre 6 (*cf.* section 6.3).

Arg

La relation `arg` représente la relation d'argument entre un mot et son argument. Il y a deux types d'argument : le sujet et l'objet. Il s'agit donc de la relation `subj` ou de la relation `obj` si l'argument est respectivement le sujet ou l'objet du mot concerné. Par exemple :

– `arg(nhớ,người yêu)` ; `arg(nhớ,cô)` dans « `Cô ấy nhớ người yêu` » (Son petit ami lui manque)
– `arg(còn,nhà)` ; `arg(còn,gạo)` dans « `Nhà còn gạo` » (Il reste encore du riz)

Comme nous l'avons déjà vu au chapitre 2, le vietnamien ne comporte pas de variation morphologique. La caractérisation fonctionnelle des verbes sert de base pour décrire les structures syntaxiques des phrases. Dans la plupart des cas, l'ordre des composants des phrases est sujet–verbe–objet comme nous l'avons donné dans des exemples ci-dessus. Pourtant, on rencontre assez fréquemment en vietnamien un type de phrases dont le sujet agentif est absent, et le sujet grammatical est le complément d'objet du prédicat verbal. [66] Par exemple :

– Sous-catégorisation *sujet-verbe-objet* : « `Người ta đang xây bảo tàng` » (On est en train de construire un musée). Ici, `người ta=on`, `xây=construire`, `bảo tàng=musée` et `đang` est un adverbe de temps signifiant le présent.
– Phrases du type *objet-verbe* :
  – « `Bảo tàng đang xây` » (Le musée est en train d'être construit).
  – « `Bảo tàng xây rồi` » (Le musée a été construit).
  – « `Bảo tàng xây cạnh trường tôi` » (Le musée est construit à côté de mon école).
– On rencontre également des phrases du type *verbe-sujet*, par exemple « `Sinh ra cái mặt tôi là giời` » (C'est dieu qui me prend naissance), « `Từ đằng cuối bãi tiến lại hai cậu bé` » (Les deux garçons sont venus de la fin de la berge) [132].

Nous avons vu également que le vietnamien est une langue avec préférence du thème où le sujet d'une phrase ne peut pas être identifié en s'appuyant sur sa position dans la phrase ni sur la morphologie (*cf.* section 2.4.2). Pour ces raisons, la distinction entre le sujet et l'objet d'une phrase vietnamienne n'est pas une tâche évidente, notamment dans un processus de distinction automatique. Nous ne distinguons donc pas, pour l'instant, les deux relations `subj` et `obj` dans les évaluations. Nous utilisons la relation générale `arg` pour représenter ces relations.

66. Dans ces cas, le locuteur n'aborde pas l'agent de l'action, mais insiste sur les informations concernant le temps, la manière ou l'aspect de l'action. Pour cette raison, le prédicat verbal de ces phrases doit être modifié par un complément circonstanciel [128].

Mod

Les relations de modifieur représentent la modification d'un mot qui est son gouverneur. Selon les catégories syntaxiques du modifieur, nous distingons les relations de modifieur suivantes.

1. `modN` : relation de modifieur nominal. Il s'agit souvent d'un nom commun (concret ou abstrait) qui modifie un nom individuel. Par exemple :
   – `modN(chiếc,kẹo)` [67] dans « `Tất cả những chiếc kẹo ấy` » (Tous les bonbons)
   – `modN(quả,táo)` dans « `Một quả táo` » (Une pomme)
2. `modM` : relation de modifieur numéral. Il s'agit d'un numéral qui quantifie un nom. Par exemple :
   – `modM(quả,một)` dans « `Một quả táo` » (Une pomme)
   – `modM(người,mười)` dans « `Mười người` » (Dix personnes)
3. `modA` : relation de modifieur adjectival. Il s'agit d'une relation dans laquelle un mot (typiquement un nom) est modifié par un adjectif. Par exemple :
   – `modA(cô gái, trẻ)` dans « `Một cô gái trẻ ra mở cửa` » (Une jeune fille ouvre la porte)
4. `modR` : relation de modifieur adverbial. Il s'agit d'une relation dans laquelle un adverbe modifie son gouverneur (typiquement un verbe ou un adjectif). Par exemple :
   – `modR(đến,chưa)` dans « `Nam chưa đến` » (Nam n'est pas encore arrivé).
   – `modR(đẹp,rất)` dans « `Cô ấy rất đẹp` » (Elle est très belle)
   – `modR(ăn,đã)`, `modR(ăn,rồi)` dans « `Tôi đã ăn tối rồi` » (J'ai déjà dîné)
5. `modE` : relation de modifieur prépositionnel. Il s'agit d'une relation entre une préposition et un verbe. Par exemple :
   – `modE(chuyển,xuống)` dans « `Họ sẽ chuyển hàng xuống thuyền` » (Ils livreront les marchandises au bateau)
6. `modV` : relation de modifieur verbal. Il s'agit d'une relation entre un verbe et un verbe. Par exemple :
   – `modV(được,khen)` dans « `Nó được khen` » (Il est félicité)
   – `modV(đi,buôn lậu)` dans « `Chúng đi buôn lậu` » (Ils font le commerce de contrebande)
7. `modL` : relation de modifieur déterminant. Il s'agit d'une relation entre la tête d'un groupe nominal et son déterminant. Par exemple :
   – `modL(ngôi,những)` dans « `Kia là những ngôi nhà ngói đỏ` » (Ce sont des maisons au toit rouge)
8. `modP` : relation de modifieur pronominal. Il s'agit d'une relation entre la tête d'un groupe nominal et un pronom indicatif. Par exemple :
   – `modP(cuốn,ấy)` dans « `Cuốn sách ấy hay` » (Ce livre est intéressant)
   – `modP(chỗ,này)` dans « `Chỗ này là đồn bót xưa` » (Cette place est un vieux poste militaire).
9. `modC` : relation de modifieur de coordination subordonnée. Il s'agit d'une relation entre la tête d'un groupe et une coordination subordonnée. Par exemple :

67. L'ordre des arguments est le suivant (gouverneur,gouverné).

– modC(nên,phải) dans « Nên anh phải làm gương » (Donc vous devez être un bon exemple)
– modC(vậy mà,lên) dans « Vậy mà cây vẫn lên xanh » (Pourtant, les plantes sont bien venantes).

Coord

La relation coord représente la dépendance entre chaque tête lexicale de deux groupes coordonnés et la conjonction. Donc, pour chaque groupe de conjonction, on crée deux relations coord. Par exemple :
– coord(và,đến), coord(và,chuyển) dans « Họ đã đến và họ sẽ chuyển hàng xuống thuyền vào ngày mai » (Ils sont arrivés et ils livreront les marchandises au bateau demain).
Nous passons maintenant à l'extraction d'analyses en dépendances à partir des arbres de dérivation TAG.

9.3.2 Algorithme d'extraction des analyses en dépendances

Il a été montré que le formalisme des grammaires d'arbres adjoints partagent plusieurs similarités importantes avec le formalisme de grammaires de dépendances [147]. Les arbres de dérivation TAG se convertissent facilement en arbres de dépendances dans le cas de grammaires lexicalisées. L'idée de base est de transformer chaque étape de dérivation (symbolisée par un arc dans l'arbre de dérivation) en une dépendance. Une opération de dérivation entre un arbre source t_1 et un arbre cible t_2 se traduit par une dépendance entre la tête de t_1 comme gouverneur et la tête de t_2 comme gouverné.

Par exemple, la Figure 9.3 montre l'analyse correcte de la phrase « Giang cho tôi một quả cam » (Giang m'a donné une orange). La partie en haut de la figure donne des arbres élémentaires pour l'analyse ; la partie en bas montre l'arbre de dérivation et son arbre dérivé correspondant où la notation <ancre> signifie l'arbre élémentaire correspondant de l'ancre lexicale ancre.

Ainsi, l'arbre de dérivation devient l'arbre de dépendances si l'on effectue une transformation simple où chaque nœud de l'arbre de dérivation est remplacé par le nœud lexical qui lui est rattaché. On obtient donc l'arbre de dépendances montré dans la Figure 9.4.

Ici, on veut extraire des dépendances typées dont chaque arc est étiqueté par un type de relation selon le schéma d'annotation présenté dans la sous-section précédente. On doit alors considérer le type d'opération effectuée à chaque nœud de l'arbre de dérivation. Si c'est une substitution, une relation arg sera créée ; si c'est une adjonction, une relation de modifieur sera créée, son étiquette peut être déterminée en considérant la catégorie syntaxique du mot qui se trouve au nœud lexical concerné de l'arbre de dérivation.

Le cas le plus difficile est la construction de relations de coordination où l'on devrait considérer trois nœuds concernés et les deux opérations de combinaison en même temps. Nous rappelons qu'un arbre auxiliaire de conjonction a une forme spécifique ayant un nœud de substitution et un nœud pied comme les arbres exemples suivants.

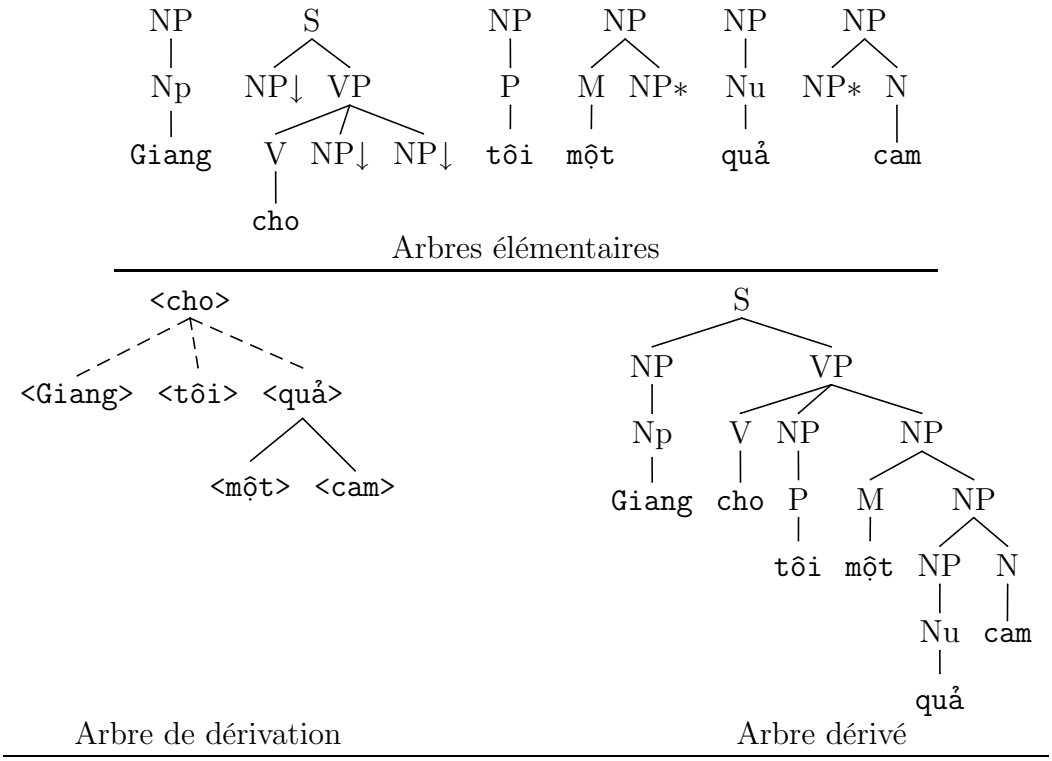


FIGURE 9.3 – Une analyse de la phrase « `Giang cho tôi một quả cam` »

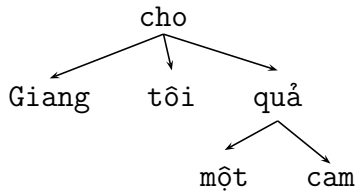
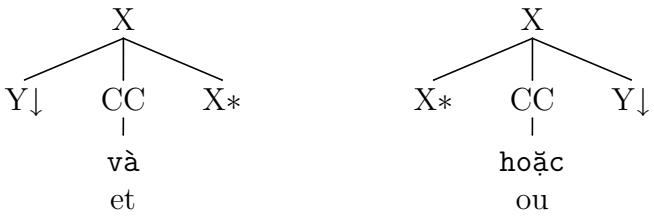


FIGURE 9.4 – Arbre de dépendance correspondant à l'analyse dans Figure 9.3



Nous avons développé un algorithme pour l'extraction automatique des analyses en dépendances à partir des arbres de dérivation donnés par l'analyseur en constituances LLP2. L'Algorithme 11 montre la procédure d'extraction en détail. Cet algorithme utilise quelques fonctions supplémentaires comme suit. La fonction LEXICAL-NODE(N) retourne la tête lexicale d'un nœud N de l'arbre de dérivation de l'entrée, tandis que la fonction POS-NODE(N) retourne la catégorie syntaxique préterminal (*part-of-speech*) de la tête lexicale. Les fonctions IS-SUBST() et IS-ADJ() sont invoquées à chaque nœud de l'arbre de dérivation pour vérifier s'il s'agit d'une substitution ou d'une adjonction qui a été ef-

Algorithm 11 EXTRACT-RELATIONS(N)

Require: A derivation tree N
Ensure: a set $\mathcal{R}$ of dependency relations

```
1: w_n ← LEXICAL-NODE(N) ;
2: t_n ← POS-NODE(N) ;
3: for K ∈ N.kids do
4:    w_k ← LEXICAL-NODE(K) ;
5:    t_k ← POS-NODE(K) ;
6:    if K.IS-SUBST() then
7:       if t_n = CC then
8:          R ← R ∪ NEW-RELATION(coord, w_n, w_k) ;
9:       else
10:         R ← R ∪ NEW-RELATION(arg, w_n, w_k) ;
11:      end if
12:   else if K.IS-ADJ() then
13:      if t_k ∈ {A, N, R, V, E, L, M, P, C} then
14:         R ← R ∪ NEW-RELATION(modt_k, w_n, w_k) ;
15:      end if
16:      if t_k = CC then
17:         R ← R ∪ NEW-RELATION(coord, w_k, w_n) ;
18:      end if
19:   end if
20:   {Recursively extract relations from tree K}
21:   EXTRACT-RELATIONS(K) ;
22: end for
23: return R ;
```

fectuée à ce nœud. Enfin, la fonction NEW-RELATION(type,w_1, w_2) crée et retourne une nouvelle relation de type `type` entre deux unités lexicales w_1 et w_2.

Par exemple, l'application de l'Algorithme 11 sur l'arbre de dérivation de l'analyse en Figure 9.3 nous donne les 5 relations suivantes : `arg(cho,Giang)`, `arg(cho,tôi)`, `arg(cho,quả)`, `modM(quả,một)`, `modN(quả,cam)`.

Les représentations de l'analyse syntaxique en constituants et en dépendances sont utilisées dans les différentes procédures d'évaluation d'analyses syntaxiques présentées dans la section suivante.

9.4 Différentes procédures d'évaluation d'analyses syntaxiques

En plus des résultats intrinsèques fournis par un analyseur, l'évaluation quantitative des analyseurs syntaxiques joue un rôle essentiel dans un composant syntaxique et ses applications. Une évaluation quantitative des analyseurs syntaxiques est alors nécessaire pour évaluer leur efficacité face à un type particulier de phrases ou encore face à un phénomène

linguistique particulier.

De nombreux protocoles d'évaluation ont été proposés. Parmis ces différentes procédures d'évaluation des analyseurs syntaxiques, on peut distinguer trois approches différentes [121] :

1. Approche fondée sur la notion de « compétences linguistiques » où l'évaluation repose sur un ensemble de phrases traitant un, voire plusieurs phénomènes linguistiques. Dans ce cadre, l'évaluation repose sur le nombre de phénomènes correctement traités voire le nombre d'analyses retournées.
2. Approche fondée sur la notion de « corpus non annoté » où l'évaluation repose sur le nombre d'analyses retournées sans considérer le fait que cette analyse soit ou non correcte.
3. Approche fondée sur la notion de « corpus annoté » où cette fois-ci la qualité de l'analyse retournée est prise en compte.

En ce qui concerne la première approche, l'existence d'un ensemble de phrases de test est la question cruciale. Cet ensemble sert à évaluer la capacité d'un analyseur sur des phénomènes linguistiques différents. Notons que des phrases agrammaticales appartiennent également à l'ensemble de phrases de test car elles permettent de contrôler la surgénération d'une grammaire.

La réutilisabilité et la spécificité d'un ensemble de phrases de test font partie de l'objectif du projet TSNLP (*Test Suites for Natural Language Processing*) de la communauté européenne [106]. Ce projet a proposé une méthodologie pour la construction systématique et progressive de phrases de test, ainsi qu'un schéma d'annotation détaillé et neutre tant vis-à-vis des théories linguistiques spécifiques que des types d'applications particuliers.

En particulier pour le français, un protocole d'évaluation des analyseurs syntaxiques a été récemment proposé dans le cadre d'une campagne d'évaluation des analyseurs syntaxiques du projet PASSAGE [49] qui est en continuité avec la campagne d'évaluation EASY [142, 141] du projet EVALDA. [68] Dans le cadre de ces projets, 4.000 phrases de test ont été construites comme données de référence pour évaluer les analyseurs participants.

Pourtant, la construction d'un ensemble de phrases de test pour une langue est une tâche très coûteuse qui requiert des années de travail continu de plusieurs experts linguistes informaticiens, ce qui dépasse le cadre de notre travail de thèse.

La deuxième approche d'évaluation propose une évaluation très brute des analyseurs syntaxiques car elle ne repose que sur le nombre d'analyses retournées, correctes ou non. Nous avons quand même réalisé une telle évaluation pour notre analyseur. En utilisant une grammaire d'arbres adjoints de taille moyenne qui contient 767 arbres élémentaires (165 schèmes) pour analyser 100 phrases de longueur de 15 mots ou moins [69], l'analyseur nous donne une moyenne de $49,6$ analyses (en constituances) pour une phrase. Il y a

68. En fait, l'objectif principal du projet PASSAGE est de développer un corpus arboré de taille importante pour le français (environ 100 millions mots) en utilisant les sorties de 10 analyseurs syntaxiques différents du français.

69. Ces phrases sont tirées du corpus arboré du vietnamien.

seulement 14 phrases ayant une analyse unique. Cette expérimentation montre une ambiguïté très élevée de l'analyse syntaxique du vietnamien. Nous allons également justifier cette ambiguïté de manière plus détaillée en section 9.5.

Dans la troisième approche, on compare les sorties de l'analyseur évalué sur la base d'un corpus annoté, c'est-à-dire que la qualité de l'analyse retournée est prise en compte. Cette approche a été employée pour obtenir une bonne évaluation d'analyseurs syntaxiques différents. Plusieurs procédures fondées sur cette approche ont été proposées ces dernières années, notamment le projet PARSEVAL [14] qui a été parmi les premiers à proposer une telle évaluation. Même si pendant longtemps cette procédure a été considérée comme une bonne base pour l'évaluation, elle a été remise en cause ces dernières années. En effet, à l'origine elle ne permettait pas de donner une image précise de la qualité des sorties des analyseurs car l'évaluation se faisait sur la délimitation des constituants et non sur les étiquettes associées à ceux-ci. Rapidement, d'autres évaluations ont été menées dans ce projet PARSEVAL prenant en compte la qualité des informations retournées par le biais des mesures de précision et de rappel. On note que ces mesures peuvent être effectuées sur des analyses partielles, qui permet de voir le résultat d'une analyse de manière plus détaillée.

Cette procédure a servi de base à de nombreuses autres procédures comme l'évaluation proposée par Lin [109] fondée sur l'évaluation des dépendances ou encore l'évaluation plus complète proposée par Caroll *et al.* [25] et celle proposée par Briscoe *et al.* [77] fondée à la fois sur l'évaluation des constituants et des relations grammaticales. Dans le cadre du projet PASSAGE [49], les analyseurs syntaxiques différents du français ont été également évalués à la fois sur leur capacité d'analyse en constituants et en dépendances.

A l'étude de ces différentes procédures d'évaluation, il est évident que la comparaison des sorties d'un analyseur avec un corpus annoté permet de tester la qualité des informations qu'il retourne. L'objectif de l'évaluation est non seulement de permettre de tester la pertinence de l'analyse retournée mais également de l'évaluer sur les différents phénomènes linguistiques rencontrés. A l'heure actuelle, on a vu l'émergence d'un consensus sur les procédures d'évaluation : une bonne procédure d'évaluation doit permettre non seulement d'évaluer des analyseurs fondés sur la reconnaissance de constituants mais également des analyseurs fondés sur l'extraction de relations syntaxiques. Par conséquent, le formalisme d'annotation doit être suffisamment souple pour permettre l'appariement de n'importe quelle analyse dans ce formalisme [121].

9.5 Expérimentations

Nous présentons dans cette section les expériences de notre travail sur l'analyse syntaxique mené sur un corpus arboré vietnamien.

9.5.1 Mesures d'évaluation des résultats

Nous avons constaté dans la section précédente que le vietnamien est une langue à l'ambiguïté syntaxique élevée, chaque phrase peut avoir de nombreuses analyses syntaxiques possibles (correctes ou incorrectes).

La première question que l'on peut se poser est « comment doit-on évaluer les résultats ? » ; plus précisément, comment peut-on comparer des sorties de l'analyseur avec un corpus annoté. Notons qu'il s'agit à la fois de l'évaluation d'analyses en constituants et de celle d'analyses en dépendances puisque pour chaque analyse en constituants, il y a exactement une analyse en dépendances qui est extraite automatiquement.

A titre d'illustration, nous reprenons la phrase exemple présentée précédemment dont l'analyse correcte est montrée dans la Figure 9.3. Outre cette analyse, l'analyseur produit deux autres analyses montrées dans les Figure 9.5 et Figure 9.6.

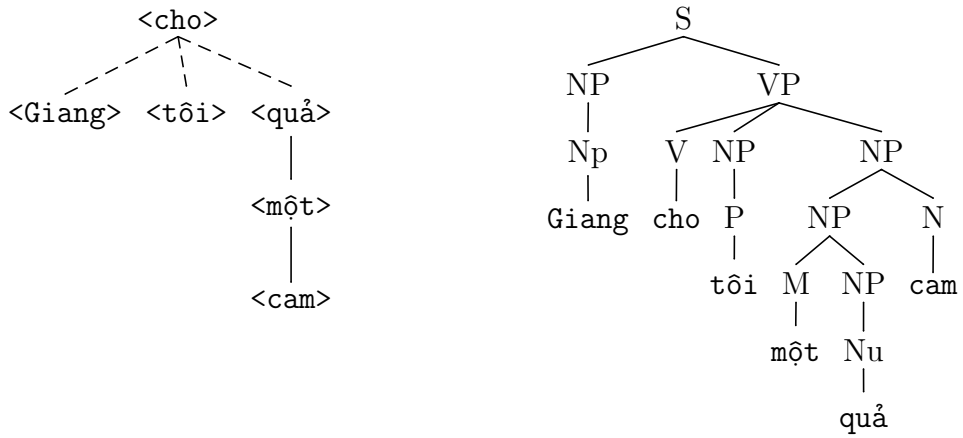


FIGURE 9.5 – Deuxième analyse de la phrase exemple

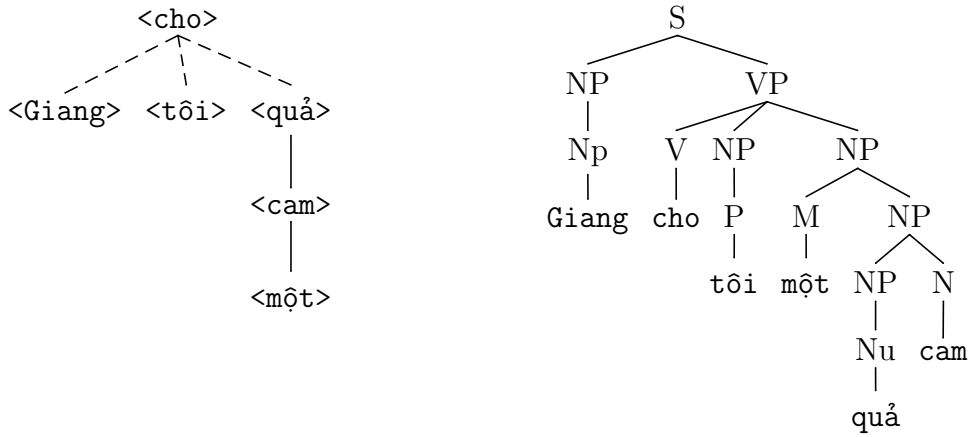


FIGURE 9.6 – Troisième analyse de la phrase exemple

Les relations syntaxiques correspondant à ces deux analyses en constituants sont produites dans les deux premières colonnes de la Table 9.1, avec les relations syntaxiques de référence. Il est intéressant de noter que dans la troisième analyse, on a le même arbre dérivé que celui de l'analyse correcte, tandis que l'arbre de dérivation est différent de celui de l'analyse de référence.

Le résultat de l'analyse syntaxique est évalué par deux mesures : *la précision par arbre* (ou T-précision) et la *précision par dépendance* (D-précision). Etant donné un corpus C

Ensemble 2	Ensemble 3	Ensemble de référence
arg(cho,Giang)	arg(cho,Giang)	arg(cho,Giang)
arg(cho,tôi)	arg(cho,tôi)	arg(cho,tôi)
arg(cho,quả)	arg(cho,quả)	arg(cho,quả)
modM(quả,một)	modN(quả,cam)	modM(quả,một)
modN(một,cam)	modM(cam,một)	modN(quả,cam)

TABLE 9.1 – Ensembles différents des relations syntaxiques de la phrase

et une grammaire G, la précision par arbre est le nombre de phrases de C pour lesquelles l'arbre correct a été construit par l'analyseur divisé par le nombre de phrases de C [70].

La T-précision est une mesure assez grossière, car elle ne permet pas de voir si un résultat est partiellement correct. Nous utilisons également des mesures plus précises des résultats d'une analyse syntaxique définies pour les arbres syntagmatiques des campagnes PARSEVAL. Ces dernières distinguent *précision*, *rappel* et *croisement*. Etant donné un arbre syntagmatique d'une phrase produit par l'analyseur, que l'on appellera arbre *hypothèse*, et l'arbre correct de cette phrase, appelé arbre *référence*, les trois mesures vont comparer les syntagmes de l'hypothèse et de la référence. Un syntagme est défini par son étiquette et par son extension, c'est-à-dire les mots qu'il regroupe. La précision d'une hypothèse est définie comme le nombre de syntagmes de la référence présents dans l'hypothèse. Le rappel est le nombre de syntagmes de l'hypothèse présents dans la référence. Le croisement est le nombre de syntagmes de l'hypothèse qui croisent des syntagmes de la référence ($[_H\,[_R\,]_H\,]_R$).

Lorsqu'il y a plusieurs arbres d'analyse pour une phrase (comme c'est souvent le cas), nous choisissons *l'arbre de dérivation dont l'arbre dérivé comprend le plus petit nombre de nœuds* car cette analyse correspond à l'arbre le plus spécifique [71].

La comparaison d'arbres de dépendances est plus simple que la comparaison d'arbres syntagmatiques, du fait que tous les arbres de dépendances d'une même phrase comportent le même nombre de dépendances ($n-1$ dépendances pour une phrase composée de n mots). Etant donné un arbre de dépendances hypothèse et un arbre de dépendances référence, on définit la D-précision de l'hypothèse comme le rapport du nombre de dépendances de l'hypothèse présentes dans la référence sur le nombre total de dépendances de l'hypothèse (ou de la référence car les deux sont égaux).

9.5.2 Corpus

Les expériences décrites dans cette section ont été réalisées sur un sous ensemble du corpus arboré vietnamien [127] présenté brièvement dans la sous-section 8.4.1 du chapitre précédent.

Le corpus arboré contient actuellement 10.163 phrases (225.085 mots). La longueur moyenne des phrases du corpus est de $22,14$ mots. Il y a 9.314 phrases de longueur 40 mots ou

70. On ne compte pas les analyses échouées.
71. En cas d'égalités avec ce critère, la première analyse retournée par l'analyseur est choisie.

moins[72] (91, 65% du corpus). La répartition des phrases selon leur longueur est représentée dans la Figure 9.7. On voit que la plupart des phrases ont une longueur entre 10 et 30 mots.

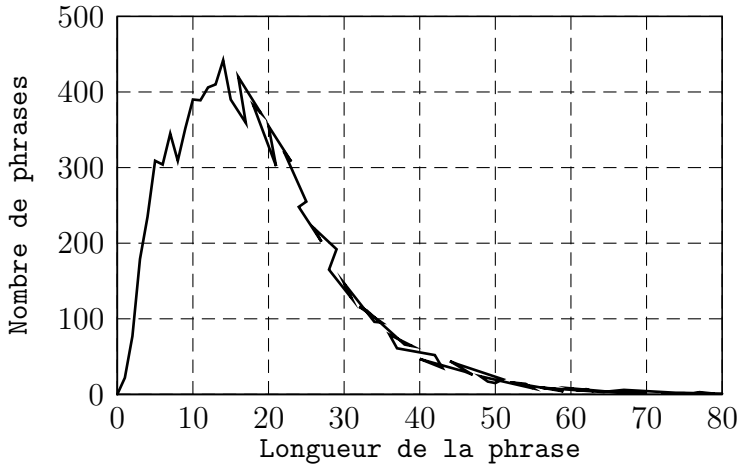


FIGURE 9.7 – Répartition du nombre de phrases du corpus selon leur longueur

Nous choisissons un sous-ensemble du corpus arboré contenant 8.808 phrases de longueur 30 mots ou moins comme corpus utilisé pour expérimenter l'analyse syntaxique du vietnamien. Ce corpus a été divisé en deux parties : un corpus d'apprentissage (95% du corpus complet) et un corpus de test (5%). Le corpus d'apprentissage a servi à l'extraction des grammaires. Le corpus de test a été utilisé pour évaluer les performances de l'analyseur. Les tailles des deux corpus sont reproduites dans la Table 9.2.

	Apprentissage	**Test**
Nombre de phrases	8.367	441
Nombre de mots	129.059	6.913

TABLE 9.2 – Tailles des corpus d'apprentissage et de test

La taille de la grammaire LTAG extraite du corpus d'apprentissage est montrée dans la Table 9.3.

Type	**Nombre d'arbres**	**Nombre de schèmes**
Arbres d'épine	19.708	741
Arbres de modifieur	15.868	860
Arbres de conjonction	79	57
Total	**35.655**	**1.658**

TABLE 9.3 – Taille de la grammaire LTAG extraite du corpus d'apprentissage.

Les deux sous-sections qui suivent présentent les performances de l'analyseur sur le corpus de test en deux versions : sans ou avec un étiqueteur syntaxique intégré.

72. Nous ne comptons pas les ponctuations présentes dans la phrase.

9.5.3 Performances de l'analyseur sans étiquetage

Nous présentons dans cette section les expériences et les résultats de l'analyseur sur le corpus de test. Les phrases sont analysées sans recourir à un étiqueteur syntaxique. Par défaut, chaque occurrence de mot est étiquetée par toutes les catégories qui lui ont été associées dans le corpus d'apprentissage. Si c'est un nouveau mot, il est étiqueté comme un nom (l'étiquette `N`).

Tout d'abord, nous évaluons les performances de l'analyse en constituants. Les résultats sont montrés dans la Table 9.4 [73]. En plus des mesures de précision et de rappel, d'autres

T-précision	Tous	≤ 10 mots
Précision	67,98	71,28
Rappel	68,40	71,39
F-mesure	68,19	71,33
Concordance parfaite	13,00	17,57
Croisement moyen	2,66	1,80
Non-croisement	23,00	29,73
Moins de 3 croisements	55,00	68,92
Précision d'étiquetage	87,72	87,34

TABLE 9.4 – Performances de l'analyse en constituants

mesures qui sont utiles pour l'analyse de résultats sont également listées :
- La concordance parfaite est le pourcentage de phrases ayant la F-mesure de 100% (toutes les mesures de précision et de rappel sont 100%). Il y a 13% de phrases de test qui ont la concordance parfaite, tandis que ce taux pour les phrases de 10 mots ou moins est de 17,57%.
- Le croisement moyen est le nombre total de croisements divisé par le nombre de phrases du corpus de test.
- Le taux de non-croisement est le pourcentage de phrases ayant zéro croisement syntagmatique. Il y a 23% de phrases de test qui n'ont pas de croisement (29,73% pour les phrases de 10 mots ou moins). Il y a 55% (respectivement 68,92%) de phrases de test qui ont moins de 3 croisements.
- La précision d'étiquetage est le pourcentage de catégories syntaxiques préterminales qui sont correctes. Il est intéressant de noter que la précision d'étiquetage est légèrement diminuée quand des phrases de test plus courtes sont utilisées.

Nous avons évalué la performance de l'analyse en dépendances en deux versions, avec ou sans type [74]. Dans la première évaluation, deux dépendances typées $\texttt{type}_1(u_1, v_1)$ et $\texttt{type}_2(u_2, v_2)$ sont considérées égales si les trois parties correspondantes des dépendances sont toutes égales : $\texttt{type1}_1 \equiv \texttt{type}_2, u_1 \equiv u_2, v_1 \equiv v_2$. Dans la deuxième évaluation, on

73. Les résultats de l'évaluation présentés sont calculés automatiquement par le programme EVALB–un outil couramment utilisé pour l'évaluation de l'analyse syntaxique en constituants, librement distribué à l'adresse *http ://nlp.cs.nyu.edu/evalb/*.

74. Les performances de l'analyse en dépendances sont évaluées sur un sous-ensemble contenant 100 phrases du corpus de référence ; ces phrases ont été annotées en dépendances manuellement par nous-même.

ne compare que l'égalité des deux paires des mots concernées sans utiliser le type des dépendances. Les D-précisions des deux évaluations sont fournies dans la Table 9.5.

D-précision	**Avec type**	**Sans type**
Précision	70,83	74,02
Concordance parfaite	15,87	23,37

TABLE 9.5 – Performances de l'analyse en dépendances

Afin d'avoir une vue précise de l'exactitude d'un type particulier, nous avons calculé les scores de précision, de rappel et de F-mesure pour les types différents. Les résultats sont fournis dans la Table 9.6.

Type	**Précision**	**Rappel**	**F-mesure**
`arg`	87,57	79,02	83,08
`coord`	100,00	100,00	100,00
`modA`	48,57	62,96	54,84
`modC`	46,67	43,75	45,16
`modE`	50,00	56,52	53,06
`modL`	72,73	47,06	57,14
`modM`	80,00	53,33	64,00
`modN`	50,00	66,67	57,14
`modR`	64,10	60,98	62,50
`modV`	52,63	62,50	57,14

TABLE 9.6 – Performances de l'analyse en dépendances par type

Nous voyons que l'analyseur fonctionne correctement sur les structures de coordination ; sa performance sur la reconnaissance des dépendances de type argument est nettement meilleure que sur celle de type modifieur. Ces résultats justifient une ambiguïté plus élevée de l'opération d'adjonction (qui est reliée aux arbres auxiliaires) que celle de l'opération de substitution (qui est reliée aux arbres initiaux).

Nous observons que l'analyseur n'a pas pu analyser environ 16,6% du corpus de test. Nous pensons que le fait qu'une phrase n'est pas analysable peut être dû à deux raisons. Premièrement, il s'agit de la couverture insuffisante de la grammaire utilisée : la grammaire extraite du corpus d'apprentissage ne contient pas la structure syntaxique (les schèmes élémentaires) de la phrase à analyser. Deuxièmement, notre choix heuristique d'avoir étiqueté tous les nouveaux mots comme un nom commun peut effectivement introduire des erreurs préalables à l'analyse, ce qui pourrait mener à des échecs d'analyse. Nous n'avons pas, pour l'instant, d'analyse précise de ces causes.

L'ambiguïté et le temps d'analyse étant fortement dépendants de la longueur des phrases. Nous présentons dans la Figure 9.8 les ambiguïtés et les temps d'analyse moyens et maximaux pour les phrases de longueurs identiques de moins de 15 mots. Les graphiques de

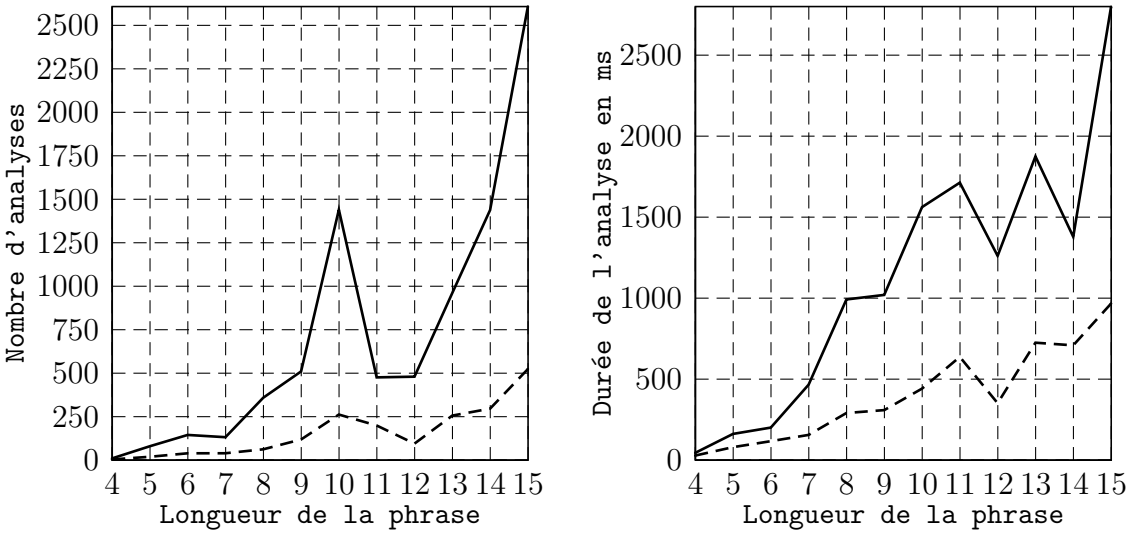


FIGURE 9.8 – Ambiguïté et temps d'analyse, moyens et maximaux, selon la longueur des phrases

cette figure permettent d'observer que le nombre d'analyses évolue de manière exponentielle par rapport à la longueur des phrases. [75]

9.5.4 Performances de l'analyseur couplé à un étiqueteur

Les résultats de la section précédente ont permis d'évaluer la grammaire et les performances de l'analyseur. Les conditions dans lesquelles les expériences ont été effectuées sont assez grossières car l'analyseur a dû essayer toutes les catégories syntaxiques possibles de chaque mot de la phrase à analyser. Les expériences décrites dans cette section sont plus proches des conditions réelles. Les phrases sont traitées, dans un premier temps, par un étiqueteur grammatical produisant une seule solution – à chaque mot est assignée une catégorie syntaxique unique. On a donc une seule séquence de mots/catégories et elle est fournie en entrée à l'analyseur syntaxique qui produit un nombre donné de solutions. L'étiquetage a été effectué en utilisant l'étiqueteur `vnTagger`, présenté au chapitre 6.

De façon similaire à la sous-section précédente, nous commencerons par donner les résultats de l'analyse en constituants, puis les résultats de l'analyse en dépendances, et enfin l'ambiguïté et le temps d'analyse selon la longueur des phrases.

La Table 9.7 présente la T-précision du système. En intégrant un étiqueteur, la précision d'étiquetage globale du système a été augmentée nettement, de $87,72\%$ à $95,25\%$ [76]. Cela a pu aider à augmenter tous les scores du système, notamment la concordance parfaite de l'analyseur, de $13,00\%$ à $16,67\%$ (et celle pour les phrases de longueur 10 mots ou moins est de $20,69\%$).

La performance de l'analyse en dépendances a été évaluée en deux versions, avec type ou sans type comme dans la section précédente. Les D-précisions des deux versions sont

75. Pour certaines phrases de longueur importante, l'analyseur n'arrive pas à donner de résultat après un temps limite fixé à 3 minutes.

76. Nous rappelons que le corpus de test ne contient que des phrases de longueur 30 mots ou moins.

T-précision	Tous	≤ 10 mots
Précision	69, 15	71, 60
Rappel	69, 52	72, 30
F-mesure	69, 33	71, 95
Concordance parfaite	16, 67	20, 69
Croisement moyen	2, 39	1, 69
Non-croisement	27, 78	32, 76
Moins de 3 croisements	54, 17	65, 52
Précision d'étiquetage	95, 25	95, 43

TABLE 9.7 – Performances de l'analyse en constituants couplée à un étiqueteur

fournies dans la Table 9.8.

D-précision	Avec type	Sans type
Précision	71, 81	73, 21
Concordance parfaite	20, 00	25, 45

TABLE 9.8 – Performances de l'analyse en dépendances couplée à un étiqueteur

Les résultats détaillés de l'analyse pour les différents types de dépendances sont fournis dans la Table 9.9.

Nous observons une amélioration légère des performances du système par rapport à la version sans étiquetage. Pourtant, le gain le plus important de l'analyseur couplé à un étiqueteur est une réduction forte de l'ambiguïté et du temps d'analyse, montré dans la Figure 9.9. L'étiqueteur aide à réduire l'ambiguïté de l'analyse cinq fois en moyenne. Il a permis de réduire le temps de l'analyse en moyenne trois fois par rapport au temps requis par le système sans étiquetage préalable. Cependant, nous observons que l'intégration de l'étiqueteur à l'analyeur a donné un taux plus élevé de phrases que le système n'a pas pu analyser, à savoir 40% du corpus de test. En effet, cette augmentation est prévisible car l'analyseur n'a exploré qu'une seule catégorie syntaxique pour chaque mot donnée par l'étiqueteur (la catégorie la plus probable). Nous rappelons également que la précision globale au niveau des phrases de l'étiqueteur est d'environ 32% (*cf.* section 6.3, chapitre 6), c'est-à-dire qu'il y a seulement environ un tiers de fois où l'étiqueteur peut donner l'étiquetage correct pour tous les mots d'une phrase à analyser.

9.5.5 Discussions

Nous avons vu dans les sections précédentes que les expériences effectuées ont permis d'évaluer le système syntaxique à base de grammaires LTAG pour le vietnamien. Les meilleurs résultats obtenus sont de 73, 21% (précision par dépendances) et de 69, 33% (F-mesure de la précision par arbre) pour un corpus de test.

Nous notons que ce sont les tout premiers résultats d'analyse syntaxique du vietnamien basée sur une grammaire LTAG. En effet, à notre connaissance, il existe un seul travail d'analyse syntaxique du vietnamien qui a été publié pour l'instant. Il s'agit d'une étude

Type	Précision	Rappel	F-mesure
arg	87, 18	80, 95	83, 95
coord	100, 00	100, 00	100, 00
modA	59, 09	65, 00	61, 90
modC	66, 67	60, 00	63, 16
modE	35, 71	35, 71	35, 71
modL	100, 00	50, 00	66, 67
modM	81, 82	75, 00	78, 26
modN	58, 54	68, 57	63, 16
modR	47, 06	42, 11	44, 44
modV	58, 33	87, 50	70, 00

TABLE 9.9 – Performances de l'analyse en dépendances par type couplée à un étiqueteur

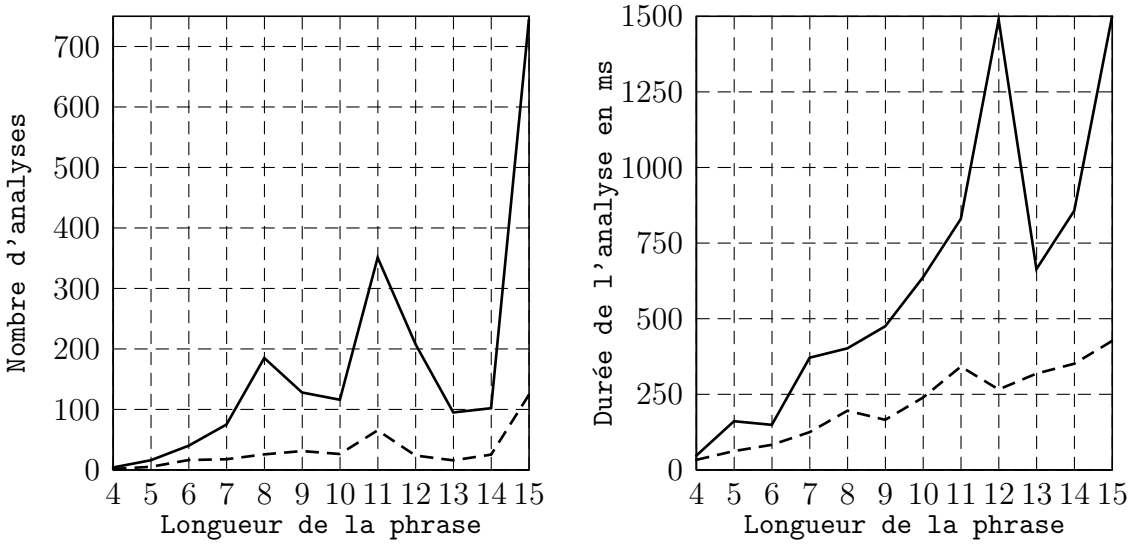


FIGURE 9.9 – Ambiguïté et temps d'analyse, moyens et maximaux, selon la longueur des phrases, avec un étiqueteur intégré

empirique de l'analyse statistique du vietnamien en se basant sur des modèles statistiques de grammaires hors-contextes lexicalisées [96]. Ce travail explore l'application des modèles d'analyse de Michael Collins [39] pour l'analyse syntaxique du vietnamien, son meilleur résultat d'analyse en constituants est de 78% sur un corpus de test ; aucun résultat d'analyse en dépendances n'a été publié. En ce qui concerne le résultat d'analyse en constituants, leur système est légèrement supérieur au nôtre. Cependant, ces résultats ne sont pas directement comparables car les modèles sont entraînés et testés sur des corpus différents.

Nos premiers résultats d'analyse syntaxique du vietnamien sont encourageants malgré qu'ils soient, pour le moment, assez décevants et significativement inférieurs aux résultats d'analyse syntaxique de l'état de l'art pour des langues bien étudiées comme l'anglais (où on atteint une T-précision de $91, 10\%$ [26] et une D-précision de $92, 93\%$ [92] sur le Penn Treebank) ou le français (où on atteint une T-précision de $86, 41\%$ [20] et une D-précision de $85, 55\%$ sur le FTB [21]). Pourtant, nous pensons que nous pouvons améliorer encore

les résultats du système puisque les expériences ont révélé un certain nombre de sources d'erreurs, certaines desquelles pourraient être traitées, améliorant ainsi les résultats. Il y a trois sources majeures d'erreurs qui sont présentées ci-dessous.

La principale source d'erreurs que nous avons identifiée, à l'issue des expériences, est la sélection de l'analyse. En effet, nous avons choisi une seule analyse pour chaque phrase selon une méthode très simple. S'il y a plusieurs arbres d'analyse pour une phrase (c'est souvent le cas), nous avons choisi l'arbre de dérivation dont l'arbre dérivé comprend le plus petit nombre de nœuds. Bien que l'arbre retenu corresponde à l'analyse la plus spécifique, il est évident qu'une telle méthode de sélection d'arbre est purement heuristique et assez fragile. Il existe des cas où les arbres choisis ne correspondent pas aux analyses correctes des phrases données. L'amélioration de la méthode de sélection du meilleur arbre d'analyse est une condition nécessaire et cruciale pour augmenter les performances du système. Dans le futur, nous voulons développer et évaluer des méthodes plus efficaces pour déterminer les meilleurs arbres de dérivations de phrases. Dans une telle perspective, le recours à des méthodes différentes de classifications statistiques ou symboliques (non probabilistes) peut être envisagé.

La deuxième source d'erreurs est l'étiquetage grammatical. Dans les expérimentations effectuées avec l'analyse couplée à un étiqueteur, nous avons pris en compte une seule (la meilleure) solution de l'étiqueteur `vnTagger`. Nous avons vu que l'étiqueteur commet souvent des erreurs au niveau de la phrase, notamment pour les phrases longues. Une erreur d'étiquetage se traduit généralement par une erreur (ou plus) d'analyse. L'amélioration des résultats d'étiquetage est donc une autre condition nécessaire pour augmenter la performance du système global.

La troisième source d'erreurs concerne la couverture de la grammaire utilisée dans les expérimentations. De manière générale, la proportion de phrases de test comportant au moins un mot pour lequel la catégorie n'est pas connue par la grammaire est assez élevée, à environ 15%. Par conséquent, l'analyse correcte de ces phrases ne pourrait pas être construite par l'analyseur. Une solution évidente à ce problème est l'accroissement de la taille du corpus d'apprentissage, ce qui revient à l'accroissement de la taille du corpus arboré du vietnamien. Mais de tels corpus sont chers à élaborer. De plus, un accroissement important de la taille de la grammaire peut mener à un problème spécifique à l'analyseur syntaxique symbolique LLP2 que nous avons utilisé dans lequel le temps d'analyse nécessaire d'une phrase longue pourrait dépasser un *timeout* élevé. Ici, on revoit un problème général du développement d'un analyeur syntaxique pour une langue naturelle, à savoir le compromis entre finesse et efficacité. En effet, les analyseurs à description syntaxique superficielle peuvent être rapides, mais c'est difficile à obtenir pour les systèmes reposant sur des grammaires à large couverture proposant des analyses détaillées. Nous pensons que l'enjeu de l'efficacité du temps et de la performance de l'analyse syntaxique symbolique constitue en lui-même une question intéressante. Une exploration plus détaillée de ce sujet fera l'objet de recherches futures.

9.5.6 Logiciels

Nous avons développé un logiciel intitulé `vnLTAGParser` qui implemente un analyseur syntaxique à base de grammaires LTAG pour l'analyse de textes vietnamiens. Le logiciel `vnLTAGParser` est librement distribué sous la licences GNU/GPL [77]. L'analyseur `vnLTAGParser` a été intégré au logiciel `vnToolkit`, une suite d'outils pour les traitements du texte vietnamien (*cf.* Annexe B).

9.6 Bilan

Nous avons présenté dans ce chapitre le développement d'un analyseur syntaxique en constituants et en dépendances pour le vietnamien et les résultats de l'analyse effectués sur un corpus de test. Nous avons défini pour la première fois un schéma d'annotation en dépendances pour le vietnamien, un analyseur syntaxique à base de grammaires d'arbres adjoints pour cette langue, ainsi qu'un algorithme d'extraction des relations syntaxiques à partir des arbres de dérivation en sortie de l'analyseur. Nous pensons que les résultats et les expériences obtenus constituent une base de réflexion utile pour l'élaboration de projets d'ingénierie des langues de grande ampleur concernant le traitement automatique du vietnamien.

77. *http ://www.loria.fr/ lehong/tools/vnLTAGParser.php*

Conclusion et perspectives

Dans cette thèse, nous avons réalisé un composant syntaxique pour la langue vietnamienne qui comprend trois parties principales : une chaîne modulaire de prétraitements ; une grammaire à large couverture à base de grammaires d'arbres adjoints ; et un analyseur syntaxique profond du vietnamien. Nous profitons de cette conclusion pour résumer les résultats les plus importants et aussi pour proposer un certain nombre de problèmes pour les travaux à venir.

Conclusion

Dans la première partie, nous présentons une étude de base de la langue vietnamienne afin de clarifier les difficultés particulières liées au traitement automatique de cette langue.

Dans la deuxième partie, nous étudions d'une part les fondements théoriques des modèles d'apprentissage supervisé, plus précisément des modèles log-linéaires, et d'autre part l'exploration des connaissances utiles représentant des particularités du vietnamien. Ces connaissances ont été intégrées aux modèles d'apprentissage, qui ont donné de bons résultats pour les tâches de segmentation de textes en phrases et l'étiquetage morpho-syntaxique. Nous avons proposé une méthode hybride et un algorithme pour résoudre efficacement le problème de la segmentation de textes en unité lexicales.

Dans la troisième partie, nous avons développé un système qui permet l'acquisition automatique d'une grammaire à large couverture à partir d'un corpus arboré du vietnamien. Les arbres élémentaires de la grammaire forment les structures syntaxiques du vietnamien. Nous avons construit un analyseur syntaxique profond du vietnamien en adaptant et enrichissant un analyseur syntaxique originellement développé pour le français. Nous avons conçu un schéma d'annotation en dépendances permettant d'encoder les relations grammaticales du vietnamien ainsi qu'un algorithme permettant d'extraire des analyses en dépendances à partir des arbres de dérivation donnés par l'analyseur. Nous avons étudié systématiquement les différentes procédures d'évaluation d'analyses syntaxiques afin d'effectuer des évaluations significatives de notre système. Notre système d'analyse en particulier et ses modules couplés en général atteignent des performances prometteuses dans les tâches du traitement automatique du vietnamien à l'heure actuelle.

Les méthodes et le système que nous avons développés constituent une bonne base pour faciliter la mise à disposition de corpus annotés du vietnamien, enjeu majeur du développement du TAL pour une langue. L'ensemble des logiciels que nous avons implémentés et

mis à disposition pour le téléchargement libre est une autre contribution fructueuse pour la communauté du traitement du vietnamien.

Au départ, ces travaux bénéficient d'expériences accumulées au LORIA sur les TAG pour le français, mais au final, certains travaux, spécifiquement développés pour le vietnamien, pourront être bénéfiques pour améliorer les résultats pour le français, par exemple l'étiquetage avec les modèles log-linéaires, le module d'extraction des dépendances à partir des arbres de dérivation et la plateforme de traitement multilingue Nancy Parsing Toolkit [140].

Perspectives

Les résultats dans cette thèse pourront évidemment être améliorés. De nombreux problèmes restent ouverts et feront l'objet de nos recherches futures.

Au niveau du prétraitement de textes vietnamiens, nous avons vu qu'il existe deux types d'ambiguïtés de segmentation en unités lexicales : l'ambiguïté de chevauchement et l'ambiguïté de combinaison. Notre méthode est capable de résoudre efficacement le premier type d'ambiguïté mais pas le deuxième type. Nous avons vu également que dans la majorité des cas, les erreurs de segmentation sont dues à des mots inconnus. Une erreur de segmentation pourrait effectivement introduire des erreurs d'étiquetage morpho-syntaxique. Vu que notre méthode de segmentation peut proposer non seulement la meilleure segmentation mais toutes les segmentations possibles d'une phrase, nous pourrions alors introduire un modèle probabiliste commun qui combine à la fois la segmentation en mots et l'étiquetage morpho-syntaxique. Nous espérons que la combinaison de deux tâches en un modèle raisonnable nous permettrait d'améliorer encore les résultats du prétraitement. Ce problème intéressant est une direction prioritaire de notre recherche future concernant les niveaux de prétraitement de textes.

En ce qui concerne l'analyse syntaxique, nous avons vu une complexité élevée de la tâche d'analyse syntaxique du vietnamien, notamment du point de vue de l'ambiguïté. Nous avons aussi identifié trois sources d'erreurs d'analyse ainsi que des directions majeures pour améliorer les résultats du traitement. Nous voudrions insister ici sur le point le plus important concernant la stratégie de sélection de l'analyse. Dans les travaux à venir, nous envisagerons de développer et d'évaluer des méthodes plus efficaces que la méthode actuelle afin de déterminer les meilleures analyses, tant du point de vue de l'analyse en constituants que de l'analyse en dépendances. Dans une telle perspective, le recours à des méthodes récentes d'analyse syntaxique probabiliste profonde ou peu profonde, y compris le super-étiquetage est une direction prioritaire puisque ces méthodes ont montré leur efficacité dans de nombreuses tâches de prédiction structurelle. Il s'agit de méthodes d'appentissage supervisé et semi-supervisé qui constituent un domaine de recherche actif au sein de la communauté internationale du TAL à l'heure actuelle.

En ce qui concerne le développement d'une grammaire d'arbres adjoints du vietnamien en particulier et du composant syntaxique du vietnamien en général, bien que la grammaire actuelle ait une couverture relativement large, elle reste néanmoins encore insuffisante au regard des résultats expérimentaux obtenus. Une perspective à moyen et long terme de

ces travaux porte à la fois sur l'accroissement du corpus arboré du vietnamien et sur la validation de manière plus détaillée de la grammaire afin d'assurrer la pertinence et la robustesse du composant syntaxique. La question de l'équilibre entre l'efficacité du temps de calcul et la performance du système d'analyse par rapport à la taille importante de la grammaire utilisée est un autre problème qui constitue notre sujet de recherche futur. Dans cette perspective, le recours à un modèle de méta-grammaire permettant une représentation compacte de grammaires LTAG est une direction envisagée. Comme nous l'avons constaté précédemment, avec le système d'extraction automatique de grammaires et la disponibilité des corpus arborés du vietnamien et du français, nous projetons de réaliser une comparaison quantitative des structures syntaxiques du français et de celles du vietnamien. Une comparaison de deux grammaires pourrait révéler des relations intéressantes entre les deux langues car, pour des raisons historiques, la langue vietnamienne a été enrichie non seulement dans son vocabulaire mais également dans sa syntaxe par l'influence de la grammaire française.

Enfin, durant cette thèse, nous avons participé à de nombreux projets de recherche coordonnés par des équipes de recherche en France et au Viêtnam. Nous avons été impliqué dans les projets PaULE (Passage Utilities for Language Engineering [78]) et Nancy-TAG (Tools for Tree Adjoining Grammar based parsing [79]) du LORIA. Nous avons été également impliqué dans le projet national vietnamien VLSP (Vietnamese Language and Speech Processing) regroupant des scientifiques de quelques groupes de recherche en informatique linguistique au Viêtnam qui a pour but de développer des ressources linguistiques et des outils fondamentaux pour le traitement automatique du vietnamien écrit et parlé. Nos travaux présentés dans cette thèse ont été menés en collaboration avec des informaticiens et des linguistes à l'Université nationale du Viêtnam à Hanoï, au Centre de lexicographie du Viêtnam et à l'Institut des technologies de l'information, Centre national de la recherche scientifique du Viêtnam. Nous espérons que nos expériences acquises en TAL grâce aux travaux consacrés aux langues européennes et au vietnamien pourraient aider à faire progresser la recherche et le développement ainsi que la formation sur le traitement informatique du vietnamien à une plus grande échelle dans le futur.

78. *http ://gforge.inria.fr/projects/paule*
79. *http ://gforge.inria.fr/projects/nancy-tag/*

A

Les tables spécifiques du corpus arboré vietnamien

Cette annexe décrit les tables qui constituent l'information spécifique du corpus arboré vietnamien. Ces tables ont été utilisées dans les algorithmes d'extraction des grammaires LTAG présentés au chapitre 8. Nous présentons dans les sections suivantes les trois tables : la table de percolation de tête, la table d'arguments et la table des catégories syntaxiques.

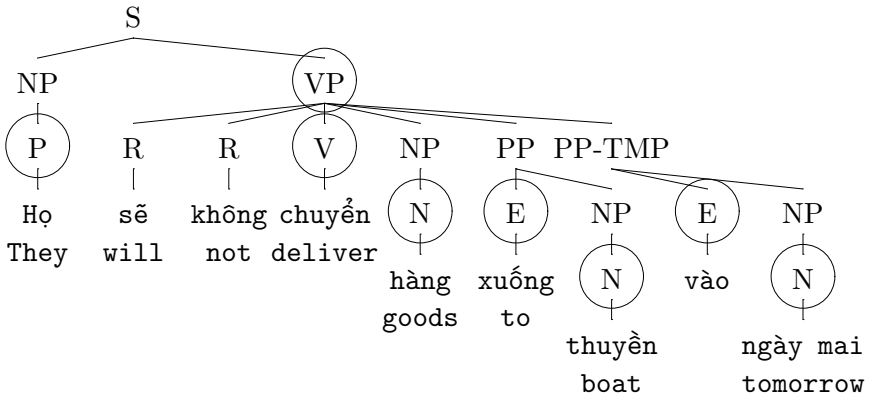
A.1 Table de percolation de têtes

Cette section décrit les règles que nous avons conçues pour sélectionner les têtes de constituants du corpus arboré ; c'est-à-dire pour un syntagme $(X\,(Y_1 \cdots Y_n))$ [80], ces règles déterminent quel composant $Y_i, i = 1, \ldots, n$ est la tête du syntagme.

Table A.1 montre les règles utilisées pour les constituants du corpus arboré. A titre d'illustration, pour la règle $X \rightarrow Y_1 \cdots Y_n$ où X est `VP`, l'algorithme va chercher à partir de la gauche de la séquence $Y_1 \cdots Y_n$ pour le premier Y_i de type `VP`, s'il ne trouve aucun `VP`, il continue à chercher pour le premier Y_i de type `V` ; s'il n'y a aucun `V` trouvé, il cherche le premier Y_i de type `A`, *etc.* S'il ne trouve aucun constituant, il sélectionne par défaut le premier constituant–Y_1.

Nous reprenons une phrase exemple du chapitre 8 « `Họ sẽ không chuyển hàng xuống thuyền vào ngày mai` » (Ils ne livreront pas les marchandises au bateau demain) dans laquelle les têtes des syntagmes sont entourées dans la figure ci-dessous.

80. Ce syntagme corresponde à une règle hors-contexte $X \rightarrow Y_1 \cdots Y_n$



Catégories	**Direction**	**Liste de priorité**
S	→	S VP AP NP
SBAR	→	SBAR S VP AP NP
SQ	→	SQ VP AP NP
NP	→	NP Nc Nu Np N P
VP	→	VP V A AP N NP S
AP	→	AP A N S
RP	←	RP R T NP
PP	→	PP E VP SBAR AP QP
QP	→	QP M
XP	→	XP X
YP	→	YP Y
MDP	→	MDP T I A P R X
WHNP	→	WHNP NP Nc Nu Np N P
WHAP	→	WHAP A N V P X
WHRP	→	WHRP P E T X
WHPP	→	WHPP E P X
WHXP	→	XP X

TABLE A.1 – Règles de percolation de têtes pour le corpus vietnamien.

A.2 Table d'arguments

Cette section décrit les règles utilisées pour sélectionner les arguments de constituants du corpus arboré. Pour un syntagme $(X\,(Y_1 \cdots Y_n))$ ces règles déterminent quels composants $Y_i, i = 1, \ldots, n$ sont les arguments de X (les autres composants sont des modifieurs de X).

Table A.2 montre les règles utilisées pour déterminer les arguments de constituants. Notons que en plus de cette table, nous avons utilisé également des étiquettes fonctionnelles du corpus arboré pour distinguer les arguments des modifieurs. Le corpus utilise, le cas échéant, les étiquettes `SUB`, `DOB` et `IOB` pour marquer les arguments de groupes ; tandis

que les étiquettes `TMP`, `LOC`, `DIR`, `MNR`, `PRP`, `CND`, `CNC` et `ADV` pour marquer les modifieurs de groupes.

Catégories	**Liste des arguments**
S	S
NP	VP V
AP	NP N VP V S
VP	NP N V S
V	SBAR NP N P VP V PP
A	NP N
Nc	N
E	NP N VP V

TABLE A.2 – Règles d'arguments pour le corpus vietnamien

A.3 Table des catégories syntaxiques

Les catégories syntaxiques du corpus arboré vietnamien sont reproduites dans la Table A.3. Cette table a été introduite au chapitre 6 (*cf.* section 6.3).

No.	**Catégorie**	**Description**	**No.**	**Catégorie**	**Description**
1.	Np	Nom propre	10.	M	Numéral
2.	Nc	Nom individuel	11.	E	Préposition
3.	Nu	Nom d'unité	12.	C	Conjonction subordonnée
4.	N	Nom commun	13.	CC	Conjonction coordonnée
5.	V	Verbe	14.	I	Interjection
6.	A	Adjectif	15.	T	Mot modal, auxiliaire
7.	P	Pronom	16.	Y	Abréviation
8.	R	Adverbe	17.	Z	Morphème lié
9.	L	Déterminant	18.	X	Inconnu

TABLE A.3 – Les catégories syntaxiques du corpus arboré vietnamien

B

vnToolkit

Cette annexe a pour but de donner une description générale de la plateforme `vnToolkit`– un logiciel qui rassemble des modules et des ressources linguistiques nécessaires pour les traitements de textes vietnamiens que nous avons développé dans le cadre de notre travail de thèse.

Depuis la première mise en ligne pour de téléchargements libres, le logiciel `vnToolkit` et ses modules ont été téléchargés environ 9.300 fois. [81] Les logiciels que nous avons développés ont été utilisés dans de nombreux projets de recherche d'institutions scientifiques et d'enterprises au niveau national et international. Notre travail a contribué efficacement au développement de la jeune communauté du traitement automatique du vietnamien. Le logiciel `vnToolkit` et ses modules, les outils séparés ainsi que les ressources linguistiques associées sont disponibles en ligne sur notre site web personnel. [82] Tous les logiciels sont implémentés dans le langage de programmation Java et distribués sous la licence GNU/GPL. [83]

B.1 Les modules

Le logiciel `vnToolkit` comprend une chaîne modulaire de traitements qui permettent d'appliquer à des corpus bruts une cascade de traitements de surface et l'analyse syntaxique profonde.

Pour l'instant, le logiciel se compose des modules principaux suivants :

1. Un segmenteur en phrases
2. Un segmenteur en unités lexicales
3. Un reconnsaisseur de mots redoublés
4. Un étiqueteur morpho-syntaxique
5. Un extracteur de grammaires LTAG

81. Les statistiques du téléchargement de logiciels sont calculées jusqu'à la fin du juillet 2010, toutes les versions d'un logiciel sont cumulées.
82. *http ://www.loria.fr/∼lehong/softwares.php*
83. *http ://www.gnu.org/copyleft/gpl.html*

6. Un analyseur syntaxique

Nous décrivons brièvement ces modules dans les sous-sections qui suivent.

B.1.1 Segmenteur en phrases

Le noyau de ce module est l'outil `vnSentDetector` qui implémente l'approche du découpage automatique de textes vietnamiens en phrases que nous avons présenté dans le chapitre 4. Le logiciel est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnSentDetector.php

Depuis sa première mise en ligne, ce logiciel a été téléchargé 396 fois.

B.1.2 Segmenteur en unités lexicales

La partie centrale de ce module est l'outil `vnTokenizer` qui implémente l'approche du découpage automatique de textes vietnamiens en unités lexicales que nous avons présenté dans le chapitre 5. Le logiciel est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnTokenizer.php

Depuis sa première mise en ligne, ce logiciel a reçu 4.352 téléchargements.

B.1.3 Reconnaisseur de mots redoublés

Comme nous l'avons présenté dans le chapitre 2 (*cf.* section 2.4.2), la réduplication ou le redoublement est un phénomène courant du vietnamien. Pour une langue typiquement isolante comme le vietnamien, le redoublement est un moyen pour exprimer les sens grammaticaux de mots. Il permet notamment de modifier l'intensité des adjectifs ; la forme redoublée des mots est souvent combinée avec des mots outils pour souligner un constituant.

En étudiant la classe des mots redoublés bisyllabiques du vietnamien, nous apprenons que la formation de mots redoublés d'une large partie de cette classe observe des règles lexicales explicites, qui pourraient être modélisées efficacement par des automates à état finis optimaux. Nous avons donc introduit un modèle informatique permettant de modéliser le processus de formation des mots redoublés du vietnamien. Cette étude a été publiée dans [102], et dans un but de complétitude, elle est également incluse dans l'annexe suivante.

Le résultat principal de cette étude est le logiciel appelé `vnReduplicator` qui implémente le modèle proposé. Ce reconnaisseur de mots redoublés a été intégré au segmenteur `vnTokenizer`. Cette intégration a pu aider à améliorer la précision de la segmentation de textes vietnamiens en unités lexicales.

Le logiciel `vnReduplicator` est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnReduplicator.php

Depuis sa première mise en ligne, ce logiciel a reçu 213 téléchargements.

B.1.4 Etiqueteur morpho-syntaxique

Ce module contient l'étiqueteur morpho-syntaxique `vnTagger` qui a été présenté dans le chapitre 6. Le logiciel est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnTagger.php

Depuis sa première mise en ligne, ce logiciel a reçu 463 téléchargements.

B.1.5 Extracteur de grammaires LTAG

Ce module contient le logiciel `LExtractor` qui a été présenté dans le chapitre 8. Ce logiciel a été utilisé pour élaborer automatiquement des grammaires d'arbres adjoints pour le vietnamien. Le logiciel est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnLExtractor.php

Depuis sa première mise en ligne, ce logiciel a reçu 72 téléchargements.

B.1.6 Analyseur syntaxique

Ce module contient le logiciel `vnLTAGParser` qui a été présenté dans le chapitre 9. Cet analyseur permet d'analyser des phrases vietnamiennes et de donner des analyses en constituants et en dépendances. Les résultats d'analyse peuvent être représentés sous forme d'arbres en interface graphique ou textuelle. Ce logiciel est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnLTAGParser.php

B.1.7 Modules supplémentaires

En plus des modules principaux présentés ci-dessus, la plateforme `vnToolkit` intègre également deux modules supplémentaires qui sont utiles pour le développement de ressources linguistiques, notamment un corpus arboré.

Le premier outil est le logiciel `SynAF` qui permet la construction et l'édition des arbres syntaxiques en utilisant une interface graphique. Il permet également d'encoder les arbres résultats sous forme des fichiers XML (eXtensible Markup Language) conformes à l'encodage SynAF recommandé par l'Organisation Internationale de Standardisation (ISO) [71]. [84] La construction d'un schéma d'annotation syntaxique pour le vietnamien et du logiciel `SynAF` a été publiée dans [104]. Le logiciel `vnSynAF` est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/synaf.php

84. Syntactic Annotation Framework (SynAF) est une norme proposée par l'ISO comme un framework général pour l'encodage d'annotations syntaxiques des langues naturelles. Cette norme est gérée par le comité TC37/SC4 de l'ISO (*http ://www.tc37sc4.org*).

Depuis sa première mise en ligne, ce logiciel a reçu 1.167 téléchargements.

Le deuxième outil est le logiciel `vnLexerCounter` qui permet d'examiner des corpus bruts et de donner des statistiques utiles sur ce corpus. En effet, ce logiciel a été originellement développé suite à la demande des linguistes informaticiens du Centre de la Lexicographie du Viêtnam. [85] Il est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnLexerCounter.php

Depuis sa première mise en ligne, ce logiciel a reçu 94 téléchargements.

B.2 La plateforme vnToolkit

B.2.1 La plateforme

La plateforme n'est pas simplement un ensemble d'outils. En effet, elle est une application riche basée sur la plateforme d'application Eclipse (Eclipse Rich Client Platform). [86] C'est-à-dire que la plateforme fournit à l'utilisateur une interface unique et dynamique [87] offrant des fonctionnalités. Ces fonctionnalités sont fournies par des plugins Eclipse dédiés. Pour chaque outil inclus dans la plateforme, deux plugins ont été développés. Un plugin réalise l'action correspondant à la fonctionnalité et l'autre plugin offre une interface graphique pour présenter le résultats (par exemple l'affichage des arbres d'analyse).

La plateforme est disponible à l'adresse :

http ://www.loria.fr/∼lehong/tools/vnToolkit.php

Depuis sa première mise en ligne, la plateforme elle-même a reçu 2.552 téléchargements.

B.2.2 Gestion des ressources linguistiques

Du point de vue des ressources, le logiciel `vnToolkit` et ses modules utilisent de nombreuses ressources linguistiques nécessaires pour les tâches concernées. En général, les ressources sont soit des fichiers binaires qui encodent les modèles statistiques entraînés, soit des fichiers textuels XML qui contiennent des lexiques et des grammaires.

En particulier, l'analyseur `vnLTAGParser` adopte la structure des ressources du logiciel LLP2. L'analyseur LLP2 s'inspire de l'architecture XTAG [182, 42] qui distingue le lexique morphologique (permettant d'étiqueter les segments et d'identifier les lemmes correspondants), le lexique syntaxique (qui permet la sélection des arbres par filtrage et leur ancrage) et la grammaire (qui contient les arbres TAG).

85. Vietlex : *http ://www.vietlex.com*
86. Eclipse RCP : *http ://www.eclipse.org/rcp*
87. Par dynamique, nous signifions que le contenu de l'interface change selon la fonctionnalité sélectionnée.

C

Finite-state description of Vietnamese reduplication

This appendix presents our work on the description of reduplication, a special phenomenon of the Vietnamese language. This work was published previously in [102].

We present for the first time a computational model for the reduplication of the Vietnamese language. Reduplication is a popular phenomenon of Vietnamese in which reduplicative words are created by the combination of multiple syllables whose phonics are similar. We first give a systematical study of Vietnamese reduplicative words, bringing into focus clear principles for the formation of a large class of bi-syllabic reduplicative words. We then make use of optimal finite-state devices, in particular minimal sequential string-to-string transducers to build a computational model for very efficient recognition and production of those words. Finally, several nice applications of this computational model are discussed.

C.1 Introduction

Finite-state technology has been applied successfully for describing the morphological processes of many natural languages since the pioneering works of [84, 87]. It is shown that while finite-state approaches to most natural languages have generally been very successful, they are less suitable for non-concatenative phenomena found in some languages, for example the non-concatenative word formation processes in Semitic languages [35]. A popular non-concatenative process is reduplication – the process in which a morpheme or part of it is duplicated.

Reduplication is a common linguistic phenomenon in many Asian languages, for example Japanese, Mandarin Chinese, Cantonese, Thai, Malay, Indonesian, Chamorro, Hebrew, Bangla, and especially Vietnamese.

We are concerned with the reduplication of Vietnamese. It is noted that Vietnamese is a monosyllabic language and its word forms never change, contrary to occidental languages that make use of morphological variations. Consequently, reduplication is one popular and important word formation method which is extensively used to enrich the lexicon. This

follows that the Vietnamese lexicon consists of a large number of reduplicative words.

This paper presents for the first time a computational model for recognition and production of a large class of Vietnamese reduplicative words. We show that Vietnamese reduplication can be simulated efficiently by finite-state devices. We first introduce the Vietnamese lexicon and the structure of Vietnamese syllables. We next give a complete study about the reduplication phenomenon of Vietnamese language, bringing into focus formation principles of reduplicative words. We then propose optimal finite-state sequential transducers recognizing and producing a substantial class of these words. Finally, we present several nice applications of this computational model before concluding and discussing the future work.

C.2 Vietnamese lexicon

In this section, we first present some general characteristics of the Vietnamese language. We then give some statistics of the Vietnamese lexicon and introduce the structure of Vietnamese syllables.

The following basic characteristics of Vietnamese are adopted from [185, 186, 187, 129].

C.2.1 Language type

Vietnamese is classified in the Viet-Muong group of the Mon-Khmer branch, that belongs to the Austro-Asiatic language family. Vietnamese is also known to have a similarity with languages in the Tai family. The Vietnamese vocabulary features a large amount of Sino-Vietnamese words. Moreover, by being in contact with the French language, Vietnamese was enriched not only in vocabulary but also in syntax by the calque of French grammar.

Vietnamese is an isolating language, which is characterized by the following properties :
– it is a monosyllabic language ;
– its word forms never change, contrary to occidental languages that make use of morphological variations (plural form, conjugation, *etc.*) ;
– hence, all grammatical relations are manifested by word order and function words.

C.2.2 Vocabulary

Vietnamese has a special unit called "*tiếng*" that corresponds at the same time to a syllable with respect to phonology, a morpheme with respect to morpho-syntax, and a word with respect to sentence constituent creation. For convenience, we call these "*tiếng*" syllables. The Vietnamese vocabulary contains
– simple words, which are monosyllabic ;
– reduplicative words composed by phonetic reduplication ;
– compound words composed by semantic coordination and by semantic subodination ;
– complex words phonetically transcribed from foreign languages.
The Vietnamese lexicon edited recently by the Vietnam Lexicography Center (Vietlex[88]) contains 40, 181 words and idioms, which are widely used in contemporary spoken language, newspapers and literature. These words are made up of 7, 729 syllables. Table C.1

88. *http ://www.vietlex.com/*

Length	#	%
1	6,303	15.69
2	28,416	70.72
3	2,259	5.62
4	2,784	6.93
≥ 5	419	1.04
Total	40,181	100

TABLE C.1 – Length of words measured in syllables

No.	Tones	Notation
1.	low falling	à
2.	creaky rising	ã
3.	creaky falling	ạ
4.	mid level	a
5.	dipping	ả
6.	high rising	á

TABLE C.2 – Vietnamese tones

shows some interesting statistics of the word length measured in syllables. 6,303 syllables (about 81.55% of syllables) are words by themselves. Two-syllable words are the most frequent, consisting of nearly 71% of the vocabulary.

C.2.3 Syllables

In this paragraph, we introduce phonetic attributes of Vietnamese syllables. In addition of the monosyllabic characteristic, Vietnamese is a tonal language in that each syllable has a certain pitch characteristic. The meaning of a syllable varies with its tone. This phonetic mechanism can also be found in other languages such that Chinese or Thai.

There are six tones in Vietnamese as specified in Table C.2. The letter *a* denotes any non-accent syllable. These six tones can be roughly classified into two groups corresponding to low and high pitches in pronunciation. The first half of the table contains three low tones and the second half contains three high tones. In addition, the difference in the tone of two syllables are distinguished by flat property of tones. The 1st and 4th tones in Table C.2 are flat (*bằng*), the other tones are non-flat (*trắc*).

The structure of a Vietnamese syllable is given in Table C.3. Each syllable can be divided into three parts : onset, rhyme and tone. The onset is usually a consonant, however it may be empty. The rhyme contains a vowel (nucleus) with or without glide /w/, and an optional consonant (coda). It is noticed that the initial consonant of a syllable does not carry information of the tone, the Vietnamese tone has an effect only on the rhyme part of the syllable [168]. This result reinforces the fact that a tone is always marked by the nucleus composant of the rhyme which is a vowel. Readers who are interested in detail the phonetic composition of Vietnamese syllables may refer to [168, 178].

Tone			
Onset	Rhyme		
	Glide	Nucleus	Coda

TABLE C.3 – Phonetic structure of Vietnamese syllables

C.3 Reduplication in Vietnamese

Reduplication is one of the methods for creating multi-syllable words in Vietnamese. A reduplicative word is characterized by a phenomenon called phonetic interchange, in which one or several phonetic elements of a syllable are repeated following a certain number of specific rules.

From the point of view of sense, the reduplication in Vietnamese usually indicates a diminutive of adjectives, which can also be found in Hebrew, or a pluralization in Malay, in Thai and in Indonesian, or an intensivity as the use of partial reduplication in Japanese, Thai, Cantonese and Chamorro (an Austronesian language spoken on Guam and the Northern Mariana Islands). In this aspect, Vietnamese reduplication serves similar functions as those of reduplication in several Asian languages, as reported in an investigation of Asian language reduplication within the NEDO project [163, 162].

The Vietnamese reduplication creates an expressional sense connecting closely to the phonetic material of Vietnamese, a language of rich melody. Consequently, there are many Vietnamese reduplicative words which are difficult to interpret to foreigners, though in general, native Vietnamese speakers always use and understand them correctly [55].

Vietnamese reduplicative words can be classified into three classes basing on the number of syllables they contain : two-syllable (or bi-syllabic) reduplicative words, three-syllable (or tri-syllabic) reduplicative words and four-syllable reduplicative words. The bi-syllabic class is the most important class because of two reasons : (1) bi-syllabic reduplicative words make up more than 98% amount of reduplicative words, that is, almost reduplicative words has two syllables ; and (2) bi-syllabic reduplicative words embody principle characteristics of the reduplication phenomenon in both phone aspect and sense formation aspect. For these reasons, in this paper, we address only bi-syllabic reduplicative words and call them reduplicative words for short, if there is no confusion.

As presented in the previous section, a syllable has a strict structure containing three parts : the onset, the rhyme and the tone. Basing on the phonetic interchange of a syllable, we distinguish two types of reduplication :
– full reduplication, where the whole syllable is repeated ;
– partial reduplication, where either the onset is repeated or the rhyme and the tone are repeated.

In this work, we constraint ourselves by focusing only on the construction of an efficient computational model applied for reduplicative words which have clear and well-defined formation principles. These words can be classified into three types investigated in detail in the following subsections. In given examples, the base syllables (or root syllable, or root for short) are the ones which are underlined. The reduplication that has undefined

Reduplicant	Root	#
a	*ả*	72
a	*á*	128
à	*ã*	27
à	*ạ*	80
	Sum	307

TABLE C.4 – Statistic of the second type reduplication

or incomplete formation rules will be tackled in future works.

C.3.1 Full reduplication

In this type of reduplication, the root is identically repeated ; there is only a slight difference on stress in pronunciation. For example, *hao* <u>*hao*</u> (a little similar), *lăm* <u>*lăm*</u> (intentional), *đùng* <u>*đùng*</u> (accidentally dertermined), *lừ* <u>*lừ*</u> (silently). In the Vietnamese lexicon there are 274 reduplicative words of this type.

In principle, there appears to be many reduplicative words of this type whose roots may be whatever syllables bearing whatever tone, for instance *đỏ đỏ*, *hớ hớ*, *sững sững*, *chậm chậm*. However, in consequence of the difference of stress between the root and the reduplicant, the tone of the reduplicant is changed in order to be in harmony with the root, for the sake of more readability and audibility ("easier to read, easier to hear"). This consequence leads to the formation of reduplicative words of the second type which we call reduplication with tone according.

C.3.2 Reduplication with tone according

As presented above, the difference between tone of the root and the reduplicant is a consequence of the difference between their stress which is expressed by their tones. This creates reduplicative words of the second type ; for example, *đo* <u>*đỏ*</u> (reddish), *hơ* <u>*hớ*</u> (in the bloom of youth), *sừng* <u>*sững*</u> (statly, high and majestic), *chầm* <u>*chậm*</u> (rather slow). The tone properties (low or high pitch, flat or non-flat) are now put into use.

The prosodic influence is responsible for the creation of the reduplicant from its root. As a result, the combination of tones between two syllables is realized in the following principle : *non-flat tones of the roots are matched against a corresponding flat tones of their reduplicants*. That is, the non-flat root has to select for it the flat reduplicant belonging to the same pitch, *i.e.*, in the same row. In this type of reduplicative words, the root is stressed in pronunciation.

A detailed statistic about these reduplicative words with respect to the combination of tones is given in Table C.4. There are 307 reduplicative words of the second type.

Example	**At root**	**At reduplicant**
	Noisy phone	Nasal phone
ăm ắp	*-p*	*-m*
phơn phớt	*-t*	*-n*
vằng vặc	*-c*	*-ng*
anh ách	*-ch*	*-nh*

TABLE C.5 – Transformation rules of final consonants

Root	**Reduplicant**	**#**
-p	*-m*	52
-t	*-n*	96
-c	*-ng*	56
-ch	*-nh*	28
	Sum	232

TABLE C.6 – Statistic of the third type reduplication

C.3.3 Reduplication with final consonant according

In this type of reduplication, there is not only the difference between tones of the root and the reduplicant but also the difference between their final consonants (hence their penultimates). Some examples of this type of reduplication which we call the third reduplication type are :

- *cầm cập* (clatter, shiver), *lôm lốp* (pop pop), *xăm xắp* (a little full), *thiêm thiếp* (fall asleep), *nơm nớp* (be in a state of suspense)
- *giôn giốt* (sourish), *ngùn ngụt* (burn violently), *phơn phớt* (light red), *hun hút* (profound), *san sát* (be very close to, adjoining)
- *vằng vặc* (very clear), *nhưng nhức* (a little ache), *rừng rực* (brightly), *phăng phắc* (very silent), *chênh chếch* (a little oblique), *anh ách* (feeling bloated).

The practical observation shows that the modification of final consonant from the root to the duplicate also has a clear rule : *the noisy phone of the root is transformed to a nasal phone of the reduplicant* as shown in Table C.5.

The transformation of final consonant occurs only with the roots having as final consonant *p*, *t*, or *c*. The principle of tone combination is the same as that of the second reduplication type.

A detailed statistic about these reduplicative words is given in the Table C.6. There are 232 reduplicative words of the third type.

Briefly, the total number of reduplicative words of all the three types of reduplication is 813, making up about $813/28,416 \approx 2.86\%$ of the number of two-syllable words.

C.4 Implementation

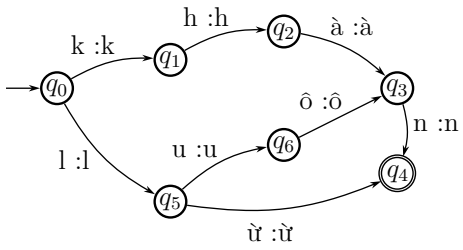
We report in this section the construction of a computational model for recognition and production of the three types of reduplication presented in the previous section. We have implemented finite-state sequential transducers (FSTs) which are able to recognize and produce corresponding types of reduplicative words. These devices operate on the same input and output alphabets, say Σ, containing all Vietnamese characters.

FSTs are formal devices for encoding regular relations. A regular relation is a mapping between two regular languages. In our cases, these languages are sets of Vietnamese root and reduplicant syllables.

We adapted nice and efficient algorithms developed by [44] to incrementally construct minimal transducers from a source of data. These algorithms are originally designed to build optimal deterministic finite-state automata on-the-fly but they can also be used to construct optimal sequential transducers. We could consider simply that the alphabet of the automata would be $\Sigma \times \Sigma$; output strings of Σ^* are associated with the final states of the lexicon and they are only outputed once corresponding valid inputs from Σ are recognized. Interested readers are invited to refer to [44] for further detail of the algorithms for building optimal automata on-the-fly.

C.4.1 First type transducer

In the first type reduplication, the root and the reduplicant is completely identical in writing; they are only distinguished by a stress in pronunciation. We can simply construct a deterministic finite-state transducer (FST) f_1 that produces reduplicants from their roots in which the output string labeled on each arc is the same as its input character; that is $f_1(x) = x$ where x is a syllable in the first type duplication. As an illustration, the following minimal FST recognizes and generates three first type reduplicative words *luôn luôn* (always), *lừ lừ* (silently), *khàn khàn* (raucous).

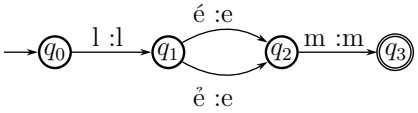


The minimal FST f_1 recognizing all 274 reduplicative words of the first type consists of 90 states in which 16 states are final ones. It has 330 transitions, the maximum number of outtransitions from a state is 28.

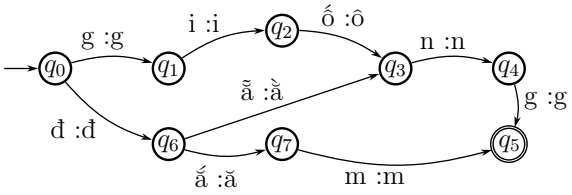
C.4.2 Second type transducer

In the second type reduplication, the root has an non-flat tone while its reduplicant has the corresponding flat tone. A root determines for it the unique reduplicant. Hence we can construct a sequential FST f_2 which is able to generate reduplicants from roots.

For instance, consider two reduplicative words of the second type *lem lém* (glib) and *lem lẻm* (voluble). They can be recognized by the minimal sequential FST f_2 such that $f_2(lém) = lem$ and $f_2(lẻm) = lem$ as depicted in the following figure :



Similarly, the minimal FST f_2 which generates three reduplicative words *giông giống* (a little similar), *đằng đẵng* (interminable) and *đăm đắm* (fixedly) is as follows :

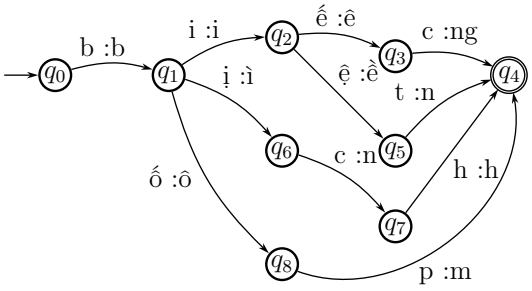


The minimal FST f_2 recognizing all 307 reduplicative words of the second type consists of 93 states in which 11 states are final ones. It has 371 transitions, the maximum number of outtransitions from a state is 22.

C.4.3 Third type transducer

The roots and reduplicants in the third type reduplication are not only combined by principles of flat and non-flat tones, they are also distinguised by last consonants. We know that in the case the root ends with *c*, its reduplicant is one character longer than it. The other three transformations of last consonants do not change the length of the reduplicants with respect to that of the roots.

Hence the FST f_3 which recognizes the third type reduplication is required to modify the tones of the reduplicants with respect to those of the roots on the one hand, and to transform last consonants of the roots on the other hand. For example, the minimal FST f_3 recognizing four reduplicative words *biêng biếc* (bluish green), *biền biệt* (leave behind no traces whatsoever), *bình bịch* (a series of thudding blows) and *bôm bốp* (pop pop) is given in the figure below :



The minimal FST f_3 recognizing all 232 reduplicative words of the third type consists of 59 states in which 2 states are final ones. It has 262 transitions, the maximum number of outtransitions from a state is 19.

Once all the three transducers have been constructed, we can unify them by making use of the standard union operation on transducers to obtain a sequential FST which is able to recognize all the three class of reduplication presented above [119, 120].

C.4.4 A software package

We have developed a Java software package named **vnReduplicator** which implements the above-mentioned computational model of Vietnamese reduplication. The core component of this package is a minimal FST which can recognize a substantial amount of reduplicative bi-syllabic words found in the Vietnamese language.

The first application of this core model which we have developed is a reduplication scanner for Vietnamese. We use the minimal FST of the core model to build a tool for fast detection of reduplication. The tool scans a given input text and produces a list of all the recognized reduplicative words. The detection process is very fast since the underlying transducer operates in optimal time in the sense that the time to recognize a syllable corresponds to the time required to follow a single path in the deterministic finite-state machine, and the length of the path is the length of the syllable measured in characters.

As an example, given the following input text

"Anh đi *biền biệt*. Cô vẫn chờ anh hơn 20 năm *đằng đẵng*."[89],

the scanner marks two reduplicative words as shown in the italic face.

We are currently investigating another useful application of the core model for a partial spell checking of Vietnamese text. It is observed that people may make typhographical errors in writing like *đằng đẳng* instead of the correct word *đằng đẵng*. In such cases, the computational model can be exploited to detect the potential errors and suggest corrections.

The reduplication model could also help improve the accuracy of Vietnamese lexical recognizers in particular and the accuracy of Vietnamese word segmentation systems in general. The reduplication scanner was integrated to `vnTokenizer`[90] – an open source and highly accurate tokenizer for Vietnamese texts [103].

C.5 Conclusion and future work

We have presented for the first time a computational model for the reduplication of the Vietnamese language. We show that a large class of reduplicative words can be modeled effectively by sequential finite-state string-to-string transducers.

The analysis of the various patterns of reduplication of the Vietnamese language has

89. He has left behind no traces whatsoever. She has been waiting for him for 20 years.
90. *http ://www.loria.fr/∼lehong/tools/vnTokenizer.php*

twofold contributions. On the one hand, it gives useful information on identification of spelling variants in Vietnamese texts. On the other hand, it gives an explicit formalization of precedence relationships in the phonology, and as a result helps ordering and modeling phonological processes before transfer of the presentation to the articulatory interface.

It is argued that the relation between morphology and phonology is an intimate one, both synchronically and diachronically. As mentioned earlier, Vietnamese reduplication is always accompanied by a modification of phone and tone for a symmetric and harmonic posture. We thus believe that the compact finite-state description of a large class of reduplication would help connect morphosyntactic attributes to individual phonological components of a set of Vietnamese word forms and contribute to the improvement of Vietnamese automatic speech recognition systems.

As mentioned earlier, the current work does not handle partial reduplication in which either the onset is repeated or the rhyme and the tone of syllables are repeated, for example *bồng bềnh* (bob), *chúm chím* (open slightly one's lips), *lẩm cẩm* (doting), *lúng túng* (perplexed, embarrassed). Partial reduplication is a topic which has been well studied for a long time by Vietnamese linguists community. It has been shown that partial reduplicative words also have certain principle formation rules [55, 188]. Hence, partial reduplicative words could also be generated and recognized by an appropriate finite-state model which encodes precisely their formation rules. This is an interesting topic of our future work in constructing a rather complete computational model for Vietnamese bi-syllabic reduplication.

Furthermore, in addition to the bi-syllabic reduplication forms, there exists also three or four syllable reduplication forms, for example *cỏn còn con* (very little), *tẹo tèo teo* (very small), or *vội vội vàng vàng* (hurry), *đủng đà đủng đỉnh* (deliberate). These reduplication forms involve the copying operation of morphological structures which is a non-regular operation. Non-regular operations are problematic in that they cannot be cast in terms of composition – the regular operation of major importance in finite-state devices, while finite-state devices cannot handle unbounded copying. However, the question of the possibility for an elegant account to reduce these specific kinds of reduplication to purely regular mechanisms would be of interest for further research to extend and improve the core reduplication components for Vietnamese. Unknown reduplicative word guessing is another interesting and useful topic since the lexicon can never cover all reduplicative words.

Bibliographie

[1] Abeillé, A. The flexibility of French idioms : a representation with lexicalized TAG. In *Idioms : Structural & psychological perspectives*, M. Everaert, A. Schenck, E.-J. van der Linden, and R. Schreuder, Eds. LEA, 1995, pp. 15–42.

[2] Abeillé, A. *Une grammaire électronique du français*. CNRS, Paris, 2002.

[3] Abeillé, A., Candito, M.-H., Daille, B., and Robichaud, B. FTAG : a lexicalized tree adjoining grammar for French. In *Tree adjoining grammars*, A. Abeillé and O. Rambow, Eds. Stanford CSLI, 2000, pp. 303–330.

[4] Abeillé, A., Clément, L., and Toussenel, F. Building a treebank for French. In *Treebanks : Building and Using Parsed Corpora*. Kluwer, Dordrecht, 2003.

[5] Abeillé, A., and Schabes, Y. Non compositional discontinuous constituents in a tree adjoining grammar. In *Discontinuous Constituency*, A. van Horck and H. Bunt, Eds. Mouton De Gruyter, 1996, pp. 113–140.

[6] Abeillée, A., and Blache, P. *Ingénierie des langues*. Hermes Science Europe, 2000, ch. Grammaires et analyseurs syntaxiques.

[7] Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. *Compilers : Principles, Techniques, and Tools, 2nd edition*. Addison Wesley, 2006.

[8] Altun, Y., Johnson, M., and Hofmann, T. Investigating loss function and optimization methods for discriminative learning of label sequences. In *Proceedings of EMNLP* (2003).

[9] Andrew, G., and Gao, J. Scalable training of l_1-regularized log-linear models. In *ICML* (2007), pp. 33–40.

[10] Bangalore, S. Performance evaluation of supertagging for partial parsing. In *Advances in probabilistic and other parsing technologies*. Kluwer Academic Publishers, 2000, pp. 203–220.

[11] Bangalore, S., and Joshi, A. K. Supertagging : An approach to almost parsing. *Computational Linguistics 25*, 2 (1999), 237–265.

[12] Becker, T. A new automation model for TAGs : 2SA. *Computational Intelligence 10*, 4 (1994), 422–430.

[13] Berger, A., Pietra, S. D., and Pietra, V. D. A maximum entropy approach to natural language processing. *Computational Linguistics 22*, 1 (1996), 39–71.

[14] Black, E., Abney, S., Flickenger, D., Gdaniec, C., Grishman, R., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M.,

Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the fourth DARPA Speech and Natural Language Workshop* (Pacific Grove, California, 1991), pp. 306–311.

[15] Borthwick, A. *A Maximum Entropy Approach to Named Entity Recognition*. PhD thesis, New York University, 1999.

[16] Brants, T. TnT - A statistical part-of-speech tagger. In *Proceedings of the Sixth Applied Natural Language Processing Conference (ANLP 2000)* (Seattle, WA, 2000), pp. 224–231.

[17] Bresnan, J., Ed. *The mental representation of grammatical relations*. Cambridge, MIT Press, 1982.

[18] Brill, E. Some advances in transformation-based part-of-speech tagging. In *AAAI* (1994), vol. 1, pp. 722–729.

[19] Butt, M., Holloway, T., Nino, M.-E., and Segond, F. *A grammar writer's cookbook*. Stanford CSLI Publications, 1999.

[20] Candito, M., Crabbé, B., and Seddah, D. On statistical parsing of French with supervised and semi-supervised strategies. In *Proceedings of the EACL 2009 Workshop on Computational Linguistic Aspects of Grammatical Inference* (Morristown, NJ, USA, 2009), Association for Computational Linguistics, pp. 49–57.

[21] Candito, M., Crabbé, B., and Denis, P. Statistical French dependency parsing : Treebank conversion and first results. In *Proceedings of LREC 2010* (Valletta, Malta, 2010).

[22] Candito, M., Crabbé, B., Denis, P., and Guérin, F. Analyse syntaxique du français : des constituants aux dépendances. In *Actes de TALN 2009* (Senlis, France, 2009).

[23] Candito, M. H. *Représentation modulaire et paramétrable de grammaires d'électroniques lexicalisées : application au français et à l'italien*. PhD thesis, Université Paris 7, 1999.

[24] Cao, X. H. *Tiếng Việt - Sơ thảo Ngữ pháp Chức năng*. NXB Giáo dục, 2004.

[25] Caroll, J., Briscoe, T., and Sanfilippo, A. Parser evaluation : a survey and a new proposal. In *Proceedings of LREC 1998* (Granada, Spain, 1998).

[26] Carreras, X., Collins, M., and Koo, T. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of COLING 2008* (Manchester, 2008).

[27] Chen, J., Bangalore, S., and Vijay-Shanker, K. Automated extraction of tree-adjoining grammars from treebanks. *Natural Language Engineering 12*, 3 (2006), 251–299.

[28] Chen, J., and Vijay-Shanker, K. Automated extraction of TAGs from the Penn treebank. In *Proceedings of the Sixth International Workshop on Parsing Technologies* (2000).

[29] Chen, S. F., and Goodman, J. T. An empirical study of smoothing techniques for language modeling. In *The Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics* (1996), pp. 310–318.

[30] Cherkassky, V., and Mulier, F. M. *Learning from data : Concepts, theory and methods*. Wiley-IEEE Press, 2007.

[31] Chiang, D. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the ACL'05* (2005), pp. 263–270.

[32] Chomsky, N. Three models for the description of language. *IEEE Transactions on Information Theory 2* (1956), 113–124.

[33] Chomsky, N. *Lectures on Government and Binding*. Foris, 1991.

[34] Clark, S., and Curran, J. R. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics 33*, 4 (2007), 493–552.

[35] Cohen-Sygal, Y., and Wintner, S. Finite-state registered automata for non-concatenative morphology. *Computational Linguistics 32*, 1 (2006), 49–82.

[36] Cohn, T., and Blunsom, P. Semantic role labeling with tree conditional random fields. In *CONLL* (2005), pp. 169–172.

[37] Collins, M. Three generative, lexicalised models for statistical parsing. In *Proceedings of ACL* (1997).

[38] Collins, M. Discriminative training methods for hidden Markov models : Theory and experiments with perceptron algorithms. In *EMNLP* (2002), pp. 1–8.

[39] Collins, M. Head-driven statistical models for natural language parsing. *Computational Linguistics 29*, 4 (2003), 589–637.

[40] Collins, M., Globerson, A., Koo, T., Carreras, X., and Bartlett, P. L. Exponentiated gradient algorithms for conditional random fields and max-margin Markov networks. *The Journal of Machine Learning Research (JMLR) 9* (2008), 1775–1822.

[41] Copestake, A. *Implementing Typed Feature Structure Grammars*. Stanford CSLI Publications, 2002.

[42] Crabbé, B. *Représentation informatique de grammaires fortement lexicalisées : An application à la grammaire d'arbre adjoints*. PhD thesis, Université Nancy 2, 2005.

[43] Crabbé, B., Gaiffe, B., and Roussanaly, A. Représentation et gestion de grammaires d'arbres adjoints lexicalisées. *Traitement Automatique des Langues 44*, 3 (2003), 67–91.

[44] Daciuk, J., Mihov, S., Watson, B., and Watson, R. Incremental construction of minimal acyclic finite-state automata. *Computational Linguistics 26*, 1 (2000).

[45] Darroch, J. N., and Ratcliff, D. Generalized iterative scaling for log-linear models. *Annals of Mathematical Statistics 43*, 5 (1972), 1470–1480.

[46] de la Clergerie, E. V. DyALog : a tabular logic programming based environment for NLP. In *Proceedings of 2nd International Workshop on Constraint Solving and Language Processing* (Barcelona, Spain, 2005).

[47] de la Clergerie, E. V. Building factorized TAGs with meta-grammars. In *Proceedings of The Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)* (Yale University, New Haven, CT, USA, 2010).

[48] de la Clergerie, E. V., and Alonso, M. A. A tabular interpretation of a class of 2-stack automata. In *Proceedings of COLING-ACL'98* (Montreal, Quebec, 1998).

[49] de la Clergerie, E. V., Hamon, O., Mostefa, D., Ayache, C., Paroubek, P., and Vilnat, A. PASSAGE : from french parser evaluation to large sized treebank. In *Proceedings of LREC 2008* (Marrakech, Marocco, 2008).

[50] de la Clergerie, E. V., Sagot, B., Nicolas, L., and Guénot, M.-L. FRMG : évolution d'un analyseur syntaxique TAG du français. In *Workshop ATALA de IWPT 2009* (Paris, 2009).

[51] de Marneffe, M.-C., MacCartney, B., and Manning, C. D. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006* (Genoa, Italy, 2006).

[52] Denis, P., and Sagot, B. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art POS tagging with less human effort. In *Proceedings of PACLIC 2009* (2009).

[53] Deyi, X., Qun, L., and Shouxun, L. Maximum entropy based phrase reordering model for statistical machine translation. In *Proceedings of the 44th Annual Meeting of the ACL* (2006), pp. 521–528.

[54] Dinh, Q. T., Nguyen, T. M. H., Vu, X. L., Rossignol, M., Le-Hong, P., and Nguyen, C. T. Word segmentation of Vietnamese texts : a comparison of approaches. In *Proceedings of LREC 2008* (Marrakech, Morocco, 2008).

[55] Diệp, Q. B., and Hoàng, V. T. *Ngữ pháp Tiếng Việt (Vietnamese Grammar).* NXB Giáo dục, Hà Nội, Việt Nam, 1999.

[56] Doran, C., Hockey, B., Sarkar, A., and Srinivas, B. Evolution of the XTAG system. In *Tree adjoining grammars*, A. Abeillé and O. Rambow, Eds. Stanford CSLI, 2000, pp. 371–404.

[57] Earley, J. An efficient context-free parsing algorithm. *Communications of the ACM* (1970).

[58] Erbach, G., and Uszkoreit, H. Grammar engineering : problems and prospects – report on the Saarbrücken grammar engineering workshop. Tech. rep., Saarbrücken, Germany, 1990.

[59] Finkel, J. R., Kleeman, A., and Manning, C. Efficient, feature-based, conditional random field parsing. In *ACL* (2008), pp. 959–967.

[60] Frank, R. *Phrase Structure Composition and Syntactic Dependencies.* MIT Press, Boston, 2002.

[61] Frederick, J., and Mercer, R. L. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice* (Amsterdam, The Netherlands, May 1980).

[62] Gao, J., Andrew, G., Johnson, M., and Toutanova, K. A comparative study of parameter estimation methods for statistical natural language learning. In *ACL* (2007), pp. 824–831.

[63] Gardent, C., and Parmentier, Y. Large scale semantic construction for tree adjoining grammars. In *LACL 2005, LNAI 3492* (Berlin, 2005), Springer-Verlag, pp. 131–146.

[64] Gazdar, G., Ewan, K., Pullum, G., and Sag, I. *Generalized Phrase Structure Grammar*. Oxford, Blackwell, 1985.

[65] Goodman, J. Sequential conditional generalized iterative scaling. In *ACL* (2002), pp. 9–16.

[66] Grover, C., Caroll, J., and Briscoe, T. *The Alvey Natural Language Tools Grammar*. Cambridge, 1993.

[67] Habash, N., and Rambow, O. Extracting a tree adjoining grammar from the Penn Arabic treebank. In *Proceedings of TALN'04* (Morocco, 2004).

[68] Haudricourt, A. G. La place du vietnamien dans les langues austrosiatique. *Bulletin de la Société de Linguistique de Paris 49*, 1 (1953).

[69] Haudricourt, A. G. De l'origine des tons en vietnamien. *Journal Asiatique 242*, 1 (1954).

[70] Huang, F.-L., Hsieh, C.-J., Chang, K.-W., and Lin, C.-J. Iterative scaling and coordinate descent methods for maximum entropy. In *ACL-IJCNLP* (2009), pp. 285–288.

[71] ISO. Language Resource Management – syntactic annotation framework. Tech. rep., ISO, TC37/SC4, 2005.

[72] ISO. Language Resource Management - word segmentation of written texts for mono-lingual and multi-lingual information processing - Part I : General principles and methods. Tech. rep., ISO, TC37/SC4, 2006.

[73] Jackendoff, R. S. *X-bar Syntax : A study of phrase structure*. MIT Press, 1977.

[74] Jianfeng, G., Mu, L., Andi, W., and Chang-Ning, H. Chinese word segmentation and named entity recognition : A pragmatic approach. *Computational Linguistics 31*, 4 (2005), 531–574.

[75] Johansen, A.-D. Extraction des grammaires LTAG à partir d'un corpus étiquetté syntaxiquement. Master's thesis, Université Paris 7, 2004.

[76] Johansson, R., and Nugues, P. Dependency-based syntactic–semantic analysis with propbank and nombank. In *CoNLL 2008 : Proceedings of the Twelfth Conference on Computational Natural Language Learning* (Manchester, England, August 2008), Coling 2008 Organizing Committee, pp. 183–187.

[77] John, T. B., Carroll, J., Graham, J., and Copestake, A. Relational evaluation schemes. In *Proceedings of LREC 2002* (Las Palmas, Canary, 2002).

[78] Joshi, A. K., Levy, L. S., and Takahashi, M. Tree adjunct grammars. *Journal of the Computer and System Sciences 10* (1975), 136–165.

[79] Joshi, A. K., and Schabes, Y. *Handbooks of Formal Languages and Automata*. Springer-Verlag, 1997, ch. Tree Adjoining Grammars.

[80] Jousse, F. *Transformatin d'arbres XML avec des modèles probabilistes pour l'annotation*. PhD thesis, Université Charles de Gaulle – Lille 3, 2007.

[81] Jousse, F., Gilleron, R., Tellier, I., and Tommasi, M. Champs conditionnels aléatoires pour l'annotation d'arbres. In *Proceedings of CAP* (2006).

[82] Kahane, S. Grammaire de dépendances formelles et théorie Sens-Texte. In *Actes TALN 2001* (Tours, France, 2001).

[83] Kallmeyer, L., Lichte, T., Maier, W., Parmentier, Y., and Dellert, J. Developping an MCTAG for German with an RCG-based parser. In *Proceedings of LREC 2008* (Marrakech, Morocco, 2008).

[84] Kaplan, R., and Kay, M. Regular models of phonological rule systems. *Computational Linguistics 20*, 3 (1994), 331–378.

[85] Kazama, J., and Tsujii, J. Evaluation and extension of maximum entropy models with inequality constraints. In *EMNLP* (2003).

[86] Kilgarriff, A., Reddy, S., and Pomikálek, J. Corpus factory. In *Proceedings of Asialex* (Bangkok, Thailand, 2009).

[87] Kimmo, K. Two-level morphology : A general computational model for word-form recognition and production. Tech. rep., The Department of General Linguistics, University of Helsinki, 1983.

[88] King, T. H., Crouch, R., Riezler, S., Dalrymple, M., and Kaplan, R. M. The PARC 700 dependency bank. In *Proceedings of 4th International Workshop on Linguistically Interpreted Corpora* (Budapest, Hungary, 2003).

[89] Kinyon, A., and Prolo, C. A. A classification of grammar development strategies. In *Proceedings of the Workshop on Grammar Engineering and Evaluation* (Taipei, Taiwan, 2002), pp. 43—49.

[90] Kiss, T., and Strunk, J. Unsupervised multilingual sentence boundary detection. *Computational Linguistics 32*, 4 (2006), 485–525.

[91] Klein, D., and Manning, C. D. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics* (Sapporo, Japan, July 2003), Association for Computational Linguistics, pp. 423–430.

[92] Koo, T., and Collins, M. Efficient third-order dependency parsers. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (Uppsala, Sweden, July 2010), Association for Computational Linguistics, pp. 1–11.

[93] Kroch, A. S., and Joshi, A. K. The linguistic relevance of tree adjoining grammars. Tech. rep., University of Pennsylvania, 1985.

[94] Lafferty, J., McCallum, A., and Pereira, F. Conditional random fields : Probabilistic models for segmenting and labeling sequence data. In *ICML* (2001), pp. 282–289.

[95] Lavergne, T., Cappé, O., and Yvon, F. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics* (Uppsala, Sweden, July 2010), Association for Computational Linguistics, pp. 504–513.

[96] Le, A.-C., Nguyen, P.-T., Vuong, H.-T., Pham, M.-T., and Ho, T.-B. An experimental study on lexicalized statistical parsing for Vietnamese. In *Proceedings of the International Conference on Knowledge and Systems Engineering* (2009), IEEE Computer Society, pp. 162–167.

[97] Le-Hong, P. An empirical study of maximum entropy approach for part-of-speech tagging of Vietnamese texts. In *Proceedings of Traitement Automatique des Langues Naturelles (TALN-2010)* (Montreal, Canada, 2010).

[98] Le-Hong, P., and Ho, T. V. A maximum entropy approach to sentence boundary detection of Vietnamese texts. In *Proceedings of IEEE International Conference on Research, Innovation and Vision for the Future – RIVF 2008* (Vietnam, 2008).

[99] Le-Hong, P., Nguyen, T. M. H., Nguyen, P. T., and Phan, T. H. Automated extraction of tree adjoining grammars from a treebank for Vietnamese. *Journal of Informatics and Cybermetics* (2010). accepted for publication.

[100] Le-Hong, P., Nguyen, T. M. H., Nguyen, P. T., and Roussanaly, A. Automated extraction of tree adjoining grammars from a treebank for Vietnamese. In *Proceedings of The Tenth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+10)* (Yale University, New Haven, CT, USA, 2010).

[101] Le-Hong, P., Nguyen, T. M. H., and Roussanaly, A. A metagrammar for Vietnamese LTAG. In *Proceedings of The Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+9)* (Tubingen, Germany, 2008).

[102] Le-Hong, P., Nguyen, T. M. H., and Roussanaly, A. Finite-state description of Vietnamese reduplication. In *Proceedings of the 7th Workshop on Asian Language Resources, ACL-IJCNLP* (Suntec City, Singapore, 2009).

[103] Le-Hong, P., Nguyen, T. M. H., Roussanaly, A., and Ho, T. V. A hybrid approach to word segmentation of Vietnamese texts. In *Proceedings of the 2nd International Conference on Language and Automata Theory and Applications*, M.-V. Carlos, Ed. Springer, LNCS 5196, Tarragona, Spain, 2008.

[104] Le-Hong, P., Phan, T. H., Nguyen, T. M. H., and Van, T. L. Building a syntactic annotation framework for Vietnamese. In *Proceedings of the Third National Symposium on Research, Development and Application of Information and Com- munication Technology* (Hanoi, Vietnam, 2008).

[105] Le-Hong, P., and Roussanaly, A. Une approche entropie maximale pour le découpage en phrases des textes en français. Tech. rep., LORIA, 2009.

[106] Lehmann, S., Oepen, S., Regnier-Prost, S., Netter, K., Lux, V., Klein, J., Falkedal, K., Fouvry, F., Estival, D., Dauphin, E., Compagnion, H., Baur, J., Balkan, L., and Arnold, D. TSNLP : Test suites for natural language processing. In *Proceedings of the 16th conference on Computational linguistics* (Morristown, NJ, USA, 1996), ACL, pp. 711–716.

[107] Li, C. N., and Thompson, S. A. Subject and topic : a new typology of language. In *Subject and topic*. London/New York : Academic Press, 1976, pp. 457–489.

[108] Li, R., Tao, X., Tang, L., and Hu, Y. Using maximum entropy model for Chinese text categorization. In *LNCS - Advanced Web Technologies and Applications*. Springer, 2004, pp. 578–587.

[109] Lin, D. A dependency-based method for evaluating broad-coverage parsers. In *In Proceedings of IJCAI-95* (1995), pp. 1420–1425.

[110] Lopez, P. *Analyse d'énonces oraux pour le dialogue homme-machine à l'aide de grammaires lexicalisées d'arbres*. PhD thesis, Université Henri Poincaré, 1999.

[111] Luo, X. A maximum entropy chinese character-based parser. In *Proceedings of Conference on Empirical Methods in Natural Language Processing* (2003), vol. 10, pp. 192–199.

[112] Magerman, D. M. Statistical decision tree models for parsing. In *Proceedings of ACL* (1995).

[113] Malouf, R. A comparison of algorithms for maximum entropy parameter estimation. In *CONLL* (2002).

[114] Marcus, M. M., Santorini, B., and Marcinkiewicz, M. A. Building a large annotated corpus of English. *Computational Linguistics 19*, 2 (1993).

[115] Martín-Vide, C., and Mitrana, V., Eds. *Electronic Dictionaries and Acyclic Finite-State Automata : A State of The Art* (2003), vol. 9 of *Topics in Computer Mathematics*, Taylor and Francis.

[116] McCallum, A., Freitag, D., and Pereira, F. Maximum entropy Markov models for information and segmentation. In *Proceedings of ICML* (2000).

[117] McCallum, A., and Li, W. Early results for named entity recognition with conditional random fields. In *Proceedings of CoNLL-2003* (Edmonton, Canada, 2003).

[118] McDonald, R., and Pereira, F. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006* (2006).

[119] Mohri, M. On some applications of finite-state automata theory to natural language processing. *Natural Language Engineering 2*, 1 (1996), 61–80.

[120] Mohri, M. Finite-state transducers in language and speech processing. *Computational Linguistics 23* (1997).

[121] Monceaux, L., and Vilnat, A. Evaluation, projection et combinaison d'analyses syntaxiques robustes. *Traitement Automatique des Langues 44*, 3 (2003), 187–214.

[122] Moreau, E., and Tellier, I. The Crotal SRL system : a generic tool based on tree-structured CRF. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL 2009) : Shared Task* (Boulder, Colorado, June 2009), Association for Computational Linguistics, pp. 91–96.

[123] Moreau, E., Tellier, I., Balvet, A., and Laurence, G. Annotation fonctionnelle de corpus arborés avec des champs aléatoires conditionnels. In *Actes de TALN 2009* (Senlis, France, 2009).

[124] Nasr, A. *Analyse syntaxique probabiliste pour grammaires de dépendances extraites automatiquement*. Habilitation à diriger des recherches, Université Paris 7, 2004.

[125] Nederhof, M.-J. An alternative LR algorithm for TAGs. In *Proceedings of ACL-COLING'98* (1998), pp. 946–952.

[126] Neumann, G. A uniform method for automatically extracting stochastic lexicalized tree grammar from treebank and HPSG. In *Treebanks : Building and Using Parsed Corpora*. Kluwer, Dordrecht, 2003.

[127] Nguyen, P. T., Xuan, L. V., Nguyen, T. M. H., Nguyen, V. H., and Le-Hong, P. Building a large syntactically-annotated corpus of Vietnamese. In *Proceedings of the 3rd Linguistic Annotation Workshop, ACL-IJCNLP* (Singapore, 2009).

[128] Nguyen, T. M. H. *Outils et ressources linguistiques pour l'alignement de textes multilingues français-vietnamiens*. PhD thesis, Université Henri Poincaré, Nancy I, 2006.

[129] Nguyen, T. M. H., Romary, L., Rossignol, M., and Vu, X. L. A lexicon for Vietnamese language processing. *Language Resources and Evaluation 40*, 3–4 (2006).

[130] Nguyen, T. M. H., Vu, X. L., and Le-Hong, P. A case study of the probabilistic tagger QTAG for tagging Vietnamese texts. In *Proceedings of the 1st National Conference ICT RDA* (2003).

[131] Nguyễn, T. C. *Ngữ pháp Tiếng Việt (Vietnamese Grammar)*. NXB Đại học Quốc gia Hà Nội, Hà Nội, Vietnam, 1998.

[132] Nguyễn, T. G., Đoàn, T. T., and Nguyễn, M. T. *Dẫn luận ngôn ngôn ngữ học*. NXB Giáo Dục, 2007.

[133] Nigam, K., Lafferty, J., and McCallum, A. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering* (1999), pp. 61–67.

[134] Ninomiya, T., Miyao, Y., and Matsuzaki, T. A log-linear model with an n-gram reference distribution for accurate HPSG parsing. In *In Proc. IWPT 2007* (2007).

[135] Nivre, J., Hall, J., and Nilsson, J. MaltParser : A data-driven parser generator for dependency parsing. In *Proceedings of LREC 2006* (2006).

[136] Nocedal, J. Updating quasi-Newton matrice with limited storage. *Mathematics of Computation 35*, 151 (1980), 773–782.

[137] Nocedal, J., and Wright, S. J. *Numerical Optimization*. Springer, New York, 1999.

[138] Park, J. Extraction of tree adjoining grammars from a treebank for Korean. In *COLING ACL'06 Student Research Workshop* (Morristown, NJ, USA, 2006), pp. 73–78.

[139] Parmentier, Y. *SemTAG : Une plate-forme pour le calcul sémantique à partir de grammaires d'arbres adjoints*. PhD thesis, Université Henri Poincaré, Nancy I, 2007.

[140] Parmentier, Y. The Nancy Parsing Toolkit : an Overview. Tech. rep., LORIA, 2009.

[141] Paroubek, P., Pouillot, L. G., Robba, I., and Vilnat, A. EASY : Campagne d'évaluation des analyseurs syntaxiques. In *Proceedings of TALN 2005* (Dourdan, France, 2005), pp. 3–12.

[142] Paroubek, P., Robba, I., Vilnat, A., and Ayache, C. EASY, evaluation of parsers of French : what are the results ? In *Proceedings of LREC 2008* (Marrakech, Marocco, 2008).

[143] Pietra, S. D., Pietra, V. D., and Lafferty, J. Inducing features of random fields. *IEEE PAMI 19*, 4 (1997), 380–393.

[144] Pinto, D., McCallum, A., Wei, X., and Croft, W. B. Table extraction using conditional random fields. In *Proceedings of SIGIR* (Toronto, Canada, 2003).

[145] Pollard, C., and Sag, I., Eds. *Head-driven Phrase Structure Grammar*. University of Chicago Press, 1994.

[146] Prolo, C. An efficient LR parser generator for TAGs. In *Proceedings of IWPT* (Trento, Italy, 2000).

[147] Rambow, O., and Joshi, A. A formal look at dependency grammars and phrase-structure grammars, with special consideration of word-order phenomena. In *Current Issues in Meaning-Text Theory*. Pinter, London, 1994.

[148] Ratnaparkhi, A. A maximum entropy model for part-of-speech tagging. In *EMNLP* (1996), pp. 133–142.

[149] Ratnaparkhi, A. A simple introduction to maximum entropy models for natural language processing. Tech. rep., IRCS Report 97-98, University of Pennsylvania, PA, USA, 1997.

[150] Reynar, J., and Ratnaparkhi, A. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing* (1997), pp. 16–19.

[151] Romary, L. Outils d'accès à des resources linguistiques. In *Ingénierie des Langues*. Hermes Science Europe, 2000.

[152] Rosenfeld, R. A maximum entropy approach to adaptive statistical language modeling. *Computer, Speech and Language 10* (1996), 187–228.

[153] Roussanaly, A., Crabbé, B., and Perrin, J. Premier bilan de la participation du LORIA à la campagne d'évaluation EASY. In *Proceedings of TALN 2005* (Dourdan, France, 2005).

[154] Sarkar, A. Practical experiments in parsing using TAGs. In *Proceedings of TAG+5* (Paris, 2000).

[155] Schabes, Y. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, University of Pennsylvania, 1990.

[156] Schabes, Y., and Joshi, A. K. An Early type parsing algorithm for tree adjoining grammars. In *Proceedings of ACL'88* (Buffalo, 1988).

[157] Schabes, Y., and Vijay-Shanker, K. Deterministic left to right parsing of tree adjoining languages. In *Proceedings of ACL'90* (Pittsburgh, 1990).

[158] Schabes, Y., and Waters, R. C. Tree insertion grammar : A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics* (1995).

[159] Schmid, H. Tokenizing. In *Corpus Linguistics. An International Handbook*, A. Lüdeling and M. Kytö, Eds. Mouton de Gruyter, Berlin, 2007.

[160] Sha, F., and Pereira, F. Shallow parsing with conditional random fields. In *Proceedings of HLT-NAACL* (2003).

[161] Sutton, C., and McCallum, A. *Introduction to Statistical Relational Learning*. MIT Press, 2006, ch. An Introduction to Conditional Random Fields for Relational Learning.

[162] Tokunaga, T. Developing international standards of language resources for semantic web applications. Tech. rep., Research Report of the International Joint Research Program (NEDO Grant) for FY 2007, 2008. http ://www.tech.nedo.go.jp/PDF/100013569.pdf.

[163] Tokunaga, T., Kaplan, D., Huang, C.-R., Hsieh, S.-K., Calzolari, N., Monachini, M., Soria, C., Shirai, K., Sornlertlamvanich, V., Charoenporn, T., and Xia, Y. Adapting international standard for Asian language technologies. In *Proceedings of The 6th International Conference on Language Resources and Evaluation (LREC 2008)* (2008).

[164] Tou, N. H., Kiat, L. J., and Wenyua, G. A maximum entropy approach to Chinese word segmentation. In *Proceedings of the Fourth SIGHAN Workshop on Chinese Language Processing* (Korea, 2005), pp. 161–164.

[165] Toutanova, K., Haghighi, A., and Manning, C. Joint learning improves semantic role labeling. In *ACL* (2005), pp. 589–596.

[166] Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. Feature-rich part-of-speech tagging with a cyclic dependency network. In *HLT-NAACL* (2003).

[167] Toutanova, K., and Manning, C. D. Enriching the knowledge sources used in a maximum entropy part-of-speech tagger. In *EMNLP/VLC* (2000), pp. 63–71.

[168] Tran, D. D., Castelli, E., Serignat, J. F., Trinh, V. L., and Le, X. H. Linear f_0 contour model for Vietnamese tones and Vietnamese syllable synthesis with TD-PSOLA. In *Proceedings of TAL2006* (La Rochelle, France, 2006).

[169] Tsuruoka, Y., Tsujii, J., and Ananiadou, S. Fast full parsing by linear-chain conditional random fields. In *EACL* (Athens, Greece, 2009), pp. 790–798.

[170] Tsuruoka, Y., Tsujii, J., and Ananiadou, S. Stochastic gradient descent training for L_1-regularized log-linear models with cumulative penalty. In *ACL-IJCNLP* (2009), pp. 477–485.

[171] Uchimoto, K., Sekine, S., and Isahara, H. Japanese dependency structure analysis based on maximum entropy models. In *Proceedings of the Ninth Conference on European Chapter of the Association for Computational Linguistics* (Norway, 1999), pp. 196–203.

[172] Uchimoto, K., Sekine, S., and Isahara, H. The unknown word problem : A morphological analysis of Japanese using maximum entropy aided by dictionary.

In *Conference on Empirical Methods in Natural Language Processing* (Pittsburgh, USA, 2001).

[173] Van-Noord, G. Head-corner parsing for TAG. *Computational Intelligence 10*, 4 (1994).

[174] Vapnik, V. *Estimation of Dependences Based on Empirical Data*. Springer, 2006.

[175] Vijay-Shanker, K. *A study of tree adjoining grammars*. PhD thesis, University of Pennsylvania, 1987.

[176] Vijay-Shanker, K., and Joshi, A. K. Some computational properties of tree adjoining grammars. In *Proceedings of ACL 23rd* (Chicago, Illinois, 1985).

[177] Vijay-Shanker, K., and Joshi, A. K. Feature structure based tree adjoining grammars. In *Proceedings of COLING 1988* (Budapest, 1988).

[178] Vu, T. T., Nguyen, D. T., Luong, M. C., and Hosom., J.-P. Vietnamese large vocabulary continuous speech recognition. In *Proceedings of Eurospeech 2005* (Lisboa, 2005).

[179] Wong, P., and Chan, C. Chinese word segmentation based on maximum matching and word binding force. In *Proceedings of the 16th Conference on Computational Linguistics* (Copenhagen, Danmark, 1996).

[180] Xia, F. *Automatic grammar generation from two different perspectives*. PhD thesis, University of Pennsylvania, 2001.

[181] Xia, F., Palmer, M., and Joshi, A. A uniform method of grammar extraction and its applications. In *Proceedings of the joint SIGDAT conference on empirical methods in NLP and very large corpora* (Morristown, NJ, USA, 2000), pp. 53–62.

[182] XTAG-Research-Group. A lexicalized tree adjoining grammar for English. Tech. rep., Institute for Research in Cognitive Science, University of Pennsylvania, 2001.

[183] Yoon, S. W. Using a meta-grammar for LTAG Korean grammar. In *Proceedings of TAG+7* (Vancouver, BC, CA, 2004).

[184] Zhao, J., and Wang, X. L. Chinese POS tagging based on maximum entropy model. In *Proceedings of the First International Conference on Machine Learning and Cybernetics* (Beijing, 2002).

[185] Đoàn, T. T. *Ngữ âm tiếng Việt (Vietnamese Phonetics)*. NXB Đại học Quốc gia Hà Nội, Hà Nội, Việt Nam, 2003.

[186] Đoàn, T. T., Nguyễn, K. H., and Phạm, N. Q. *A Concise Vietnamese Grammar (For Non-native Speakers)*. Thế Giới Publishers, Hà Nội, Việt Nam, 2003.

[187] Đạt Hữu, Trần, T. D., and Đào, T. L. *Cơ sở tiếng Việt (Basis of Vietnamese)*. NXB Giáo dục, Hà Nội, Việt Nam, 1998.

[188] Ủy ban Khoa học Xã hội Việt Nam, Ed. *Ngữ pháp tiếng Việt (Vietnamese Grammar)*. Nhà xuất bản Khoa học Xã hội – Hà Nội, Việt Nam, 1983.

Résumé

Cette thèse s'inscrit dans le domaine du traitement automatique des langues naturelles et plus spécifiquement dans celui du traitement du vietnamien. Le travail présenté dans la thèse porte sur la construction d'outils et de ressources linguistiques pour les tâches fondamentales de traitement automatique du vietnamien, notamment la construction d'une grammaire à large couverture et un analyseur syntaxique pour cette langue. Nous développons une chaîne modulaire de prétraitements pour le vietnamien dont le rôle est d'appliquer à des corpus bruts une cascade de traitements de surface. Il s'agit d'un segmenteur en phrases, d'un segmenteur en unités lexicales, d'un reconnaisseur de mots redoublés et d'un étiqueteur morpho-syntaxique. Préalables nécessaires à une possible analyse, ces traitements peuvent également servir à préparer d'autres tâches. La modélisation de la grammaire vietnamienne est effectuée en utilisant le formalisme des grammaires d'arbres adjoints lexicalisées (Lexicalized Tree Adjoining Grammars ou LTAG). Nous développons un système qui extrait automatiquement une grammaire LTAG à partir d'un corpus arboré du vietnamien. Les arbres élémentaires de la grammaire forment les structures syntaxiques de la langue vietnamienne. Nous adaptons et enrichissons un analyseur syntaxique du français pour construire un analyseur syntaxique profond pour le vietnamien. Nous présentons les fondements théoriques des différents modules et systèmes, leurs évaluations quantitatives. Nos systèmes atteignent des performances prometteuses dans les tâches du traitement automatique du vietnamien à l'heure actuelle.

Mots clés : traitement automatique des langues, analyse syntaxique, étiquetage syntaxique, segmentation, vietnamien, grammaire d'arbres adjoints, apprentissage supervisé, modèle de l'entropie maximale, automate à états finis.

Abstract

This thesis deals with the construction of linguistic resources and tools for the automatic processing of the Vietnamese language. The central research topic of the thesis is the development of a syntactic component including a broad-coverage grammar and a deep syntactic parser for this language. We have developed a modular and customizable chain aimed to apply to raw texts a cascade of surface processing steps including automatic sentence detection, word segmentation and part-of-speech tagging. Necessarily preliminary steps before parsing, they can be also used to prepare other tasks. The Vietnamese grammar is modeled using the Lexicalized Tree Adjoining Grammar (LTAG) formalism. We have developed a system which extracts automatically a grammar LTAG from a treebank for Vietnamese. The tree templates of this grammar cover the most frequent syntactic structures of the Vietnamese language. We have implemented a deep syntactic parser for Vietnamese which is able to give both constituency and dependency analysis of a sentence. We describe theoretical foundations of the system and its modules, their quantitative evaluations. Our system has good performances on related tasks, some modules have the best result ever published for the Vietnamese language.

Keywords : natural language processing, syntactic parsing, part-of-speech tagging, word segmentation, Vietnamese, tree-adjoining grammars, supervised learning, maximum entropy model, finite-state automata.